RO-SERIE

Hardware-Beschreibung

2012 Juli

INDEX

1. Einleitung 13

1.1. Vorwort 13

1.2. Kundenzufriedenheit 13

1.3. Kundenresonanz 13

2. Hardware Beschreibung 15

2.1. Ethernet Interface 15

2.1.1. Hardware Beschreibung 15

2.1.1.1. Übersichtsbild 15

2.1.1.2. Technische Daten 17

2.1.1.3. Steckverbinder auf dem Modul 18

2.1.1.3.1. Spannungsversorgung 18

2.1.1.3.2. Ethernet Interface 18

2.1.1.4. Taster auf dem Modul 19

2.1.1.5. Kontroll LED's 20

2.1.1.5.1. Definition der LEDs 20

2.1.2. Konfiguration des Moduls 21

2.1.2.1. Konfiguration über das DELIB Configuration Utility 21

2.1.2.2. Konfiguration über den internen Web-Server des Moduls 26

2.1.2.3. Auslieferungszustand 27

2.1.3. Grundkonfiguration wiederherstellen 28

2.1.3.1. IP Adresse zurücksetzen 28

2.1.3.2. Firmware zurücksetzen 28

2.2. CAN Interface 29

2.2.1. Hardware Beschreibung 29

2.2.1.1. Übersichtsbild 29

2.2.1.2. Technische Daten 30

2.2.1.3. Steckverbinder auf dem Modul 31

2.2.1.3.1. Spannungsversorgung 31

2.2.1.3.2. CAN Interface 31

2.2.1.4. Kontroll LED's 32

2.2.1.4.1. Definition der LEDs 32

2.2.2. Konfiguration des Moduls 33

INDEX

2.2.2.1. DIP-Schalter 33
2.2.2.2. Der Vorzugsmodus 34
2.2.2.3. Software-Modus 35
2.2.2.4. DIP-Schalter-Modus 37
2.2.2.4.1. Einstellen der Übertragungsgeschwindigkeit (Bitrate)
2.2.2.4.2. Einstellen der CAN-Moduladresse 38
2.3. RS-232/RS-485 Interface 40
2.3.1. Hardware Beschreibung 40
2.3.1.1. Übersichtsbild 40
2.3.1.2. Technische Daten 41
2.3.1.3. Wählen der Schnittstellenvariante RS-232 oder RS-485 42
2.3.1.4. Steckverbinder auf dem Modul 44
2.3.1.4.1. Spannungsversorgung 44
2.3.1.4.2. RS-232/RS-485 Interface 44
2.3.1.4.2.1RS-232 Pinbelegung 45
2.3.1.4.2.2RS-485 Pinbelegung 45
2.3.1.5. Kontroll LED's 46
2.3.1.5.1. Definition der LEDs 46
2.3.2. Konfiguration des Moduls 47
2.3.2.1. DIP-Schalter 47
2.3.2.2. Der Vorzugsmodus 48
2.3.2.3. Echo aktivieren 49
2.3.2.4. Einstellen der Baudrate 49
2.3.2.5. Einstellen der Moduladresse (nur bei RS-485) 50
2.4. USB Interface 51
2.4.1. Hardware Beschreibung 51
2.4.1.1. Übersichtsbild 51
2.4.1.2. Technische Daten 53
2.4.1.3. Steckverbinder auf dem Modul 54
2.4.1.3.1. Spannungsversorgung 54
2.4.1.3.2. USB Interface 54
2.4.1.4. Kontroll LED's 55
2.4.1.4.1. Definition der LEDs 55
2.5. Digitale Ein-/Ausgabemodule 56
2.5.1. Hardware Beschreibung 56
2.5.1.1. RO-O16 57

INDEX

2.5.1.1.1. Übersichtsbild 57
2.5.1.1.2. Technische Daten 58
2.5.1.1.3. 16-Bit Zähler 59
2.5.1.1.4. Erfassen von schnellen Eingangsimpulsen
2.5.1.1.5. Galvanische Trennung durch Optokoppler
2.5.1.1.6. Steckverbinder auf dem Modul 60
2.5.1.1.6.1Leitungsanschluss 60
2.5.1.1.6.2Visuelle Kontrolle der Eingänge
2.5.1.1.6.3Pinbelegung 61
2.5.1.1.7. Eingangsspannungsbereich variierbar 61
2.5.1.1.7.1Ändern der Eingangsspannung
2.5.1.2. RO-R16 63
2.5.1.2.1. Übersichtsbild 63
2.5.1.2.2. Technische Daten 64
2.5.1.2.3. Timeout-Schutz 65
2.5.1.2.4. Steckverbinder auf dem Modul 65
2.5.1.2.4.1Relais Ausgänge (galvanisch getrennt, max 1A)
2.5.1.2.4.2Leitungsanschluss 66
2.5.1.2.4.3Visuelle Kontrolle der Ausgänge
2.5.1.2.4.4Pinbelegung 66
2.5.1.3. RO-M16 67
2.5.1.3.1. Übersichtsbild 67
2.5.1.3.2. Technische Daten 68
2.5.1.3.3. Timeout-Schutz 69
2.5.1.3.4. Steckverbinder auf dem Modul 69
2.5.1.3.4.1Optokoppler Ausgänge (galvanisch getrennt, max
2.5.1.3.4.2Leitungsanschluss 70
2.5.1.3.4.3Pinbelegung 70
2.5.1.4. RO-O8-R8 71
2.5.1.4.1. Übersichtsbild 71
2.5.1.4.2. Technische Daten 72
2.5.1.4.3. Optokoppler Eingänge 73
2.5.1.4.3.116-Bit Zähler 73
2.5.1.4.3.2Erfassen von schnellen Eingangsimpulsen
2.5.1.4.3.3Galvanische Trennung durch Optpkoppler
2.5.1.4.4. Relais Ausgänge 74
2.5.1.4.4.1Timeout-Schutz 74
2.5.1.4.5. Steckverbinder auf dem Modul 75
2.5.1.4.5.1Leitungsanschluss und Pinbelegung
2.5.1.4.5.2Visuelle Kontrolle der Ein- und Ausgänge

INDEX

2.6. Analoge Ein-/Ausgabemodule 77

2.6.1. Hardware Beschreibung 77
2.6.1.1. RO-AD16-DA4 77
2.6.1.1.1. Übersichtsbild 78
2.6.1.1.2. Technische Daten 79
2.6.1.1.3. Timeout-Schutz 80
2.6.1.1.4. Pinbelegung 81
2.6.1.1.4.1A/D Steckverbinder (18pol) 81
2.6.1.1.4.2D/A Steckverbinder (10pol) 81
2.6.1.2. RO-AD16 82
2.6.1.2.1. Übersichtsbild 82
2.6.1.2.2. Technische Daten 83
2.6.1.2.3. Pinbelegung 84
2.6.1.2.3.1A/D Steckverbinder (18pol) 84
2.6.1.3. RO-AD16_ISO 85
2.6.1.3.1. Übersichtsbild 85
2.6.1.3.2. Technische Daten 86
2.6.1.3.3. Pinbelegung 87
2.6.1.3.3.1A/D Steckverbinder (18pol) 87
2.6.1.4. RO-DA4 88
2.6.1.4.1. Übersichtsbild 88
2.6.1.4.2. Technische Daten 89
2.6.1.4.3. Timeout-Schutz 90
2.6.1.4.4. Pinbelegung 90
2.6.1.4.4.1D/A Steckverbinder (10pol) 90
2.6.1.5. RO-DA2_ISO 91
2.6.1.5.1. Übersichtsbild 91
2.6.1.5.2. Technische Daten 92
2.6.1.5.3. Timeout-Schutz 93
2.6.1.5.4. Pinbelegung 94
2.6.1.5.4.1D/A Steckverbinder (10pol) 94

2.7. Stepper Modul 95

2.7.1. Hardware Beschreibung 95
2.7.1.1. Übersichtsbild 95
2.7.1.2. Technische Daten 96
2.7.1.3. Ansteuerung 96
2.7.1.4. Pinbelegung 97
2.7.1.4.1. Stepper Steckverbinder (10pol) 97

INDEX

2.8. RO-CNT8 98

2.8.1. Technische Daten 98
2.8.2. Counter Modul Modi 99
2.8.3. Blockschaltbild 100
2.8.4. Pinbelegung 101
2.8.4.1. Pinbelegung RO-CNT8 101

2.9. RO-CNT/IGR 102

2.9.1. Technische Daten 102
2.9.2. Blockschaltbild 103
2.9.3. Auswertung Phi 0/Phi 90° (Grafisch) 105
2.9.4. LEDs 106
2.9.5. Eingänge 107
2.9.5.1. Referenzieren 107
2.9.5.2. Visuelle Kontrolle der Eingänge 107
2.9.6. Pinbelegung 108
2.9.6.1. Pinbelegung RO-CNT/IGR 108

2.10. RO-PT100-4 / RO-PT100-8 109

2.10.1. Einführung 109
2.10.2. Technische Daten 110
2.10.3. Blockschaltbild 111
2.10.4. Pinbelegung 112
2.10.4.1. Pinbelegung RO-PT100 112

2.11. Abmessung der RO-Serie 113

2.11.1. Top view 113
2.11.2. Side view 114

3. Firmware Update 116

3.1. Über DEDITEC-Firmware 116

3.2. DEDITEC Flasher 117

3.2.1. Flashen der RO-Serie 118
3.2.2. Flashen anderer Produkte (nicht RO-Serie) 120

4. Software 122

4.1. Benutzung unserer Produkte 122

INDEX

4.1.1. Ansteuerung über grafische Anwendungen 122
4.1.2. Ansteuerung über unsere DELIB Treiberbibliothek 122
4.1.3. Ansteuerung auf Protokollebene 122
4.1.4. Ansteuerung über mitgelieferte Testprogramme 123
4.2. DELIB Treiberbibliothek 124

4.2.1. Übersicht 124
4.2.1.1. Programmieren unter diversen Betriebssystemen 124
4.2.1.2. Programmieren mit diversen Programmiersprachen 125
4.2.1.3. Schnittstellenunabhängiges programmieren 125
4.2.1.4. SDK-Kit für Programmierer 125
4.2.2. Unterstützte Betriebssysteme 126
4.2.3. Unterstützte Programmiersprachen 126
4.2.4. Installation DELIB-Treiberbibliothek 127
4.2.5. DELIB Configuration Utility 131
4.2.5.1. Konfiguration RO-ETH-Serie 132
4.2.5.2. Konfiguration RO-CAN-Serie 137
4.2.5.3. Konfiguration RO-SER-Serie 139
4.2.5.4. Konfiguration RO-USB-Serie und USB-Module 140
4.2.5.4.1. Beispiel zur Konfiguration identischer USB-Module
4.3. DELIB Sample Sources (Programmierbeispiele) 144

4.3.1. Installation DELIB Sample Sources 145
4.3.2. Benutzung der DELIB Sample Sources 149
4.3.2.1. Schritt 1 - Produktauswahl 150
4.3.2.2. Schritt 2 - Kategorieauswahl 151
4.3.2.3. Schritt 3 - Programmiersprachenauswahl 152
4.3.2.4. Schritt 4 - Quellcode 153
4.4. Einbinden der DELIB in Programmiersprachen 156

4.4.1. Einbinden der DELIB in Visual-C/C++ 156
4.4.2. Einbinden der DELIB in Visual-C# 159
4.4.3. Einbinden der DELIB in Visual-C# unter Windows 64 Bit 160
4.4.4. Einbinden der DELIB in Delphi 164
4.4.5. Einbinden der DELIB in Visual-Basic (VB) 165
4.4.6. Einbinden der DELIB in Visual-Basic.NET (VB.NET) 166
4.4.7. Einbinden der DELIB in MS-Office (VBA) 167
4.4.8. Einbinden der DELIB in LabVIEW 169

INDEX

4.4.8.1. Einbinden der DELIB in LabVIEW 169
4.4.8.2. Verwendung der VIs in LabVIEW 178
4.5. Testprogramme 180

4.5.1. Übersicht Testprogramme 180
4.5.2. Analog Input Demo 181
4.5.3. Analog Output Demo 183
4.5.4. Counter Demo 185
4.5.5. Counter-48 Bit Demo 187
4.5.6. Digital Input-Output Demo 189
4.5.7. Digital-PWM-Output Demo 191
4.5.8. Pulse-Generate Output Demo 193
4.5.9. Stepper Demo 194
4.5.10. Temperature Read Demo 196
4.6. DELIB CLI (command-line interface) 197

4.6.1. Anpassung für USB-Module (nur Linux) 199
4.6.2. Anpassungen für RO-ETH-Module (nur Linux) 200
4.6.3. DELIB CLI Beispiele 201
5. DELIB API Referenz 206

5.1. Verwaltungsfunktionen 206

5.1.1. DapiOpenModule 206
5.1.2. DapiCloseModule 207
5.1.3. DapiGetDELIBVersion 208
5.1.4. DapiSpecialCMDGetModuleConfig 209
5.1.5. DapiOpenModuleEx 213
5.2. Fehlerbehandlung 214

5.2.1. DapiGetLastError 214
5.2.2. DapiGetLastErrorText 215
5.3. Digitale Eingänge lesen 216

5.3.1. DapiDIGet1 216
5.3.2. DapiDIGet8 217
5.3.3. DapiDIGet16 218
5.3.4. DapiDIGet32 219
5.3.5. DapiDIGet64 220
5.3.6. DapiDIGetFF32 221

INDEX

5.3.7. DapiDIGetCounter 222
5.3.8. DapiSpecialCounterLatchAll 223
5.3.9. DapiSpecialCounterLatchAllWithReset 224
5.3.10. Dapi_Special_DI_FF_Filter_Value_Get 225
5.3.11. Dapi_Special_DI_FF_Filter_Value_Set 226
5.4. Digitale Ausgänge verwalten 227

5.4.1. DapiDOSet1 227
5.4.2. DapiDOSet8 228
5.4.3. DapiDOSet16 229
5.4.4. DapiDOSet32 230
5.4.5. DapiDOSet64 231
5.4.6. DapiDOSet1_WithTimer 232
5.4.7. DapiDOReadback32 233
5.4.8. DapiDOReadback64 234
5.5. Digitale Zähler Funktionen 235

5.5.1. DapiCnt48ModeSet 235
5.5.2. DapiCnt48ModeGet 238
5.5.3. DapiCnt48CounterGet32 239
5.5.4. DapiCnt48CounterGet48 240
5.5.5. DapiSpecialCNT48ResetSingle 241
5.5.6. DapiSpecialCNT48ResetGroup8 242
5.5.7. DapiSpecialCNT48LatchGroup8 243
5.6. Pulsgenerator Ausgänge verwalten 244

5.6.1. DapiPulseGenSet 244
5.7. A/D Wandler Funktionen 245

5.7.1. DapiADSetMode 245
5.7.2. DapiADGetMode 247
5.7.3. DapiADGet 248
5.7.4. DapiADGetVolt 249
5.7.5. DapiADGetmA 250
5.8. D/A Ausgänge verwalten 251

5.8.1. DapiDASetMode 251
5.8.2. DapiDAGetMode 252
5.8.3. DapiDASet 253
5.8.4. DapiDASetVolt 254

INDEX

5.8.5. DapiDASetmA 255
5.8.6. DapiSpecialCmd_DA 256
5.9. PT100 Funktionen 258

5.9.1. DapiTempGet 258
5.10. Schrittmotoren Funktionen 259

5.10.1. Befehle mit DapiStepperCommand 259
5.10.1.1. DAPI_STEPPER_CMD_GO_POSITION 259
5.10.1.2. DAPI_STEPPER_CMD_GO_POSITION_RELATIVE 260
5.10.1.3. DAPI_STEPPER_CMD_SET_POSITION 261
5.10.1.4. DAPI_STEPPER_CMD_SET_FREQUENCY 262
5.10.1.5. DAPI_STEPPER_CMD_GET_FREQUENCY 263
5.10.1.6. DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY 264
5.10.1.7. DAPI_STEPPER_CMD_STOP 265
5.10.1.8. DAPI_STEPPER_CMD_FULLSTOP 266
5.10.1.9. DAPI_STEPPER_CMD_DISABLE 267
5.10.1.10. DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC 268
5.10.1.11. DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC 273
5.10.1.12. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_SAVE 281
5.10.1.13. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_LOAD 282
5.10.1.14. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT 283
5.10.1.15. DAPI_STEPPER_CMD_GO_REFSWITCH 284
5.10.1.16. DAPI_STEPPER_CMD_GET_CPU_TEMP 286
5.10.1.17. DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE 287
5.10.2. Status abfragen mit DapiStepperGetStatus 288
5.10.2.1. DAPI_STEPPER_STATUS_GET_ACTIVITY 288
5.10.2.2. DAPI_STEPPER_STATUS_GET_POSITION 289
5.10.2.3. DAPI_STEPPER_STATUS_GET_SWITCH 290
5.10.3. DapiStepperCommandEx 291
5.11. Ausgabe-Timeout verwalten 292

INDEX

5.11.1. DapiSpecialCMDTimeout 292
5.11.2. DapiSpecialCMDTimeoutGetStatus 293
5.12. Register Schreib-Befehle 294

5.12.1. DapiWriteByte 294
5.12.2. DapiWriteWord 295
5.12.3. DapiWriteLong 296
5.12.4. DapiWriteLongLong 297
5.13. Register Lese-Befehle 298

5.13.1. DapiReadByte 298
5.13.2. DapiReadWord 299
5.13.3. DapiReadLong 300
5.13.4. DapiReadLongLong 301
5.14. Programmier-Beispiel 302

6. Anhang 305

6.1. Revisionen 305

6.2. Urheberrechte und Marken 306

Einleitung

1. Einleitung

1.1. Vorwort

Wir beglückwünschen Sie zum Kauf eines hochwertigen DEDITEC Produktes!

Unsere Produkte werden von unseren Ingenieuren nach den heutigen geforderten Qualitätsanforderungen entwickelt. Wir achten bereits bei der Entwicklung auf flexible Erweiterbarkeit und lange Verfügbarkeit.

Wir entwickeln modular!
Durch eine modulare Entwicklung verkürzt sich bei uns die Entwicklungszeit und - was natürlich dem Kunden zu Gute kommt - ein fairer Preis!

Wir sorgen für eine lange Lieferverfügbarkeit!
Sollten verwendete Halbleiter nicht mehr verfügbar sein, so können wir schneller reagieren. Bei uns müssen meistens nur Module redesigned werden und nicht das gesamte Produkt. Dies erhöht die Lieferverfügbarkeit.

1.2. Kundenzufriedenheit

Ein zufriedener Kunde steht bei uns an erster Stelle!
Sollte mal etwas nicht zu Ihrer Zufriedenheit sein, wenden Sie sich einfach per Telefon oder mail an uns.
Wir kümmern uns darum!

1.3. Kundenresonanz

Die besten Produkte wachsen mit unseren Kunden. Für Anregungen oder Vorschläge sind wir jederzeit dankbar.

Hardware Beschreibung

II

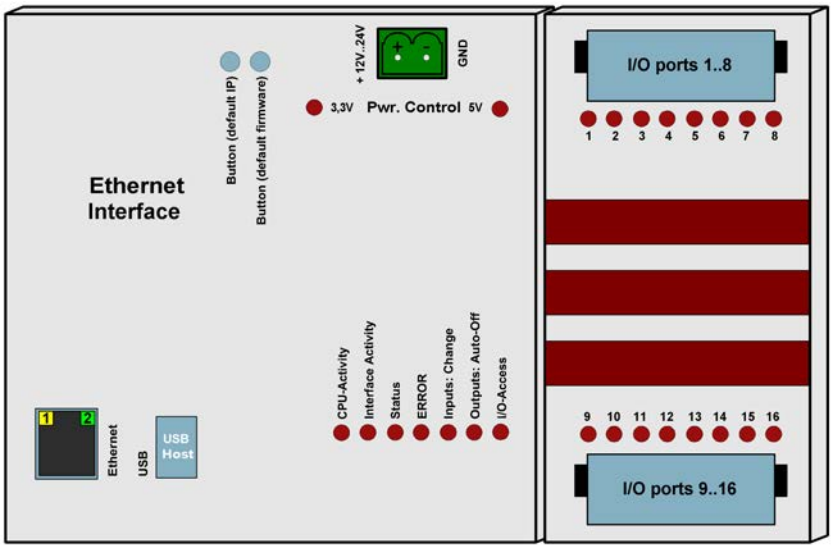
2. Hardware Beschreibung

2.1. Ethernet Interface

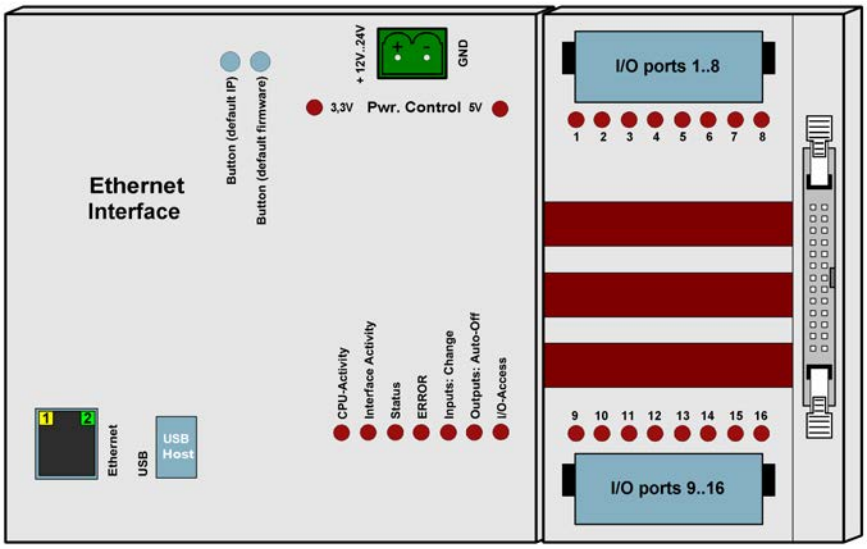
2.1.1. Hardware Beschreibung

2.1.1.1. Übersichtsbild

Die Abbildung zeigt das Steuermodul mit Ethernet Interface (links) in Kombination mit einem Ein- /Ausgabemodul (rechts).



Die Abbildung zeigt das Steuermodul mit Ethernet Interface (links) in Kombination mit einem I/O-Modul des flexiblen Steckverbindersystems (rechts).



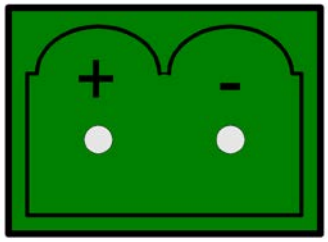
2.1.1.2. Technische Daten

- Single Spannungsversorgung +7V..+24V DC
- 10/100 Mbit/sec Ethernet Interface
- Zugriff auf Ein-/Ausgänge über TCP/IP
- WEB Interface
- Über Web-Interface einfach konfigurierbar
- 9 Kontroll LED‘s
- Anschluss über RJ45 Buchse
- Die Timeout Funktion bietet die Möglichkeit, die Ausgänge z.B. aus Sicherheitsgründen abzuschalten.
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen I/O Modulen der RO Serie kombiniert werden
- Windows Treiber Bibliothek DELIB

2.1.1.3. Steckverbinder auf dem Modul

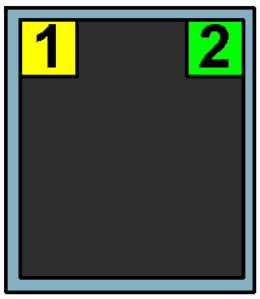
2.1.1.3.1. Spannungsversorgung

Der Eingangsspannungsbereich kann zwischen +7V und +24V DC betragen. Ein passender Steckverbinder liegt jedem Modul bei.



2.1.1.3.2. Ethernet Interface

Der Netzwerkanschluss erfolgt über eine RJ45 Buchse.



LED	
1	Activity
2	10/100 Mbit

2.1.1.4. Taster auf dem Modul

Linker Taster:

IP Adresse auf Default Werte zurücksetzen

(siehe Kapitel "IP Adresse zurücksetzen")

Rechter Taster:

Firmware in den Auslieferungszustand zurücksetzen

(siehe Kapitel "Firmware zurücksetzen")

2.1.1.5. Kontroll LED's

Auf dem RO-ETH Modul befinden sich eine Reihe von Kontroll LED's. Sie dienen zur einfachen optischen Zustandsanzeige von diversen Funktionen.

Nach dem Einschalten des Moduls muss folgende Blinksequenz im Normalbetrieb auftreten:

- ca. 20 Sek nach Einschalten des Moduls blinken LED 1 und 2 zweimal kurz hintereinander auf. -> Betriebssystem erfolgreich geladen.
- Anschließend leuchtet die 3. LED dauerhaft und LED 1 blinkt. -> Modul ist betriebsbereit.

2.1.1.5.1. Definition der LEDs

LED	Bezeichnung	Erklärung
oben	3,3V	Interne 3,3V Versorgungsspannung vorhanden.
oben	5V	Interne 5V Versorgungsspannung vorhanden.
1	CPU Activity	2x blinken + lange Pause. Betriebssystem meldet: Status OK
2	Interface Activity	Kommunikation über Ethernet aktiv.
3	Status	LED leuchtet -> Modul ist betriebsbereit
4	ERROR	Fehler bei der ETH Übertragung (näheres siehe Dokument "ETH Protokoll").
5	Inputs: Change	Zustandswechsel zwischen 2 Auslesetakten wurde erfasst.
6	Outputs: Auto-Off	Sämtliche Ausgänge wurden auf Grund des Timeout sicherheitshalber abgeschaltet.
7	I/O Access	Zugriff der CPU auf Ein- und Ausgänge der angeschlossenen Module.

2.1.2. Konfiguration des Moduls

2.1.2.1. Konfiguration über das DELIB Configuration Utility

Diese Methode ermöglicht eine einfache Konfiguration des Produkts. Folgende Grundwerte können hiermit verändert werden.

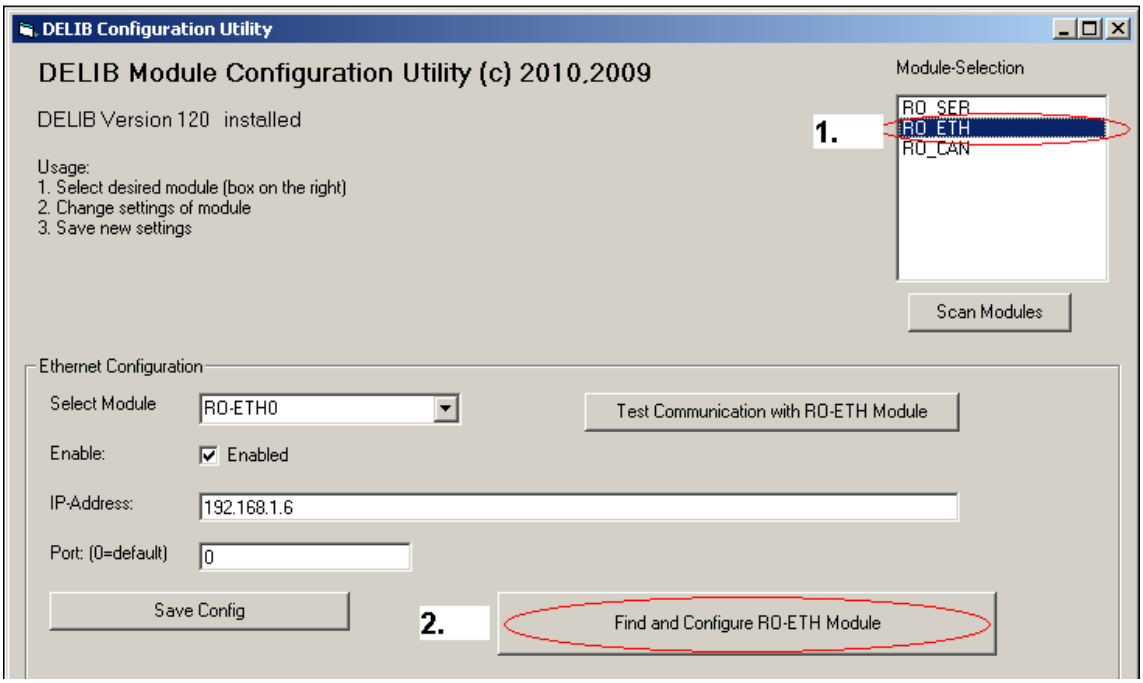
- Modulname
- IP-Adresse
- Netz-Maske
- Standard-Gateway
- DNS-Server

Außerdem lassen sich mit diesem Tool alle DEDITEC Ethernet-Geräte im Netzwerk anzeigen.

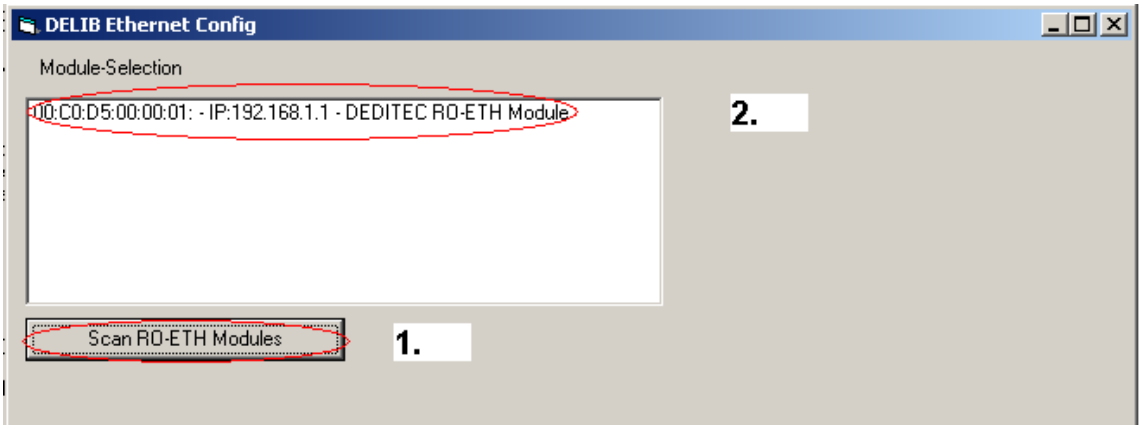
Wie das geht, sehen Sie auf den folgenden Seiten...

Starten Sie das DELIB Configuration Utility:

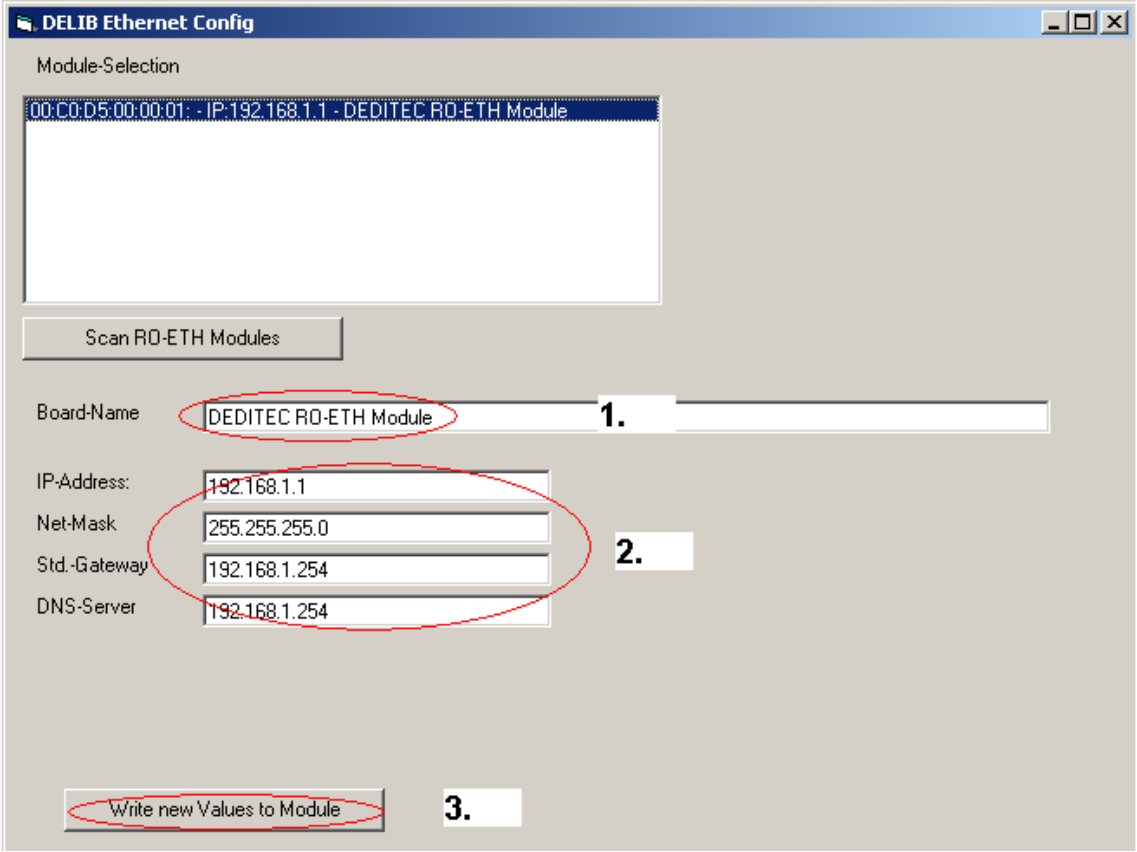
Zu finden unter: Start -> Programme -> DEDITEC -> DELIB -> DELIB Configuration Utility



1. Module Selection: RO-ETH auswählen
2. Find and configure RO-ETH Module



1. Scan RO-ETH Modules: Somit finden Sie alle DEDITEC ETH Module am lokalen Ethernet Strang. Hierbei benutzen wir ein Ethernet Protokoll, welches nicht geroutet wird. Deshalb sind nur Module zu konfigurieren, die am Bus angeschlossen sind. Vorteil dieser Methode ist, dass auch Module gefunden werden, die nicht im gleichen Sub-Netz des Rechners liegen, von dem aus konfiguriert wird.
2. Klicken Sie auf das Modul, welches Sie konfigurieren wollen



Hier lässt sich der Modulname nach Ihren Wünschen ändern

1. Hier können Modulname, IP-Adresse, Netz-Maske, Standard-Gateway und DNS-Server verändert werden.
2. Write new Values to Module. Hiermit wird die Konfiguration ins Modul geschrieben.

Hinweis:

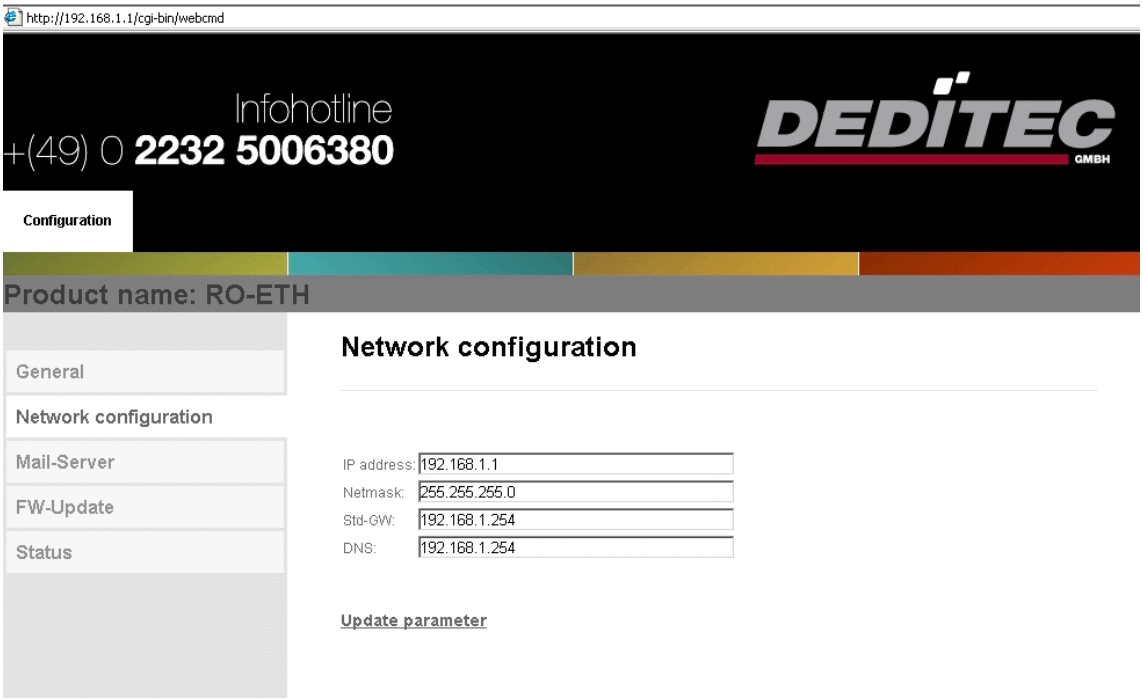
Bei der Konfiguration des RO-ETH Moduls ist zu beachten, dass die von Ihnen ausgewählte IP-Adresse im gleichen IP-Segment liegt, wie der Steuer-PC. Natürlich darf auch keine bereits belegte IP-Adresse benutzt werden.

Wenn die Standard-IP-Adresse des Moduls nicht aus dem Adressbereich des Netzwerks stammt, dann ist das Modul vorerst nicht über TCP/IP erreichbar. Erreichbarkeitsprobleme treten auch auf, wenn diese IP-Adresse bereits belegt ist. Anhand des sehr einfach zu bedienenden Utilities können die IP-Adresse und die Netzmaske des Ethernet-Moduls dennoch konfiguriert werden. Alternativ kann das Modul auch direkt am PC angeschlossen werden um auf direktem Wege die IP-Adresse und die Netzmaske einzustellen. Nachdem die Erreichbarkeit gegeben ist, erfolgt die weitere Konfiguration bequem über ein Browser auf dem integrierten Web-Server des Ethernet-Moduls.

Fragen Sie hierzu Ihren System Administrator.

2.1.2.2. Konfiguration über den internen Web-Server des Moduls

Das RO-ETH-Modul hat einen eigenen Web-Server über den die Konfiguration ebenfalls vorgenommen werden kann.



2.1.2.3. Auslieferungszustand

Im Auslieferungszustand hat das Ethernet Modul folgende Einstellungen:
IP Adresse: **192.168.1.1**

Der Auslieferungszustand kann jederzeit durch Betätigen des linken Tasters wiederhergestellt werden. (siehe Kapitel "IP Adresse zurücksetzen")

IP Adresse	**192.168.1.1**
Subnetz Maske	**255.255.255.0**
Standard Gateway	**192.168.1.254**

2.1.3. Grundkonfiguration wiederherstellen

2.1.3.1. IP Adresse zurücksetzen

Als Default Wert wird folgende IP Adresse verwendet: **192.168.1.1**

Linker Taster: IP Adresse auf Default Werte zurücksetzen (192.168.1.1):

Um die IP Adresse zurückzusetzen, gehen Sie wie folgt vor:

- Taster mindestens. 5 Sek lang drücken
- Danach müssen die linken beiden LED‘s “CPU Activity” und “Interface Activity” viermal hintereinander aufblinken (Übernahmebestätigung)
- anschließend hat das Modul folgende Einstellungen:

IP Adresse	**192.168.1.1**
Subnetz Maske	**255.255.255.0**
Standard Gateway	**192.168.1.254**

2.1.3.2. Firmware zurücksetzen

Um die Firmware auf den Default Wert zurückzusetzen, gehen Sie wie folgt vor:

Rechter Taster: Firmware in den Auslieferungszustand zurücksetzen

Um die Firmware in den Auslieferungszustand zurückzusetzen, gehen Sie wie folgt vor:

- Taster mindestens 10 Sek lang drücken
- Danach müssen die linken drei LED‘s “CPU Activity”, “Interface Activity” und “Status” viermal hintereinander aufblinken (Übernahmebestätigung)
- Anschließend startet das Modul neu.

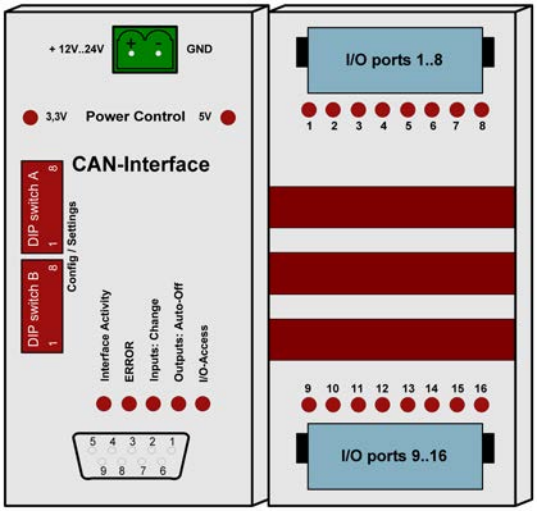
Die Firmware und Konfiguration des Auslieferzustandes, sind jetzt wieder aktiv!

2.2. CAN Interface

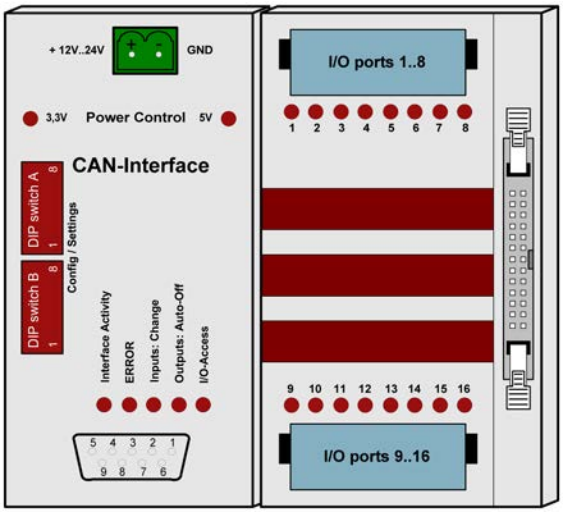
2.2.1. Hardware Beschreibung

2.2.1.1. Übersichtsbild

Die Abbildung zeigt das Steuermodul mit CAN Interface (links) in Kombination mit einem Ein- /Ausgabemodul (rechts).



Die Abbildung zeigt das Steuermodul mit CAN Interface (links) in Kombination mit einem I/O-Modul des flexiblen Steckverbindersystems (rechts).



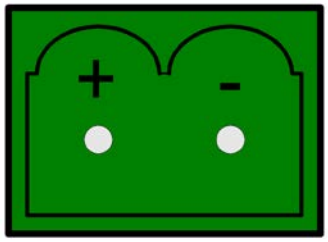
2.2.1.2. Technische Daten

- Single Spannungsversorgung +7V..+24V DC
- 7 Kontroll LED‘s
- CAN 2.0A (11Bit Adressierung)
- CAN 2.0B (29Bit Adressierung)
- Übertragungsreichweiten von bis zu 10km (bei 10kBit/s)
- Über DIP Schalter einfach konfigurierbar
- Galvanische Trennung des Interface mittels Optokoppler
- Anschluss über 9 pol. D-SUB Buchse
- Die Timeout Funktion bietet die Möglichkeit, die Ausgänge z.B. aus Sicherheitsgründen abzuschalten.
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

2.2.1.3. Steckverbinder auf dem Modul

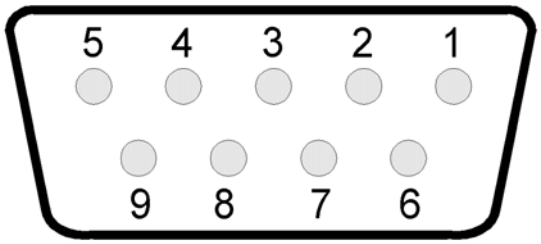
2.2.1.3.1. Spannungsversorgung

Der Eingangsspannungsbereich kann zwischen +7V und +24V DC betragen. Ein passender Steckverbinder liegt jedem Modul bei.



2.2.1.3.2. CAN Interface

Der Anschluss an den CAN Bus erfolgt über eine 9polige D-SUB Buchse und wird mittels Optokoppler galvanisch vom Modul getrennt. Ein Zwischenstecker wird mitgeliefert, mit dem sich über die RS-232 Schnittstelle des PCs das RO-CAN Modul programmieren läßt.



Pin	
1	RS-232 config
3	RS-232 config
2	CAN low
7	CAN high
5	GND

2.2.1.4. Kontroll LED's

Auf dem CAN Modul befinden sich eine Reihe von Kontroll LED's. Sie dienen zur einfachen optischen Zustandsanzeige von diversen Funktionen.

Im Dip-Schaltermodus oder Softwaremodus sollte nach dem Einschalten des Moduls folgende Blinksequenz auftreten:

alle fünf LEDs blinken kurz auf
rechte LED (I/O Access) blinkt kurz auf

Im "Vorzugsmodus" sollte nach dem Einschalten des Moduls folgende Blinksequenz auftreten:

alle fünf LEDs blinken kurz auf
rechte LED (I/O Access) blinkt kurz auf
alle fünf LEDs blinken kurz auf

2.2.1.4.1. Definition der LEDs

Bezeichnung	Erklärung
3,3V	Interne 3,3V Versorgungsspannung vorhanden.
5V	Interne 5V Versorgungsspannung vorhanden.
CAN Activity	Kommunikation über den CAN Bus aktiv.
ERROR	Fehler bei der CAN Übertragung (näheres siehe Dokument "CAN Protokoll").
Inputs: Change	Zustandswechsel zwischen 2 Auslesetakten wurde erfasst.
Outputs: Auto-Off	Sämtliche Ausgänge wurden auf Grund des Timeout sicherheitshalber abgeschaltet.
I/O Access	Zugriff der CPU auf Ein- und Ausgänge der angeschlossenen Module.

2.2.2. Konfiguration des Moduls

Um ein Modul in ein bestehendes Bus System zu integrieren, muss zunächst eine freie Moduladresse vergeben und die korrekte Übertragungsgeschwindigkeit eingestellt werden. Zum schnellen Einstieg kann man jedoch auch den Vorzugsmodus verwenden.

2.2.2.1. DIP-Schalter

Einige Einstellungen lassen sich einfach mit Hilfe von DIP Schaltern konfigurieren. Es lassen sich der ”Vorzugsmodus”, der ”Software-Modus” oder der ”DIP-Schalter-Modus” auswählen.

DIP Schalter A8	DIP Schalter A7	Erklärung
ON	ON	Vorzugsmodus -> Blinksequenz beim Start (5 LEDs, 1 rechte LED, 5 LEDs), 100KHz, CAN-ID=0x100, Response-Module-Addr=1, keine 29 Bit Adressen
ON	OFF	Nur für SERVICE-Zwecke: Applikation wird nicht gestartet. Zwangsweise im Bootloader
OFF	ON	Software-Modus: Konfiguration über Software
OFF	OFF	DIP-Schalter-Modus: Konfiguration über DIP-Schalter, Response-Module-Addr=1, CAN 2.0A

DIP Schalter	Erklärung
A6 bis A4	*) Einstellen der Übertragungsrate, à 4.5.4.1
A3 bis A1	*) Einstellen der CAN Adresse, à 4.5.4.2
B8 bis B1	*) Einstellen der CAN Adresse, à 4.5.4.2

*) für A8 und A7 = OFF

2.2.2.2. Der Vorzugsmodus

Der Vorzugsmodus dient dazu, das Gerät schnell und einfach auf festgelegte Standardwerte zu setzen. Dies ist hilfreich bei einer schnellen und einfachen Inbetriebnahme des Moduls. Eine Fehleranalyse oder erste Inbetriebnahme wird somit erleichtert.

Werden die Schalter A7 und A8 auf ”on” gestellt, gelangt man in diesen Modus. Alle anderen DIP-Schalter sind nun deaktiviert. Das Modul arbeitet dann mit folgenden Einstellungen:

Adressierung mit 11 Bit wird gewählt

100Kbit/sec Bitrate

CAN-Adresse = 0x100

Response-Modul-Addr = 1 (an diese Adresse werden die Antworten zurückgesendet)

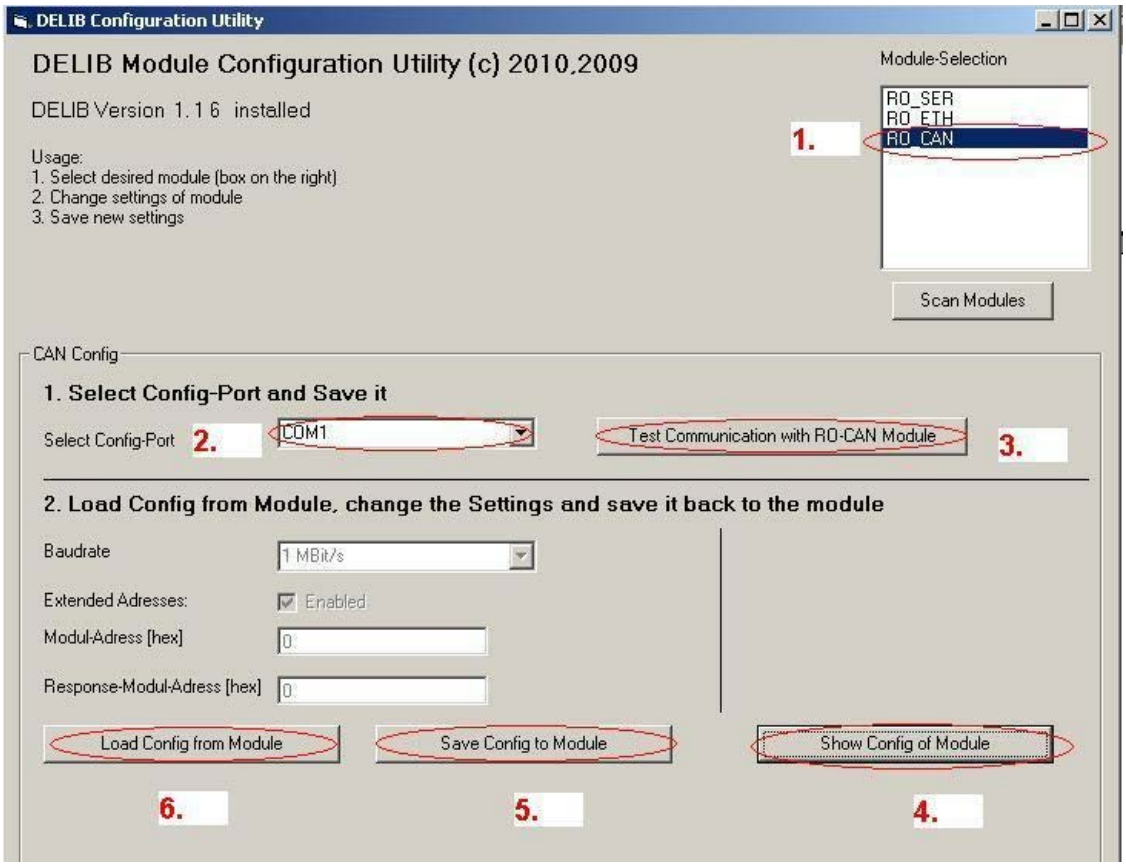
2.2.2.3. Software-Modus

Das Modul kann nur mit dem im Lieferumfang enthaltenen CAN/SER Programmier-Adapter im Software Modus konfiguriert werden. Das Konfigurieren geschieht dann über die serielle Schnittstelle am PC!

Um den Software Modus nutzen zu können, müssen die DIP Schalter auf dem Modul A7=ON und A8=OFF sein. Die DIP Schalteränderung wird nur nach Neustart des Moduls übernommen!

Modul mit DSUB-9 Kabel an eine PC RS-232 Schnittstelle wie folgt anschließen:
Serielle Schnittstelle des PC's mit CAN/SER Adapter verbinden und diesen an das RO-CAN-Interface-Modul anschließen.

Nachdem die DELIB Treiberbibliothek installiert wurde, findet man unter Start-> Programme-> DEDITEC -> DELIB, das DELIB Configuration Utility



Vorgehensweise:

1. RO-CAN Modul auswählen
2. COM Port auswählen, an dem das Modul angeschlossen wurde
3. Kommunikation mit Modul testen
4. Hier lässt sich die aktuelle Modulkonfiguration anzeigen
5. Hiermit wird die gewünschte Konfiguration in das Modul übertragen.
6. Hiermit wird die Konfiguration aus dem Modul ausgelesen

Konfigurierbar sind:

Übertragungsgeschwindigkeit
CAN 2.0A (11 Bit-Adressierung) oder CAN 2.0B (29 Bit-Adressierung)
CAN Interface-Adresse
Response-Modul-Addr (an diese werden die Antworten zurückgesendet)

Die Konfiguration erfolgt über das ”DELIB Configuration Utility” (→ 16.2).

2.2.2.4. DIP-Schalter-Modus

In diesem Modus muss das Interface-Modul über die DIP Schalter konfiguriert werden. Hierfür müssen die DIP Schalter A7=OFF und A8=OFF sein. Der Adressraum ist 11 Bit breit (CAN 2.0A). Die Modul-Adresse ist über 11 DIP-Schalter einstellbar (DIP A3..A1 und B8..B1). Die Response-Modul-Addr ist 1 (an diese Adresse werden die Antworten zurückgesendet).

2.2.2.4.1. Einstellen der Übertragungsgeschwindigkeit (Bitrate)

Je nachdem welche Reichweite der CAN Bus hat, werden unterschiedliche Übertragungsgeschwindigkeiten erreicht. Mit 3 DIP Schaltern lassen sich folgende Bitraten einstellen. Andere Bitraten sind unter Umständen nur per Software realisierbar.

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

2.2.2.4.2. Einstellen der CAN-Moduladresse

Jedes Gerät welches sich im CAN Netz befindet, benötigt eine feste Adresse um direkt angesprochen werden zu können. Mit den 11 DIP Schaltern lassen sich bis zu 2047 unterschiedliche Adressen einstellen.

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

Beispiele:

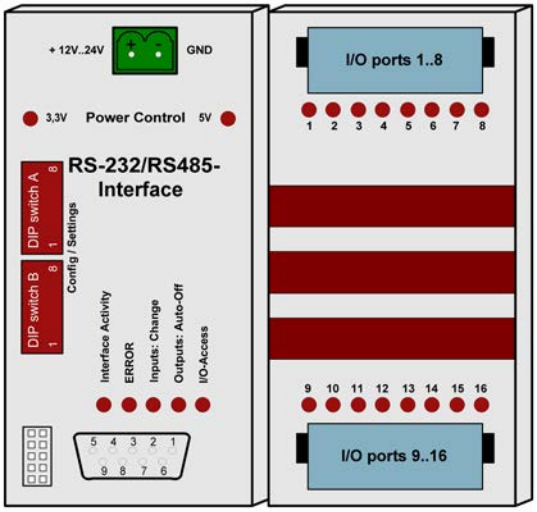
Baudrate	Adresse 0	Adresse 117	Adresse 588
DIP Schalter A3	Off	Off	Off
DIP Schalter A2	Off	Off	On
DIP Schalter A1	Off	Off	Off
DIP Schalter B8	Off	Off	Off
DIP Schalter B7	Off	On	On
DIP Schalter B6	Off	On	Off
DIP Schalter B5	Off	On	Off
DIP Schalter B4	Off	Off	On
DIP Schalter B3	Off	On	On
DIP Schalter B2	Off	Off	Off
DIP Schalter B1	Off	On	Off

2.3. RS-232/RS-485 Interface

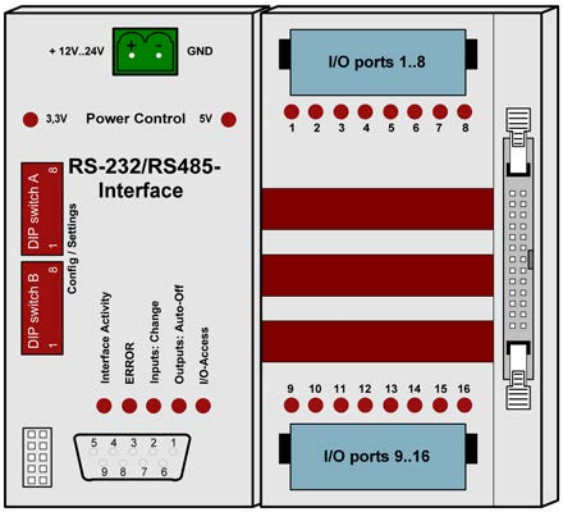
2.3.1. Hardware Beschreibung

2.3.1.1. Übersichtsbild

Die Abbildung zeigt das Steuermodul mit RS-232/RS-485 Interface (links) in Kombination mit einem Ein- /Ausgabemodul (rechts).



Die Abbildung zeigt das Steuermodul mit RS-232/RS-485 Interface (links) in Kombination mit einem I/O-Modul des flexiblen Steckverbindersystems (rechts).



2.3.1.2. Technische Daten

- Single Spannungsversorgung +7V..+24V DC
- 7 Kontroll LED‘s
- RS-232/RS-485 Schnittstelle
- Über DIP Schalter einfach konfigurierbar
- Galvanische Trennung über Optokoppler
- Anschluss über 9 pol. D-SUB Stecker
- Timeout Funktion: Die Timeout Funktion bietet die Möglichkeit, die Ausgänge z. B. aus Sicherheitsgründen abzuschalten.
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

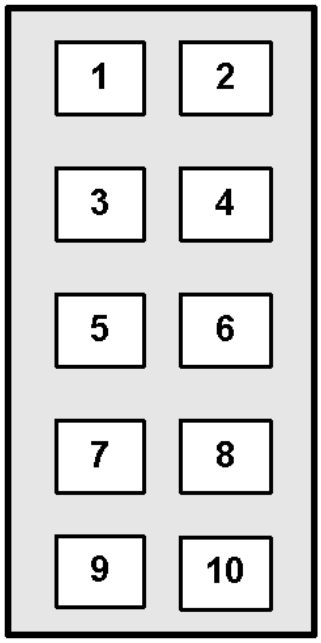
2.3.1.3. Wählen der Schnittstellenvariante RS-232 oder RS-485

Bei Auslieferung befindet sich das Schnittstellenmodul standardmäßig im RS-232 Modus. Nachfolgend wird beschrieben wie man das Modul sehr einfach auf RS-485 Betrieb umstellen kann.

Hinweis!
Vor Öffnen des Gerätes bitte folgendes beachten:

Netzstecker ziehen!
Elektronische Bauteile nicht berühren, da diese durch elektrostatische Entladung zerstört werden können!
Vor dem Arbeiten ggf. geerdete Gehäuse oder Heizkörper berühren.
Ein Seitenelement des Moduls entfernen, dazu die drei Kreuzschrauben an der Seite lösen
Leiterplatte samt Frontplatte seitlich aus dem Gehäuse herausziehen
Frontplatte abheben

Links neben dem Seriellen Interface (D-SUB Stecker) befindet sich eine 10pol. Stiftleiste mit den dazugehörigen Jumpern. In der nachfolgenden Tabelle wird gezeigt, welche Jumper gesteckt werden müssen.

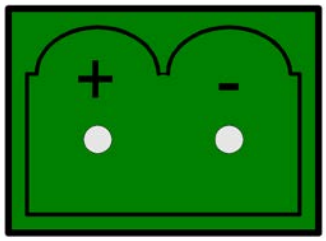
Stiftleiste	Schnittstelle	Jumper setzen
	RS-232	Pin1 & Pin3 Pin2 & Pin4
	RS-485 Abschlusswiderstand	Pin3 & Pin5 Pin4 & Pin6 Pin7 & Pin8 Pin9 & Pin10

Der Einbau erfolgt anschließend in umgekehrter Reihenfolge.

2.3.1.4. Steckverbinder auf dem Modul

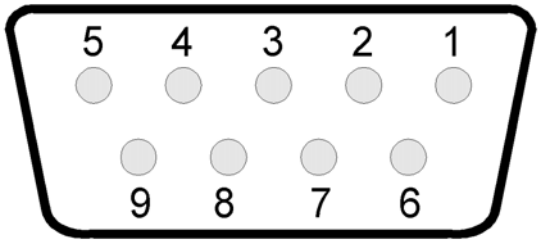
2.3.1.4.1. Spannungsversorgung

Der Eingangsspannungsbereich kann zwischen +7V und +24V DC gewählt werden. Der Anschluss kann über ein handelsübliches Steckernetzteil mit 1A Stromausgang erfolgen. Ein passender Steckverbinder liegt jedem Modul bei.



2.3.1.4.2. RS-232/RS-485 Interface

Der Anschluss an den seriellen Bus erfolgt über eine 9poligen D-SUB Stecker und wird mittels Optokoppler galvanisch vom Modul getrennt.



2.3.1.4.2.1. RS-232 Pinbelegung

Pin	
2	TX
3	RX
5	GND

2.3.1.4.2.2. RS-485 Pinbelegung

Pin	
2	RS-485 B
7	RS-485 A
5	GND

2.3.1.5. Kontroll LED's

Auf dem RS-232/RS-485 Modul befinden sich eine Reihe von Kontroll LED's. Sie dienen zur einfach optischen Zustandsanzeige von diversen Funktionen.

Im Normalbetrieb sollte nach dem Einschalten des Moduls folgende Blinksequenz auftreten:

alle fünf LEDs blinken kurz auf
rechte LED (I/O Access) blinkt kurz auf

Im "Vorzugsmodus" sollte nach dem Einschalten des Moduls folgende Blinksequenz auftreten:

alle fünf LEDs blinken kurz auf
rechte LED (I/O Access) blinkt kurz auf
alle fünf LEDs blinken kurz auf

2.3.1.5.1. Definition der LEDs

Bezeichnung	Erklärung
3,3V	Interne 3,3V Versorgungsspannung vorhanden.
5V	Interne 5V Versorgungsspannung vorhanden.
RS-232/RS-485 Activity	Kommunikation über den seriellen Bus aktiv.
ERROR	Fehler bei der seriellen Übertragung (näheres siehe Dokument "Serielles Protokoll").
Inputs: Change	Zustandswechsel zwischen 2 Auslesetakten wurde erfasst.
Outputs: Auto-Off	Sämtliche Ausgänge wurden auf Grund des Timeout sicherheitshalber abgeschaltet.
I/O Access	Zugriff der CPU auf Ein- und Ausgänge der angeschlossenen Module.

2.3.2. Konfiguration des Moduls

2.3.2.1. DIP-Schalter

Einige Einstellungen lassen sich einfach mit Hilfe von DIP Schaltern konfigurieren. Es lassen sich die Moduladresse, die Baudrate, der ”Vorzugsmodus” oder schnittstellenspezifische Einstellungen vornehmen.

DIP Schalter A8	DIP Schalter A7	Erklärung
ON	ON	Vorzugsmodus (115K Baudrate, Modul-Adresse = 0, Echo = OFF)
ON	OFF	Nur für SERVICE-Zwecke: Applikation wird nicht gestartet. Zwangsweise im Bootloader bleiben
OFF	ON	Einstellung von DIP A4..A1und B8..B1 benutzen
OFF	OFF	Einstellung von DIP A4..A1und B8..B1 benutzen

DIP Schalter A6	Erklärung
ON	Echo = ON, serielle empfangene Zeichen werden zurückgegeben (Echo = OFF, wenn DIP A8 und A7 = ON)
OFF	Echo = OFF

DIP Schalter	Erklärung
A5	Reserviert
A4 bis A1	Einstellen der Baudrate
B8 bis B1	Einstellen der seriellen Modulnummer

2.3.2.2. Der Vorzugsmodus

Der Vorzugsmodus dient dazu, das Gerät schnell und einfach auf festgelegte Standardwerte zu setzen. Dies ist hilfreich bei einer schnellen und einfachen Inbetriebnahme des Moduls. Eine Fehleranalyse oder erste Inbetriebnahme wird somit erleichtert.

Werden der Schalter DIP A8 und A7 auf ”on” gesetzt, gelangt man in diesen Vorzugsmodus. Alle anderen DIP-Schalter sind nun deaktiviert.

Das Modul kommuniziert jetzt mit einer Baudrate von 115Kbauds, die Moduladresse und “Echo” sind deaktiviert.

2.3.2.3. Echo aktivieren

Serielle empfangene Zeichen sollen auf den Bildschirm zurückgegeben werden (ON=ja, OFF=nein).

2.3.2.4. Einstellen der Baudrate

Die untere Tabelle zeigt, welche Baudraten mit den 4 DIP Schaltern (A1 bis A4) eingestellt werden können.

Baudrate	DIP A4	DIP A3	DIP A2	DIP 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

2.3.2.5. Einstellen der Moduladresse (nur bei RS-485)

Da im RS-485 Betrieb mehrere Module an einem BUS System angeschlossen werden können, ist es erforderlich jedem Modul eine eigene Adresse zuzuweisen. Diese kann mit den DIP Schaltern B1 bis B8 zwischen 0 und 255 eingestellt werden. Bei Modulnummer 0 wird die Modulnummer ignoriert.

Baudrate	Bit	Wertigkeit ON	Wertigkeit OFF
DIP Schalter B8	Bit 7	128	0
DIP Schalter B7	Bit 6	64	0
DIP Schalter B6	Bit 5	32	0
DIP Schalter B5	Bit 4	16	0
DIP Schalter B4	Bit 3	8	0
DIP Schalter B3	Bit 2	4	0
DIP Schalter B2	Bit 1	2	0
DIP Schalter B1	Bit 0	1	0

Beispiele:

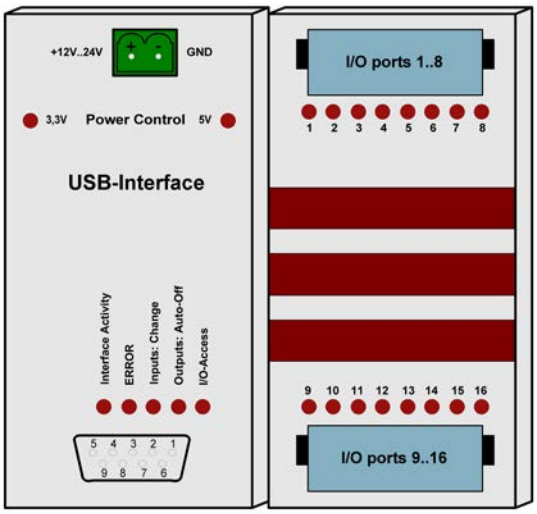
Baudrate	Adresse 0	Adresse 25	Adresse 237
DIP Schalter B8	Off	Off	On
DIP Schalter B7	Off	Off	On
DIP Schalter B6	Off	Off	On
DIP Schalter B5	Off	On	Off
DIP Schalter B4	Off	On	On
DIP Schalter B3	Off	Off	On
DIP Schalter B2	Off	Off	Off
DIP Schalter B1	Off	On	On

2.4. USB Interface

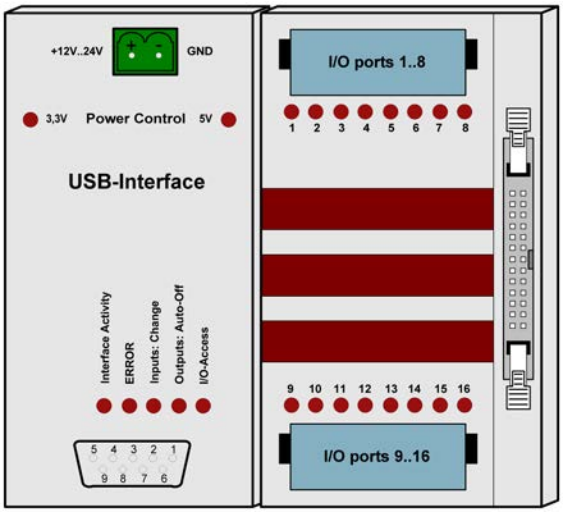
2.4.1. Hardware Beschreibung

2.4.1.1. Übersichtsbild

Die Abbildung zeigt das Steuermodul mit USB Interface (links) in Kombination mit einem Ein- /Ausgabemodul (rechts). Für die Anbindung an den USB Bus, wird ein passendes Adaptermodul in Form eines USB Sticks mitgeliefert.



Die untere Abbildung zeigt das Steuermodul mit USB Interface (links) in Kombination mit einem I/O-Modul des flexiblen Steckverbindersystems (rechts). Für die Anbindung an den USB Bus, wird ein passendes Adaptermodul in Form eines USB Sticks mitgeliefert.



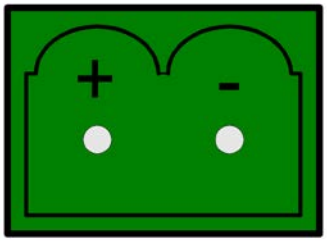
2.4.1.2. Technische Daten

- Single Spannungsversorgung +7V..+24V DC
- 7 Kontroll LED‘s
- USB Schnittstelle
- Mögliche Reichweite von über 100m!
- USB 2.0 und USB 1.1
- Übertragungsraten: 12 MBit/s oder 1,5 MBit/s.
- Galvanische Trennung über Optokoppler
- Anschluss über 9 pol. D-SUB Stecker
- Timeout Funktion: Die Timeout Funktion bietet die Möglichkeit, die Ausgänge z. B. aus Sicherheitsgründen abzuschalten.
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

2.4.1.3. Steckverbinder auf dem Modul

2.4.1.3.1. Spannungsversorgung

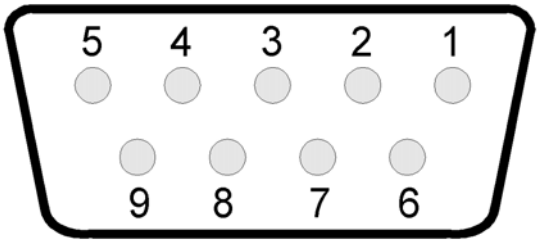
Der Eingangsspannungsbereich kann zwischen +7V und +24V DC gewählt werden. Der Anschluss kann über ein handelsübliches Steckernetzteil mit 1A Stromausgang erfolgen. Ein passender Steckverbinder liegt jedem Modul bei.



2.4.1.3.2. USB Interface

Der Anschluss an den USB-Bus erfolgt über ein speziell entwickeltes Adaptermodul in Form eines USB Sticks mit passendem Anschlusskabel. Auf dem Stick befindende Optokoppler gewährleisten gleichzeitig eine galvanische Trennung zum PC.

Über einen 9poligen D-SUB Stecker wird der Adapter mit dem eigentlichen USB Interface des RO-Moduls verbunden.



2.4.1.4. Kontroll LED's

Auf dem USB Modul befinden sich eine Reihe von Kontroll LED's. Sie dienen zur einfach optischen Zustandsanzeige von diversen Funktionen.

Im Betrieb sollte nach dem Einschalten des Moduls folgende Blinksequenz auftreten:

alle fünf LED blinken kurz auf
rechte LED (I/O Access) blinkt kurz auf
alle fünf LED blinken kurz auf

2.4.1.4.1. Definition der LEDs

Bezeichnung	Erklärung
3,3V	Interne 3,3V Versorgungsspannung vorhanden.
5V	Interne 5V Versorgungsspannung vorhanden.
USB Activity	Kommunikation über den USB-Bus aktiv.
ERROR	Fehler bei der USB-Übertragung (näheres siehe Dokument "USB Protokoll").
Inputs: Change	Zustandswechsel zwischen 2 Auslesetakten wurde erfasst.
Outputs: Auto-Off	Sämtliche Ausgänge wurden auf Grund des Timeout sicherheitshalber abgeschaltet.
I/O Access	Zugriff der CPU auf Ein- und Ausgänge der angeschlossenen Module.

2.5. Digitale Ein-/Ausgabemodule

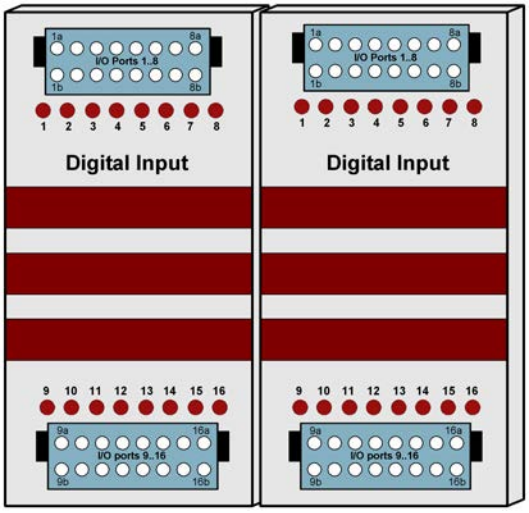
2.5.1. Hardware Beschreibung

Die Ein-/Ausgangsmodule basieren auf der Verwendung von zwei 16pol. Steckverbindern mit jeweils 8 unterschiedlichen Schaltkreisen. Jeder Zustand dieser insgesamt 16 Schaltkreise wird durch eine LED signalisiert. Die Durchnummerierung jedes Moduls erfolgt immer von links oben nach rechts unten (siehe Übersichtsbild).

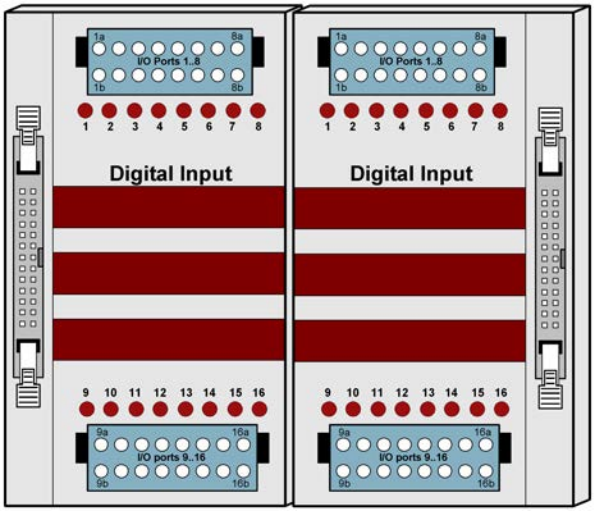
2.5.1.1. RO-O16

2.5.1.1.1. Übersichtsbild

Die Abbildung zeigt zwei Module nebeneinander mit entsprechender Durchnummerierung der Anschlussklemmen.



Untere Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit 32 Eingängen und entsprechender Durchnummerierung der Anschlussklemmen. An den äußeren Enden des Moduls befindet sich jeweils eine 26pol. Wannenstiftleiste mit Verriegelungshebeln. Somit können mehrere Moduleinheiten mit Hilfe eines Flachbandkabels miteinander verbunden werden.



2.5.1.1.2. Technische Daten

- Variabler Eingangsspannungsbereich min. 5V, max. 30V AC
- 16 Bit-Zähler für die ersten 16 Eingangskanäle
- Erfassung von Impulsen zwischen 2 Auslesetakten
- LED Zustandsanzeige der Eingänge
- Galvanische Trennung über Optokoppler
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

2.5.1.1.3. 16-Bit Zähler

Die ersten 16 Eingangskanäle verfügen über je einen 16 Bit-Zähler. Hiermit können Ereignisse wie z.B. Lichtschranken, Drehkreuze oder Taster vom Modul gezählt werden. Somit lassen sich auch einfache Logische Schaltungen realisieren, die nach erreichen eines Sollwertes einen oder mehrere Ausgänge schalten können. Die Softwaremäßige Umsetzung solcher Logikschaltungen entnehmen Sie bitte dem Handbuch “RO-Serie”.

2.5.1.1.4. Erfassen von schnellen Eingangsimpulsen

Schnelle Zustandswechsel an den Eingängen, die innerhalb von größeren Auslesezyklen auftreten, werden durch eine zusätzliche Logik erfasst und können separat per Software ausgelesen werden. Ein solcher "Zustandswechsel" wird (für alle Eingänge gemeinsam) mit der LED “Inputs: Change”, auf dem Steuermodul signalisiert. Die LED erlischt erst wieder, wenn die Softwareregister für die Eingangszustandsänderung vom Anwender ausgelesen wurden. Näheres siehe Handbuch “Registerbelegung”.

2.5.1.1.5. Galvanische Trennung durch Optokoppler

Wechselspannungs geeignete Eingangs-Optokoppler sorgen zum einem für eine galvanische Trennung des Moduls zu den angeschlossenen Anlagen und zum anderen verhindert man eine Beschädigung des Moduls bei verpoltem Anschluss oder auftretenden Spannungspitzen o.ä. im Steuerstromkreis.

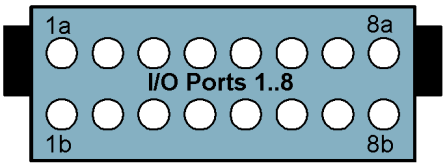
2.5.1.1.6. Steckverbinder auf dem Modul

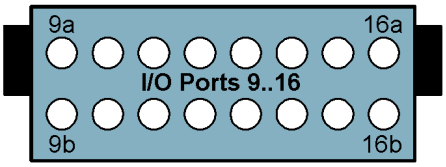
Als Anschlussklemmen kommen servicefreundliche Steckleisten mit Verriegelungsschutz und Auswerfmechanik zum Einsatz. Diese sind fehlsteckgeschützt und ermöglichen ein schnelles, nachträgliches Umstecken der angeschlossenen Anlagen. Der Leitungsanschluss selbst erfolgt über ein schraubenloses Stecksystem. Hierfür benötigtes Betätigungswerkzeug wird bei jedem Modul mitgeliefert.

2.5.1.1.6.1. Leitungsanschluss

Der Leitungsanschluss erfolgt jeweils an den Ports mit gleicher Nummerierung.
z.B.: 1a & 1b usw..
Die Optokopplereingänge sind für Wechselspannung geeignet. Auf korrekte Polarität der Eingangsbeschaltung muss deshalb nicht geachtet werden.

Die Abbildung zeigt zwei Klemmleisten mit entsprechender Nummerierung der Anschluss-Ports.





2.5.1.1.6.2. Visuelle Kontrolle der Eingänge

Über eine LED wird der Zustand jedes Eingangs direkt angezeigt. Signale an den Eingängen sind somit einfacher zu erkennen und Fehler in der Verdrahtung lassen sich dadurch schneller beheben.

2.5.1.1.6.3. Pinbelegung

Port	Pin	Port	Pin
1	1a & 1b	9	9a & 9b
2	2a & 2b	10	10a & 10b
3	3a & 3b	11	11a & 11b
4	4a & 4b	12	12a & 12b
5	5a & 5b	13	13a & 13b
6	6a & 6b	14	14a & 14b
7	7a & 7b	15	15a & 15b
8	8a & 8b	16	16a & 16b

2.5.1.1.7. Eingangsspannungsbereich variierbar

Die Eingänge sind standardmäßig für einen Spannungsbereich von 15V bis 30V ausgelegt. Diese kann (auch nachträglich) für einen Bereich von 5V bis 15V angepasst werden.

Eingangsspannungsbereich	5V – 15V	15V – 30V
Widerstandswert	1K	2K2

2.5.1.1.7.1. Ändern der Eingangsspannung

Um den Eingangsspannungsbereich anzupassen, müssen pro Steckverbinder 2 Vorwiderstände ausgetauscht werden. D.h., man kann jeweils 4 Eingangskanäle (1-4, 5-8, 9-12 und 13-16) mit unterschiedlicher Spannung einspeisen. Die einzelnen Schritte hierfür werden im Folgenden erläutert.

Hinweis!
Vor Öffnen des Gerätes bitte folgendes beachten:

Netzstecker ziehen!
Elektronische Bauteile nicht berühren, da diese durch elektrostatische Entladung zerstört werden können! Vor dem Arbeiten ggf. geerdete Gehäuse oder Heizkörper berühren.

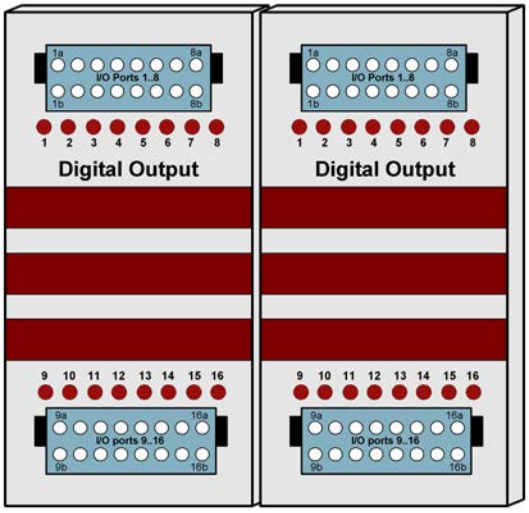
Seitenelement des Moduls entfernen, dazu die drei Kreuzschrauben lösen
Leiterplatte samt Frontplatte seitlich aus dem Gehäuse herausziehen
Frontplatte abheben
Die Eingangsmodule verfügen alle über einreihige Sockelleisten, in denen die Eingangswiderstände für den Spannungsbereich eingesteckt sind
Widerstände nun vorsichtig aus den Sockelstreifen herausziehen
Neue Widerstände einsetzen

Einbau erfolgt in umgekehrter Reihenfolge

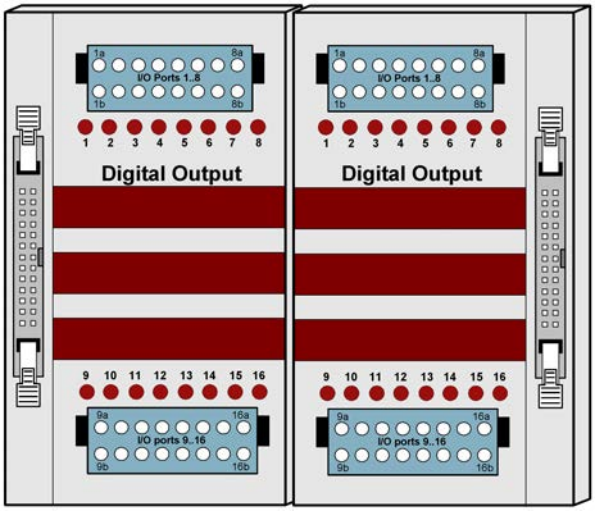
2.5.1.2. RO-R16

2.5.1.2.1. Übersichtsbild

Die Abbildung zeigt zwei Module nebeneinander mit entsprechender Durchnummerierung der Anschlussklemmen.



Untere Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit 32 Ausgängen und entsprechender Durchnummerierung der Anschlussklemmen. An den äußeren Enden des Moduls befindet sich jeweils eine 26pol. Wannenstiftleiste mit Verriegelungshebeln. Somit können mehrere Moduleinheiten mit Hilfe eines Flachbandkabels miteinander verbunden werden.



2.5.1.2.2. Technische Daten

- Timeout-Schutz
- LED Zustandsanzeige der Ausgänge
- Galvanische Trennung über Optokoppler
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden
- Max. Schaltspannung: 36V
- Max. Schaltstrom: 1A
- Max. Schaltleistung: 20W
- Schaltzyklen laut Hersteller: 10 Mio.

2.5.1.2.3. Timeout-Schutz

Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden. Das Abschalten der Ausgänge wird durch eine LED signalisiert.

2.5.1.2.4. Steckverbinder auf dem Modul

Als Anschlussklemmen kommen, servicefreundliche Steckleisten mit Verriegelungsschutz und Auswerfmechanik zum Einsatz. Diese sind fehlsteckgeschützt und ermöglichen ein schnelles, nachträgliches Umstecken der angeschlossenen Anlagen. Der Leitungsanschluss selbst erfolgt über ein schraubenloses Stecksystem. Hierfür benötigtes Betätigungswerkzeug wird bei jedem Modul mitgeliefert.

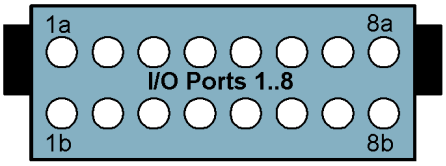
2.5.1.2.4.1. Relais Ausgänge (galvanisch getrennt, max 1A)

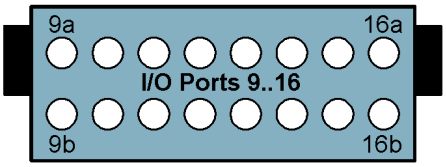
Durch den Einsatz von Relais lassen sich Spannungen von bis zu 36V schalten. Die maximale Strombelastbarkeit beträgt 1A bei einer maximalen Schaltleistung von 20W.

Außerdem sorgen die Relais für eine sichere galvanische Trennung des Moduls von den angeschlossenen Anlagen.

2.5.1.2.4.2. Leitungsanschluss

Der Leitungsanschluß erfolgt jeweils an den Ports mit gleicher Nummerierung. z.B.: 1a & 1b usw..
Auf korrekte Polarität muß bei den Relaisausgängen nicht geachtet werden.





2.5.1.2.4.3. Visuelle Kontrolle der Ausgänge

Über eine LED wird der Zustand jedes Ausgangs direkt angezeigt. Signale an den Ausgängen sind somit einfacher zu erkennen und Fehler in der Verdrahtung lassen sich dadurch schneller beheben.

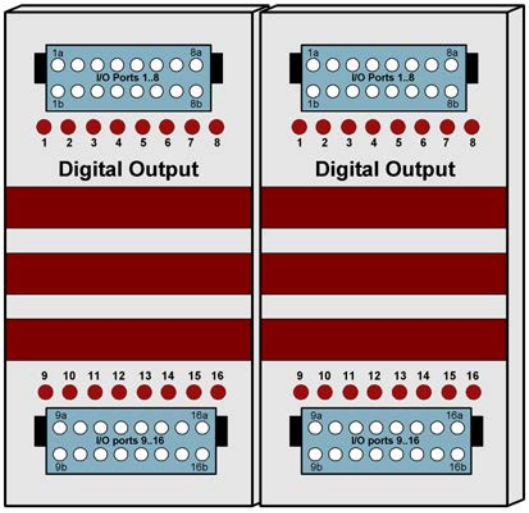
2.5.1.2.4.4. Pinbelegung

Port	Pin	Port	Pin
1	1a & 1b	9	9a & 9b
2	2a & 2b	10	10a & 10b
3	3a & 3b	11	11a & 11b
4	4a & 4b	12	12a & 12b
5	5a & 5b	13	13a & 13b
6	6a & 6b	14	14a & 14b
7	7a & 7b	15	15a & 15b
8	8a & 8b	16	16a & 16b

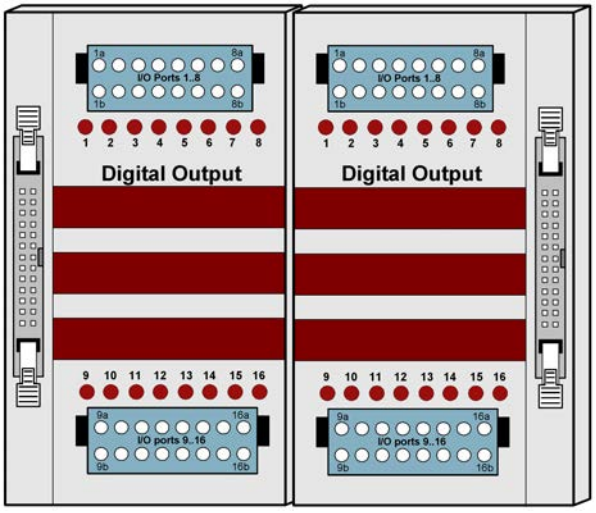
2.5.1.3. RO-M16

2.5.1.3.1. Übersichtsbild

Die Abbildung zeigt zwei Module nebeneinander mit entsprechender Durchnummerierung der Anschlussklemmen.



Untere Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit 32 Ausgängen und entsprechender Durchnummerierung der Anschlussklemmen. An den äußeren Enden des Moduls befindet sich jeweils eine 26pol. Wannenstiftleiste mit Verriegelungshebeln. Somit können mehrere Moduleinheiten mit Hilfe eines Flachbandkabels miteinander verbunden werden.



2.5.1.3.2. Technische Daten

- Timeout-Schutz
- LED Zustandsanzeige der Ausgänge
- Galvanische Trennung über Optokoppler
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 16 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden
- Max. Schaltstrom: 2A DC
- Max. Schaltspannung: 30V DC
- Max. Schaltleistung: 40W

2.5.1.3.3. Timeout-Schutz

Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden. Das Abschalten der Ausgänge wird durch eine LED signalisiert.

2.5.1.3.4. Steckverbinder auf dem Modul

Als Anschlussklemmen kommen, servicefreundliche Steckleisten mit Verriegelungsschutz und Auswerfmechanik zum Einsatz. Diese sind fehlsteckgeschützt und ermöglichen ein schnelles, nachträgliches Umstecken der angeschlossenen Anlagen. Der Leitungsanschluss selbst erfolgt über ein schraubenloses Stecksystem. Hierfür benötigtes Betätigungswerkzeug wird bei jedem Modul mitgeliefert.

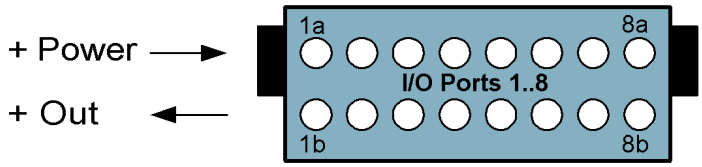
2.5.1.3.4.1. Optokoppler Ausgänge (galvanisch getrennt, max 2A DC)

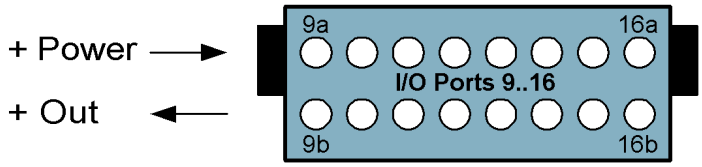
Sämtliche Ausgänge werden mittels Hochstromoptokoppler realisiert. Die Ansteuerung über Optokoppler sorgt für eine sichere galvanische Trennung des Moduls von den angeschlossenen Anlagen.

Bei den Optokopplerausgängen muss auf die richtige Polarität beim Anschluss geachtet werden (siehe Bild unten)!

2.5.1.3.4.2. Leitungsanschluss

Der Leitungsanschluss erfolgt jeweils an den Ports mit gleicher Nummerierung.
z.B.: 1a & 1b usw..
Bei den Optokopplerausgängen muss auf die korrekte Polarität der Beschaltung geachtet werden, da der Ausgangsport sonst beschädigt wird!
An Port “a” wird die positive Spannung angelegt und an Port ”b” die geschaltete positive Spannung.





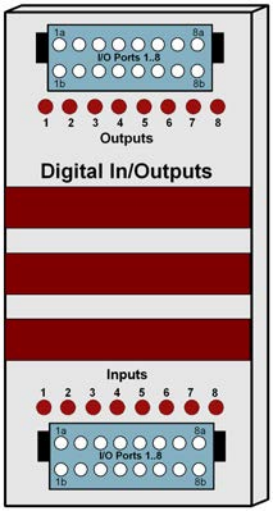
2.5.1.3.4.3. Pinbelegung

Port	Pin	Port	Pin
1	1a & 1b	9	9a & 9b
2	2a & 2b	10	10a & 10b
3	3a & 3b	11	11a & 11b
4	4a & 4b	12	12a & 12b
5	5a & 5b	13	13a & 13b
6	6a & 6b	14	14a & 14b
7	7a & 7b	15	15a & 15b
8	8a & 8b	16	16a & 16b

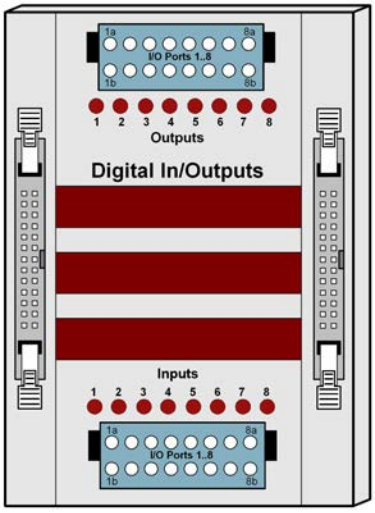
2.5.1.4. RO-O8-R8

2.5.1.4.1. Übersichtsbild

Die Abbildung zeigt ein RO-O8-R8 Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Untere Abbildung zeigt ein RO-O8-R8 Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen. An den äußeren Enden des Moduls befindet sich jeweils eine 26pol. Wannenstiftleiste mit Verriegelungshebeln. Somit können mehrere Moduleinheiten mit Hilfe eines Flachbandkabels miteinander verbunden werden.



2.5.1.4.2. Technische Daten

Optokoppler Eingänge:

- Variabler Eingangsspannungsbereich min. 5V, max. 30V AC
- 16 Bit-Zähler für 8 Eingangskanäle mit einer Abtastrate von >10 kHz
- Erfassung von Impulsen zwischen 2 Auslesetakten
- LED Zustandsanzeige der Eingänge
- Galvanische Trennung über Optokoppler
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 8 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

Relais Ausgänge:

- Timeout-Schutz
- LED Zustandsanzeige der Ausgänge
- Galvanische Trennung über Relais
- Komfortables Steckverbindersystem mit Auswerfmechanik
- In 8 facher Abstufung erweiterbar
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden
- Max. Schaltspannung: 36V
- Max. Schaltstrom: 1A
- Max. Schaltleistung: 20W
- Schaltzyklen des Relais (laut Hersteller: 10 Mio.)

2.5.1.4.3. Optokoppler Eingänge

2.5.1.4.3.1. 16-Bit Zähler

Die 8 Eingangskanäle verfügen über je einen 16 Bit-Zähler mit einer Abtastrate von >10 kHz. Hiermit können Ereignisse wie z.B. Lichtschranken, Drehkreuze oder Taster vom Modul gezählt werden. Somit lassen sich auch einfache Logische Schaltungen realisieren, die nach Erreichen eines Sollwertes einen oder mehrere Ausgänge schalten können.

2.5.1.4.3.2. Erfassen von schnellen Eingangsimpulsen

Schnelle Zustandswechsel an den Eingängen, die innerhalb von größeren Auslesezyklen auftreten, werden durch eine zusätzliche Logik erfasst und können separat per Software ausgelesen werden. Ein solcher "Zustandswechsel" wird (für alle Eingänge gemeinsam) mit der LED “Inputs: Change”, auf dem Steuermodul signalisiert. Die LED erlischt erst wieder, wenn die Softwareregister für die Eingangszustandsänderung vom Anwender ausgelesen wurden. Näheres siehe Handbuch “Registerbelegung”.

2.5.1.4.3.3. Galvanische Trennung durch Optpkoppler

Eingangs-Optokoppler (AC-Eingangs-Optokoppler) sorgen zum einem für eine galvanische Trennung des Moduls zu den angeschlossenen Anlagen und zum anderen verhindert man eine Beschädigung des Moduls bei verpoltem Anschluss oder auftretenden Spannungspitzen o.ä. im Steuerstromkreis. AC-Eingangs-Optokoppler besitzen keine Polarität. Die Anschluss Richtung der Eingänge (+ und -) kann vertauscht werden.

2.5.1.4.4. Relais Ausgänge

2.5.1.4.4.1. Timeout-Schutz

Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden. Das Abschalten der Ausgänge wird durch eine LED signalisiert.

2.5.1.4.5. Steckverbinder auf dem Modul

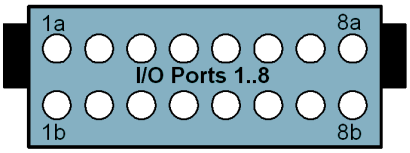
Als Anschlussklemmen kommen servicefreundliche Steckleisten mit Verriegelungsschutz und Auswerfmechanik zum Einsatz. Diese sind fehlsteckgeschützt und ermöglichen ein schnelles, nachträgliches Umstecken der angeschlossenen Anlagen. Der Leitungsanschluss selbst erfolgt über ein schraubenloses Stecksystem. Hierfür benötigtes Betätigungswerkzeug wird bei jedem Modul mitgeliefert.

2.5.1.4.5.1. Leitungsanschluss und Pinbelegung

Relais Ausgänge (Stecker oben):

Der Leitungsanschluß erfolgt jeweils an den Ports mit gleicher Nummerierung. z.B.: 1a & 1b usw..

Auf korrekte Polarität muß bei den Relaisausgängen nicht geachtet werden.



Port	Pin
Relais 1	1a & 1b
Relais 2	2a & 2b
Relais 3	3a & 3b
Relais 4	4a & 4b
Relais 5	5a & 5b
Relais 6	6a & 6b
Relais 7	7a & 7b
Relais 8	8a & 8b

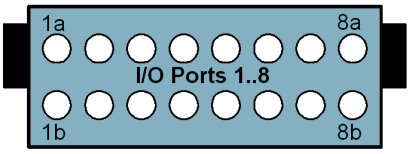
Optokoppler Eingänge (Stecker unten):

Der Leitungsanschluss erfolgt jeweils an den Ports mit gleicher Nummerierung.

z.B.: 1a & 1b usw..

Die Optokopplereingänge sind für Wechselspannung geeignet. Auf korrekte Polarität der Eingangsbeschaltung muss deshalb nicht geachtet werden.

Die Abbildung zeigt die Klemmleiste mit entsprechender Nummerierung der Anschluss-Ports.



Port	Pin
Opto-IN 1	1a & 1b
Opto-IN 2	2a & 2b
Opto-IN 3	3a & 3b
Opto-IN 4	4a & 4b
Opto-IN 5	5a & 5b
Opto-IN 6	6a & 6b
Opto-IN 7	7a & 7b
Opto-IN 8	8a & 8b

2.5.1.4.5.2. Visuelle Kontrolle der Ein- und Ausgänge

Über eine LED wird der Zustand jedes Ein- und Ausgangs direkt angezeigt. Signale an den Ein- und Ausgängen sind somit einfacher zu erkennen und Fehler in der Verdrahtung lassen sich dadurch schneller beheben.

2.6. Analoge Ein-/Ausgabemodule

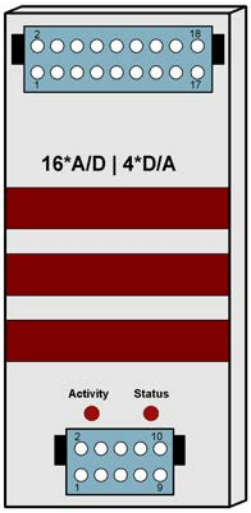
2.6.1. Hardware Beschreibung

2.6.1.1. RO-AD16-DA4

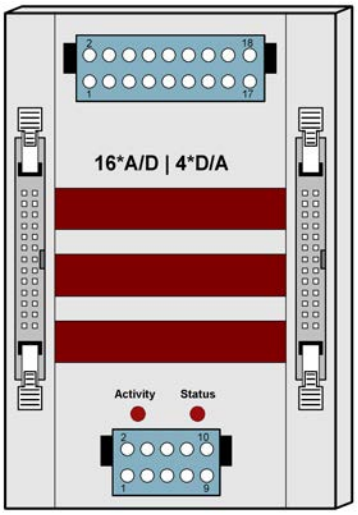
Mit 16 A/D Eingängen stellt dieses Modul eine gute Basis zur Umwandlung von Spannungen in digitale Werte dar. Zusätzlich dank der 4 D/A Ausgänge ist es möglich digitale Werte in Spannungen umzuwandeln.

2.6.1.1.1. Übersichtsbild

Die Abbildung zeigt ein Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Die Abbildung zeigt ein A/D-Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen.



2.6.1.1.2. Technische Daten

- Timeout-Schutz
- Komfortables Steckverbindersystem mit Auswerfmechanik
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

A/D Eingänge

Modus U: (Spannung)
Unipolar: 0-5V, 0-10V
Bipolar: +5V, +10V

Modus I: (Strom)
Bereich: 0-20mA (optional)

D/A Ausgänge

Timeout Schutz

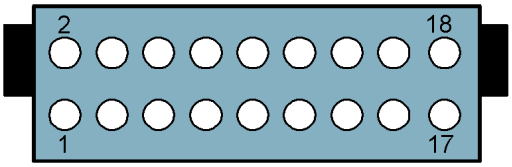
Modus U: (Spannung)
Unipolar: 0-5V, 0-10V
Bipolar: +5V, +10V

2.6.1.1.3. Timeout-Schutz

Der Timeout-Schutz muss per Software separat aktiviert werden. Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden. Das Abschalten der Ausgänge wird durch eine LED signalisiert.

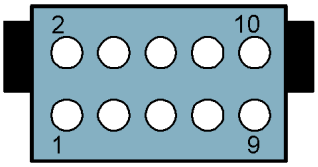
2.6.1.1.4. Pinbelegung

2.6.1.1.4.1. A/D Steckverbinder (18pol)



Pin		Pin	
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

2.6.1.1.4.2. D/A Steckverbinder (10pol)



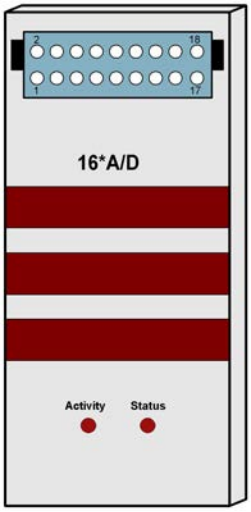
Pin		Pin	
1	AGND	2	DA0
3	AGND	4	DA1
5	AGND	6	DA2
7	AGND	8	DA3
9	AGND	10	AGND

2.6.1.2. RO-AD16

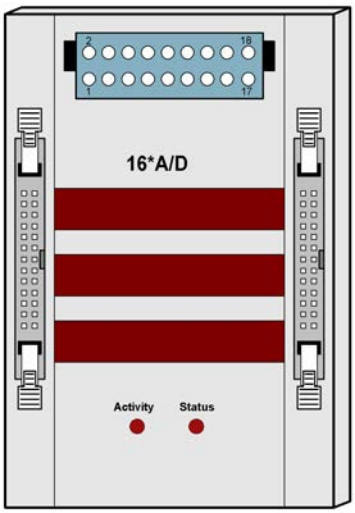
Mit 16 A/D Eingangs-Kanäle stellt dieses Modul eine gute Basis zur Umwandlung von Spannungen in digitale Werte dar.

2.6.1.2.1. Übersichtsbild

Die Abbildung zeigt ein Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Die Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen.



2.6.1.2.2. Technische Daten

- Komfortables Steckverbindersystem mit Auswerfmechanik
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

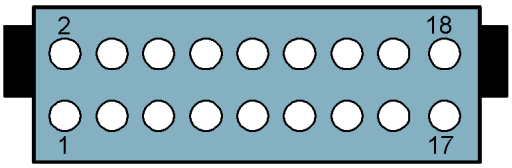
A/D Eingangs-Kanäle

Modus U: (Spannung)
Unipolar: 0-5V, 0-10V
Bipolar: +5V, +10V

Modus I: (Strom)
Bereich: 0-20mA (optional)

2.6.1.2.3. Pinbelegung

2.6.1.2.3.1. A/D Steckverbinder (18pol)



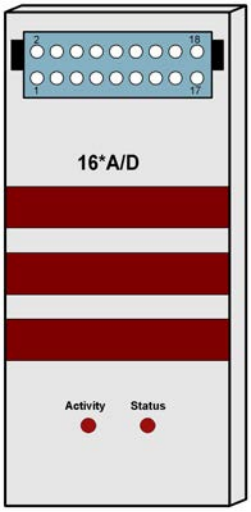
Pin		Pin	
1	AGND	2	AGND
3	AD1	4	AD0
5	AD3	6	AD2
7	AD5	8	AD4
9	AD7	10	AD6
11	AD9	12	AD8
13	AD11	14	AD10
15	AD13	16	AD12
17	AD15	18	AD14

2.6.1.3. RO-AD16_ISO

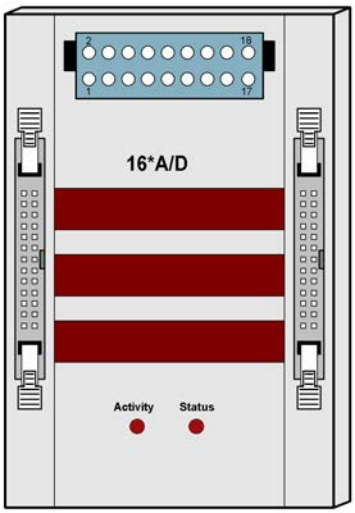
Mit 16 A/D Eingangs-Kanälen (galvanisch getrennt) stellt dieses Modul eine gute Basis zur Umwandlung von Spannungen in digitale Werte dar.

2.6.1.3.1. Übersichtsbild

Die Abbildung zeigt ein Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Die Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen.



2.6.1.3.2. Technische Daten

- Komfortables Steckverbindersystem mit Auswerfmechanik
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

A/D Eingangs-Kanäle

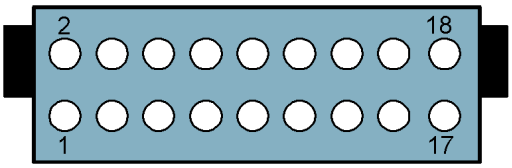
Galvanisch getrennnt

Modus U: (Spannung)
Unipolar: 0-5V, 0-10V
Bipolar: +5V, +10V

Modus I: (Strom)
Bereich: 0-20mA (optional)

2.6.1.3.3. Pinbelegung

2.6.1.3.3.1. A/D Steckverbinder (18pol)



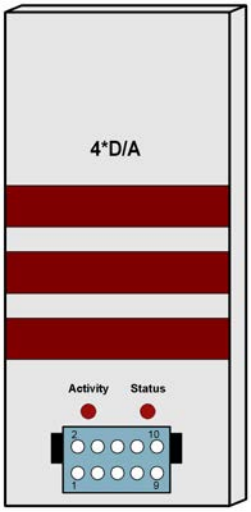
Pin		Pin	
1	AGND	2	AGND
3	AD1	4	AD0
5	AD3	6	AD2
7	AD5	8	AD4
9	AD7	10	AD6
11	AD9	12	AD8
13	AD11	14	AD10
15	AD13	16	AD12
17	AD15	18	AD14

2.6.1.4. RO-DA4

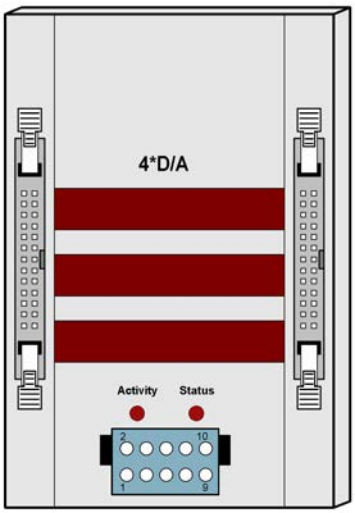
Mit 4 D/A Ausgängen stellt dieses Modul eine gute Basis zur Umwandlung von digitalen Werten in Spannungen dar.

2.6.1.4.1. Übersichtsbild

Die Abbildung zeigt ein Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Die Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen.



2.6.1.4.2. Technische Daten

- Timeout-Schutz
- Komfortables Steckverbindersystem mit Auswerfmechanik
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

D/A Ausgänge

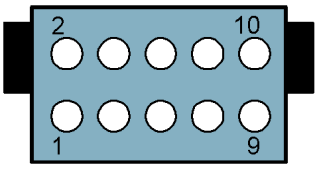
Unipolar: 0-5V, 0-10V
Bipolar: +5V, +10V
Timeout Schutz

2.6.1.4.3. Timeout-Schutz

Der Timeout-Schutz muss per Software separat aktiviert werden. Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden. Das Abschalten der Ausgänge wird durch eine LED signalisiert.

2.6.1.4.4. Pinbelegung

2.6.1.4.4.1. D/A Steckverbinder (10pol)



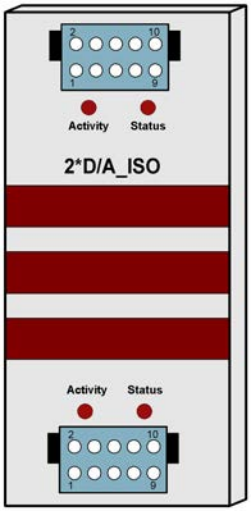
Pin		Pin	
1	AGND	2	DA0
3	AGND	4	DA1
5	AGND	6	DA2
7	AGND	8	DA3
9	AGND	10	AGND

2.6.1.5. RO-DA2_ISO

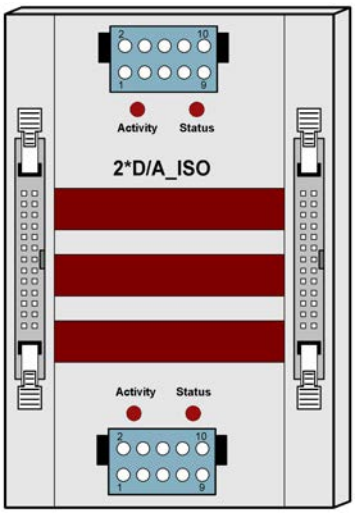
Mit 2 galvanisch getrennten D/A Ausgängen stellt dieses Modul eine gute Basis zur Umwandlung von digitalen Werten in Spannungen dar.

2.6.1.5.1. Übersichtsbild

Die Abbildung zeigt ein Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Die Abbildung zeigt ein I/O-Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen.



2.6.1.5.2. Technische Daten

- Timeout-Schutz
- Komfortables Steckverbindersystem mit Auswerfmechanik
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

D/A Ausgänge

Timeout Schutz

Modus U: (Spannung)
Unipolar: 0V-5V, 0V-10V
Bipolar: +5V, +10V

Modus I: (Strom)
0-20mA, 4-20mA, 0-24mA

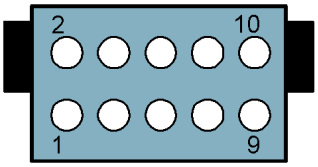
2.6.1.5.3. Timeout-Schutz

Der Timeout-Schutz muss per Software separat aktiviert werden. Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden. Das Abschalten der Ausgänge wird durch eine LED signalisiert.

2.6.1.5.4. Pinbelegung

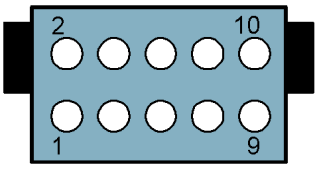
2.6.1.5.4.1. D/A Steckverbinder (10pol)

Steckverbinder oben:



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

Steckverbinder unten:



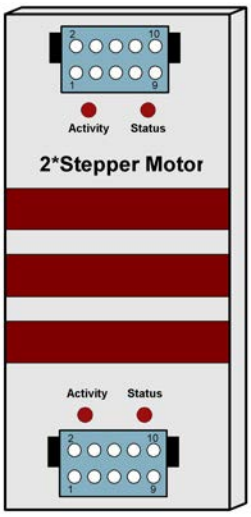
Pin		Pin	
1	VOUT_B	2	+Vsense_B
3	VOUT_B	4	+Vsense_B
5	AGND	6	-Vsense_B
7	AGND	8	-Vsense_B
9	AGND	10	IOUT_B

2.7. Stepper Modul

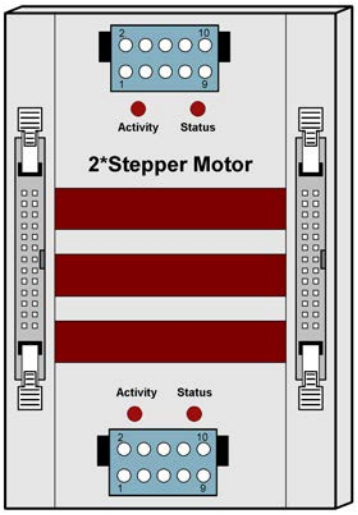
2.7.1. Hardware Beschreibung

2.7.1.1. Übersichtsbild

Die Abbildung zeigt ein Modul mit entsprechender Durchnummerierung der Anschlussklemmen.



Die Abbildung zeigt ein Stepper-Modul des flexiblen Steckverbindersystems mit entsprechender Durchnummerierung der Anschlussklemmen.



2.7.1.2. Technische Daten

Allgemeines:

- Modul für zwei Schrittmotoren
- Komfortables Steckverbindersystem mit Auswerfmechanik
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

Einstellbare Parameter

- Start-/ Stopfrequenz
- Maximale Schrittfrequenz
- Beschleunigungsrampe
- Bremsrampe
- Phasenstrom
- Haltestrom
- Haltezeit

2.7.1.3. Ansteuerung

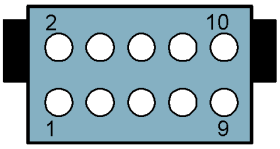
Alle Parameter lassen sich bequem über die DELIB Library einstellen.

Zwei Referenzschalter dienen zum Anfahren zu Referenzpositionen. Weitere zwei Endschalter sorgen für ein sicheres Anhalten und ausschließliches Fahren in entgegengesetzter Richtung, wenn diese betätigt werden.

2.7.1.4. Pinbelegung

2.7.1.4.1. Stepper Steckverbinder (10pol)

Pinbelegung eines Steckverbinders und zugleich eines Schrittmotors:



Pin		Pin	
1	24 V (Motorversorgung)	2	0 V (Motorversorgung)
3	Phase 1 (+)	4	Referenzschalter 2*)
5	Phase 1 (-)	6	Referenzschalter 1*)
7	Phase 2 (+)	8	Endschalter 2*)
9	Phase 2 (-)	10	Endschalter 1*)

*) Die Schalter sind gegen 24 V zu verschalten.

2.8. RO-CNT8

2.8.1. Technische Daten

- 8 Eingangszähler
- max 20 MHz Eingangsfrequenz
- 32-Bit Zähler
- 15V-30V galv. getrennte Eingänge / Umkonfigurierbar auf 5V-12V
- Jeder Eingangskanal kann mit einem Filter versehen werden
- Jeder Eingangskanal zeigt mit einer LED an, dass ein Eingangzustand erfasst wurde

- 2 Ausgangszähler
- Max 5 MHz Ausgangsfrequenz
- 32-Bit Zähler
- Es kann die Anzahl der zu erzeugenden Impulse eingestellt werden (0 = unendlich viele)
- Einstellung der low-Zeit (10ns Schritte)
- Einstellung der high-Zeit (10ns Schritte)
- PWM Ausgänge mit z.B. folgenden Modi:
- 1kHz - Frequenz PWM Verhältnis mit 0,01% Auflösung
- 10kHz - Frequenz PWM Verhältnis mit 0,1% Auflösung
- 100kHz - Frequenz PWM Verhältnis mit 1% Auflösung

- Anzeige-LEDs Activity LED (kurzes Aufblinken bei Schnittstellenkommunikation)
- Status LED (kurzes Aufblinken bei Zugriff auf D/A Wandler)

- Steckverbinder 10 pol. steckbare Klemmleisten für jeweils einen D/A-Ausgang

2.8.2. Counter Modul Modi

Das Counter Modul besitzt folgende 4 Modi:

Modus 0: Zählen von Impulsen Mit diesem Modus lassen sich Impulse anhand von einer steigenden Flanke zählen. Ein zurücksetzen ist hierbei per Software möglich.

Modus 1: Zählen von Impulsen mit "reset bei read"

Mit diesem Modus lassen sich Impulse anhand einer steigenden Flanke zählen. Beim Auslesen des Zählers wird der Zähler gleichzeitig auf Null zurückgesetzt.

Modus 2: Periodendauer Bestimmung

In diesem Modus wird die Periodendauer "T" des angelegten Eingangssignals gemessen. Ein interner 100MHz Zähler dient hierbei als Zeitbasis. Bei einer steigenden Flanke des Eingangssignals wird der 32 Bit Zähler in einen Zwischenspeicher gelegt (Latch) und kann dann ausgewertet werden, währen das nächste Signal wieder gezählt wird.

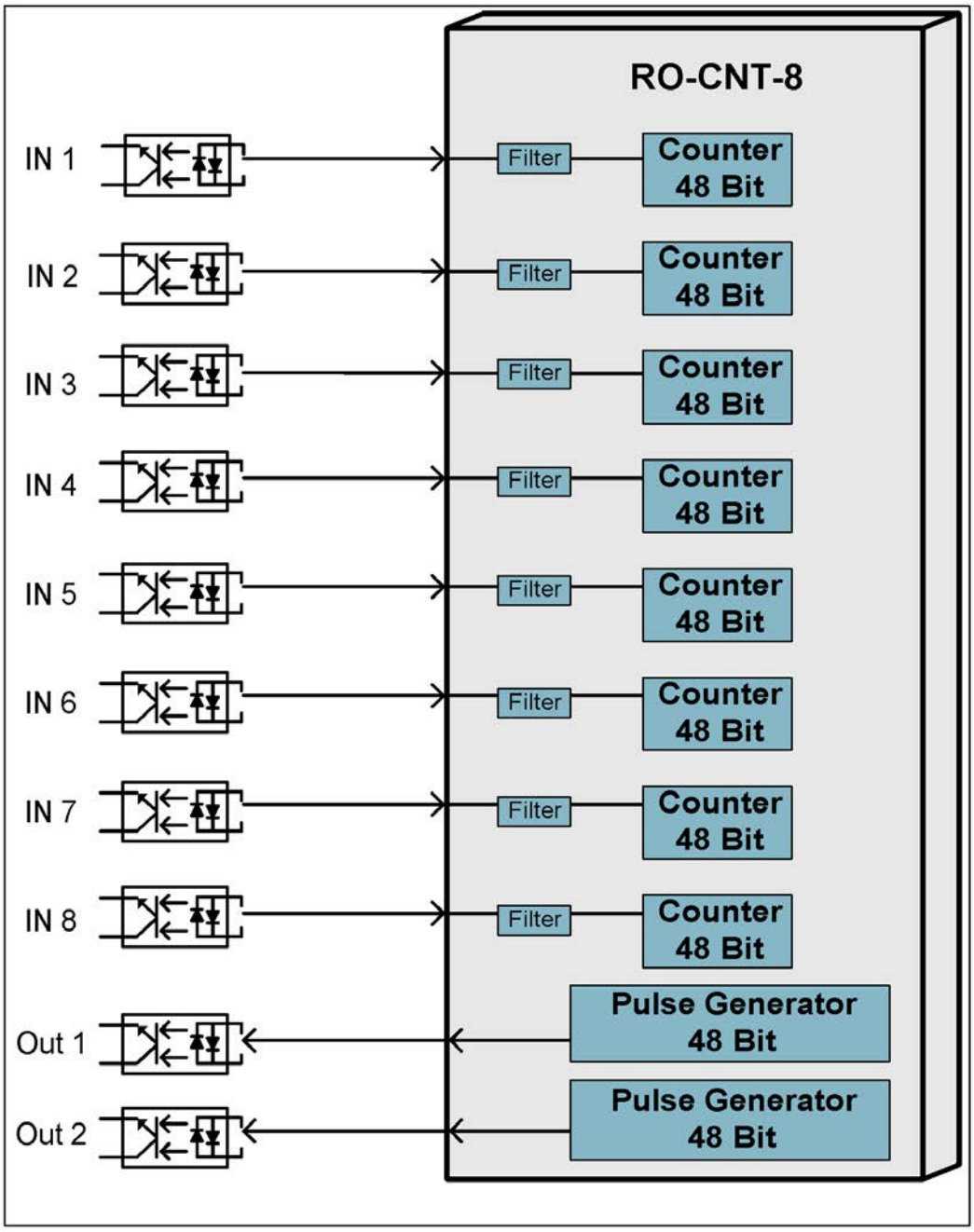
Modus 3: Anzahlmessung/Frequenzmessung

Mit diesem Modus lässt sich die Anzahl der steigenden Flanken des Eingangssignals innerhalb einer bestimmten Zeit messen. Hier lassen sich die Zeiten 1ms, 10ms, 100ms, 1sec. per Software konfigurieren. So lassen sich z.B. Frequenzmessungen durchführen.

Modus 4: PWM Messungen

Mit diesem Modus werden mit zwei unterschiedlichen Zählern die "high"-Zeit und die "low"-Zeit eines Signales gemessen. Dadurch lässt sich das high / low-Verhältnis bestimmen (PWM). Durch die Verwendung eines internen 100MHz Zählers lassen sich hier äusserst genaue PWM Verhältnisse bestimmen.

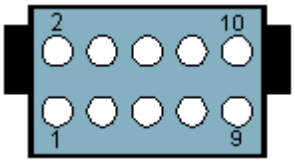
2.8.3. Blockschaltbild



2.8.4. Pinbelegung

2.8.4.1. Pinbelegung RO-CNT8

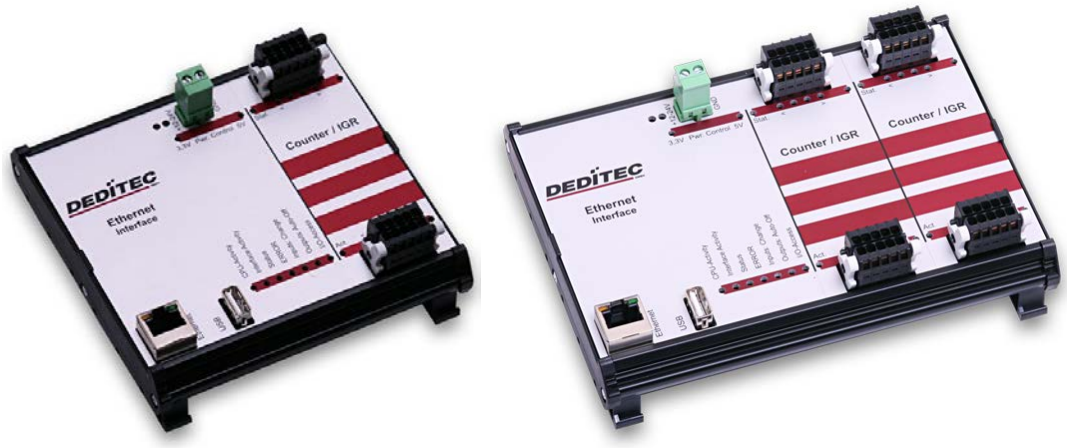
Steckverbinder 10 pol. steckbare Klemmleiste für jeweils 4 * Counter-Eingänge



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

2.9. RO-CNT/IGR

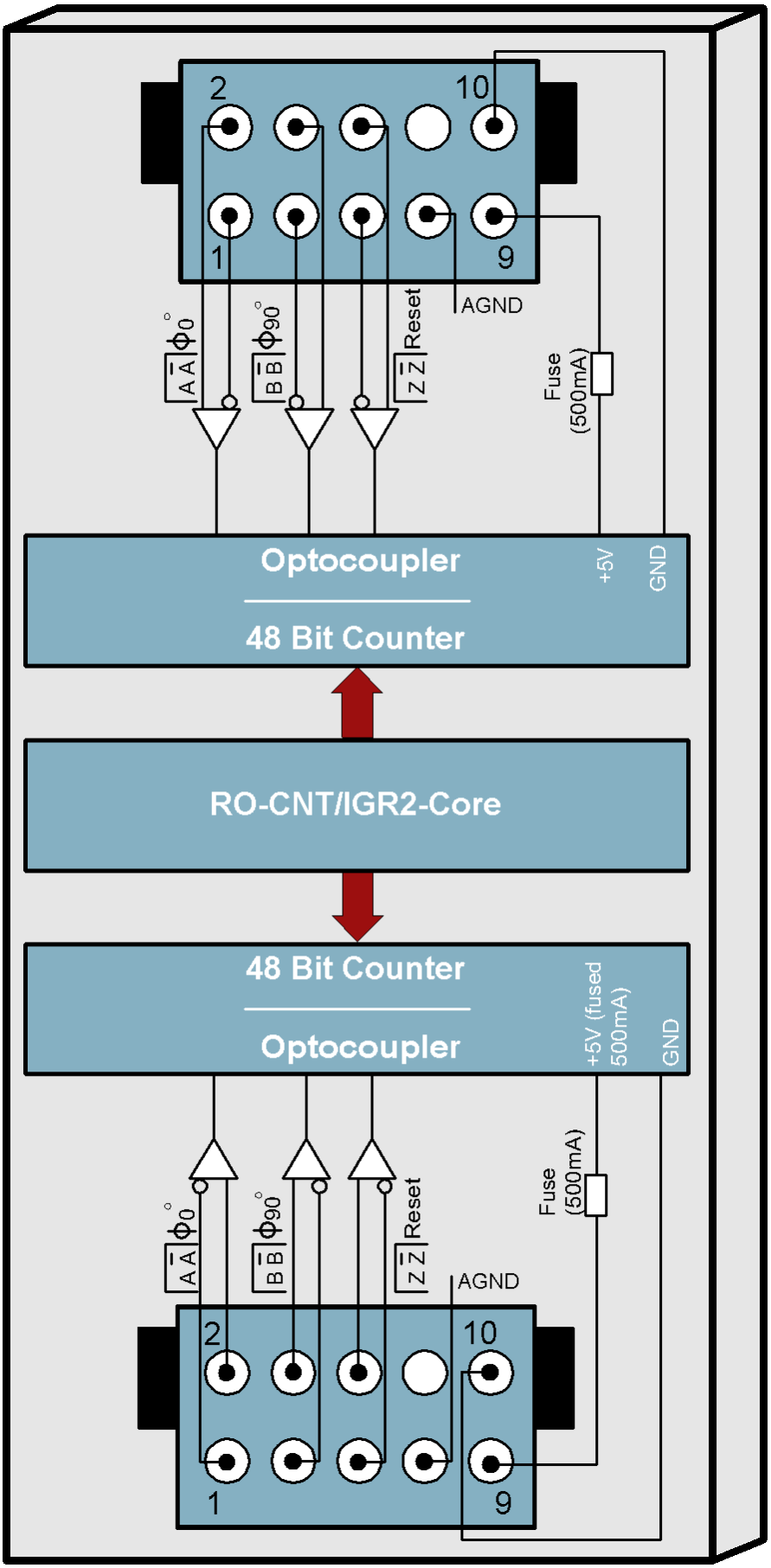
2.9.1. Technische Daten



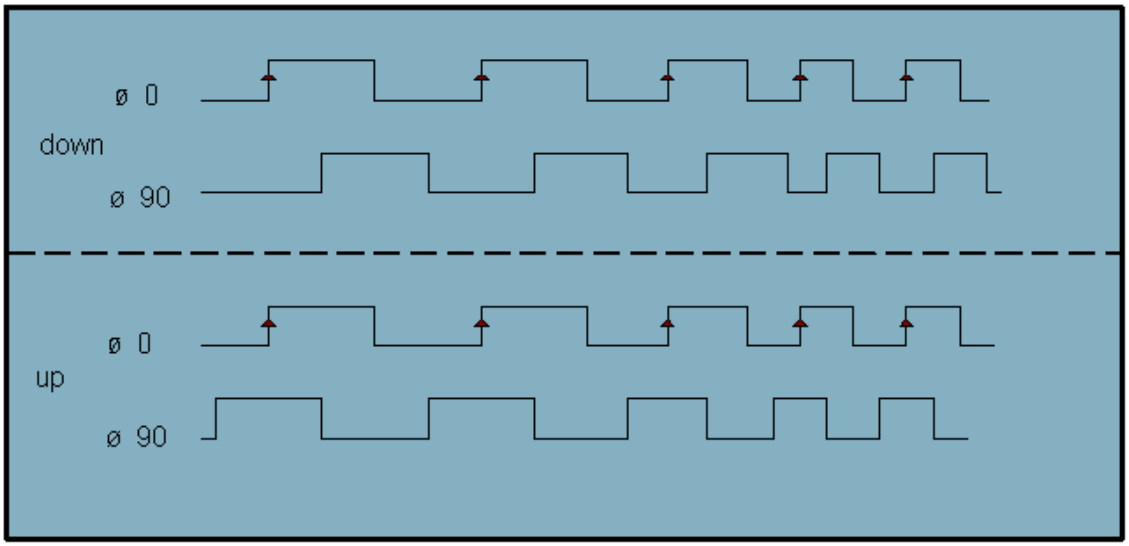
- 48Bit Counter
- Max 10MHz Phi 0/Phi 90° Frequenz
- Galvanisch getrennt (A, B, Z der Eingänge beider Inkrementalgeber Eingänge untereinander und auch zur 24V Modulversorgung)
- Anzeige-LEDs Activity LED (kurzes Aufblinken bei Schnittstellenkommunikation)
- Status LED (kurzes Aufblinken bei Zugriff auf die Inkrementalgeber)
- Richtungs LED's (Leuchten je nach ausgewählter Drehrichtung des Inkrementaldrehgebers)
- Steckverbinder 10 pol. steckbare Klemmleisten für jeweils einen Inkrementalgeber
- Betriebstemperatur 10°C...+50°C

2.9.2. Blockschaltbild

In dem Blockschaltbild wird deutlich, wie das Modul aufgebaut ist.


2
10
1
9
AGND
A A
0°
B B
90°
Z Z
Reset
Fuse
(500mA)
Optocoupler
48 Bit Counter
+5V
GND
RO-CNT/IGR2-Core
48 Bit Counter
Optocoupler
+5V (fused
500mA)
GND
Fuse
(500mA)
A A
0°
B B
90°
Z Z
Reset
AGND
2
10
1
9

2.9.3. Auswertung Phi 0/Phi 90° (Grafisch)



Diese Grafik verdeutlicht wann das Zählmodul hoch/runter zählt.

Ist bei Phi 90° der Pegel low so wird runter gezählt. Ist der Pegel high so wird hoch gezählt.

2.9.4. LEDs

Activity LED:

- kurzes Aufblinken bei Schnittstellenkommunikation

Status LED:

- kurzes Aufblinken bei Zugriff vom Interface auf die Inkrementalgeber

Richtungs LEDs:

- linke Richtungs LED blinkt kurz auf wenn das Zählermodul runter zählt.
- rechte Richtungs LED blinkt kurz auf wenn das Zählermodul hoch zählt.
- beide Richtungs LEDs blinken kurz auf wenn der Zählerstand zurückgesetzt wird.

Counter LED:

- blinkt auf wenn ein Zählvorgang statt findet.

2.9.5. Eingänge

2.9.5.1. Referenzieren

Durch das Neustarten des Zählermoduls wird die absolute Position nicht erkannt und die aktuelle Position auf 0 gesetzt. Dabei können Fehlmessungen entstehen. Durch Endlagenschaltern können Referenzfahrten eingestellt werden bei denen eine bestimmte Position angefahren wird. Dies führt zur genauen Kalibirerung des Drehgebers. Inkrementelle Drehgeber geben pro voller Umdrehung noch einen Referenzimpuls (Index) auf einem dritten Kanal aus. Einige Inkrementalgeber haben auch eine abstandskodierte Referenzmarke.

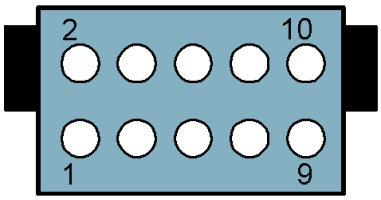
2.9.5.2. Visuelle Kontrolle der Eingänge

Über LED wird der Zustand als auch die Drehrichtung jedes Inkrementalgebers direkt angezeigt. Fehler in der Verdrahtung sind somit schneller erkennbar.

2.9.6. Pinbelegung

2.9.6.1. Pinbelegung RO-CNT/IGR

Steckverbinder 10 pol. steckbare Klemmleiste für jeweils 1 * Inkrementalgeber-Eingang



Pin	Belegung	Funktion	Galvanische Trennung
1	Inputchannel A-	Phi 0°	Galvanisch getrennt
2	Inputchannel A+		
3	Inputchannel B-	Phi 90°	
4	Inputchannel B+		
5	Inputchannel Z-	Reset	
6	Inputchannel Z+		
7	GND-Analog	GND zu Pin 1-6	
8	N.C.		
9	GND	GND zu Pin 10	keine Galvanische Trennung
10	+5V	Fused (500mA)	

2.10. RO-PT100-4 / RO-PT100-8

2.10.1. Einführung

Über den **Zwei- oder Dreileiteranschluss** lässt sich ein PT100-Sensor anschließen, mit dem eine sehr genaue Messung durch den integrierten **24 Bit A/D Wandler** möglich ist.

Je nach Sensor liegt der Temperaturmessungsbereich zwischen **-200°C und 850°C**. Der Leitungswiderstand wird durch eine zusätzliche **Fühlerleitung** ermittelt und ausgeglichen, dadurch ist der Einsatz einer viel längeren Anschlussleitung möglich.

Als Anschlussklemmen kommen servicefreundliche Steckleisten mit Verriegelungsschutz und Auswerfmechanik zum Einsatz. Diese ermöglichen ein schnelles, nachträgliches Umstecken der angeschlossenen Anlagen. Der Leitungsanschluß selbst erfolgt über ein schraubenloses Stecksystem.

2.10.2. Technische Daten

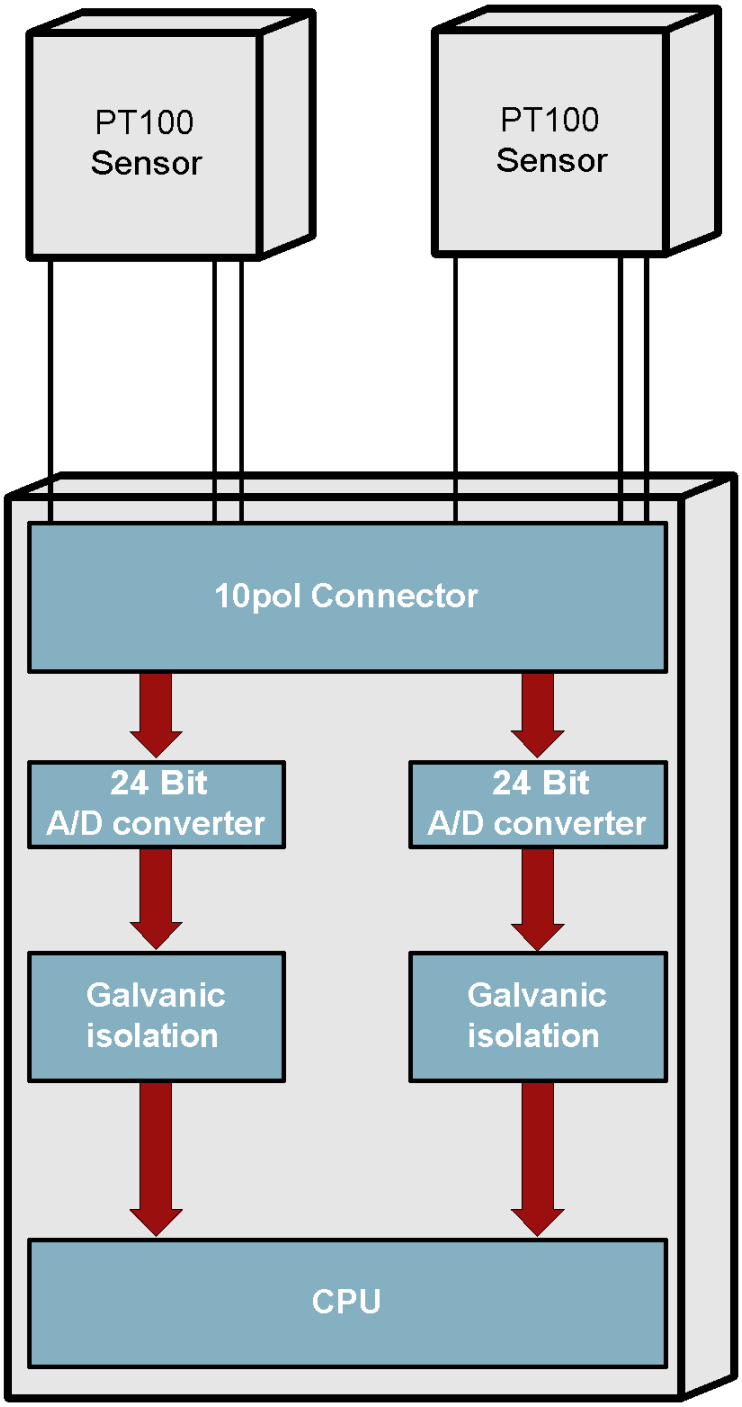
- 24 Bit A/D Wandler für eine genaue Messung
- Der Temperaturmessbereich liegt zwischen -200°C bis 850°C.
- Galvanisch getrennte Eingänge.
- Anzeige-LEDs Activity LED (kurzes Aufblinken bei Schnittstellenkommunikation)
- Status LED (kurzes Aufblinken bei Zugriff auf PT100)
- Anschluss LED 1 je Eingang (kurzes Aufblinken bei einem angeschlossenen PT100 Sensor. Leuchtet bei nicht belegten Eingängen oder bei einem defekten PT100 Sensor).
- Steckverbinder 10 pol. steckbare Klemmleiste für jeweils 2 * PT100-Eingänge
- Kann problemlos mit anderen Modulen der RO Serie kombiniert werden

Produktspezifische Daten:

Produkt	Eingänge
RO-PT100-4	4 * PT100 Eingänge
RO-PT100-8	8 * PT100 Eingänge

2.10.3. Blockschaltbild

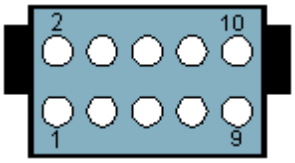
In dem Blockschaltbild wird deutlich, wie das Modul aufgebaut ist.



2.10.4. Pinbelegung

2.10.4.1. Pinbelegung RO-PT100

Steckverbinder 10 pol. steckbare Klemmleiste für jeweils 2 * PT100-Eingänge



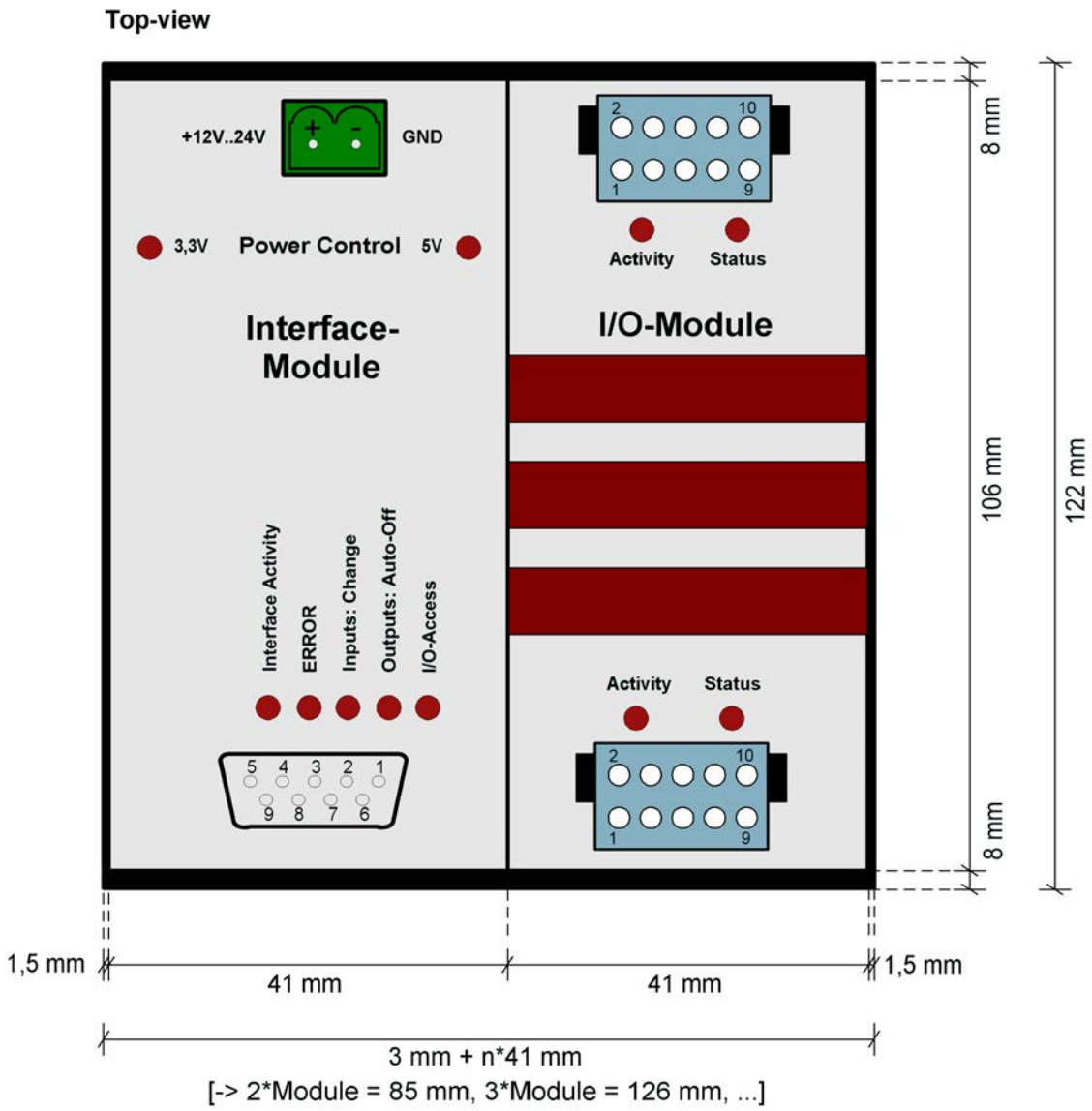
Pin	Belegung
1	AIN-_A
2	AIN+_A
3	AIN-_A
4	REFIN+_A
5	-
6	-
7	AIN-_B
8	AIN+_A
9	AIN-_B
10	REFIN+_B

Zweikanal Konfiguration: Der Temperatursensor wird an Pin 1 (AIN- A) und Pin2 (AIN+ A) angeschlossen. Pin 3 (AIN-A) und Pin 4 (REFIN + A) müssen hierbei überbrückt werden.

Dreikanal Konfiguration: Der Temperatursensor wird an Pin 1 (AIN- A) und Pin2 (AIN+ A) angeschlossen. Pin 4 (REFIN+ A) dient hierbei als Fühlerleitung und muss zwischen Temperatursensor und Pin 1 (AIN- A) angeklemmt werden.

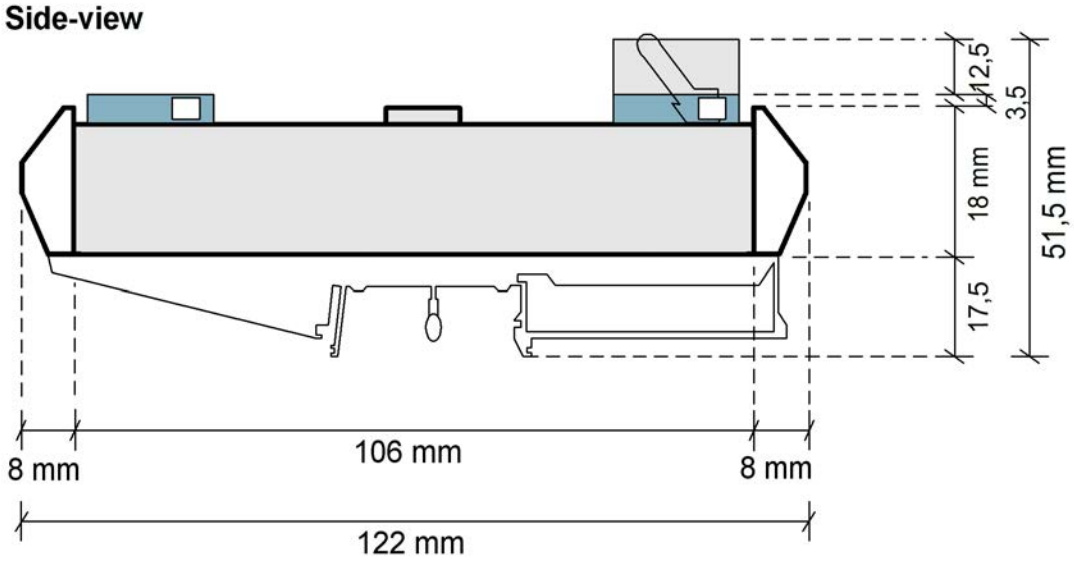
2.11. Abmessung der RO-Serie

2.11.1. Top view



Hinweis:
Die Breite des ETH-Interfaces entspricht der doppelten Breite eines CAN-, SER- oder USB-Interfaces.
So ergibt sich eine Breite von 3 mm + 82 mm + n*41 mm

2.11.2. Side view



Firmware Update

III

3. Firmware Update

3.1. Über DEDITEC-Firmware

Die meisten DEDITEC-Produkte verfügen über einen eigenen Microcontroller. Dieser Prozessor ist für die Steuerung aller Abläufe der Hardware verantwortlich.

Um die für den Prozessor benötigte **Firmware** im nachhinein zu ändern, stellen wir unser kostenloses Tool **DEDITEC-Flasher** zur Verfügung. Mit diesem Tool hat der Kunde die Möglichkeit neu veröffentlichte Firmware-Versionen, direkt bei sich vor Ort auf das Modul zu übertragen.

Da neue Firmware Versionen in der Regel neue Funktionen für Ihr Produkt "freischalten", empfehlen wir daher ein regelmäßiges Fimware-Update Ihrer DEDITEC-Produkte.

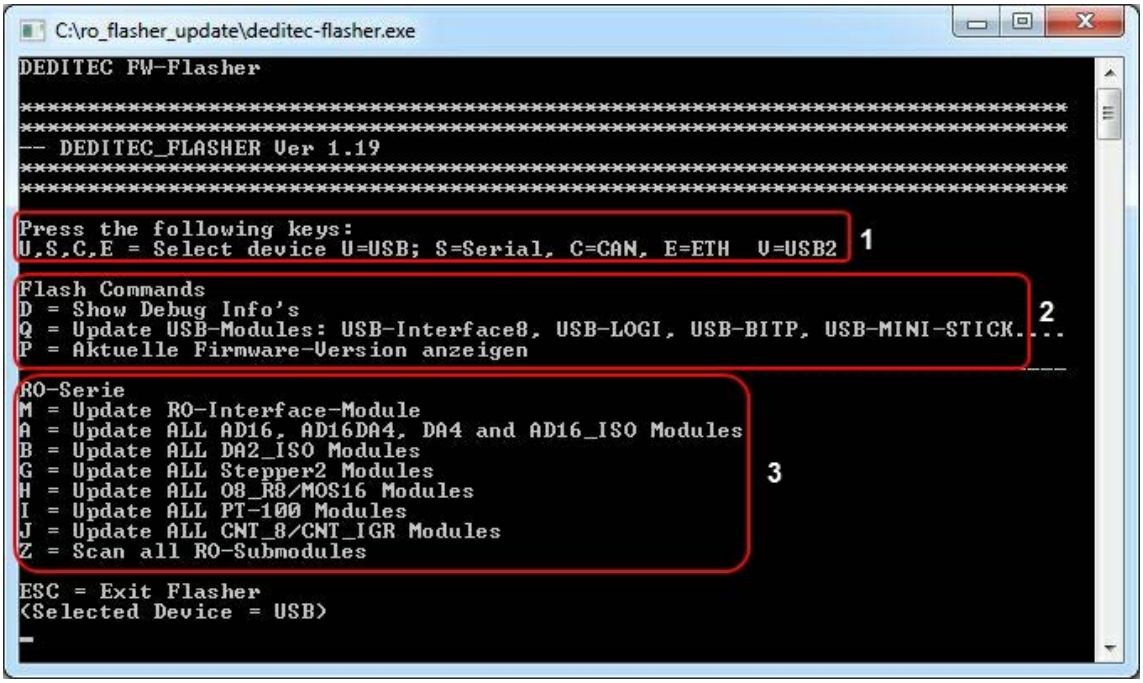
3.2. DEDITEC Flasher

Die aktuellste Firmware-Version zu Ihrem DEDITEC-Produkt finden Sie immer im Download-Bereich unserer Homepage.
(-> http://www.deditec.de/delib)

Vorgehensweise nach Download

- Entpacken Sie das ZIP-Archiv.
- Starten Sie die Anwendung **deditec-flasher.exe**

Nun öffnet sich folgende Anwendung:



Eine detailierte Beschreibung der Funktionsweise, sowie der verfügbaren Kommandos, finden Sie auf den nachfolgenden Seiten.

3.2.1. Flashen der RO-Serie

1. Interface-Auswahl für Module der **RO-Serie**:

Kommando (Taste)	Interface
U	USB-Interface
S	Serielles Interface (RS-232 / RS-485)
C	CAN-Interface
E	Ethernet-Interface
V	USB2-Interface

Hinweis:

1) Das aktuell ausgewählte Interface sehen Sie immer in der letzten Zeile: **(Selected Device = USB)**

2) Zum Flashen der RO-CAN-Serie, muss das Modul mittels des CAN/SER Adapters mit dem PC verbunden werden.

2. Zusätzliche Optionen

Kommando (Taste)	Beschreibung
D	Flasher wird im DEBUG-Modus ausgeführt. Hierbei werden zusätzliche Informationen angezeigt
P	Liest die aktuelle Firmware-Version angeschlosser DEDITEC-Module aus

3. Auswahl des zu flashenden Moduls

Kommando (Taste)	Beschreibung
M	Flasht das RO-Interface-Modul
A	Flasht alle AD16, AD16-DA4, DA4 oder AD16_ISO Submodule
B	Flasht alle DA2_ISO Submodule
G	Flasht alle STEPPER2 Submodule
H	Flasht alle O8-R8 oder M16 Submodule
I	Flasht alle PT100 Submodule
J	Flasht alle CNT8 oder CNT/IGR Submodule
Z	Scannt das Modul nach verfügbaren Submodulen

Nach erfolgreichem Update-Vorgang erscheint die Meldung **FLASH - OK!**

3.2.2. Flashen anderer Produkte (nicht RO-Serie)

USB-Produkte der "Nicht-RO-Serie", wie z.B. USB-Mini-Stick-Serie oder USB-LOGI-500, werden ausschließlich mit der **Taste Q** geflasht.

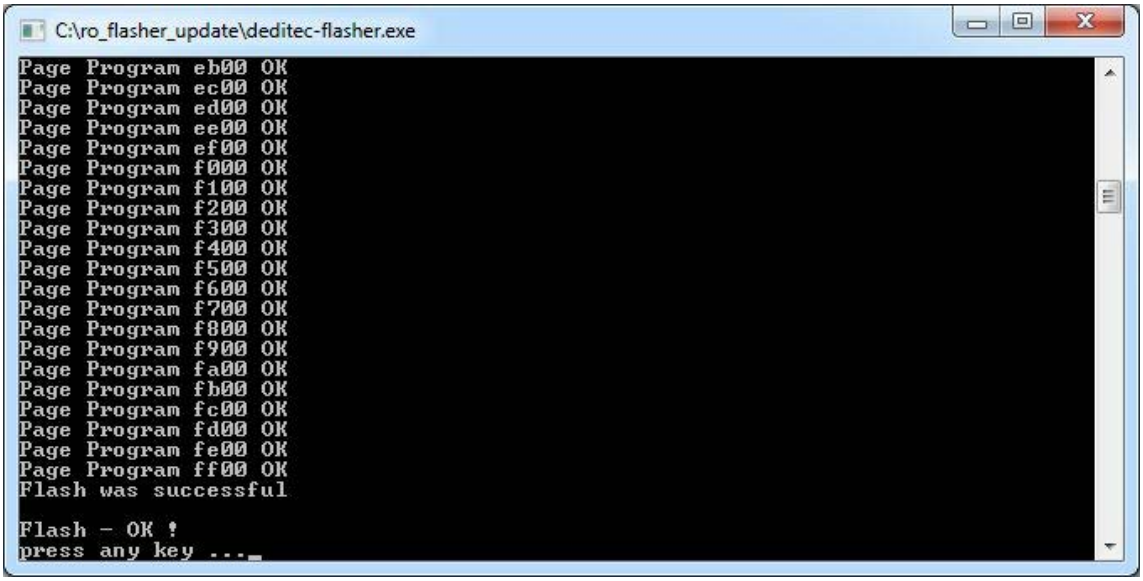
1. Produktauswahl

Kommando (Taste)	Modul
Q	USB-Mini-Stick-Serie, USB-CAN-Stick, USB-Controller-8, USB-LOGI-Serie, USB-OPTOIN-X-RELAIS-X-Serie, USB-SPI-MON, USB-Watchdog

2. Zusätzliche Optionen

Kommando (Taste)	Beschreibung
D	Flasher wird im DEBUG-Modus ausgeführt. Hierbei werden zusätzliche Informationen angezeigt
P	Liest die aktuelle Firmware-Version angeschlosser DEDITEC-Module aus

Nach erfolgreichem Update-Vorgang erscheint die Meldung **FLASH - OK!**



Software

IV

4. Software

4.1. Benutzung unserer Produkte

4.1.1. Ansteuerung über grafische Anwendungen

Wir stellen Treiberinterfaces z.B. für LabVIEW und ProfiLab zur Verfügung. Als Basis dient die DELIB Treiberbibliothek, die von ProfiLab direkt angesteuert werden kann.
Für LabVIEW bieten wir eine einfache Treiberanbindung mit Beispielen an!

4.1.2. Ansteuerung über unsere DELIB Treiberbibliothek

Im Anhang befindet sich die komplette Funktionsreferenz für die Integration unserer API-Funktionen in Ihre Software. Des Weiteren bieten wir passende Beispiele für folgende Programmiersprachen:

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

4.1.3. Ansteuerung auf Protokollebene

Das Protokoll für die Ansteuerung unserer Produkte legen wir komplett offen. So können Sie auch auf Systemen ohne Windows oder Linux unsere Produkte einsetzen!

4.1.4. Ansteuerung über mitgelieferte Testprogramme

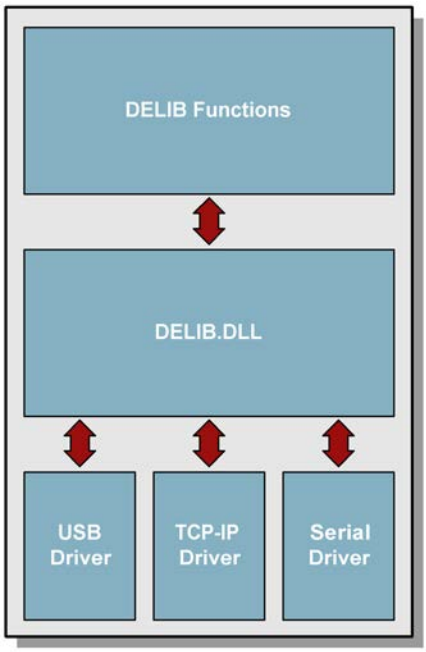
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

So können z.B. Relais direkt getestet werden oder Spannungen am A/D Wandler direkt überprüft werden.

4.2. DELIB Treiberbibliothek

4.2.1. Übersicht

Die folgende Abbildung erläutert den Aufbau der DELIB Treiberbibliothek



Die DELIB Treiberbibliothek ermöglicht ein einheitliches Ansprechen von DEDITEC Hardware, mit der besonderen Berücksichtigung folgender Gesichtspunkte:

- Betriebssystem unabhängig
- Programmiersprachen unabhängig
- Produkt unabhängig

4.2.1.1. Programmieren unter diversen Betriebssystemen

Die DELIB Treiberbibliothek ermöglicht ein einheitliches Ansprechen unserer Produkte auf diversen Betriebssystemen. Wir haben dafür gesorgt, dass mit wenigen Befehlen alle unsere Produkte angesprochen werden können. Dabei spielt es keine Rolle, welches Betriebssystem Sie verwenden. - Dafür sorgt die DELIB !

4.2.1.2. Programmieren mit diversen Programmiersprachen

Für das Erstellen eigener Anwendungen stellen wir Ihnen einheitliche Befehle zur Verfügung. Dies wird über die DELIB Treiberbibliothek gelöst.
Sie wählen die Programmiersprache !

So können leicht Anwendung unter C++, C, Visual Basic, Delphi oder LabVIEW® entwickelt werden.

4.2.1.3. Schnittstellenunabhängiges programmieren

Schreiben Sie Ihre Anwendung schnittstellenunabhängig !
Programmieren Sie eine Anwendung für ein USB-Produkt von uns. - Es wird auch mit einem Ethernet oder RS-232 Produkt von uns laufen !

4.2.1.4. SDK-Kit für Programmierer

Integrieren Sie die DELIB in Ihre Anwendung. Auf Anfrage erhalten Sie von uns kostenlos Installationsskripte, die es ermöglichen, die DELIB Installation in Ihre Anwendung mit einzubinden.

4.2.2. Unterstützte Betriebssysteme

Unsere Produkte unterstützen folgende Betriebssysteme:

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

4.2.3. Unterstützte Programmiersprachen

Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

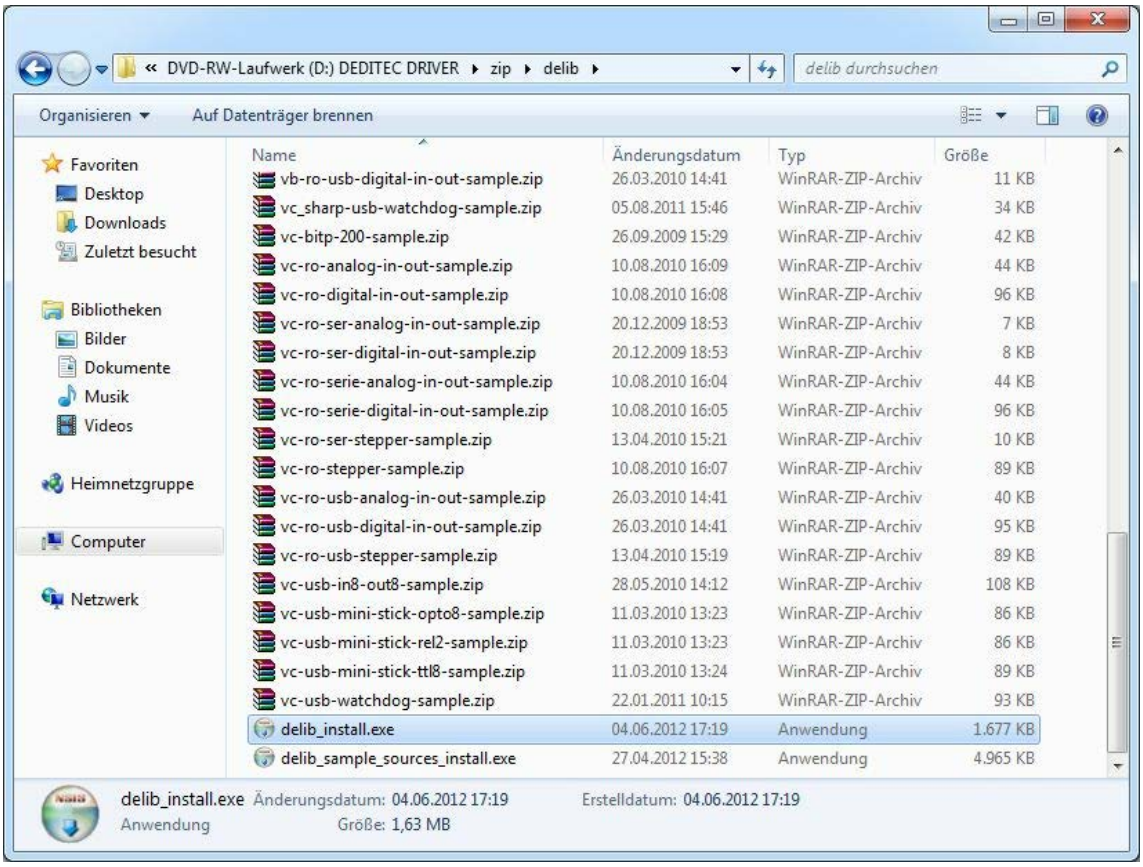
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

4.2.4. Installation DELIB-Treiberbibliothek

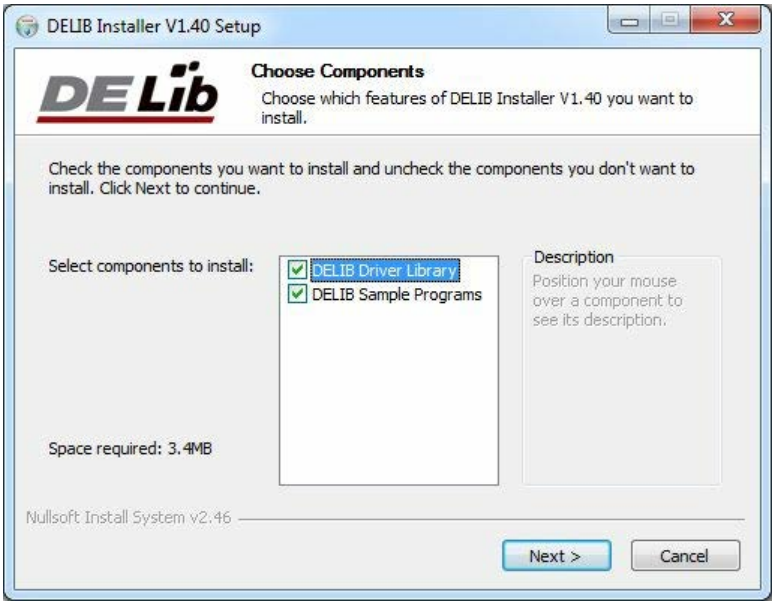
Legen Sie die DEDITEC Driver CD in das Laufwerk und starten Sie **delib_install.exe**.

Eine aktuelle Version der DELIB-Treiberbibliothek finden Sie auch im Internet unter

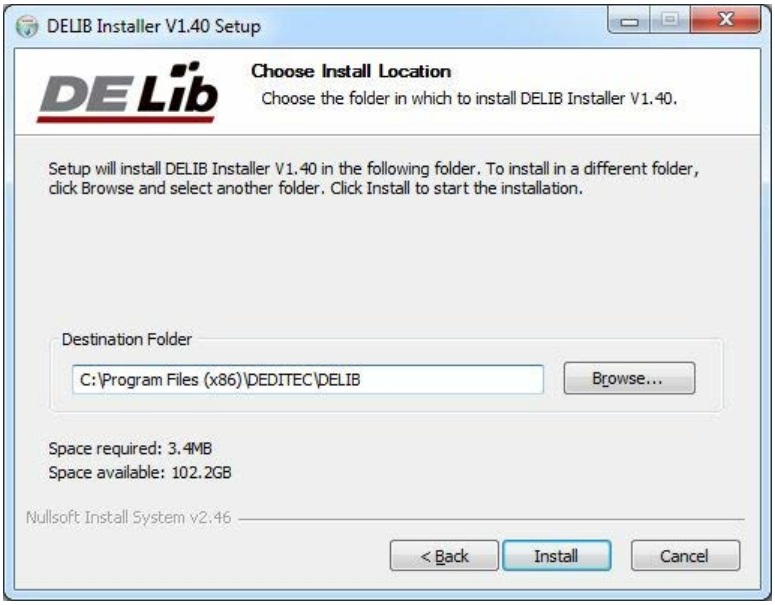
http://www.deditec.de/delib



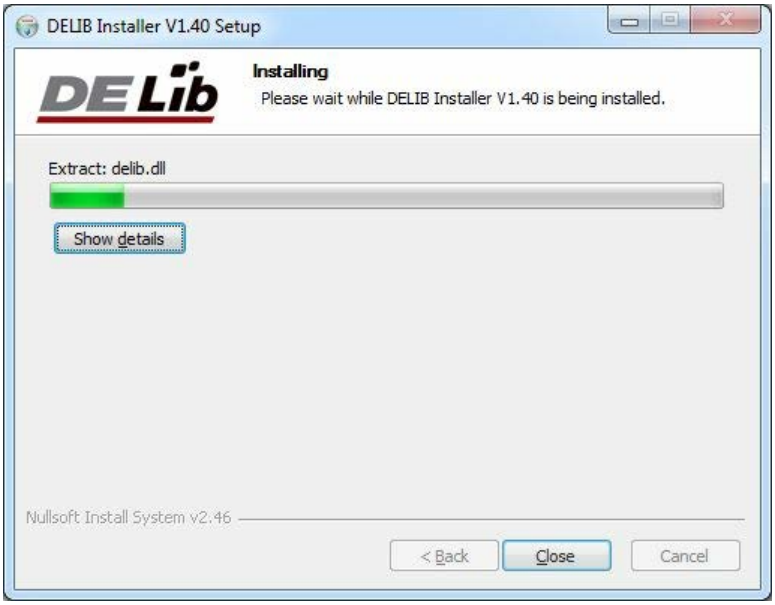
Startbild des DELIB Installers



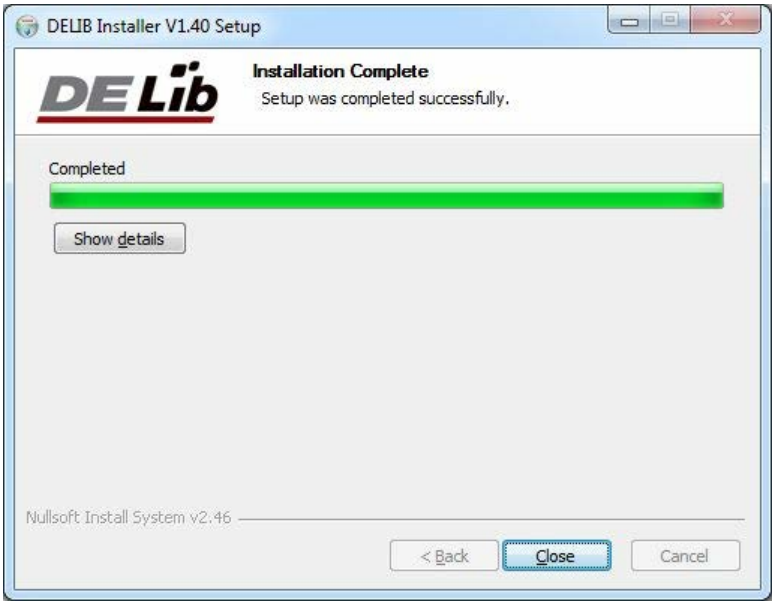
Drücken Sie **Next**.



Drücken Sie **Install**.



Die DELIB-Treiberbibliothek wird nun installiert.



Die DELIB-Treiberbibliothek wurde erfolgreich installiert. Drücken Sie **Close** um die Installation zu beenden.

Hinweis:

Nach Abschluss der Installation, kann das Modul nun über das **DELIB Configuration Utlity** (nächstes Kapitel) konfiguriert werden.

4.2.5. DELIB Configuration Utility

Das **DELIB Configuration Utility** wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → DELIB Configuration Utility.

Das **DELIB Configuration Utility** ist ein Programm zur Konfiguration und Unterteilung Identischer USB-Module im System. Zusätzlich können mit diesem Tool auch Produkte der RO-ETH-, RO-CAN- oder RO-SER-Serie konfiguriert werden, um sie beispielsweise mit den DEDITEC-Beispielprogrammen nutzen zu können.

4.2.5.1. Konfiguration RO-ETH-Serie

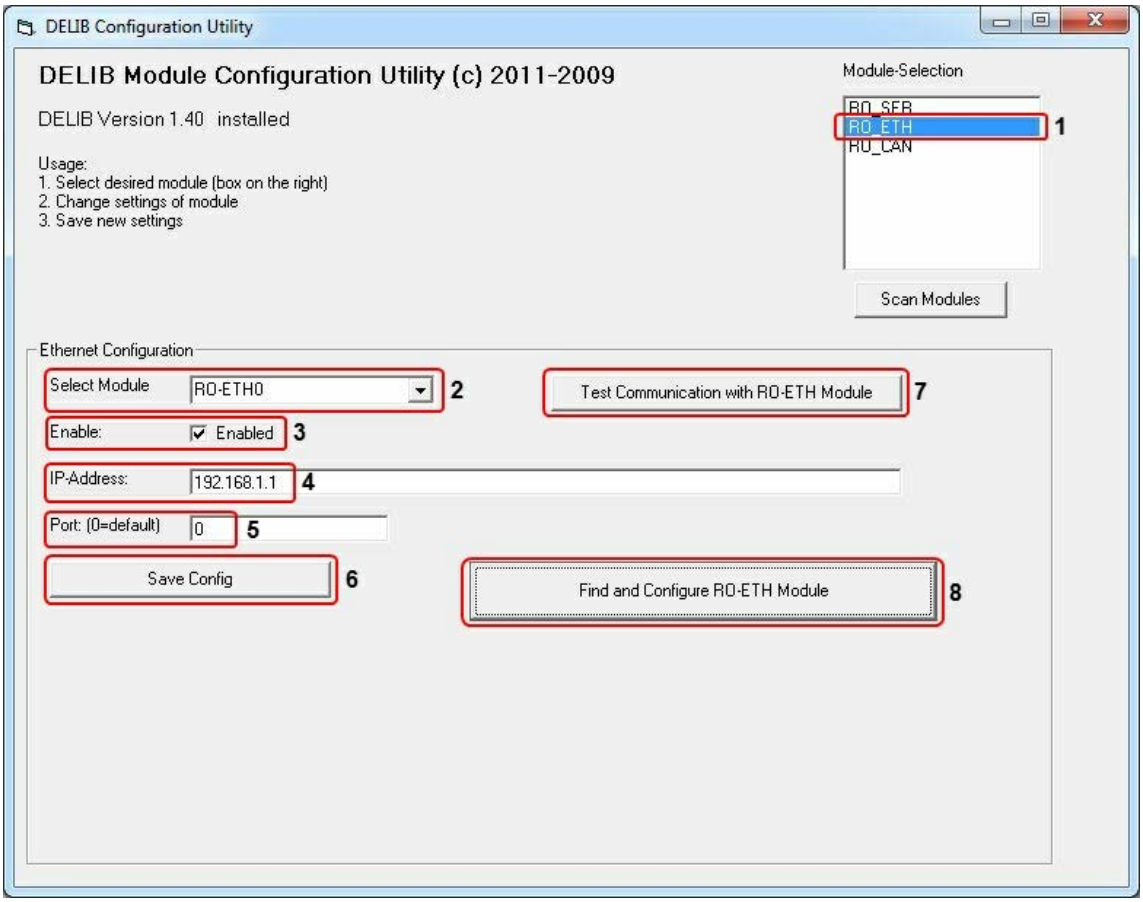
Das DELIB Configuration Utility ermöglicht eine einfache Konfiguration des RO-ETH-Moduls. Folgende Einstellungen können hiermit verändert werden:

- Modulname
- IP-Adresse
- Netz-Maske
- Standard-Gateway
- DNS-Server

Zusätzlich lassen sich mit diesem Tool alle DEDITEC Ethernet-Module im Netzwerk anzeigen.

Auf den nachfolgenden Seiten, wird Schritt für Schritt gezeigt, wie ein RO-ETH-Modul konfiguriert werden kann.

Starten Sie das DELIB Configuration Utility:



1. **RO-ETH** bei der Module-Selection auswählen
2. Über **Select Module** können bis zu 16 verschiedene RO-ETH-Module konfiguriert werden.
3. Über die Checkbox **Enabled** wird bestimmt, ob die aktuelle Konfiguration aktiv ist.
4. Hier wird die IP-Adresse des RO-ETH-Moduls definiert. Im Auslieferungszustand ist die IP-Adresse mit dem Wert **192.168.1.1** vordefiniert.
5. Hier wird der Port des RO-ETH-Moduls definiert. Im Auslieferungszustand ist der Port mit dem Wert **0** vordefiniert.
6. Mit dem Button **Save Config** wird die aktuelle Konfiguration in die Windows-Registry geschrieben bzw. gespeichert.
7. Mit dem Button **Test Communication with RO-ETH Module** kann die Kommunikation mit einem RO-ETH-Modul getestet werden. Hierbei wird die aktuelle Konfiguration verwendet.

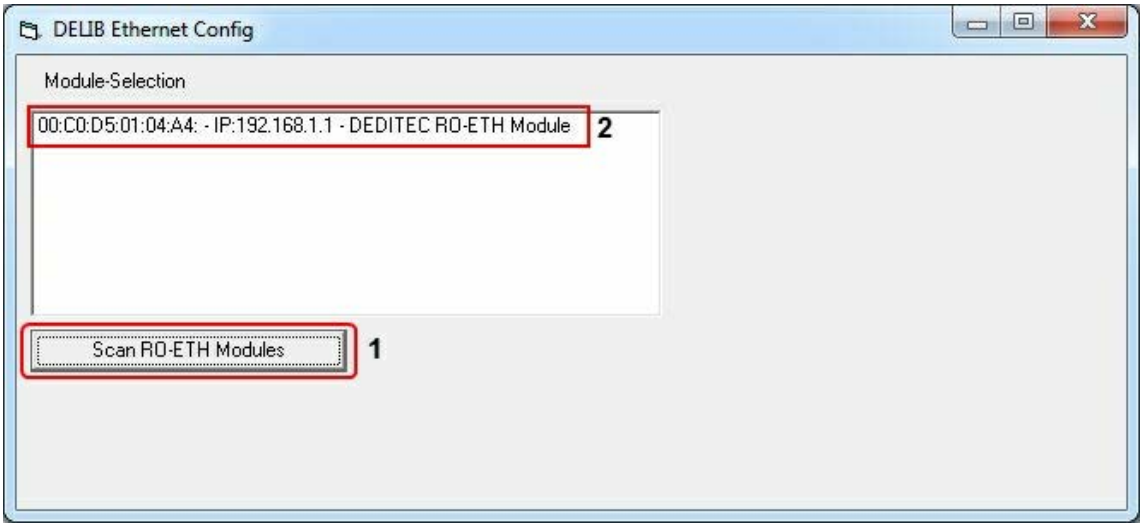
8. Mit dem Button **Find and Configure RO-ETH Module** kann das Netzwerk nach RO-ETH-Modulen durchsucht werden. Anschließend können diese konifguriert werden.

Hinweis:

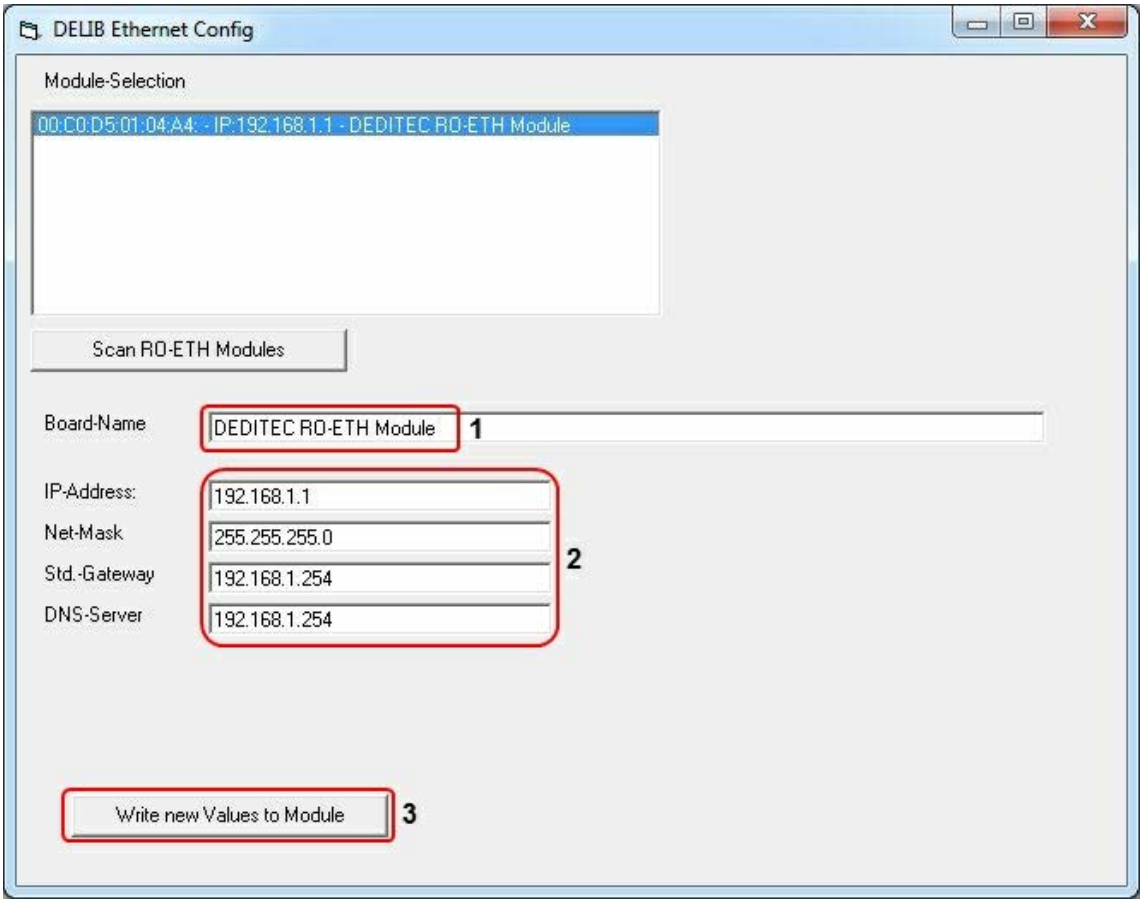
1) Da die gesamte Konfiguration in die Windows-Registry geschrieben wird, benötigen Sie Administrator Rechte.

2) Diese Konfiguration ist notwendig, um das RO-ETH-Modul beispielsweise über die mitgelieferten Beispielprogramme anzusteuern.

Find and Configure RO-ETH Module



1. Mit dem Button **Scan RO-ETH Modules** wird der lokale Ethernet-Strang nach DEDITEC-ETH-Modulen durchsucht. Hierbei benutzen wir ein Ethernet Protokoll, welches nicht geroutet wird. Deshalb sind nur Module konfigurierbar, die am Bus angeschlossen sind. Vorteil dieser Methode ist, dass auch Module gefunden werden, die nicht im gleichen Sub-Netz des Rechners liegen, von dem aus konfiguriert wird.
2. Klicken Sie auf das Modul, welches Sie konfigurieren wollen



1. Hier kann der Name des RO-ETH-Moduls verändert werden
2. Hier können IP-Adresse, Netz-Maske, Standard-Gateway und DNS-Server des RO-ETH-Moduls verändert werden.
3. Über den Button **Write new Values to Module** wird die aktuelle Konfiguration im Modul gespeichert.

Hinweis:

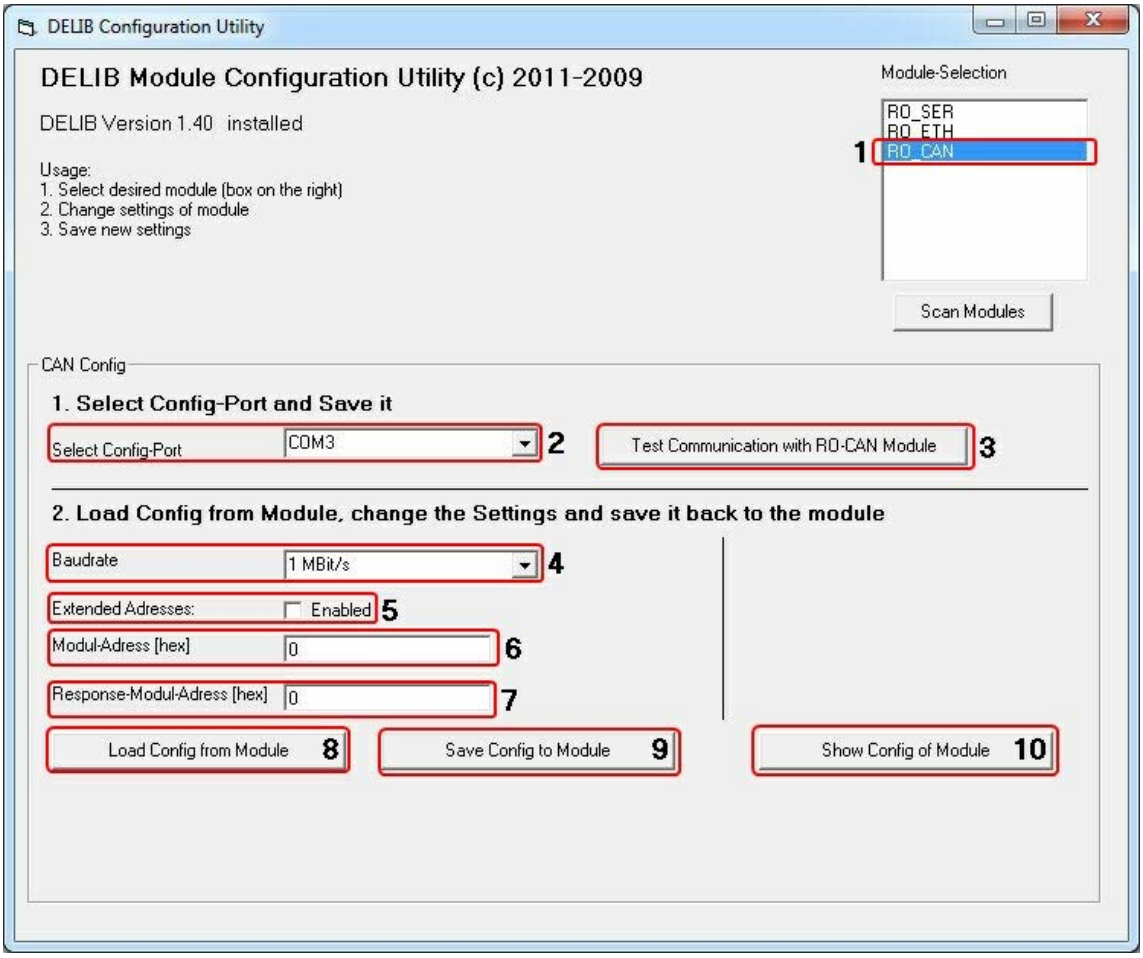
Bei der Konfiguration des RO-ETH Moduls ist zu beachten, dass die von Ihnen ausgewählte IP-Adresse im gleichen IP-Segment liegt, wie der Steuer-PC. Natürlich darf auch keine bereits belegte IP-Adresse benutzt werden.

Wenn die Standard-IP-Adresse des Moduls nicht aus dem Adressbereich des Netzwerks stammt, dann ist das Modul vorerst nicht über TCP/IP erreichbar. Erreichbarkeitsprobleme treten auch auf, wenn diese IP-Adresse bereits belegt ist. Anhand des sehr einfach zu bedienenden Utilities können die IP-Adresse und die Netzmaske des Ethernet-Moduls dennoch konfiguriert werden.

Alternativ kann das Modul auch direkt am PC angeschlossen werden um auf direktem Wege die IP-Adresse und die Netzmaske einzustellen. Nachdem die Erreichbarkeit gegeben ist, erfolgt die weitere Konfiguration bequem über ein Browser auf dem integrierten Web-Server des Ethernet-Moduls.

Fragen Sie hierzu Ihren System Administrator.

4.2.5.2. Konfiguration RO-CAN-Serie



1. **RO-CAN** bei der Module-Selection auswählen.
2. Über **Select Config-Port** stellen Sie den verwendeten COM-Port ein.
3. Mit dem Button **Test Communication with RO-CAN Module** kann die Kommunikation mit einem RO-CAN-Modul getestet werden.
4. Hier kann die **Baudrate** des Moduls konfiguriert werden.
5. Mit **Extended Adresses** kann die Adressierung der Module von CAN 2.0A (11 Bit) auf CAN 2.0B (29 Bit) gewechselt werden.
6. Die **Modul-Adress** kann hier konfiguriert werden. Bitte achten Sie darauf, den Wert in **Hexadezimal** anzugeben.
7. Die **Response-Modul-Adress** gibt an, an welche Adresse eine Nachricht gesendet wird, sobald eine Nachricht vom Modul empfangen wurde.

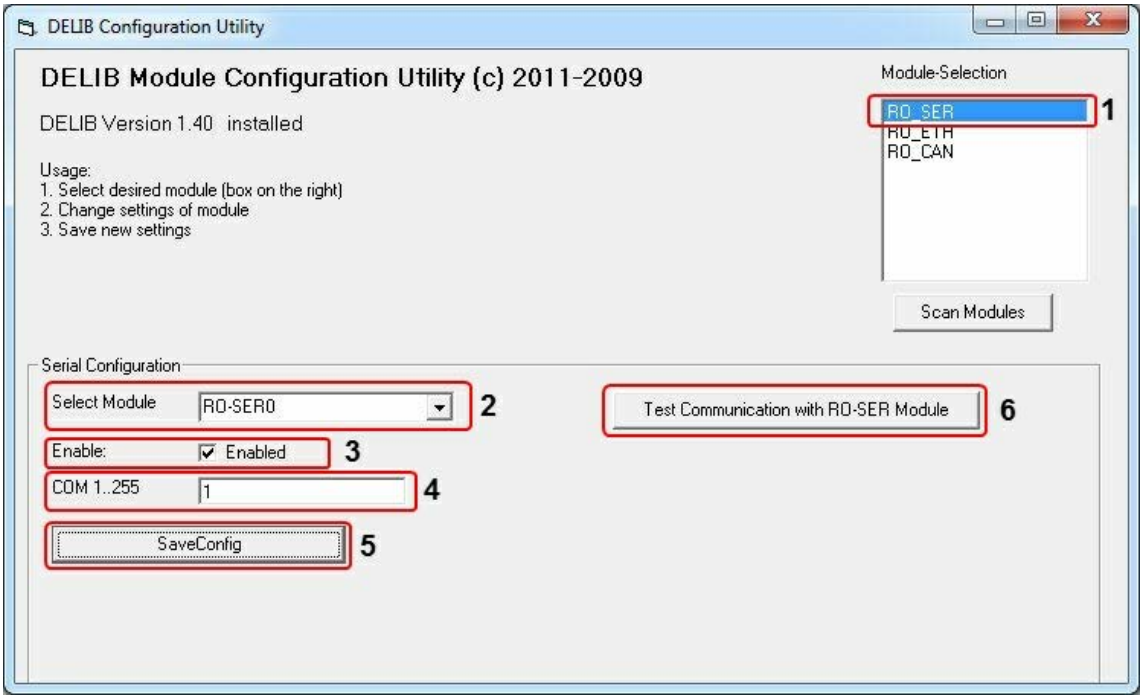
8. Mit dem Button **Load Config from Module** kann die auf dem Modul gespeicherte Konfiguration ausgelesen werden.
9. Mit dem Button **Save Config to Module** kann wird die aktuelle Konfiguration auf dem Modul gespeichert.
10. Mit dem Button **Show Config of Module** kann die auf dem Modul gespeicherte Konfiguration ausgelesen werden.

Hinweis:

1) Da die gesamte Konfiguration in die Windows-Registry geschrieben wird, benötigen Sie Administrator Rechte.

2) Diese Konfiguration ist notwendig, um das RO-CAN-Modul beispielsweise über die mitgelieferten Beispielprogramme anzusteuern.

4.2.5.3. Konfiguration RO-SER-Serie



1. **RO-SER** bei der Module-Selection auswählen.
2. Über **Select Module** können bis zu 16 verschiedene RO-SER-Module konfiguriert werden.
3. Über die Checkbox **Enabled** wird bestimmt, ob die aktuelle Konfiguration aktiv ist.
4. Hier wird der COM-Port an dem das Modul angeschlossen ist, als Zahl angegeben.
5. Mit dem Button **Save Config** wird die aktuelle Konfiguration in die Windows-Registry geschrieben bzw. gespeichert.
6. Mit dem Button **Test Communication with RO-SER Module** kann die Kommunikation mit einem RO-SER-Modul getestet werden. Hierbei wird die aktuelle Konfiguration verwendet.

Hinweis:

1) Da die gesamte Konfiguration in die Windows-Registry geschrieben wird, benötigen Sie Administrator Rechte.

2) Diese Konfiguration ist notwendig, um das RO-SER-Modul beispielsweise über die mitgelieferten Beispielprogramme anzusteuern.

4.2.5.4. Konfiguration RO-USB-Serie und USB-Module

Das DELIB Configuration Utility ermöglicht eine einfache Konfiguration der RO-USB-Serie und der USB-Module.

Um mehrere identische USB-Module (USB-Module mit gleicher Modul-ID) in einem System verwenden zu können, muss jedem Modul mit dem DELIB Configuration Utility eine eindeutige NR zugeordnet werden.
Befindet sich nur ein USB-Modul, oder mehrere USB-Module mit unterschiedlicher Modul-ID (z.B. RO-USB-O16 und USB-RELAIS-8) im System, ist keine Konfiguration nötig, da die Module über die ID eindeutig identifizierbar sind.

Auf der nachfolgenden Seite finden Sie ein Beispiel zur Konfiguation identischer USB-Module.

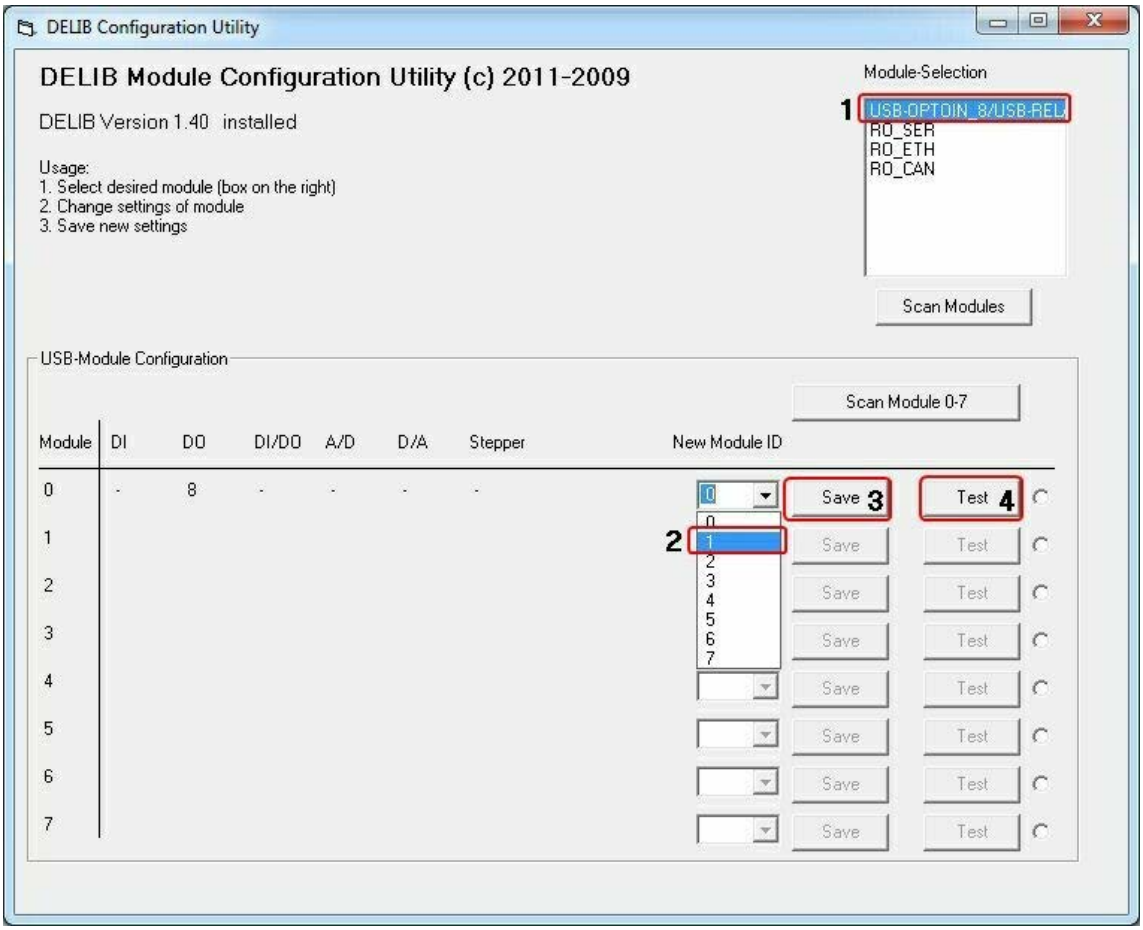
4.2.5.4.1. Beispiel zur Konfiguration identischer USB-Module

Im folgenden Beispiel, erklären wir die Konfiguration von 2 USB-RELAIS-8 Modulen.

Schritt 1

Verbinden Sie zunächst nur ein USB-RELAIS-8 mit dem PC und starten Sie das DELIB Configuration Utility.

Hier muss nun folgende Konfiguration vorgenommen werden:

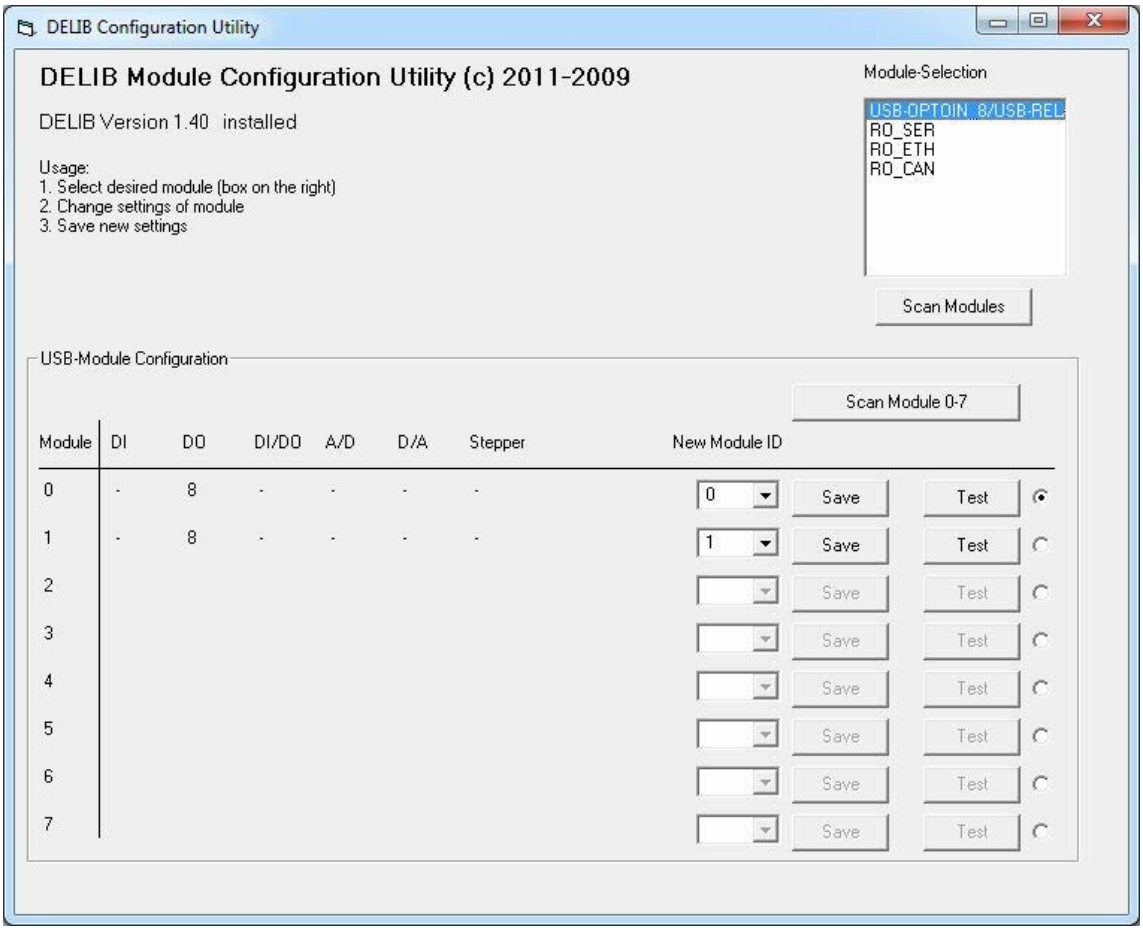


1. Wählen Sie das entsprechende USB-Modul aus.
2. Hier wird die neue Modul-NR gesetzt. Im Auslieferungszustand ist die NR bereits mit "0" vordefiniert. Ändern Sie die NR auf "1".
3. Durch einen Klick auf den Button **Save** wird die neue NR im Modul gespeichert.
4. Mit dem Button **Test** können Sie die Kommunikation mit dem Modul testen.

Schritt 2

Verbinden Sie nun das zweite Modul mit dem PC und aktualisieren Sie die Ansicht durch einen Klick auf den Button **Scan Module 0-7**.

Nun sollte folgende Ansicht erscheinen:



Nun sind beide Module konfiguriert und betriebsbereit.

Schritt 3

Nachfolgend, finden Sie Hinweise, was bei der Programmierung der beiden Module beachten werden muss.

Alle Module werden einheitlich mit dem Befehl DapiOpenModule geöffnet. Dieser Befehl ist wie folgt definiert:

ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

Ansprechen des USB-RELAIS-8 mit der NR 0

```
ulong handle;
handle = DapiOpenModule(USB_RELAIS_8, 0);      // öffnet das Modul USB-RELAIS-8 mit der
NR 0
```

Ansprechen des USB-RELAIS-8 mit der NR 1

```
ulong handle;
handle = DapiOpenModule(USB_RELAIS_8, 1);      // öffnet das Modul USB-RELAIS-8 mit der
NR 1
```

Hinweis:

Alle Module haben als Werkseinstellung die NR "0".

Wenn Sie mehrere identische USB-Module verwenden möchten, muss den Modulen nacheinander, eine eindeutige NR zugeordnet werden.

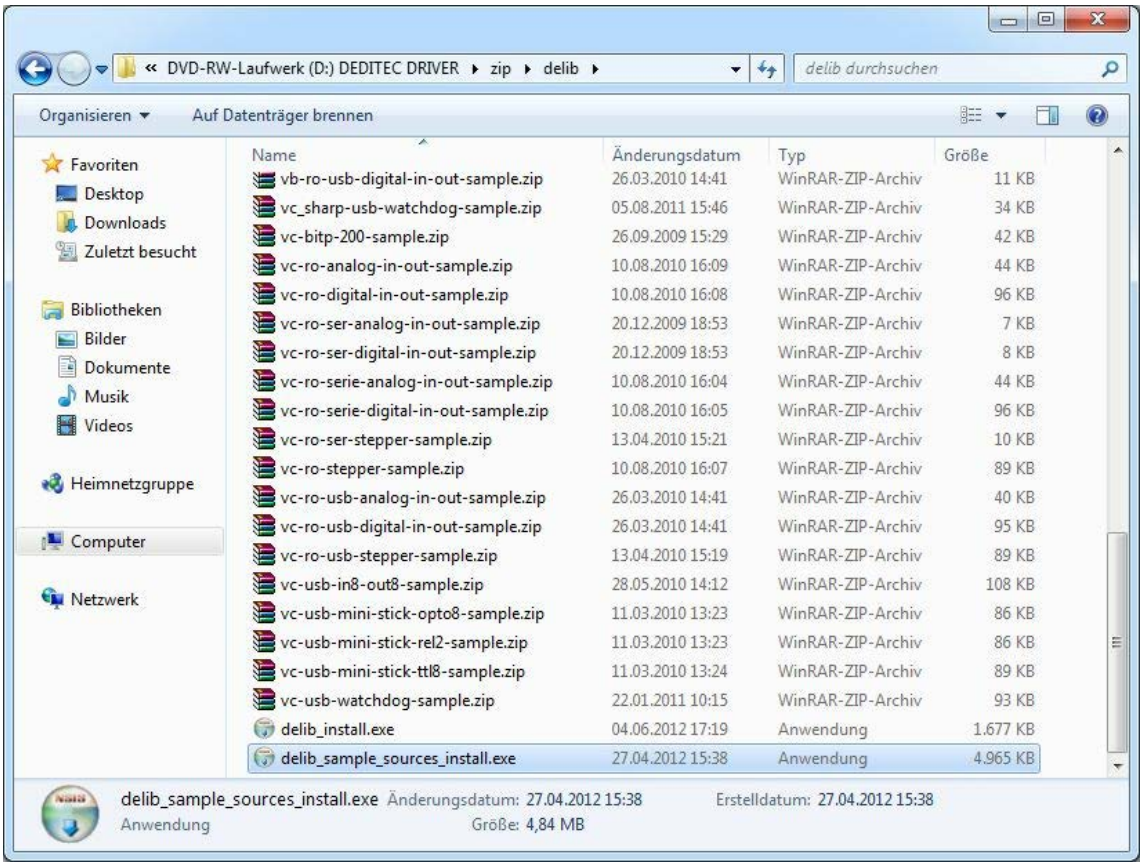
4.3. DELIB Sample Sources (Programmierbeispiele)

Die **DELIB Sample Sources** bieten Beispielprogramme inklusive Quellcode zu nahezu allen DEDITEC-Produkten.
Um den Schnelleinstieg mit unseren Modulen zu vereinfachen, finden Sie Quellcodes zu folgenden Programmiersprachen:

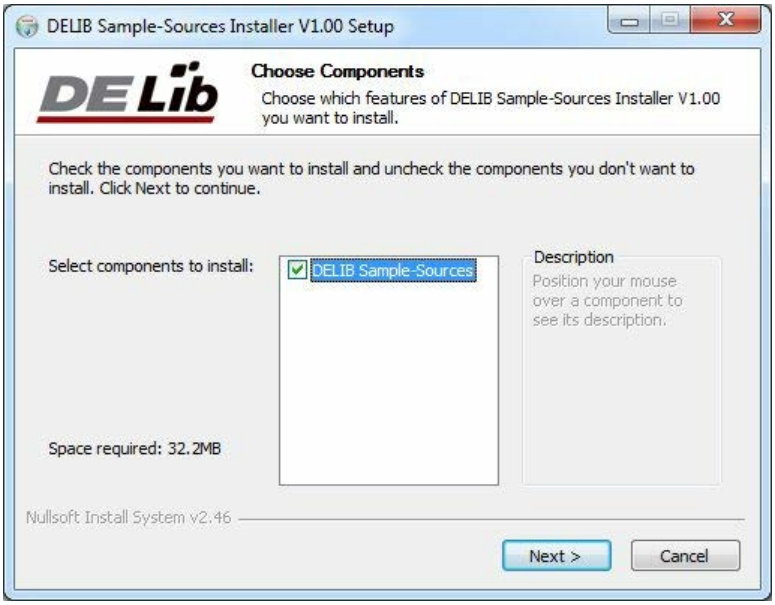
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office
- LabVIEW

4.3.1. Installation DELIB Sample Sources

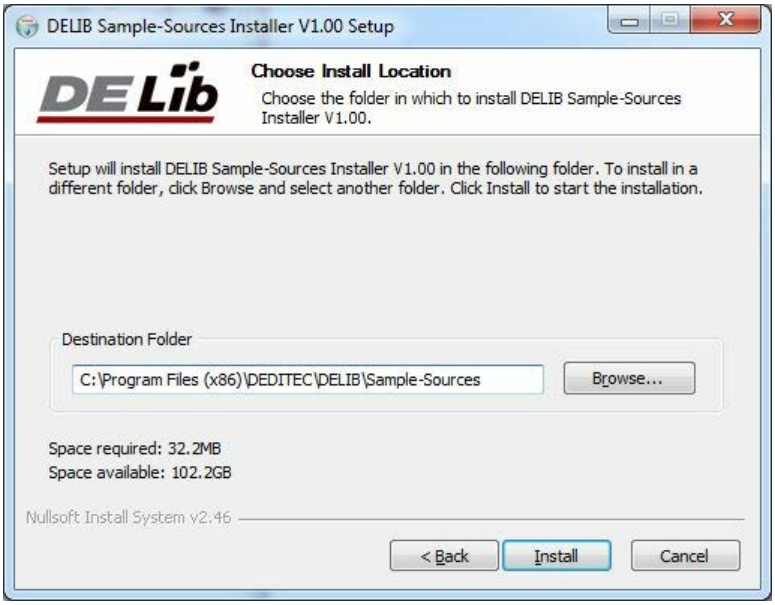
Legen Sie die DEDITEC Driver CD in das Laufwerk und starten Sie **delib_sample_sources_install.exe**.
Eine aktuelle Version der Sample Sources finden Sie auch im Internet unter
http://www.deditec.de/de/delib



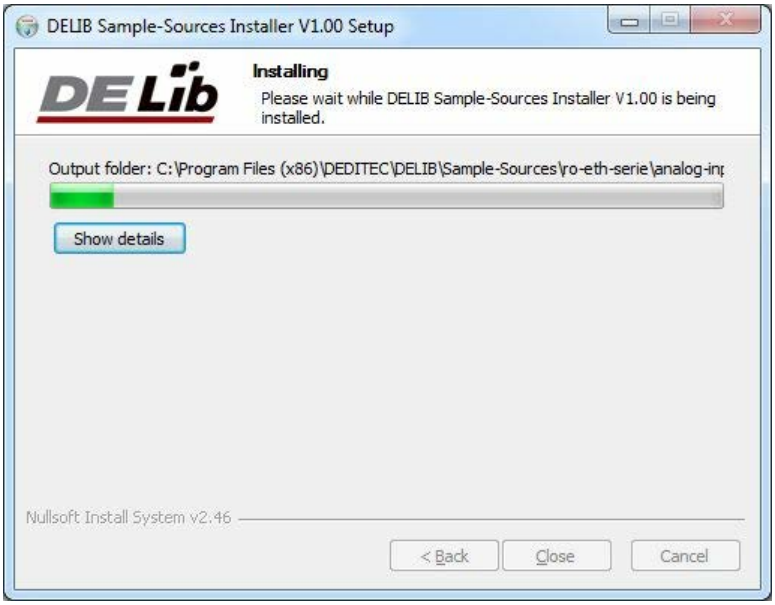
Startbild des DELIB Sample Sources Installer



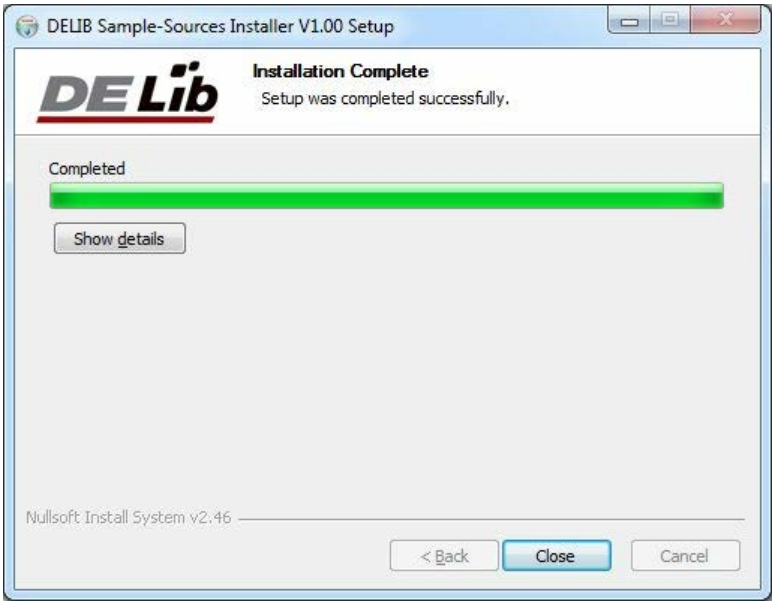
Drücken Sie **Next**.



Drücken Sie **Install**.



Die DELIB Sample Sources werden nun installiert.

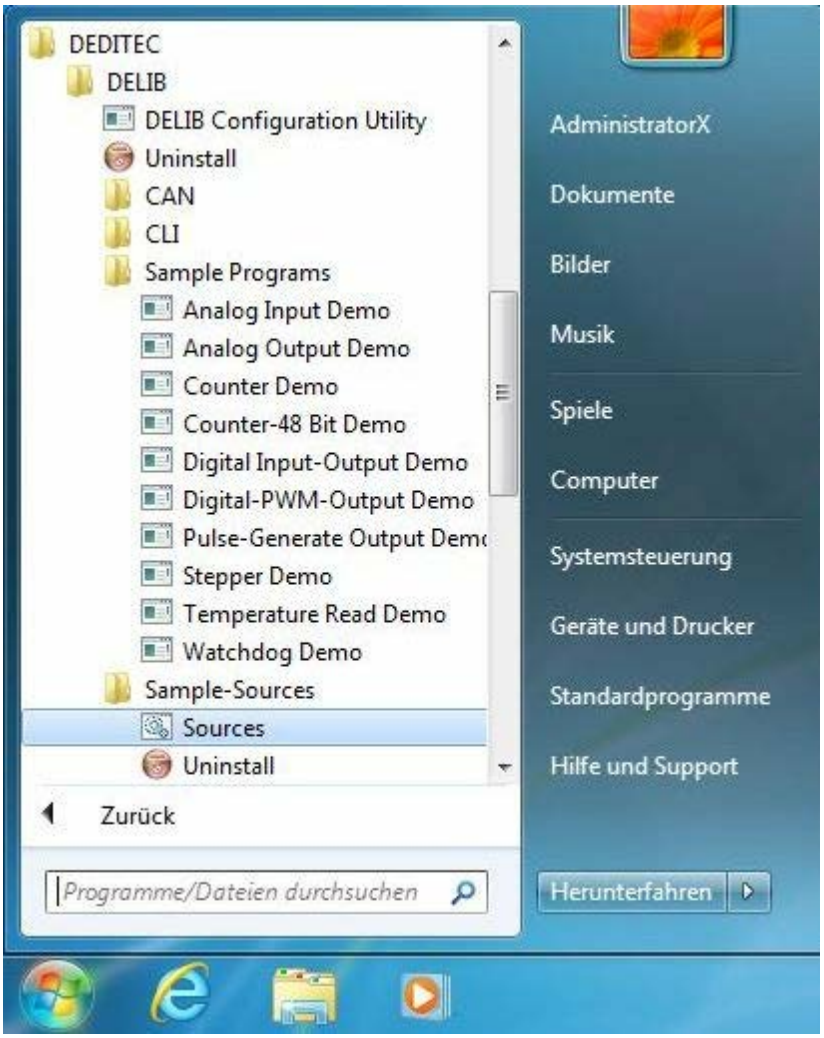


Die DELIB Sample Sources wurden erfolgreich installiert. Drücken Sie **Close** um die Installation zu beenden.

4.3.2. Benutzung der DELIB Sample Sources

Nach Installation der **DELIB Sample Sources** finden Sie diese unter

Start -> Programme -> DEDITEC -> DELIB -> Sample-Sources -> Sources

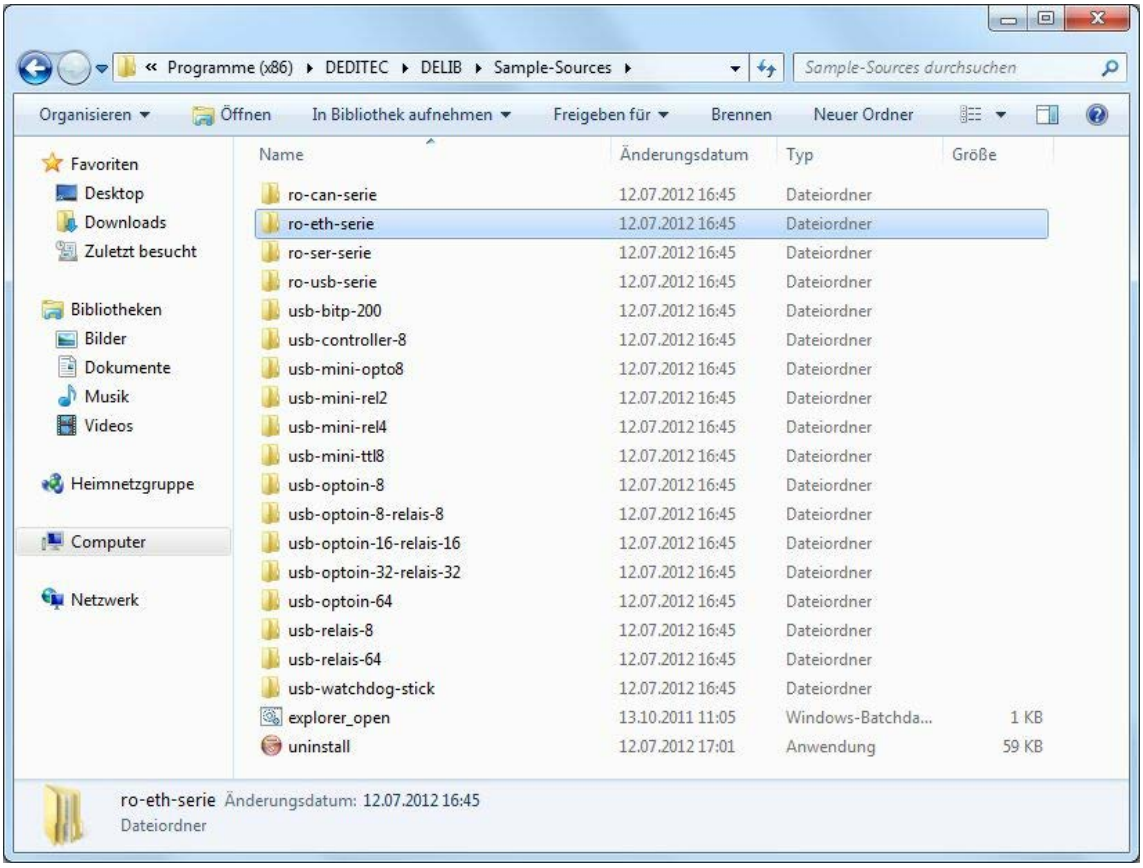


Nun öffnet sich der Windows-Explorer mit einer Übersicht aller Produkte für die ein Beispielprogramm verfügbar ist.

4.3.2.1. Schritt 1 - Produktauswahl

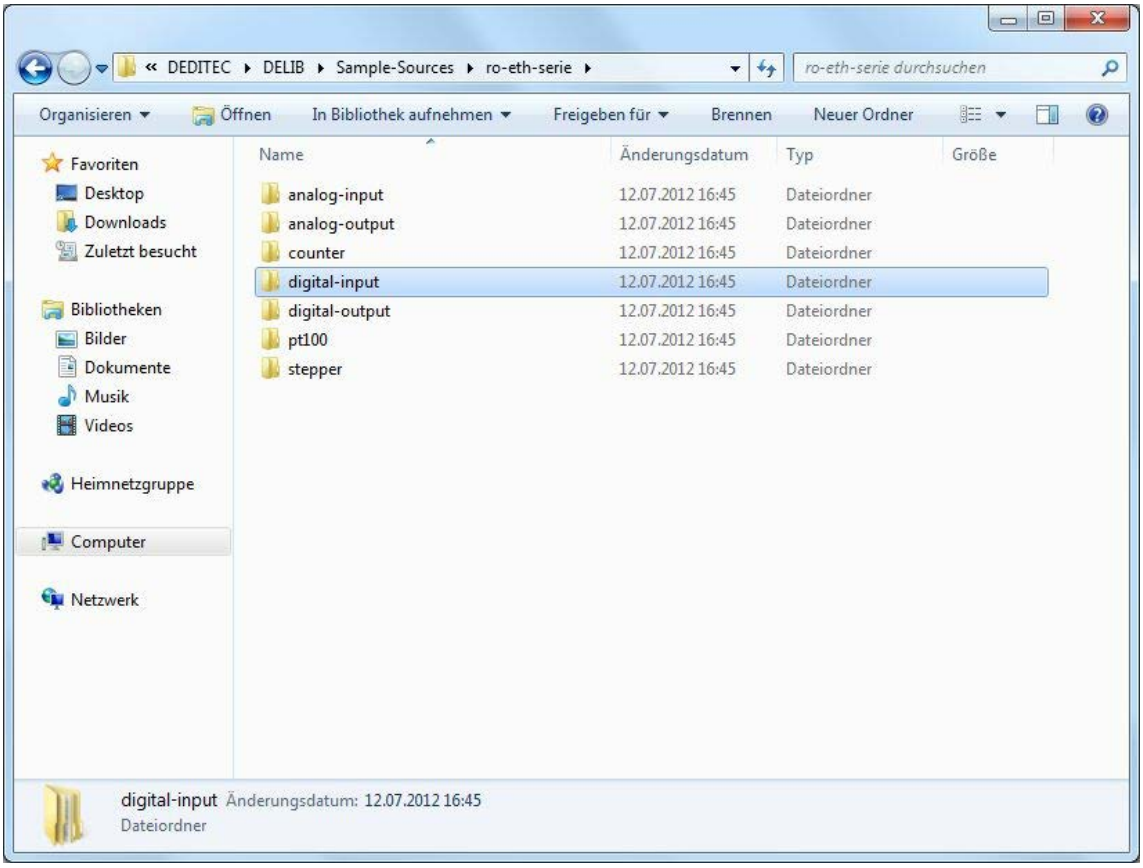
Sie benötigen beispielsweise eine Hilfestellung zur Programmierung der digitalen Eingänge eines RO-ETH-Moduls (z.B RO-ETH-O16) in der Programmiersprache Visual-C.

Da es sich um ein RO-ETH-Produkt handelt, wählen bzw. öffnen Sie den Ordner **ro-eth-serie**



4.3.2.2. Schritt 2 - Kategorieauswahl

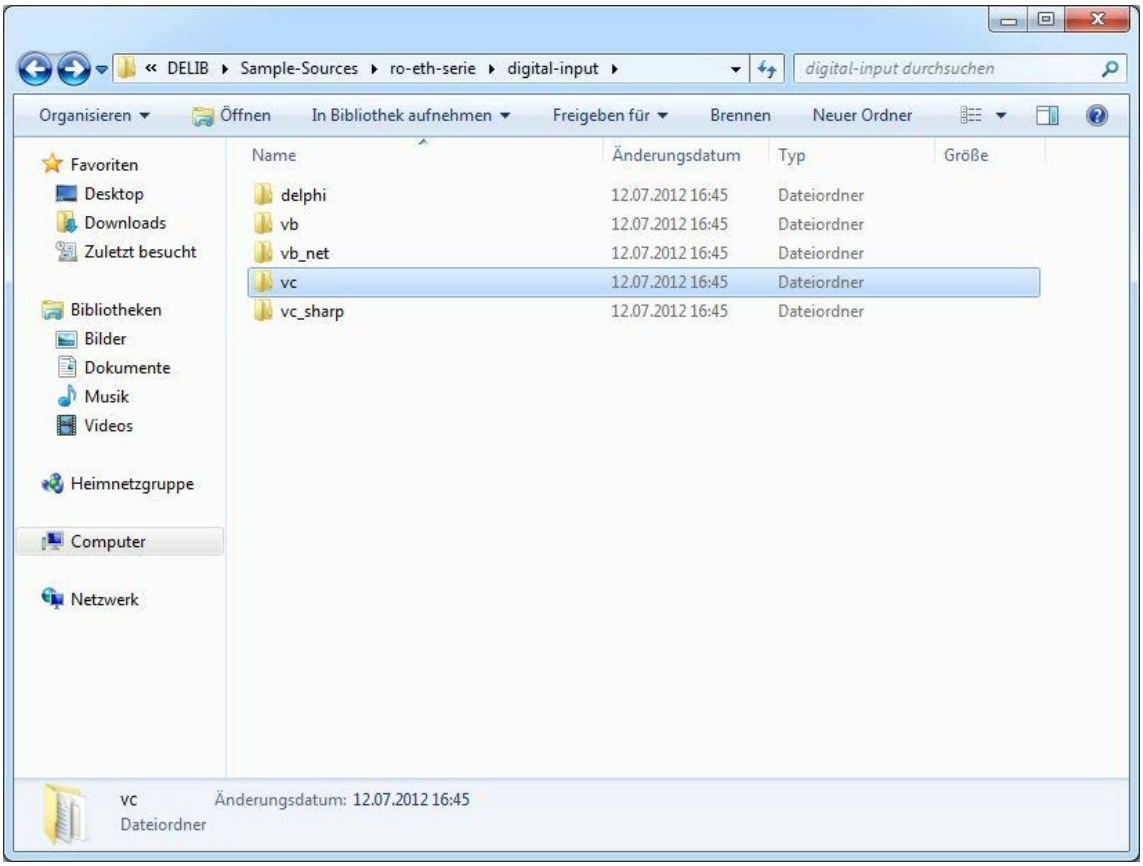
Im nächsten Schritt, finden Sie eine Übersicht der verfügbaren Kategorien für das ausgewählte Produkt.
Da wir uns in diesem Beispiel auf die digitalen Eingänge konzentrieren, wählen Sie die Kategorie **digital-input**



4.3.2.3. Schritt 3 - Programmiersprachenauswahl

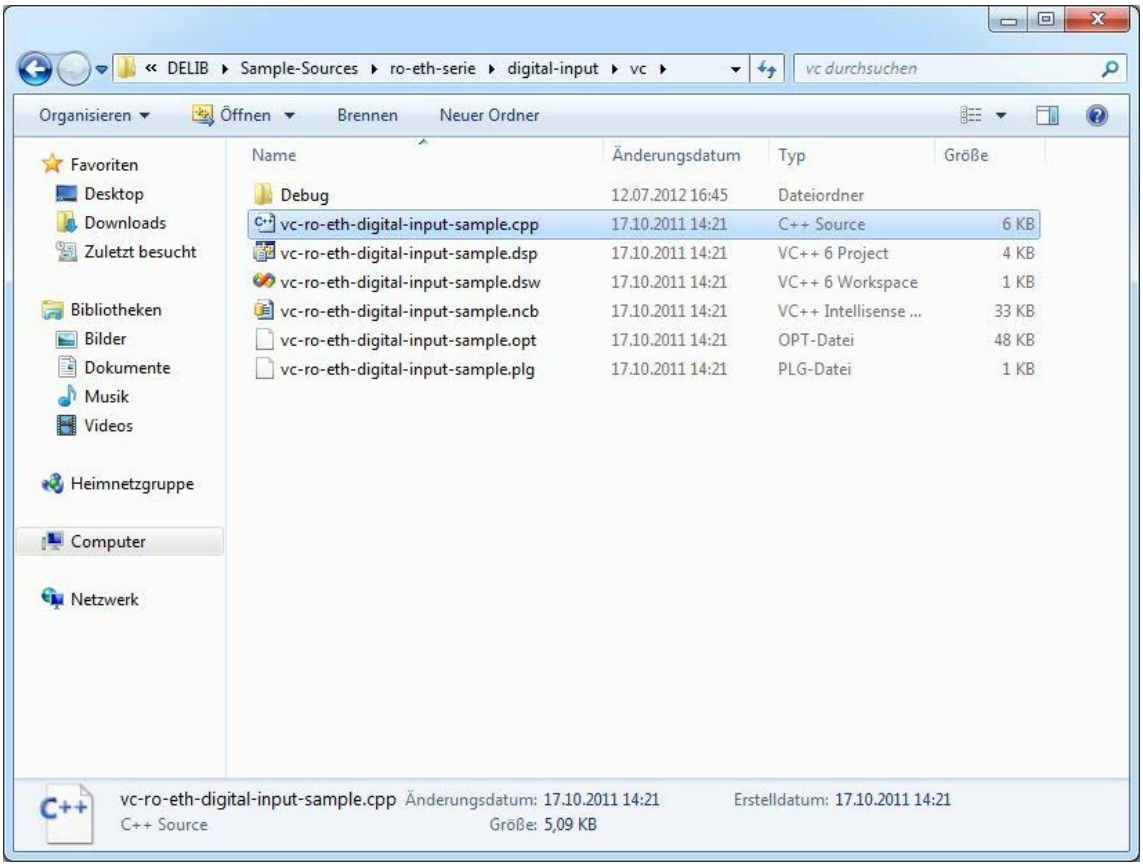
In diesem Schritt sehen Sie alle verfügbaren Programmierbeispiele der gewählten Kategore, sortiert nach Programmiersprachen.

Da wir uns in diesem Beispiel auf die Programmiersprache Visual-C konzentrieren, öffnen Sie den Ordner **vc**.



4.3.2.4. Schritt 4 - Quellcode

Nach Auswahl der Programmiersprache erhalten Sie folgende Übersicht:



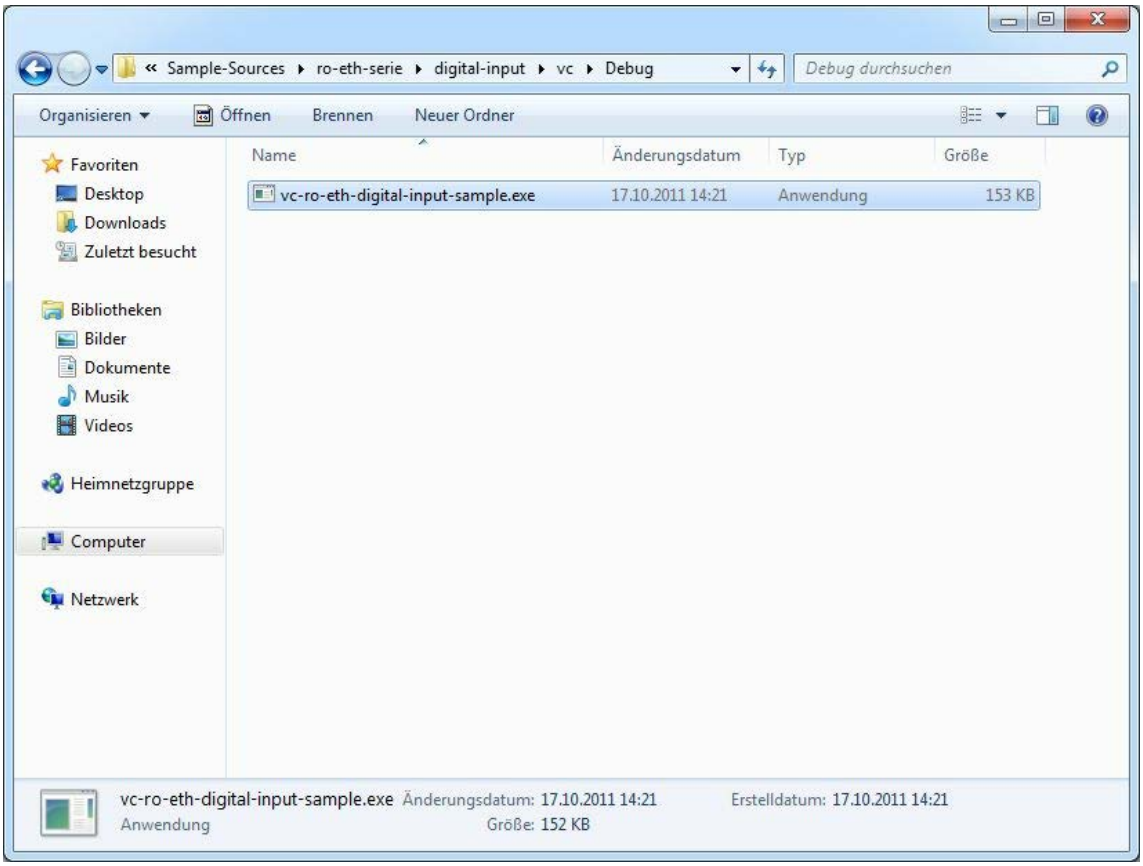
Den Quellcode des Beispielprogramm (in diesem Fall **.cpp-Datei**) können Sie nun mit einem beliebigen Text-Editor öffnen.

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);
```

Zusätzlich finden Sie im Ordner **debug**, ein bereits kompliliertes und ausführbares Programm zu diesem Projekt.

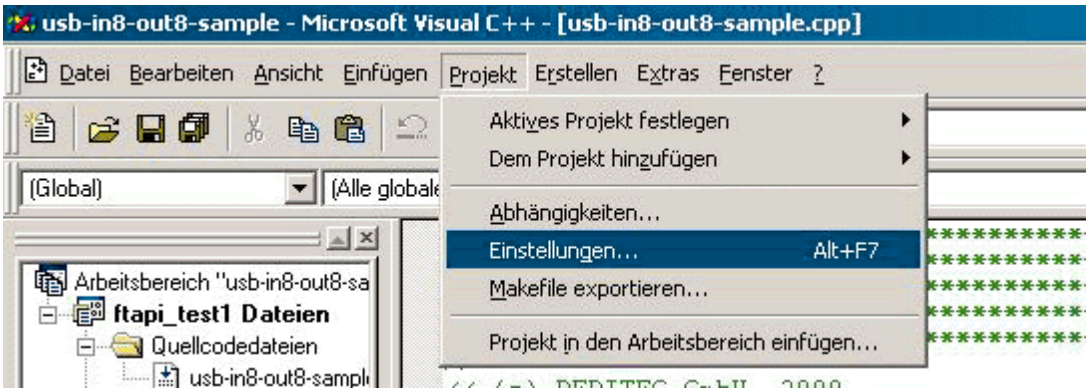


4.4. Einbinden der DELIB in Programmiersprachen

4.4.1. Einbinden der DELIB in Visual-C/C++

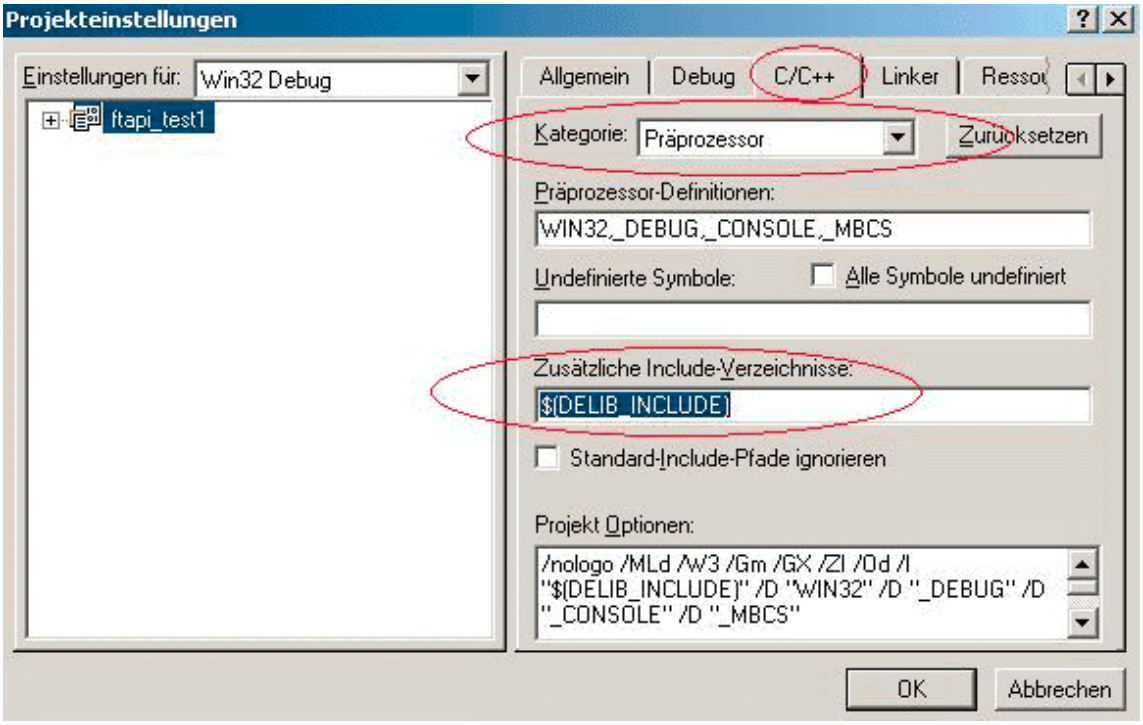
Zur Erleichterung für Verweise auf das DELIB-Include und das DELIB-Lib Verzeichnis werden bei installation der DELIB Umgebungsvariablen definiert.
DELIB_LIB = C:\Programme\DEDITEC\DELIB\lib
DELIB_INCLUDE = C:\Programme\DEDITEC\DELIB\include

Diese werden im Folgenden in den Projekteinstellungen des Compilers eingetragen.
Visual-C/C++ Starten und im Menue "Projekt -> Einstellungen" öffnen.



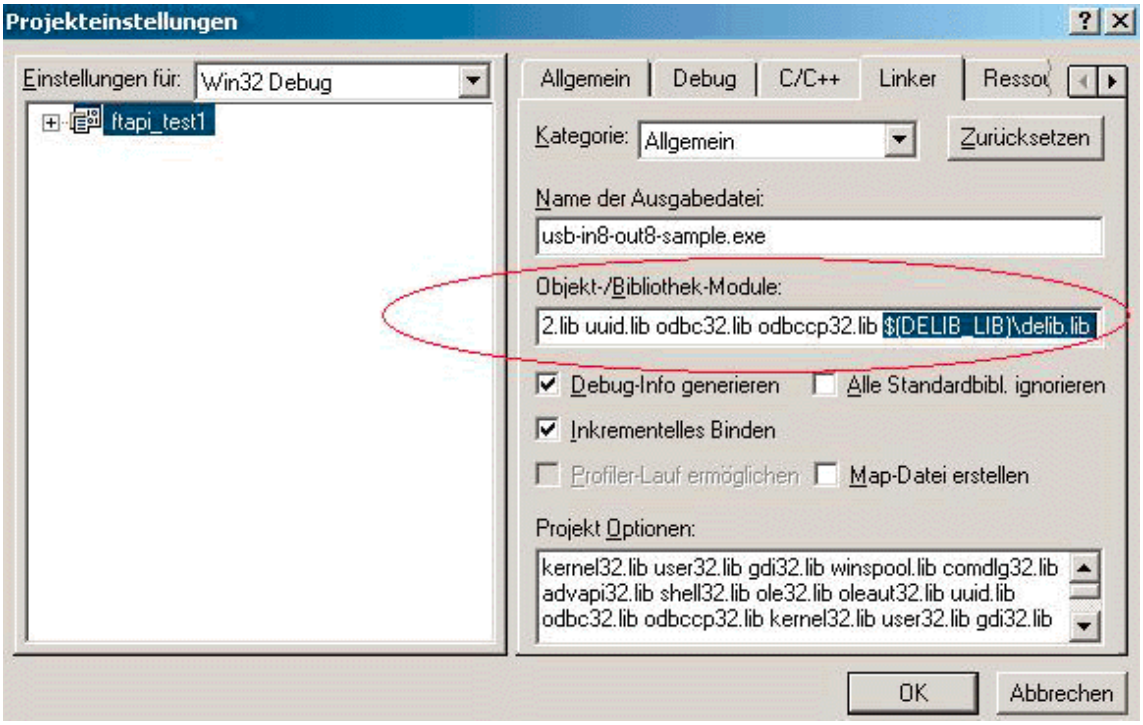
DELIB.H Eintrag in den Visual-C/C++ Projekt Einstellungen

Unter dem Reiter "C/C++" die "Kategorie" Präprozessor auswählen und unter "Zusätzliche Include Verzeichnisse" "$(DELIB_INCLUDE)" eintragen.



DELIB.LIB Eintrag in den Visual-C/C++ Projekt Einstellungen

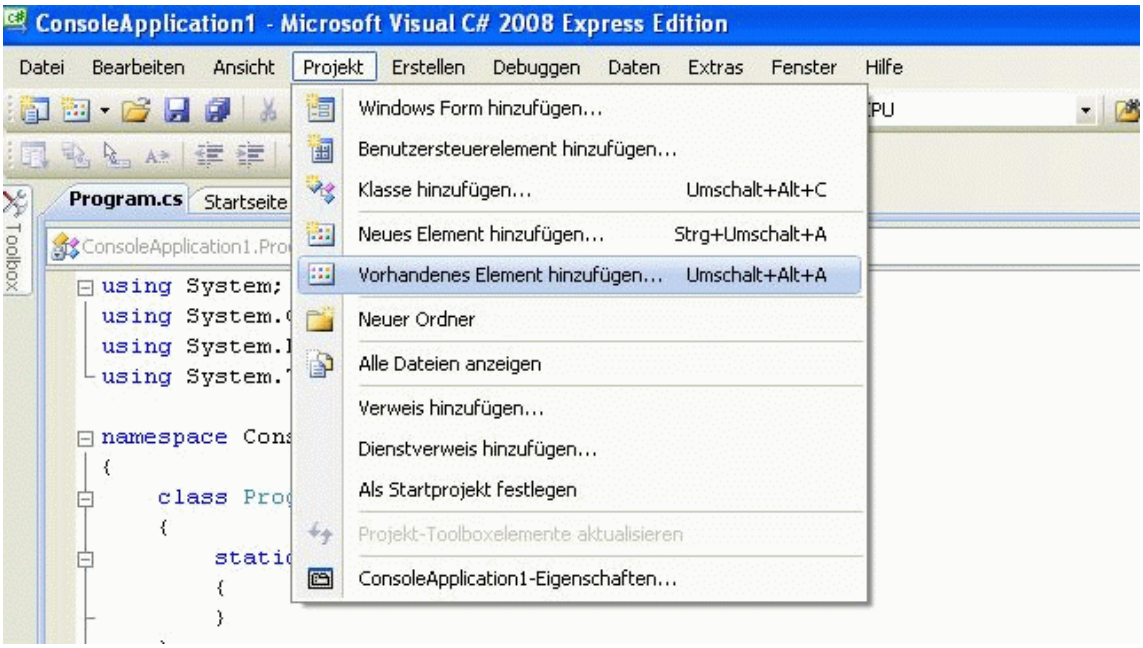
Unter dem Reiter "Linker" bei "Objekt-/Bibliothek-Module" die vorhandene Zeile mit der Endung "$(DELIB_LIB)\delib.lib" erweitern.



4.4.2. Einbinden der DELIB in Visual-C#

Die benötigte Datei für Visual-C# befindet sich im Verzeichnis **C:\Programme\DEDITEC\DELIB\include**.

Visual-C# starten und über das Menue "Projekt -> Vorhandenes Element hinzufügen" im Verzeichnis C:\Programme\DEDITEC\DELIB\include\ die Datei **delib.cs** zum Importieren öffnen.



Folgenden Verweis in Ihrem Programm hinzufügen:

```
using DeLib;
```

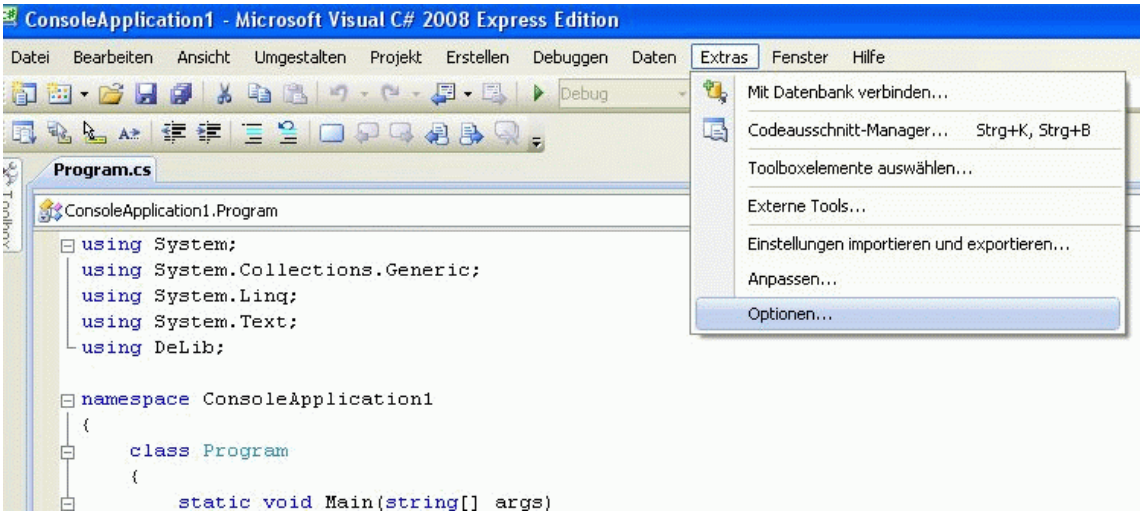
4.4.3. Einbinden der DELIB in Visual-C# unter Windows 64 Bit

Beispiel für die Einbindung der DELIB Treiberbibliothek unter Win 64 Bit

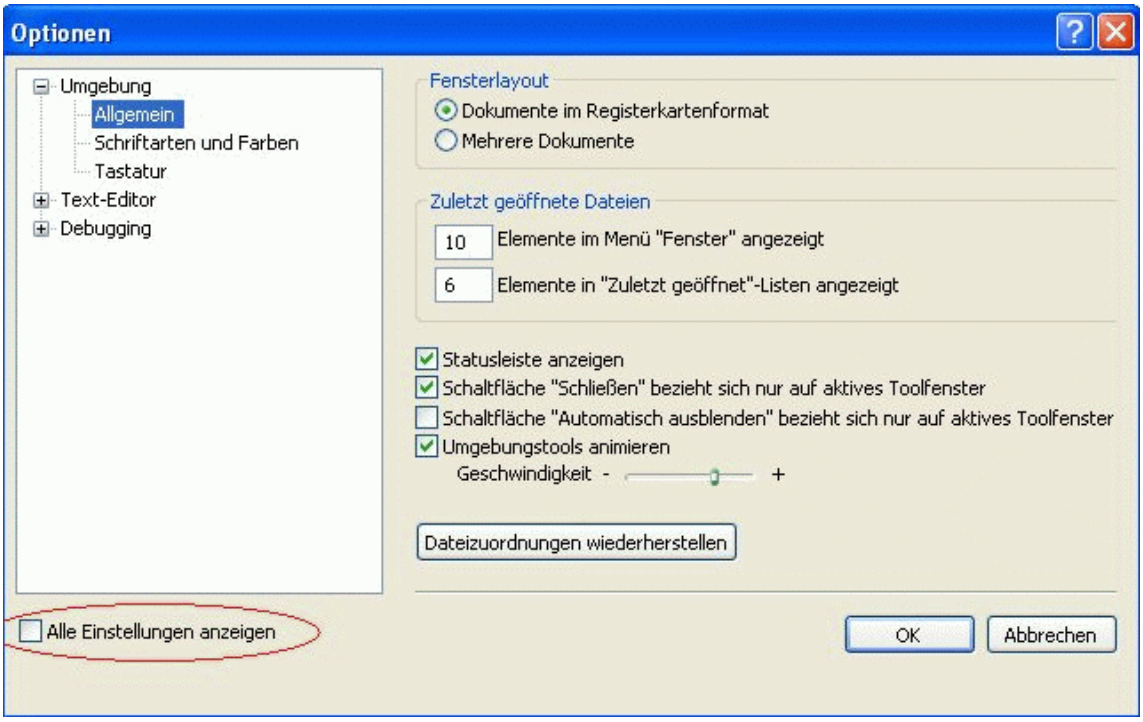
In den folgenden Abschnitten wird Schritt für Schritt die Einbindung der DELIB Treiberbibliothek in Visual-C# unter Windows 64 Bit erklärt.

Erweiterte Einstellungen sichtbar machen

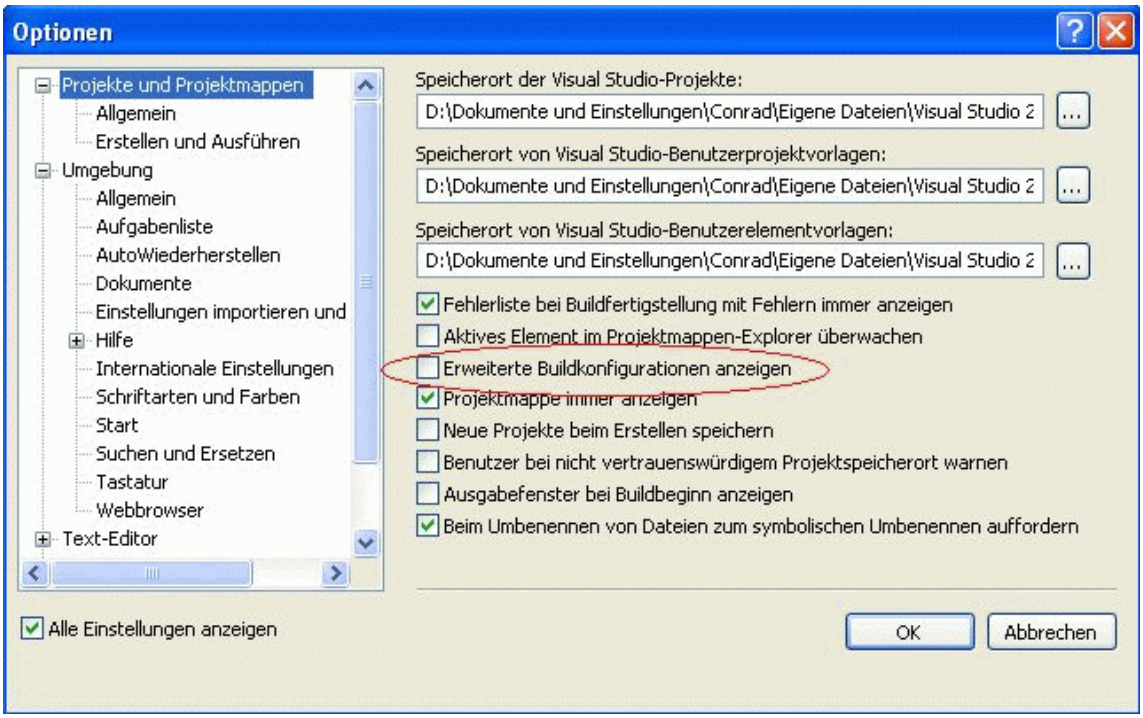
Über das Menue "Extras -> Optionen" die Optionen öffnen.



Den Haken bei "Alle Einstellungen anzeigen" setzen.



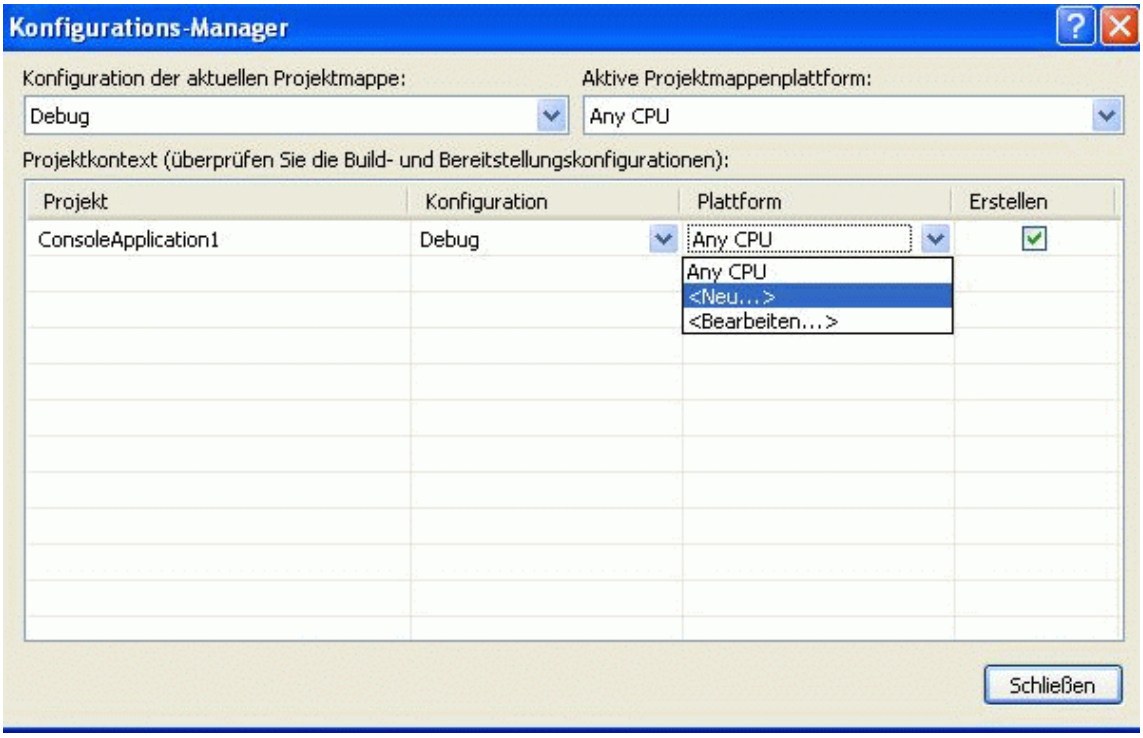
Nun unter dem Punkt "Projekte und Projektmappen" den Haken bei "Erweiterte Buildkonfigurationen anzeigen" setzen.



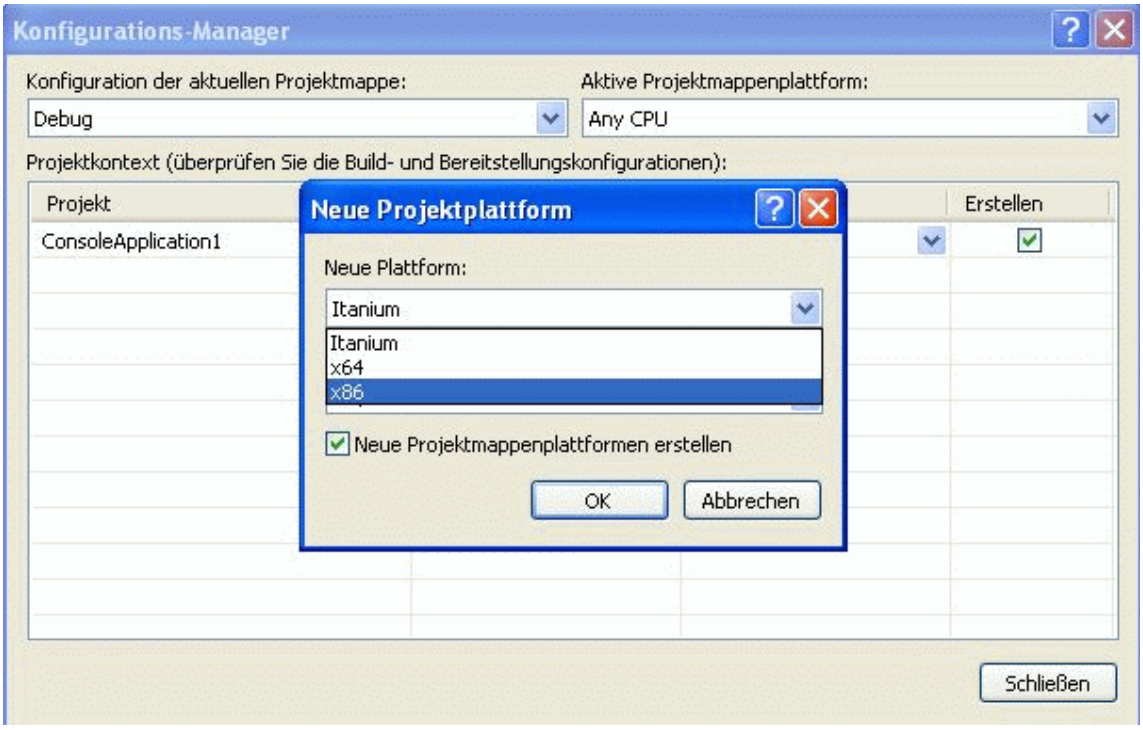
Der Konfigurations-Manager

Den Konfigurations-Manager durch Klicken auf "Any CPU -> Konfigurations-Manager..." starten.

Nun im Konfigurations-Manager unter "Plattform" "Any CPU -> Neu..." auswählen.



Nun unter "Neue Plattform" "x86" auswählen. Das Fenster mit "Ok" schließen und den Konfigurations-Manager über "Schließen" verlassen.



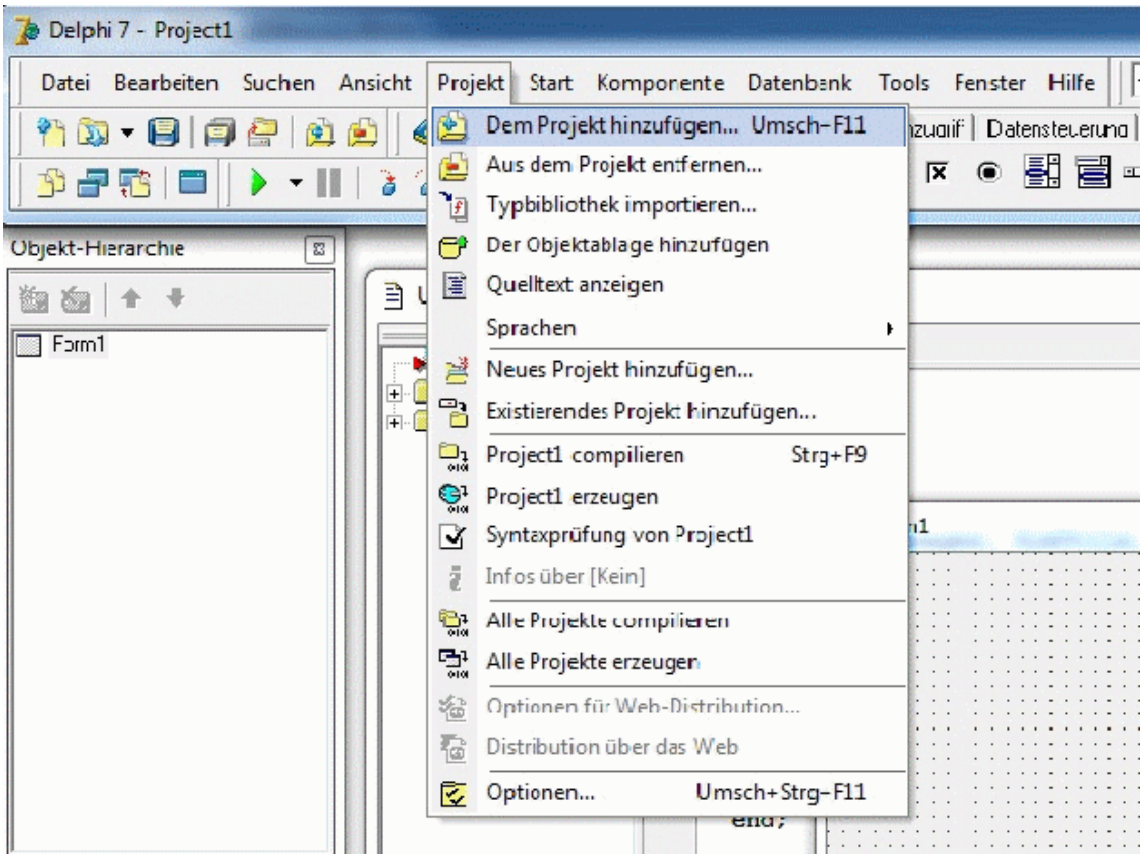
Debugging starten

Das Debugging kann nun ganz normal durch betätigen des "Start-Buttons" gestartet werden. Beachten Sie, dass "x86" bei der Plattform Konfiguration ausgewählt ist.

4.4.4. Einbinden der DELIB in Delphi

Die benötigte Datei für Delphi befindet sich im Verzeichnis **C:\Programme\DEDITEC\DELIB\include**.

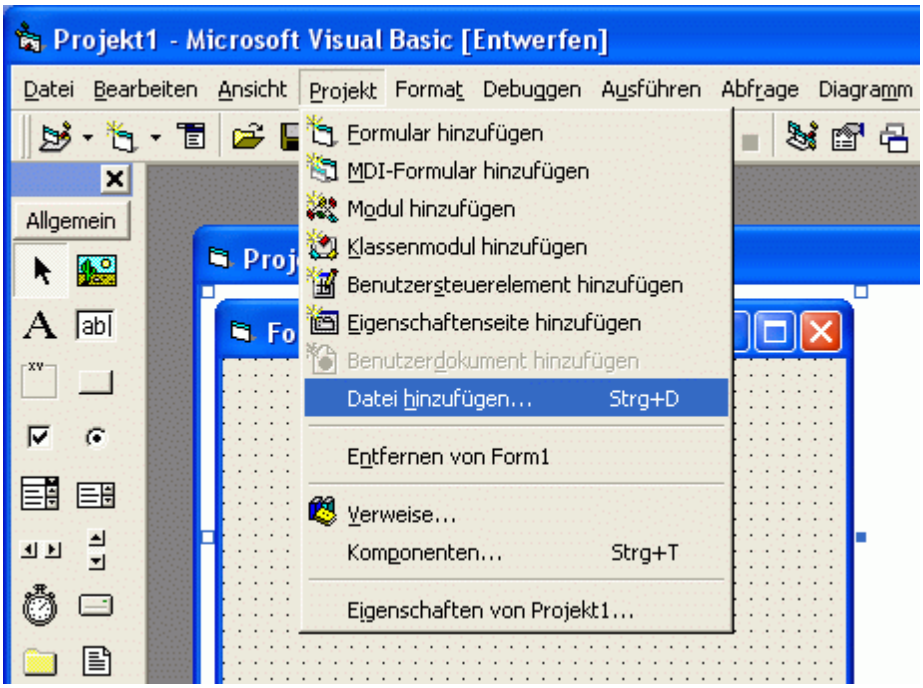
Delphi starten und über das Menue "Projekt -> dem Projekt hinzufügen" im Verzeichnis C:\Programme\DEDITEC\DELIB\include\ die Datei **delib.pas** zum Importieren öffnen.



4.4.5. Einbinden der DELIB in Visual-Basic (VB)

Die benötigte Datei für Visual-Basic befindet sich im Verzeichnis **C:\Programme\DEDITEC\DELIB\include**.

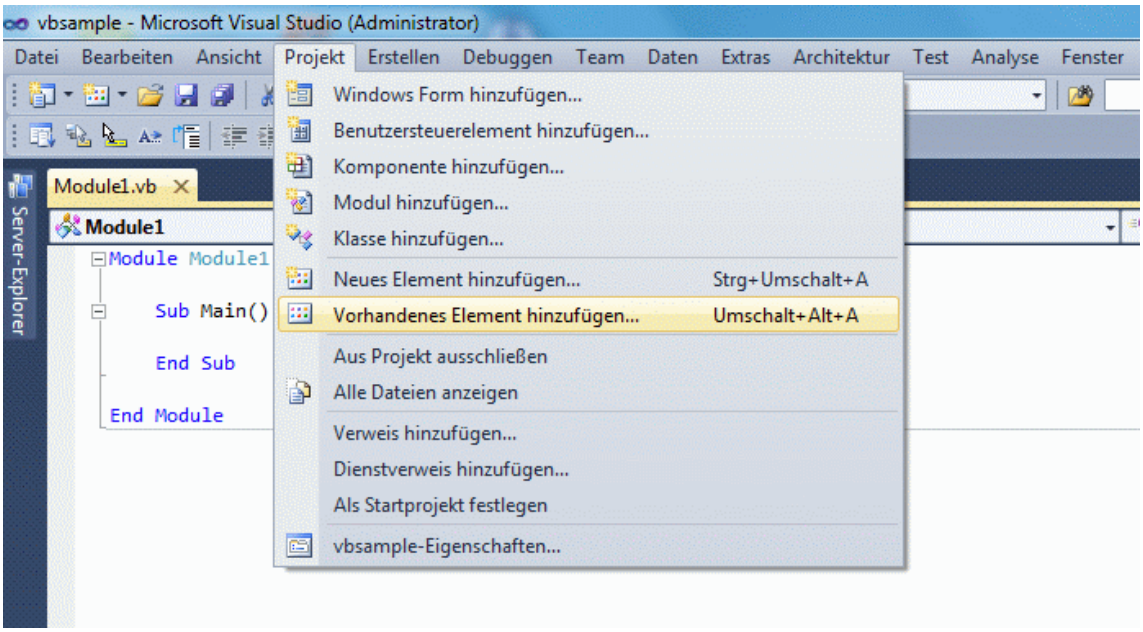
Visual Basic starten und über das Menue "Projekt -> Datei hinzufügen..." im Verzeichnis C:\Programme\DEDITEC\DELIB\include\ die Datei **delib.bas** zum Importieren öffnen.



4.4.6. Einbinden der DELIB in Visual-Basic.NET (VB.NET)

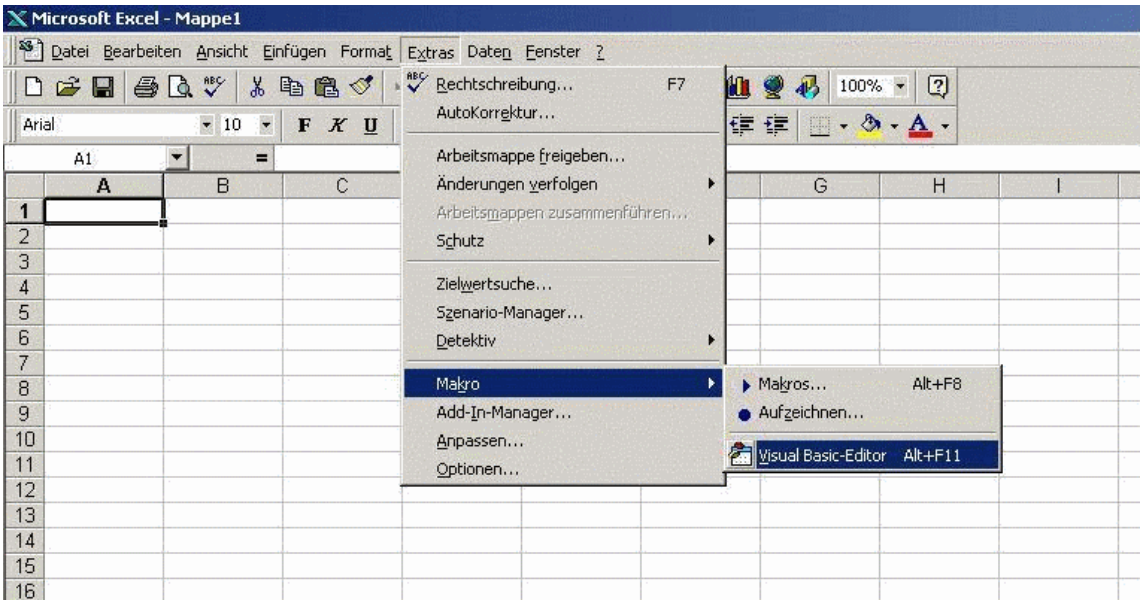
Die benötigte Datei für VB.NET befindet sich im Verzeichnis **C:\Programme\DEDITEC\DELIB\include**.

VB.NET starten und über das Menue "Projekt -> Vorhandenes Element hinzufügen" im Verzeichnis C:\Programme\DEDITEC\DELIB\include\ die Datei **delib.vb** zum Importieren öffnen.



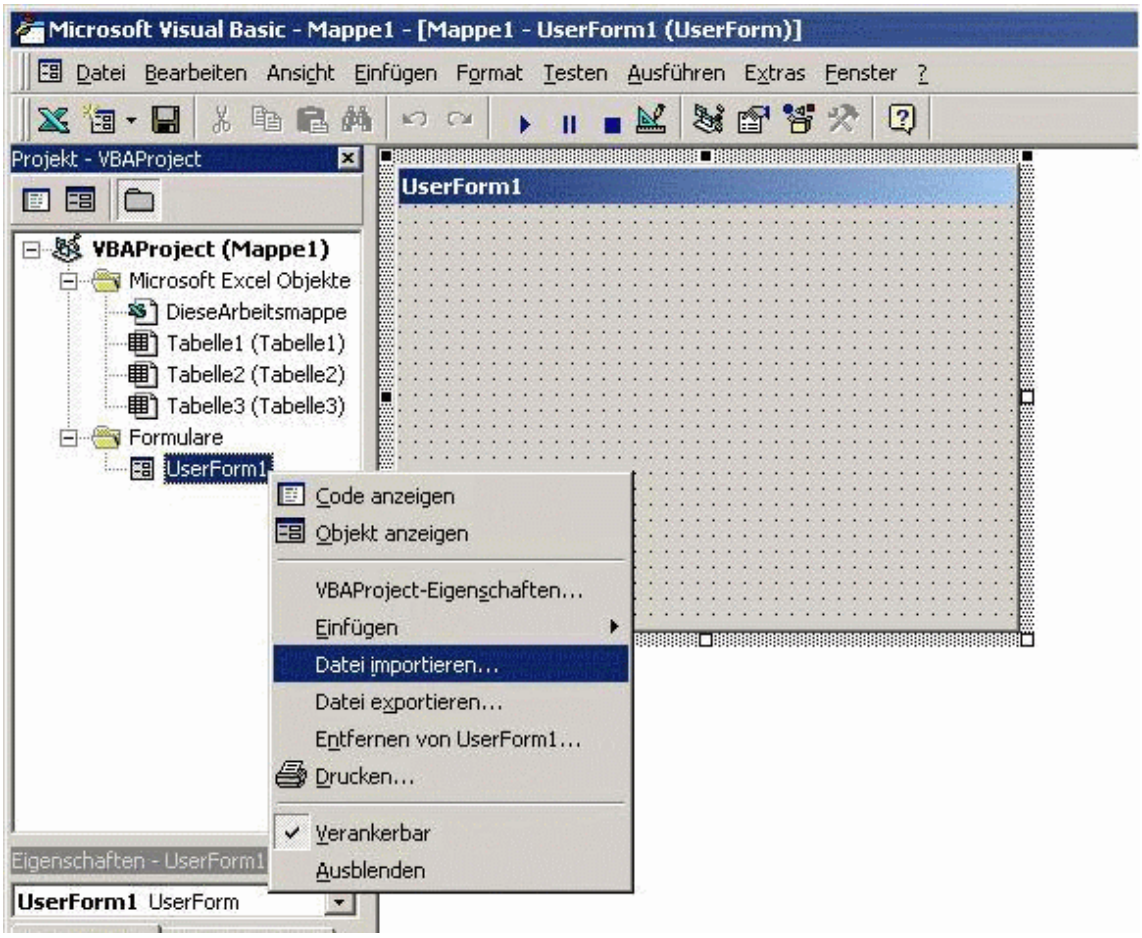
4.4.7. Einbinden der DELIB in MS-Office (VBA)

Die benötigte Datei für VBA befindet sich im Verzeichnis
C:\Programme\DEDITEC\DELIB\include.
Microsoft Excel starten und über das Menue "Extras -> Makro -> Visual Basic Editor" öffnen.



Erstellen der UserForm

Ein neues Arbeitsblatt (UserForm) über das Menue "Einfügen -> UserForm" erstellen. Oben links im Projektmanager einen Rechtsklick auf "UserForm -> Datei importieren". Im Verzeichnis C:\Programme\DEDITEC\DELIB\include die Datei **delib.bas** zum importieren öffnen.



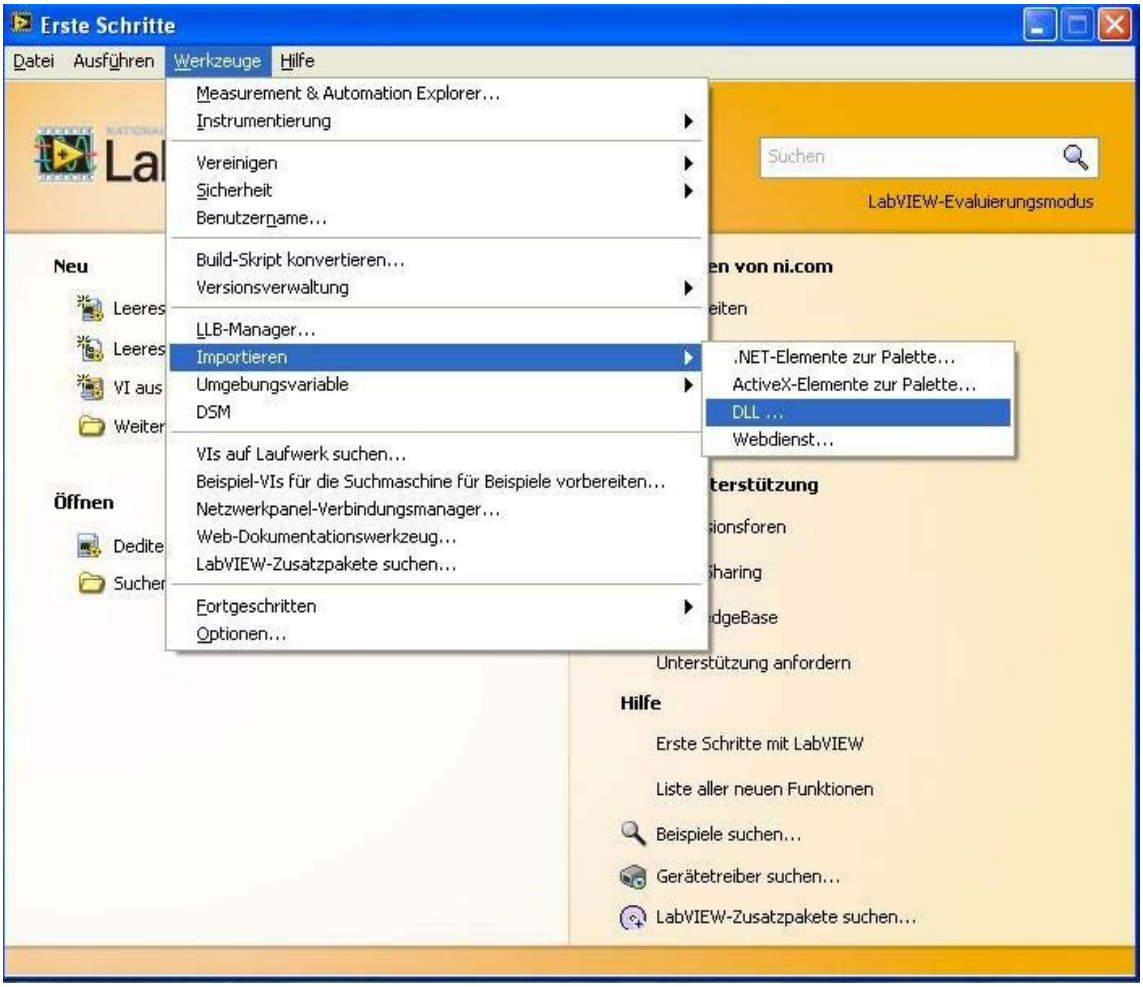
4.4.8. Einbinden der DELIB in LabVIEW

4.4.8.1. Einbinden der DELIB in LabVIEW

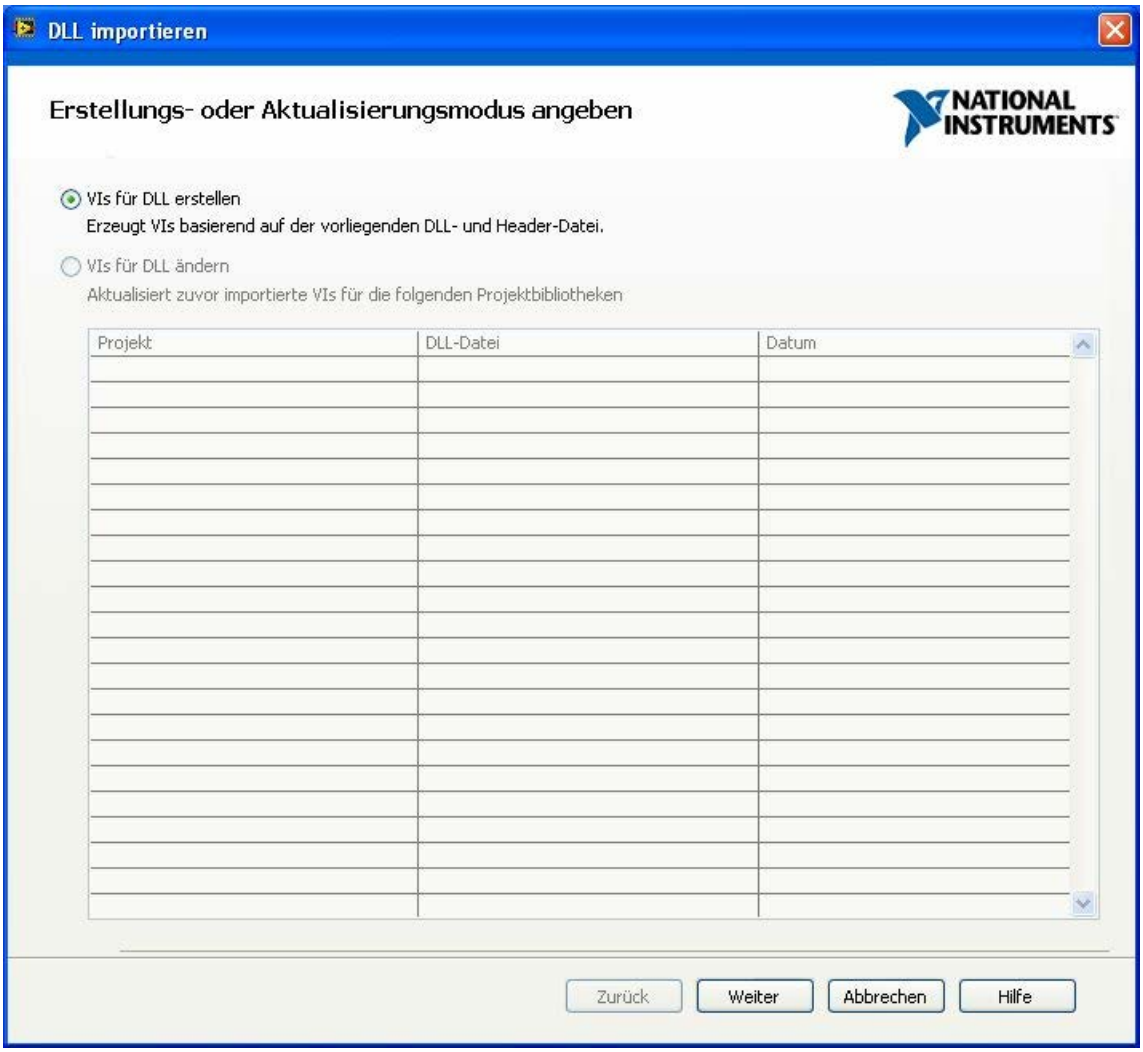
Das LabVIEW-Beispielprogramm "Deditec_Modul_Control.vi" ist keine EXE-Datei und benötigt deshalb zur Ausführung die LabVIEW Entwicklungsumgebung

Beschreibung der Einbindung der "delib.dll" in LabVIEW Version 11

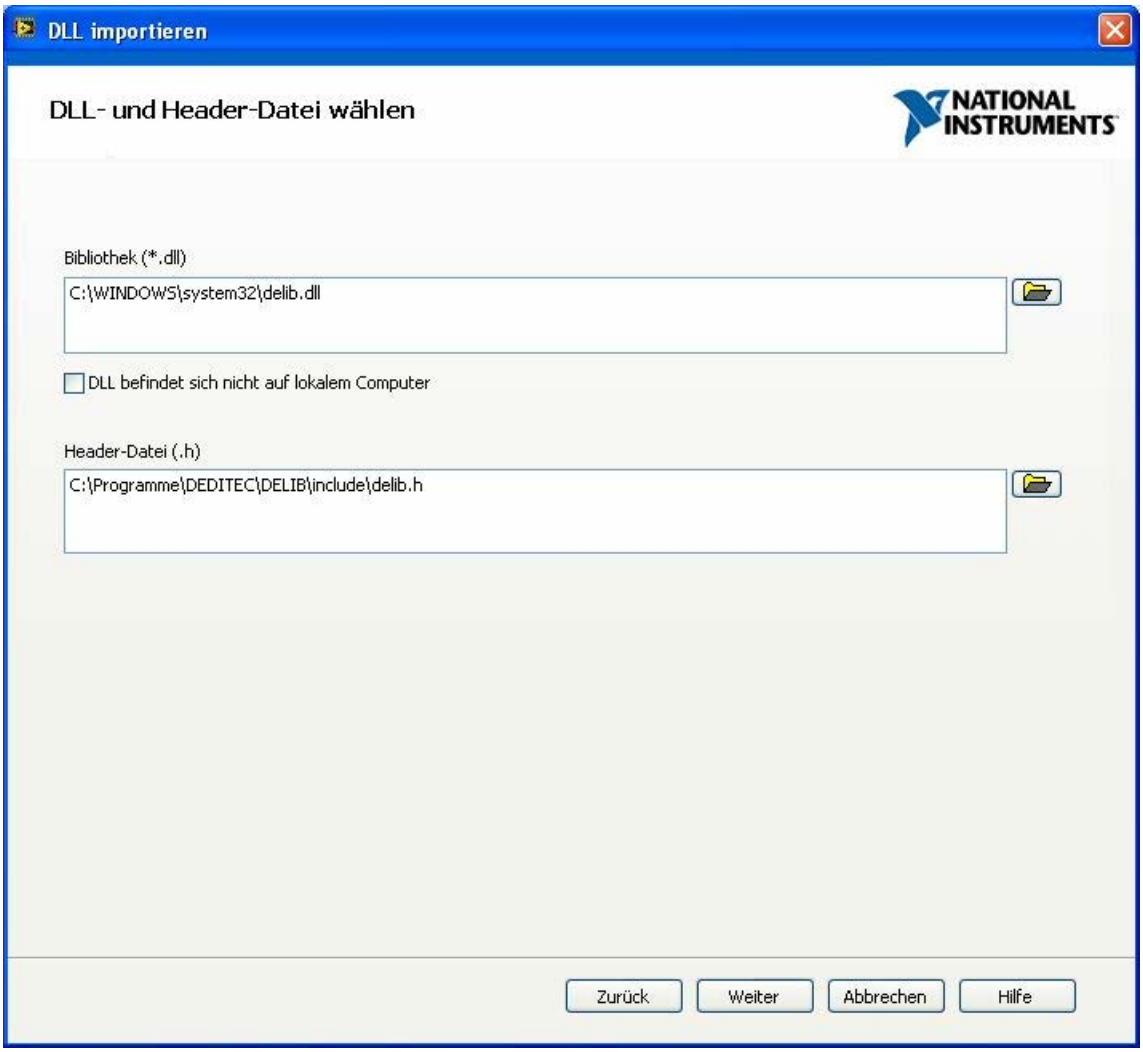
- Die benötigten Dateien für LabVIEW befinden sich im Verzeichnis **C:\Windows\System32\delib.dll** und in **C:\Programme\DEDITEC\DELIB\include\delib.h**
- LabVIEW starten und folgende Option auswählen "Werkzeuge -> Importieren -> DLL ..."



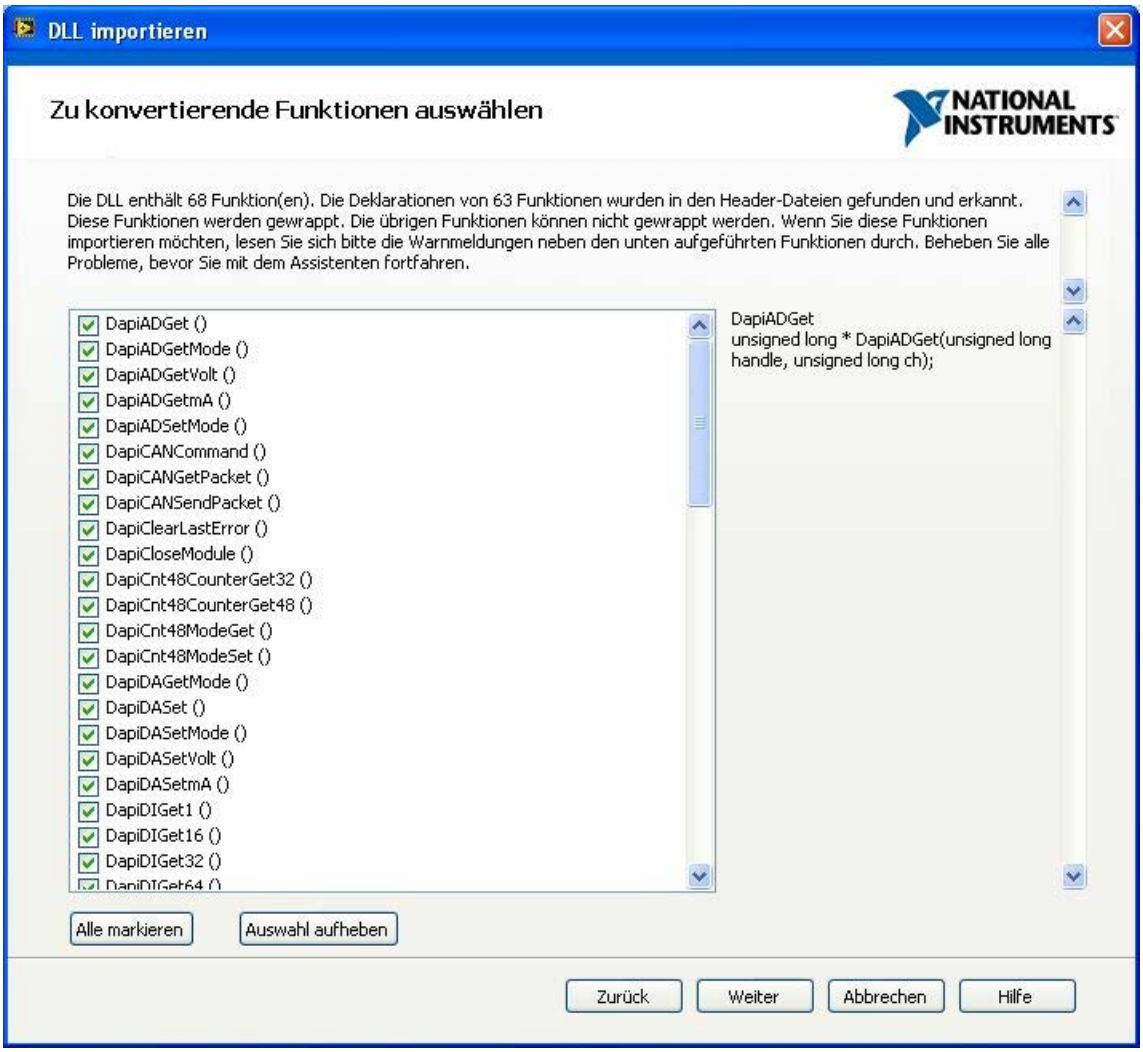
- Wählen sie den Punkt "VIs für DLL erstellen" und drücken Sie auf "Weiter"



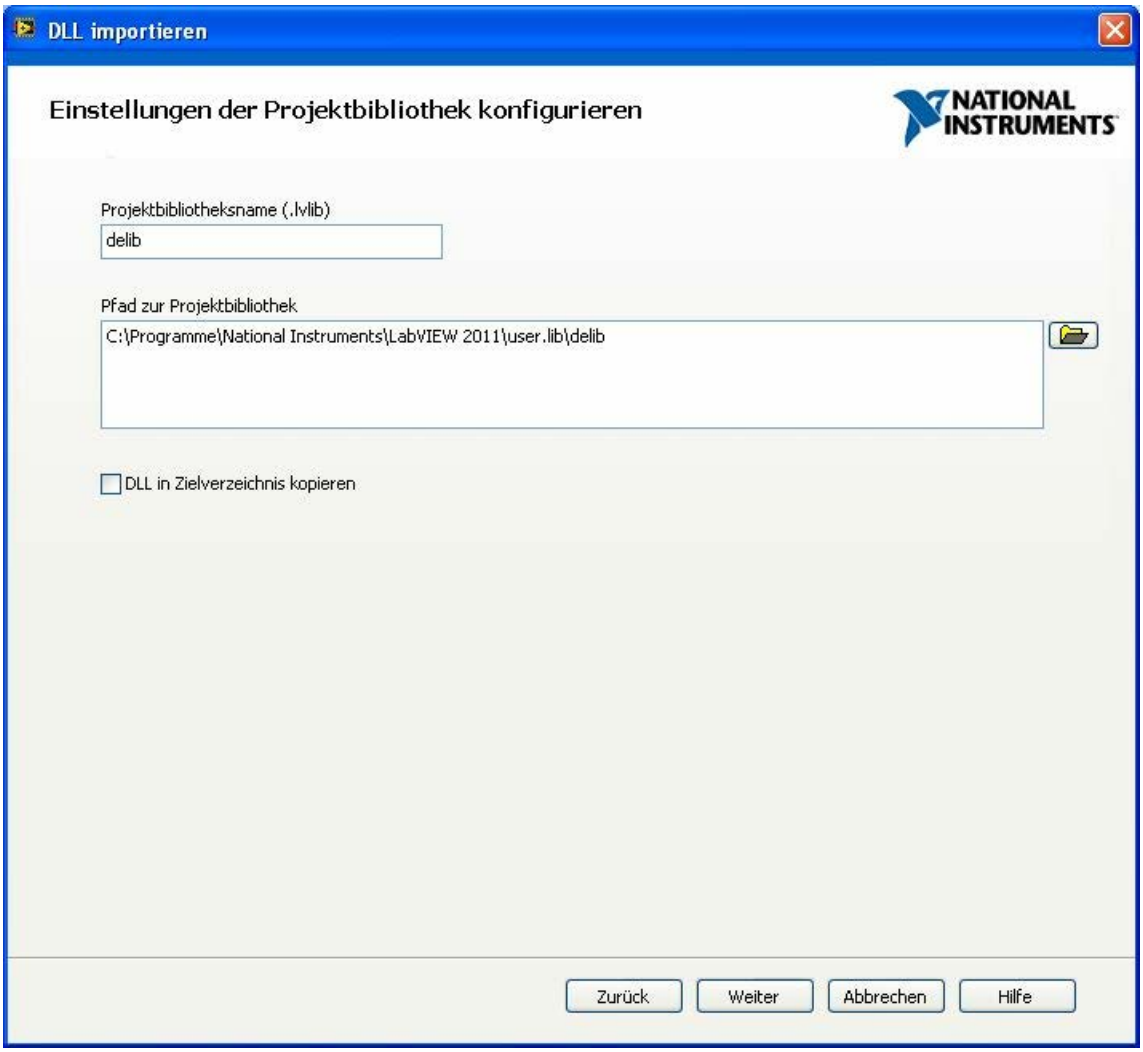
- Im nächsten Fenster über die Browse-Buttons den Speicherort der .DLL und der .H Datei angeben und mit "Weiter" fortfahren.



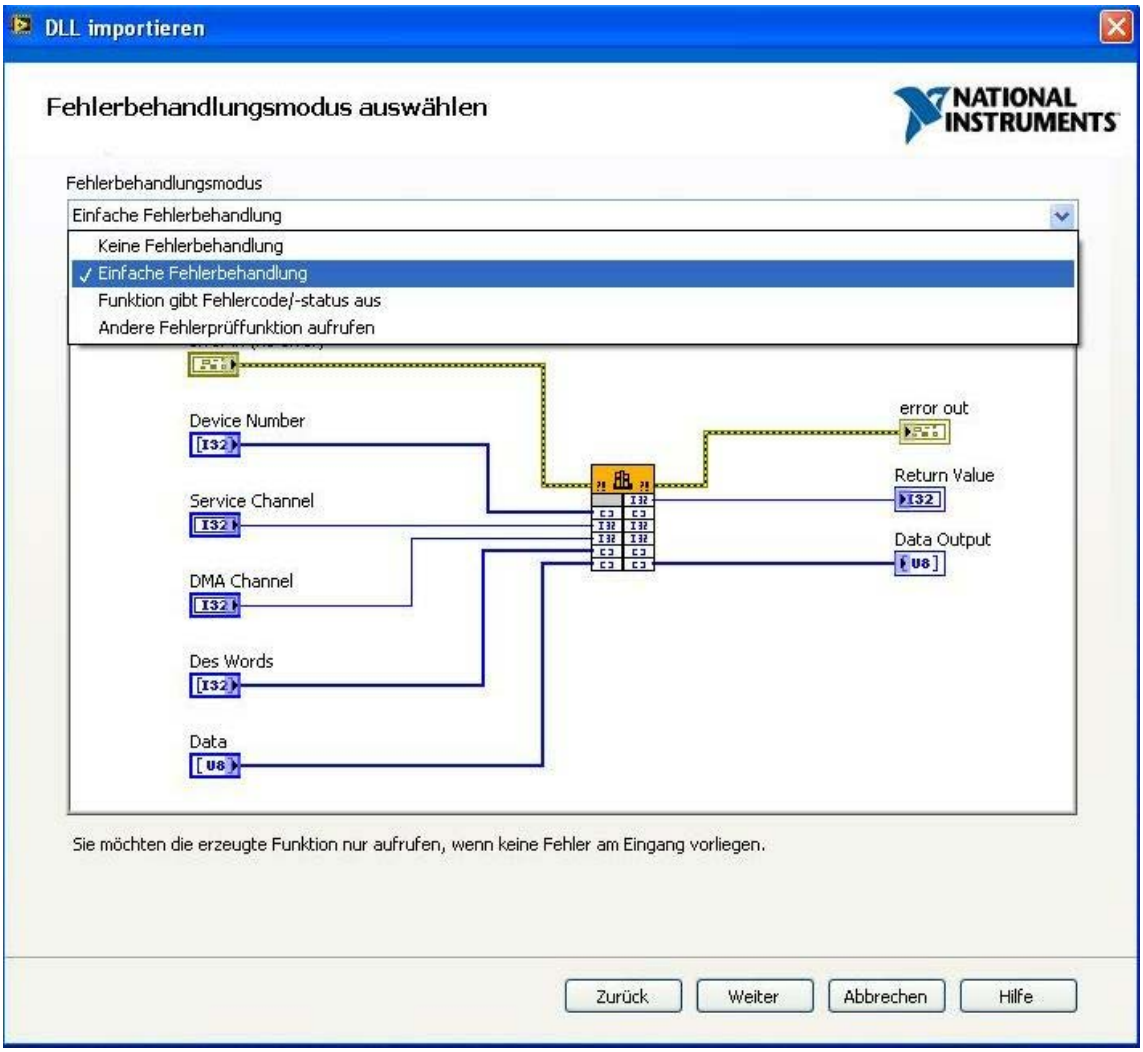
- Nochmals auf "Weiter" klicken um fortzufahren.
- Die Header-Datei wird nun analysiert. Anschließend fahren sie im folgendem Fenster wieder mit "Weiter" fort.



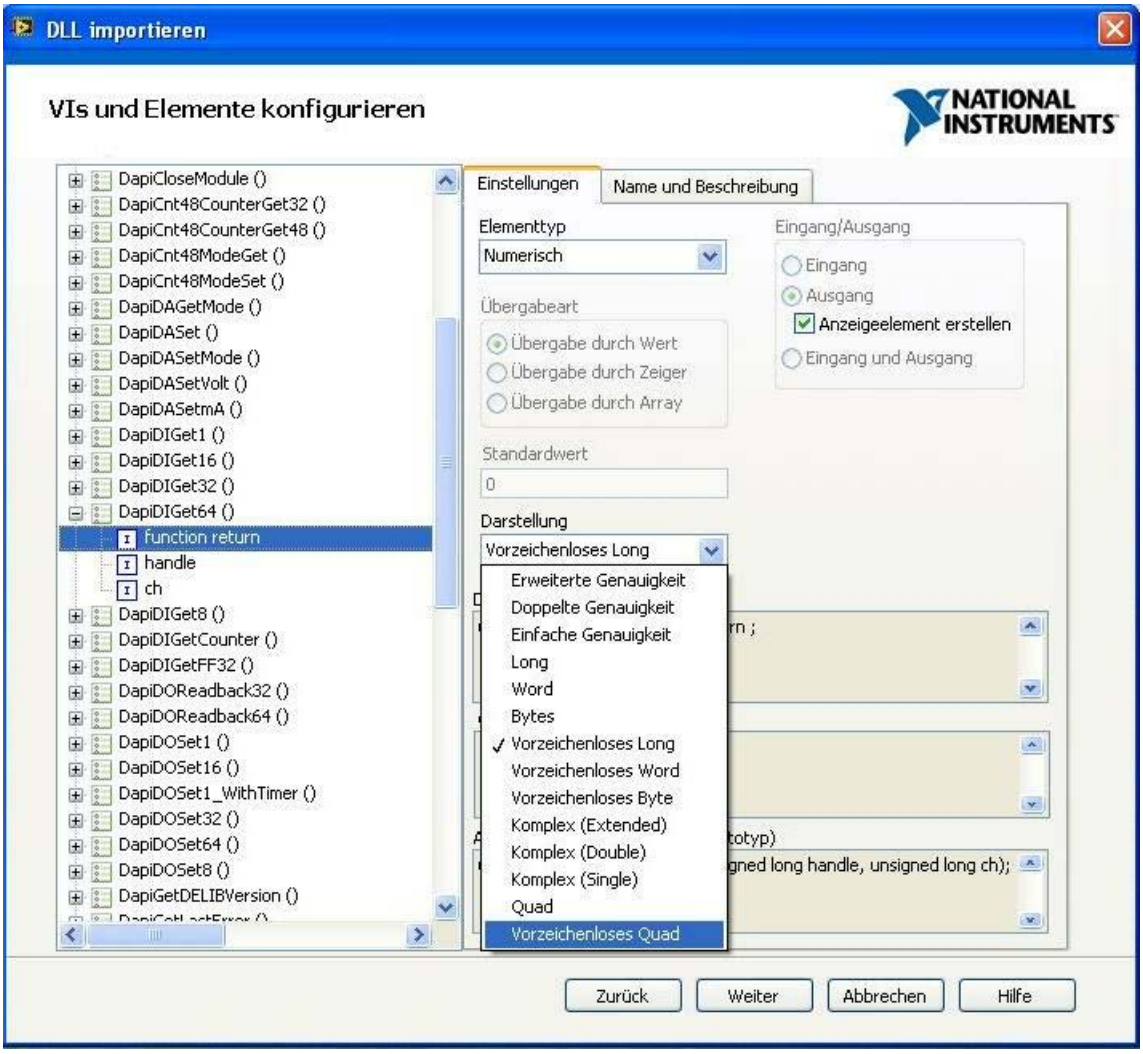
- Den weiteren Anweisungen folgen, bzw. die Konfiguration und den Speicherort für die VIs anpassen.



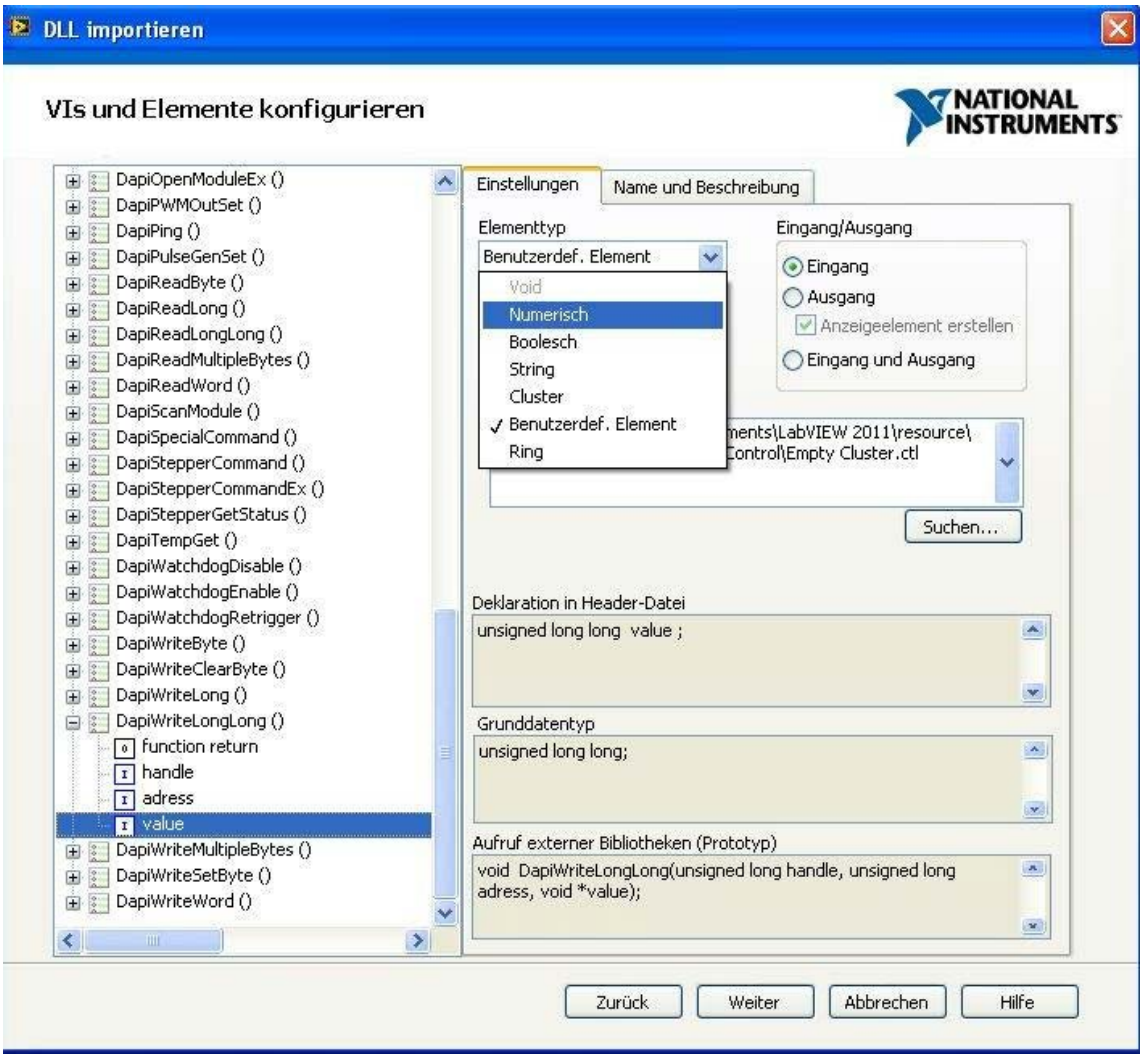
- Im folgendem Fenster wählen sie im Drop-Down Menü die Option "Einfache Fehlerbehandlung" aus und fahren mit "Weiter" fort.



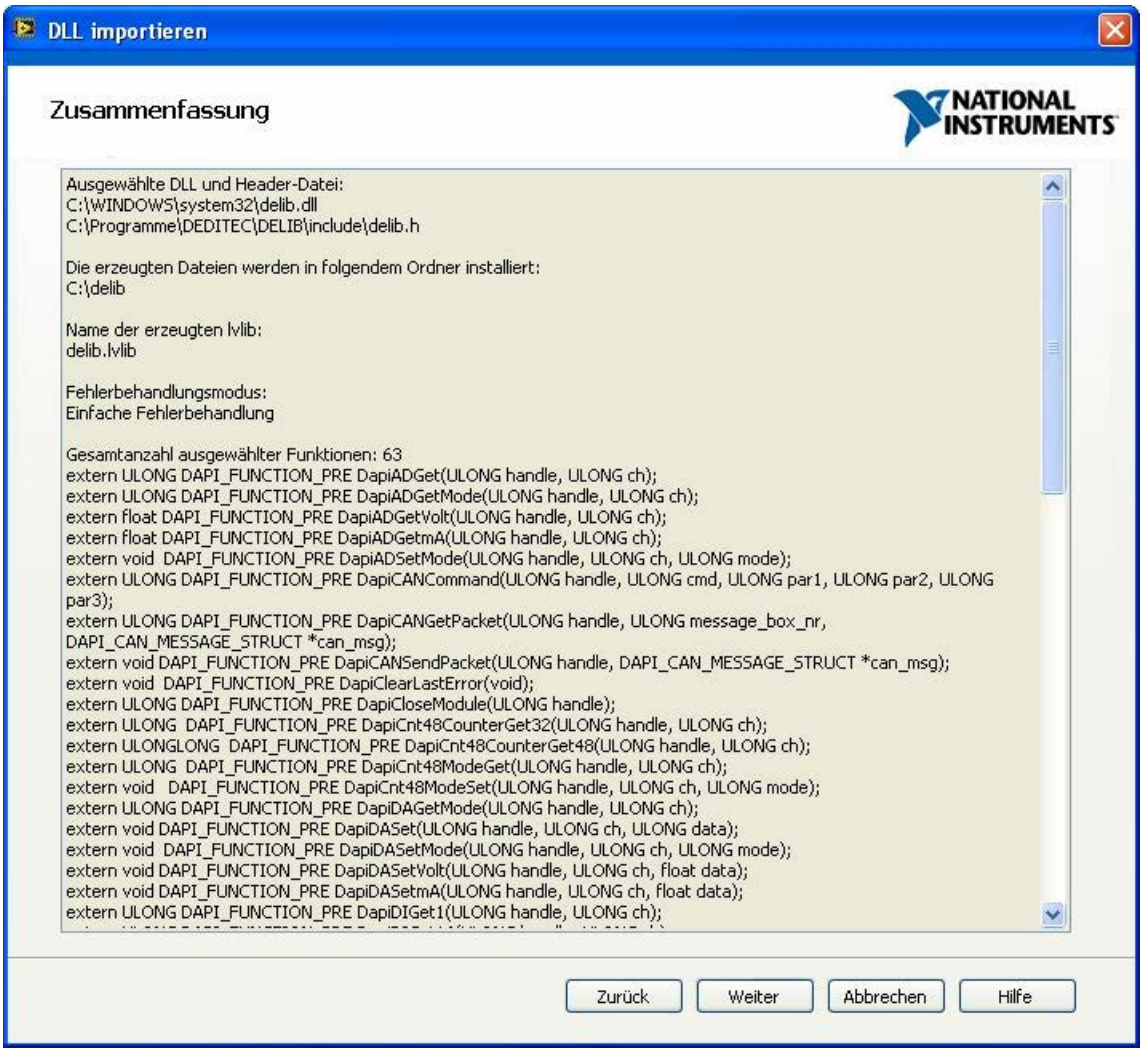
- Bei VIs die mit 64-bit Werten arbeiten, muss die Darstellung von "Vorzeichenloses Long" in "Vorzeichenloses Quad" geändert werden.
- Folgende VIs müssen bearbeitet werden:
    -> DapiCNT48CounterGet48 (function return)
    -> DapiDIGet64 (function return)
    -> DapiDOSet64 (data)
    -> DapiDOReadBack64 (function return)



- Bei manchen VIs muss zusätzlich noch der Elementtyp auf "Numerisch" geändert werden und anschließend die Darstellung auf "Vorzeichenloses Quad"
- Folgende VIs müssen bearbeitet werden:
    -> DapiWriteLongLong (value)
    -> DapiReadLongLong (function return)
- Anschließend mit "Weiter" fortfahren.



- Es erscheint eine Zusammenfassung der ausgeführten Schritte
- Zum fortfahren auf "Weiter" drücken.



- Die VIs werden nun erzeugt und können verwendet werden.

4.4.8.2. Verwendung der VIs in LabVIEW

Manche Funktion der DELIB Treiberbibliothek erwarten als Übergabeparameter einen Parameter vom Type String.

Dieses Beispiel soll zeigen, wie solche Funktionen in LabVIEW genutzt werden können.

Als Beispiel dient uns hierbei die Funktion zur Konfiguration des Spannungsbereiches eines A/D Wandlers.

Die Definition für die Funktion lautet:
void DapiADSetMode(ULONG handle, ULONG ch, ULONG mode);

Für die Funktion sind die Spannungsbereiche in der DELIB Treiberbibliothek bereits vordefiniert.

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

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

Beispielcode in C/C++:

```
DapiADSetMode(handle, 0, ADDA_MODE_UNIPOL_5V);
```

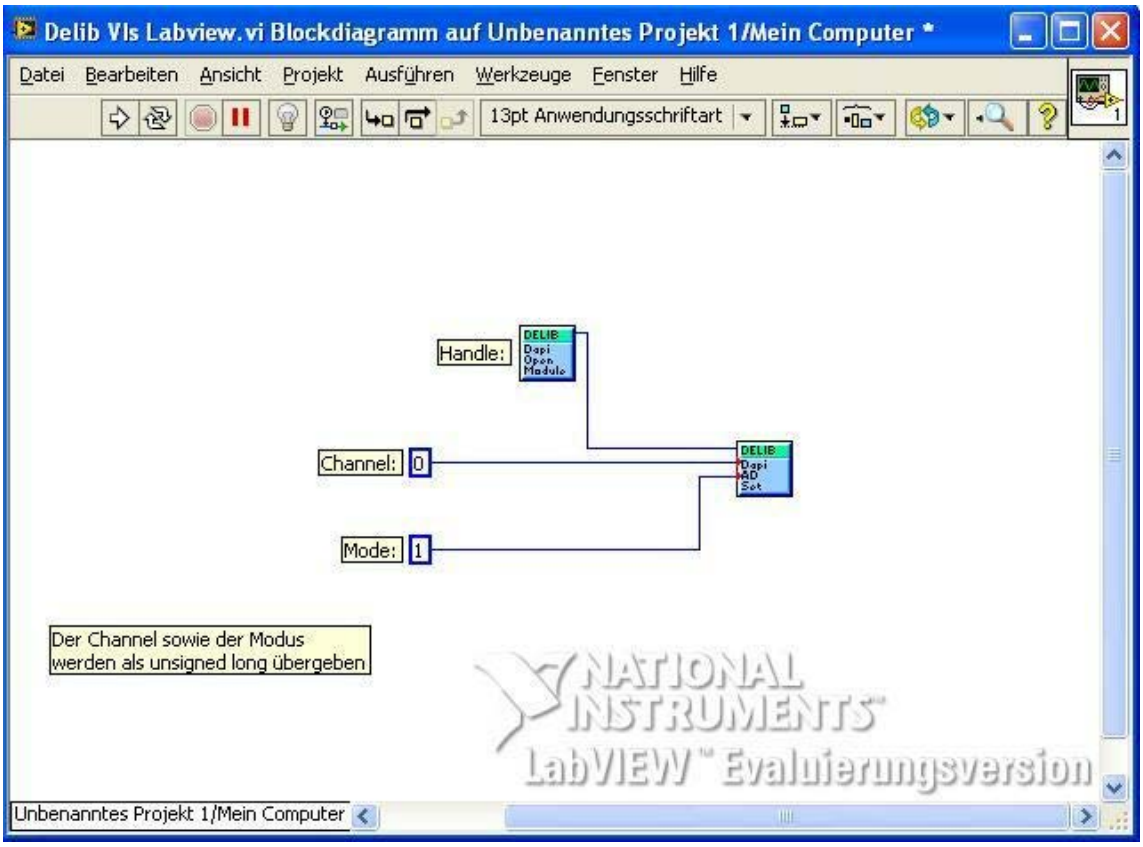
Alternativ kann man auch folgende Schreibweise verwenden:

```
DapiADSetMode(handle, 0, 1);
```

Hierbei wurde der Hexadezimalwert, den Sie aus der **delib.h** Datei entnehmen können, in Dezimal umgerechnet und als Parameter für den Modus übergeben

Die **delib.h** Datei finden sie nach der Installation der DELIB Treiberbibliothek im Verzeichnis **C:\Programme\Deditec\DELIB\Include**

In LabVIEW könnte die Funktion dann so aussehen:

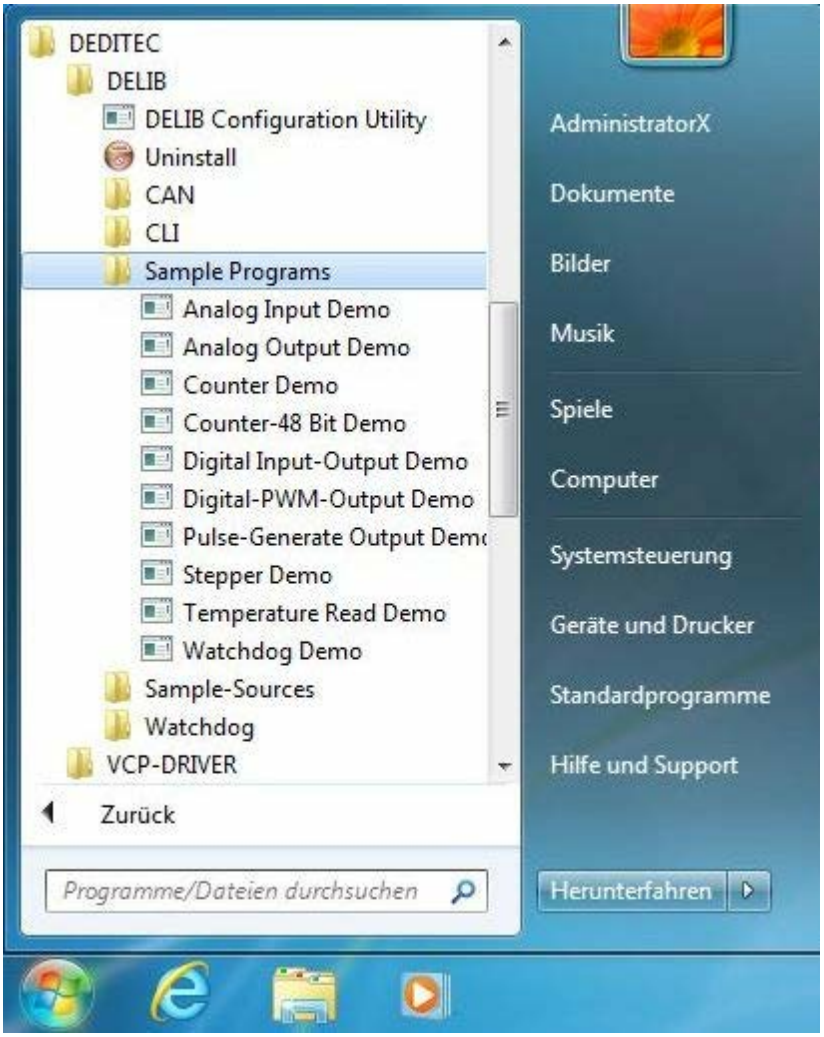


4.5. Testprogramme

4.5.1. Übersicht Testprogramme

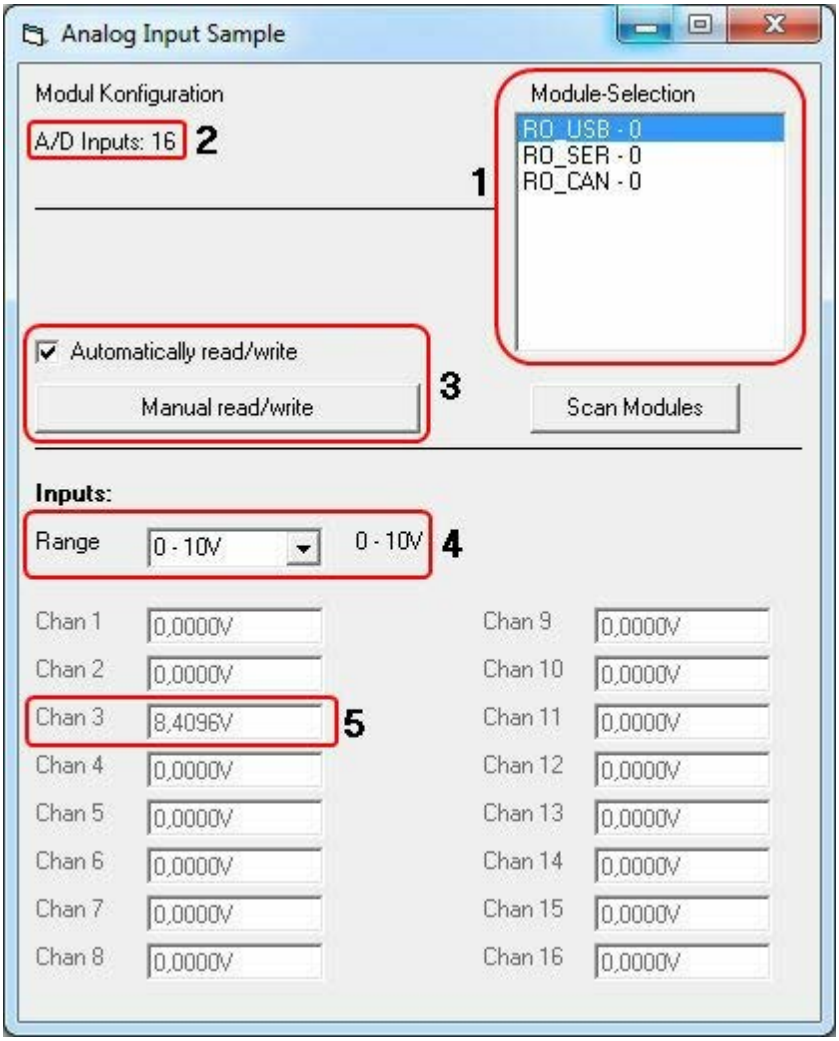
Im Lieferumfang der DELIB-Treiberbibliothek sind verschiedene Testprogramme enthalten.

Nach der Installation der DELIB-Treiberbilbliothek finden Sie die Testprogramme im Verzeichnis *"C:\Programme\DEDITEC\DELIB\programs"*
Ebenfalls können Sie die Testprogramme über *"Startmenü -> Alle Programme -> DEDITEC -> DELIB -> Sample Programs"* aufrufen.



4.5.2. Analog Input Demo

In diesem Abschnitt erläutern wir Ihnen das **Analog Input Sample** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-AD16 Modul, welches über 16 analoge Eingänge verfügt.

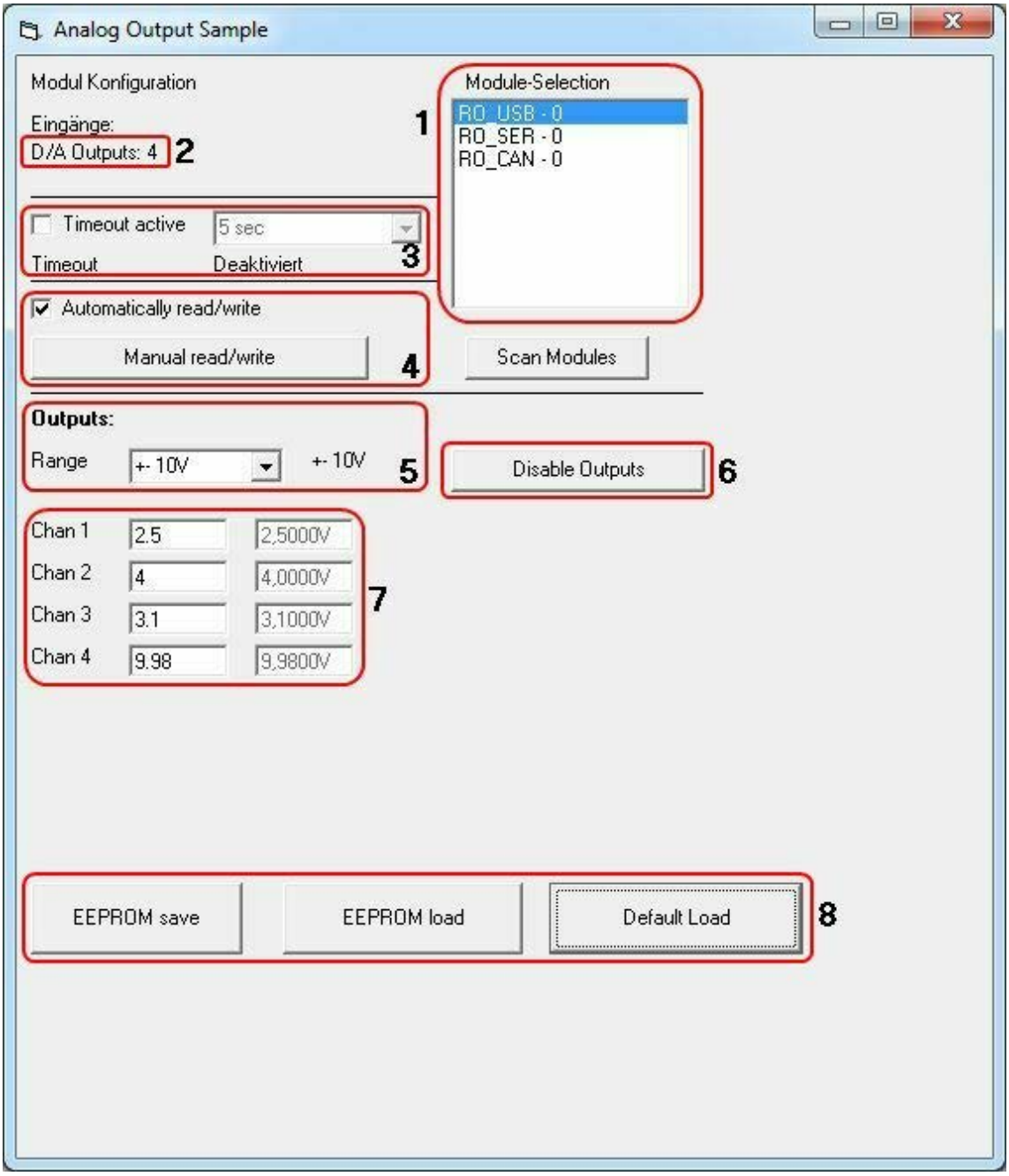
3. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob der Status der analogen Eingänge permanent ausgelesen werden soll, oder ob dies manuell über den Button **Manual read/write** geschehen soll.

4. Den Spannungs- und Strombereich den Sie messen möchten, können Sie über das Drop-Down Menü auswählen. Hierbei können nur die Modi genutzt werden, die vom Modul unterstützt werden. Bei nicht unterstützten Modi erscheint die Meldung **illegal mode**.

5. Hier sehen Sie die 16 A/D Kanäle. In diesem Beispiel haben wir 8,4V an Channel 3 angelegt.

4.5.3. Analog Output Demo

In diesem Abschnitt erläutern wir Ihnen das **Analog Output Sample** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-DA4 Modul, welches über 4 analoge Ausgänge verfügt.

3. Hier kann ein Timeout für die analogen Ausgänge definiert werden. Der Timeout bewirkt, dass alle Ausgänge ausgeschaltet werden, sobald das Modul in dem hier definierten Zeitraum nicht angesprochen werden kann. (z.B. bei einem Verbindungsabbruch)
Wichtig: Damit der Timeout getestet werden kann, muss der Haken bei **Automatically read/write** entfernt werden.

4. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob geänderte Werte direkt an das Modul gesendet werden, oder ob dies manuell über den **Manual read/write** geschehen soll.

5. Den Spannungs- und Strombereich den Sie messen möchten, können Sie über das Drop-Down Menü auswählen. Hierbei können nur die Modi genutzt werden, die vom Modul unterstützt werden. Bei nicht unterstützten Modi erscheint die Meldung **illegal mode**.

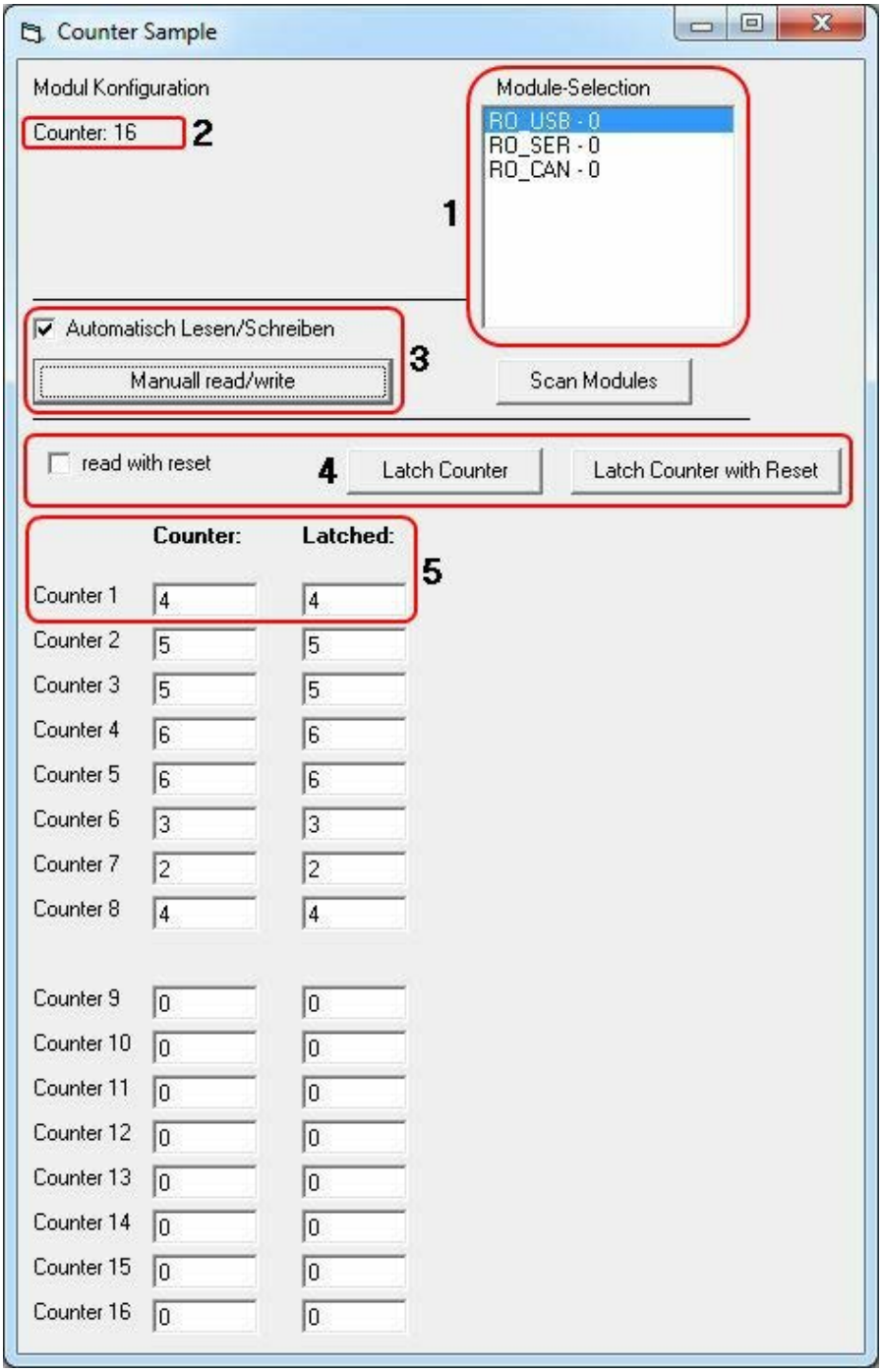
6. Mit dem Button **Disable Outputs** deaktivieren Sie die Ausgänge so, dass keine Spannungen/Ströme mehr ausgegeben werden.

7. In diesem Beispiel wird an Channel 1 2,5V, an Channel 2 4V, an Channel 3 3,1V und an Channel 4 9,98V ausgegeben.

8.Über die Buttons **EEPROM save** und **EEPROM load** können die Einstellungen aufs Modul gespeichert und geladen werden. Der Button **Default Load** stellt wie Werkseinstellungen wieder her.

4.5.4. Counter Demo

In diesem Abschnitt erläutern wir Ihnen das **Counter Demo** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-O16 Modul, welches über 16 Eingangszähler verfügt.

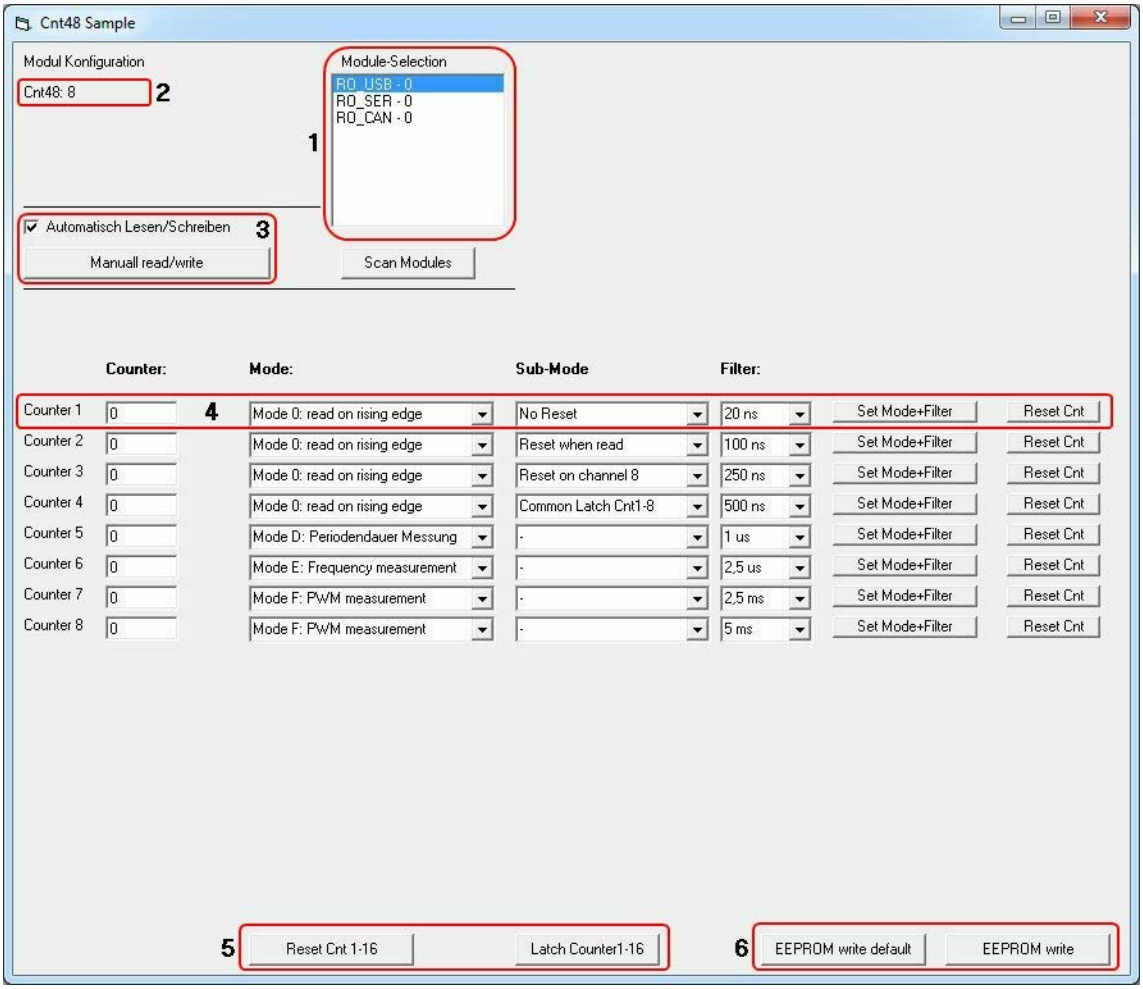
3. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob der Status der Zähler permanent ausgelesen werden soll, oder ob dies manuell über den Button **Manual read/write** geschehen soll.

4. Der Haken bei **read with reset** bewirkt, dass bei einem Auslesen der Eingangszähler diese automatisch resettet werden.
Mit dem **Latch Counter** Button wird der aktuelle Stand des Eingangszähler in einen Zwischenspeicher (**Latch**) gelegt.
Der Button **Latch Counter with Reset** legt ebenfalls den aktuellen Zählerstand in den Zwischenspeicher ab und resettet anschließend den aktuellen Zählerstand.

5. Hier wird der aktuelle Zählerstand, sowie der gespeicherte Wert im Zwischenspeicher angezeigt.

4.5.5. Counter-48 Bit Demo

In diesem Abschnitt erläutern wir Ihnen das **Counter-48 Bit Demo** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-CNT8 Modul, welches über 8 Eingangszähler verfügt.

3. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob der Status der Zähler permanent ausgelesen werden soll, oder ob dies manuell über den Button **Manual read/write** geschehen soll.

4. Der aktuelle Zählerstand eines Eingangszählers kann hier abgelesen werden. Desweiteren kann hier der Modus, Submodus und Filter für die Impulsdauer eingestellt werden. (siehe Delib-Befehl -> DapiCnt48ModeSet)

Mit dem Button **Set Mode+Filter** wird die aktuelle Konfiguration für diesen Kanal aktiv.

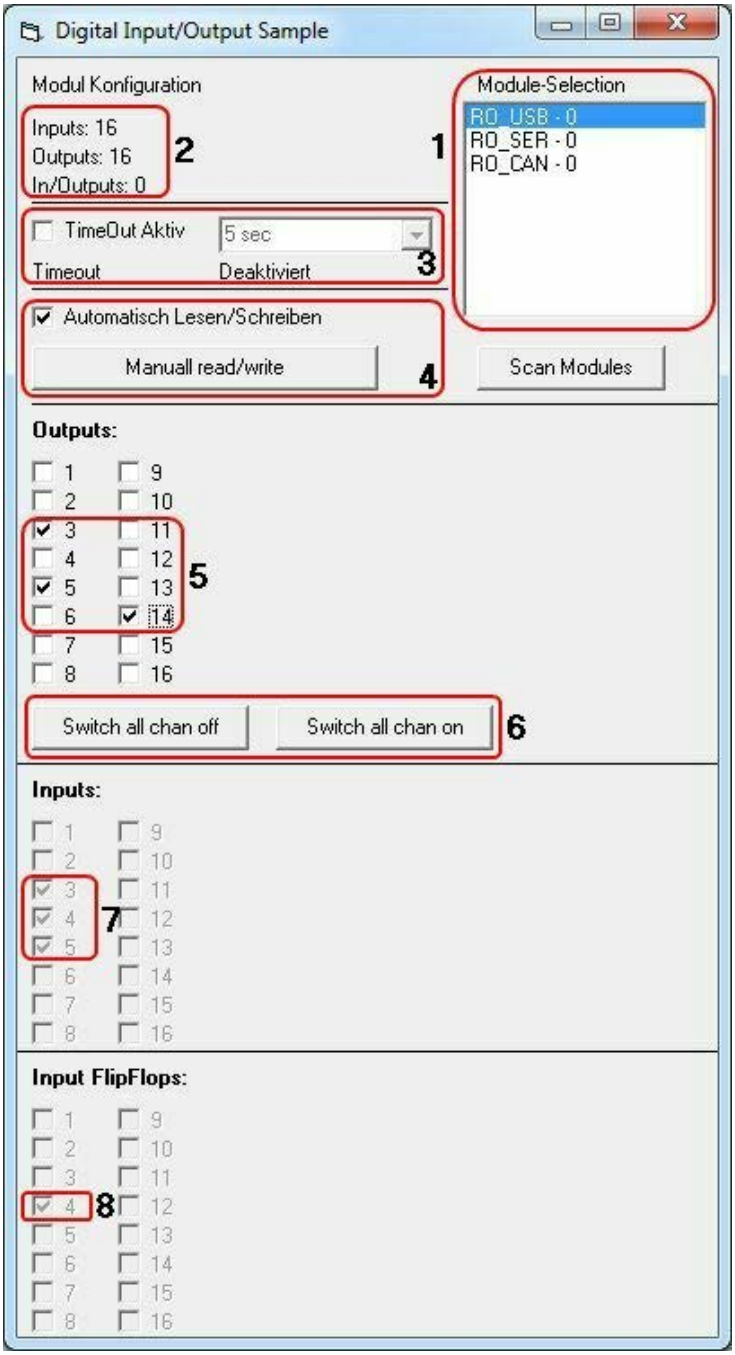
Mit dem Button **Reset Cnt** wird der Zählerstand zurückgesetzt.

5. Mit dem Button **Reset Cnt 1-16** wird der Zählerstand der Eingangszähler 1-16 zurückgesetzt. Der Button **Latch Counter 1-16** speichert den Zählerstand der Eingangszähler 1-16 in einen Zwischenspeicher (Latch).

6. Über den Button **EEPROM write** wird die aktuelle Konfiguration der Eingangskanäle im Modul gespeichert. Der Button **EEPROM write default** stellt die Werkseinstellungen wieder her.

4.5.6. Digital Input-Output Demo

In diesem Abschnitt erläutern wir Ihnen das **Digital Input-Output Sample** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-O16-R16 Modul, welches über 16 digitale Ausgänge sowie 16 digitale Eingänge verfügt.

3. Hier kann ein Timeout für die digitalen Ausgänge definiert werden. Der Timeout bewirkt, dass alle Ausgänge ausgeschaltet werden, sobald das Modul in den hier definierten Zeitraum nicht Angesprochen werden kann. (z.B. bei einem Verbindungsabbruch)
Wichtig: Damit der Timeout getestet werden kann, muss der Haken bei **Automatically read/write** entfernt werden.

4. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob permanent Werte gesetzt und gelesen werden, oder ob dies manuell über den Button **Manual read/write** geschehen soll.

5. In diesem Beispiel sind die Ausgänge 3, 5 und 14 geschaltet.

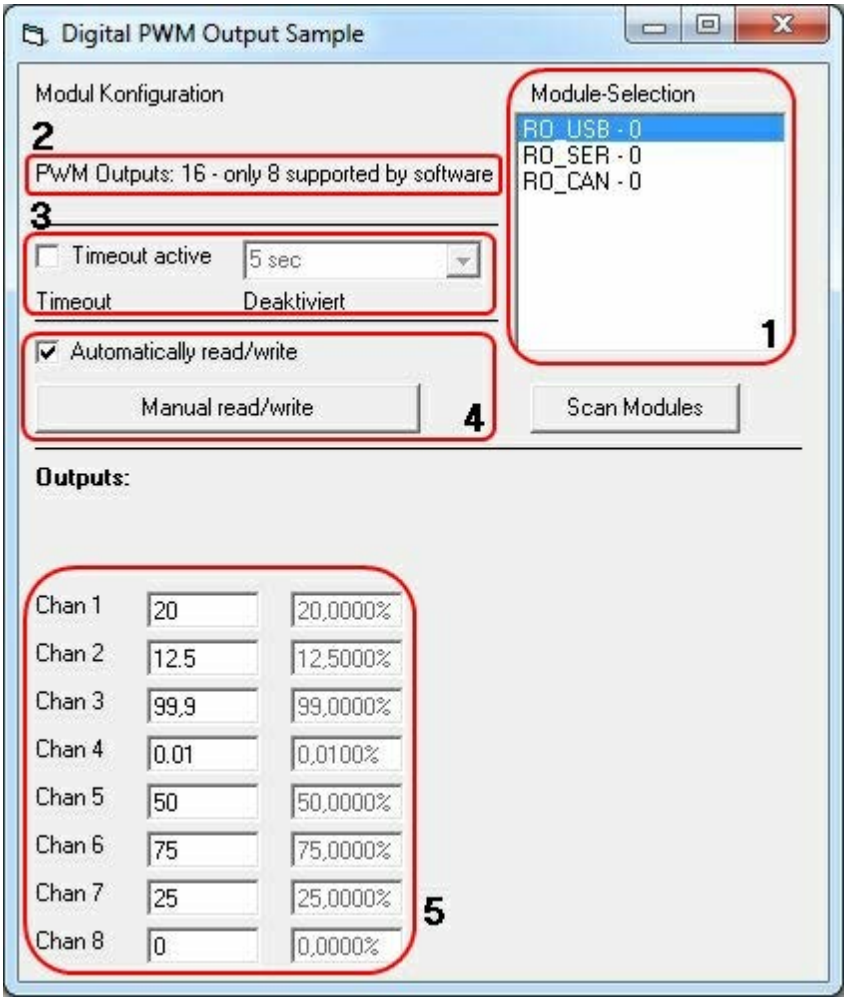
6. Mit den Buttons **Switch all chan off** und **Switch all chan on** können alle Ausgänge des Moduls ein- und ausgeschaltet werden.

7. Der Status der Eingänge kann hier abgelesen werden. Die Eingänge 3, 4 und 5 sind gesetzt.

8. Hier kann der Status der Eingangszustandsmerker (FlipFlops) abgelesen werden.

4.5.7. Digital-PWM-Output Demo

In diesem Abschnitt erläutern wir Ihnen das **Digital PWM Output Sample** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-M16 Modul, welches über 16 Ausgänge verfügt.

3. Hier kann ein Timeout für die PWM Ausgänge definiert werden. Der Timeout bewirkt, dass alle Ausgänge ausgeschaltet werden, sobald das Modul in den hier definierten Zeitraum nicht angesprochen werden kann. (z.B. bei einem Verbindungsabbruch)

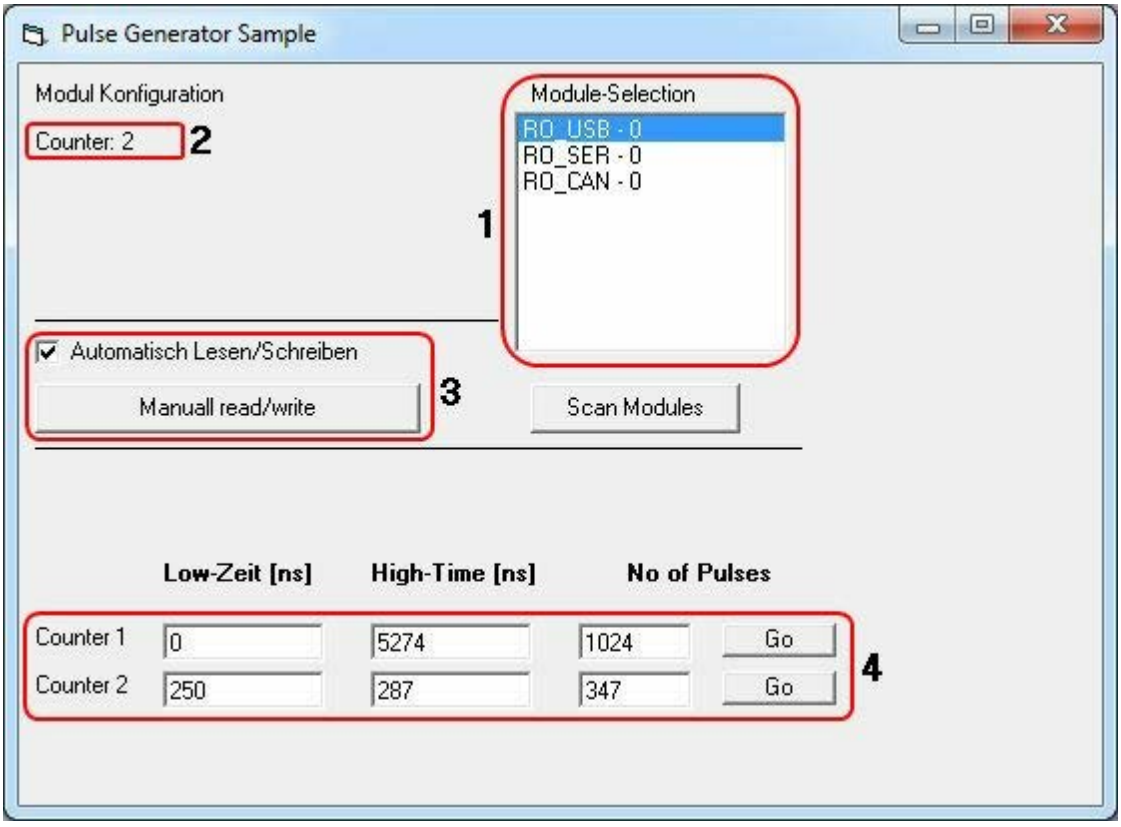
Wichtig: Damit der Timeout getestet werden kann, muss der Haken bei **Automatically read/write** entfernt werden.

4. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob geänderte Werte direkt an das Modul gesendet werden, oder ob dies manuell über den Button **Manual read/write** geschehen soll.

5. Hier können die Frequenzen für die PWM Ausgänge konfiguriert werden.

4.5.8. Pulse-Generate Output Demo

In diesem Abschnitt erläutern wir Ihnen das **Pulse Generator Sample** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

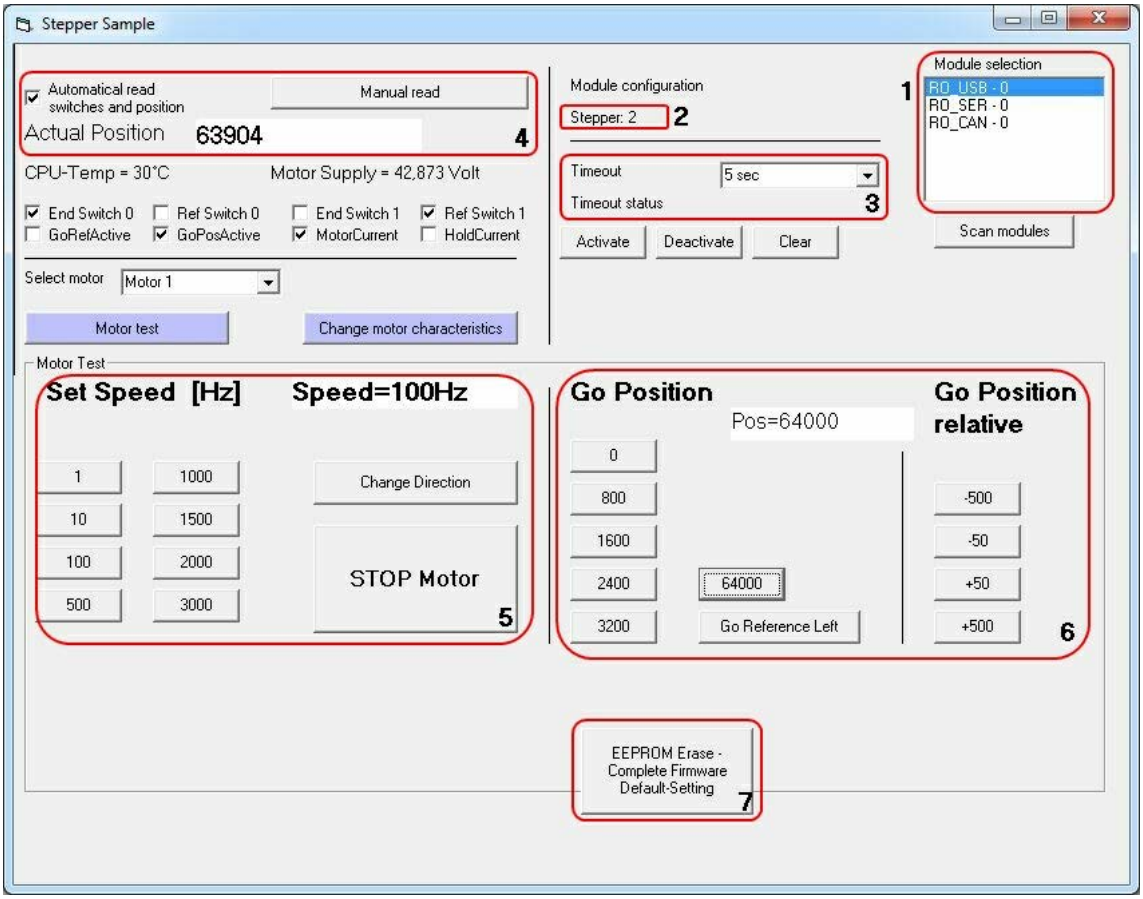
2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-CNT8 Modul, welches über 2 Ausgangszähler verfügt.

3. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob geänderte Werte direkt an das Modul gesendet werden, oder ob dies manuell über den Button **Manual read/write** Button geschehen soll.

4. Die Low-Time, High-Time und die Anzahl der zu generierenden Pulse kann hier eingestellt werden. Nach einen Klick auf den Button **Go** wird das generieren gestartet.

4.5.9. Stepper Demo

In diesem Abschnitt erläutern wir Ihnen das **Stepper Demo** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-Stepper2 Modul, welches über 2 Stepper verfügt.

3. Hier kann ein Timeout für die Stepper definiert werden. Der Timeout bewirkt, dass alle Stepper ausgeschaltet werden, sobald das Modul in den hier definierten Zeitraum nicht Angesprochen werden kann. (z.B. bei einem Verbindungsabbruch)
Wichtig: Damit der Timeout getestet werden kann, muss der Haken bei **Automatically read/write** entfernt werden.

4. Mit dem Haken bei **Automatically read switches and position** beinflussen Sie, ob der Status der Stepper permanent ausgelesen werden soll, oder manuell über den Button **Manual read**.

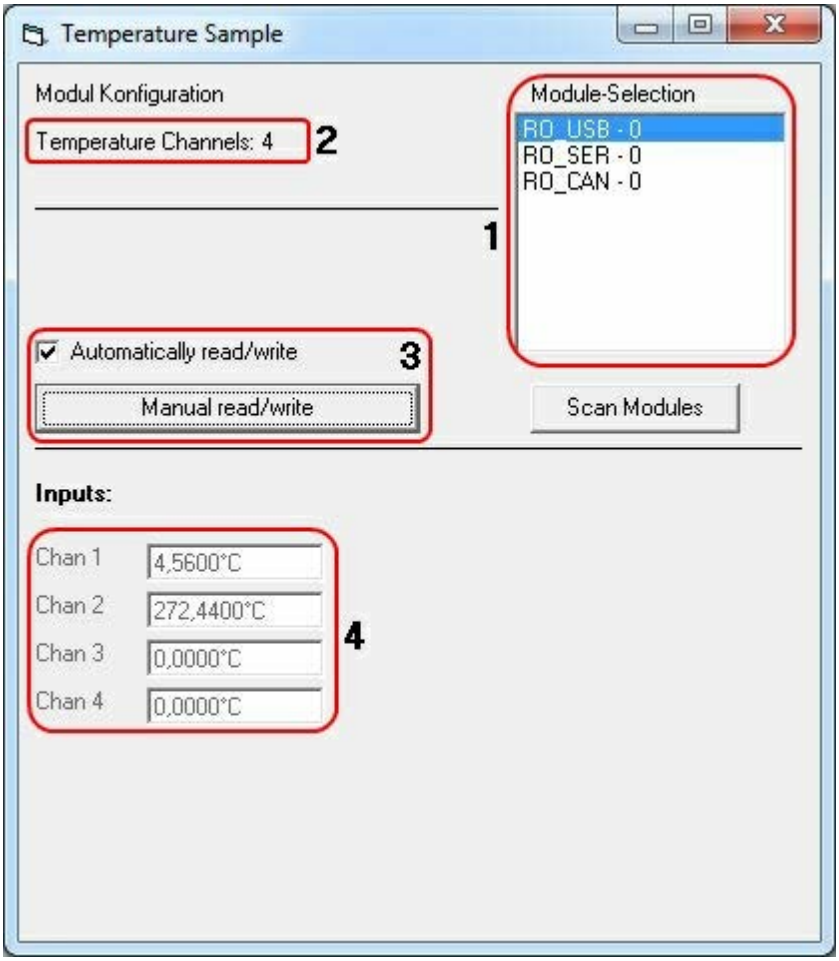
5. Die Geschwindigkeit des Stepper-Motors kann hier konfiguriert oder ausgelesen werden. Ausserdem kann der Motor mit dem Button **STOP Motor** angehalten werden.

6. Hier kann eine Position definiert werden, die vom Stepper angefahren werden soll. Hierbei kann eine absolute oder relative (zur momentanen Position) Position definiert werden.

7. Mit dem Button **EEPROM Erase - Complete Firmware Default-Setting** können Sie die Werkseinstellungen der Stepper wiederherstellen.

4.5.10. Temperature Read Demo

In diesem Abschnitt erläutern wir Ihnen das **Temperature Read Demo** Testprogramm.



1. Wählen Sie das gewünschte Modul aus. Mit dem Button **Scan Modules** können Sie nach allen angeschlossenen Modulen suchen.

2. Hier wird die Konfiguration des Moduls anzeigt. In diesem Fall, handelt es sich um ein RO-USB-PT100-4 Modul, welches über 4 PT100 Eingänge verfügt.

3. Mit dem Haken bei **Automatically read/write** beinflussen Sie, ob geänderte Werte direkt an das Modul gesendet werden, oder ob dies manuell über den Button **Manual read/write** geschehen soll.

4. Hier kann die Temperatur der PT100 Sensoren abgelesen werden.

4.6. DELIB CLI (command-line interface)

Da in manchen Programmiersprachen (wie z.B. PHP) keine DLLs eingebunden werden können, gibt es hierzu ein extra Kommandozeilen-Befehl, der direkt aus dem Programm (mit den entsprechenden Parametern) heraus aufgerufen werden kann.

Der DELIB CLI Befehl für Windows befindet sich nach der Installation der DELIB-Treiberbibliothek im Verzeichnis C:\Programme\DEDITEC\DELIB\programs\cli\ .
Der DELIB CLI Befehl für Linux befindet sich nach Entpacken des Zip-Archivs "delib-linux-cli" im Ordner "/deditec-cli/ .

Definition (Windows)
delib_cli command channel [value | unit ["nounit"]]

Definition für USB-Module (Linux)
sudo delib-cli-usb command channel [value | unit ["nounit"]]

Definition für RO-ETH-Module (Linux)
delib-cli-eth command channel [value | unit ["nounit"]]

Hinweis: Die einzelnen Parameter werden nur durch ein Leerzeichen getrennt.
Groß und Kleinschreibung wird hierbei nicht beachtet.

Parameter

<table>
<tr><th>command</th><th>channel</th><th colspan="2">value</th><th>unit</th><th>nounit</th></tr>
<tr><td>di1</td><td>0, 1, 2, ...</td><td colspan="2" rowspan="4">-</td><td rowspan="4">hex</td><td rowspan="4">nounit</td></tr>
<tr><td>di8</td><td rowspan="3">0, 8, 16, ...</td></tr>
<tr><td>di16</td></tr>
<tr><td>di24</td></tr>
<tr><td>do1</td><td>0, 1, 2, ...</td><td colspan="2">0/1 (1-Bit Befehl)</td><td rowspan="4">-</td><td rowspan="4">-</td></tr>
<tr><td>do8</td><td rowspan="3">0, 8, 16, ...</td><td>8-Bit Wert</td><td rowspan="3">(Bit 0 für Kanal 1, Bit 1 für Kanal 2, ...)</td></tr>
<tr><td>do16</td><td>16-Bit Wert</td></tr>
<tr><td>do24</td><td>24-Bit Wert</td></tr>
<tr><td>ai</td><td>0, 1, 2, ...</td><td colspan="2">-</td><td>hex, volt, mA</td><td>nounit</td></tr>
<tr><td>ao</td><td>0, 1, 2, ...</td><td colspan="2">Ganz oder Hexadezimalzahl (beginnend mit 0x).
Bei einer Ganzzahl kann an den Wert ein V für Volt oder ein mA für milli Ampere angehangen werden.</td><td>-</td><td>-</td></tr>
</table>

Return-Wert

Zustand der gelesenen digitalen Eingänge
In Kombination mit Parameter unit "hex" wird der Zustand als hex gelesen

Zustand der gelesenen analogen Eingänge
In Kombination mit Parameter unit "hex" wird der Zustand als hex gelesen
In Kombination mit Parameter unit "volt" wird die Spannung gelesen
In Kombination mit Parameter unit "mA" wird der Strom gelesen

4.6.1. Anpassung für USB-Module (nur Linux)

Damit das richtige USB-Modul unter Linux via DELIB CLI angesprochen werden kann, muss die entsprechende ID des Moduls beim Befehl "DapiOpenModule" gesetzt werden. Hierzu finden Sie nach dem Entpacken des Archivs im Ordner "/deditec-cli/source" die Datei "delib_cli_open_module_usb.c"

Standardmäßig versucht das Programm "delib-cli-usb" (siehe Quellcode), zuerst ein RO-USB-Modul zu öffnen, wird dies nicht gefunden, wird versucht ein USB-Mini-Stick zu öffnen, usw.. .

Da so unnötige Fehlermeldungen verursacht werden, können nicht benötigte Einträge einfach in Remarks gesetzt werden.

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

ULONG handle;

ULONG delib_cli_open_module()
{
	handle = DapiOpenModule(RO_USB, 0);
	if(handle==0)
	{
		handle = DapiOpenModule(USB_MINI_STICK, 0);
	}
	if(handle==0)
	{
		handle = DapiOpenModule(USB_RELAIS_8, 0);
	}
	if(handle==0)
	{
		handle = DapiOpenModule(USB_OPTOIN_8_RELAIS_8, 0);
	}
	if(handle==0)
	{
		handle = DapiOpenModule(USB_OPTOIN_16_RELAIS_16, 0);
	}
	if(handle==0)
	{
		handle = DapiOpenModule(USB_OPTOIN_32_RELAIS_32, 0);
	}

	return handle;
}
```

Hinweis:

Nach jeder Änderung im Quellcode muss das Projekt "delib-cli-usb" neu kompiliert werden. Hierzu finden Sie im Verzeichnis "/deditec-cli/" einen Shell-Script "compile_delib_cli_usb.sh" mit dem Sie das Projekt unter Linux kompilieren können.

4.6.2. Anpassungen für RO-ETH-Module (nur Linux)

Damit das RO-ETH-Modul unter Linux via DELIB CLI angesprochen werden kann, muss die IP-Adresse des Moduls gesetzt werden. Hierzu finden Sie nach dem Entpacken des Archivs im Ordner "/deditec-cli/source" die Datei "delib_cli_open_module_eth.c"

Hier muss folgender Eintrag vorgenommen werden:

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

extern char global_ip_addr[];
ULONG handle;

ULONG delib_cli_open_module()
{
	sprintf(global_ip_addr, "192.168.1.11");		// Hier muss die IP-Adresse des RO-
ETH-Moduls angegeben werden

	handle = DapiOpenModule(RO_ETH,0);

	return handle;
}
```

Hinweis:

Nach jeder Änderung im Quellcode muss das Projekt "delib-cli-eth" neu kompiliert werden. Hierzu finden Sie im Verzeichnis "/deditec-cli/" einen Shell-Script "compile_delib_cli_eth.sh" mit dem Sie das Projekt unter Linux kompilieren können.

4.6.3. DELIB CLI Beispiele

Digitale Ausgänge

Windows

delib_cli DO1 17 1
-> schaltet das 18. digitale Relais an

delib_cli DO1 3 0
-> schaltet das 4. digitale Relais aus

delib_cli DO8 0 255
-> schaltet die digitalen Relais 1 bis 8 an

delib_cli DO16 0 0
-> schaltet die digitalen Relais 1 bis 16 aus

delib_cli DO16 16 65535
-> schaltet die digitalen Relais 17 bis 32 an

delib_cli DO32 0 4294967295
-> schaltet die digitalen Relais 1 bis 32 an

Linux

sudo delib_cli _usb DO1 17 1
-> schaltet das 18. digitale Relais eines USB-Moduls an

delib_cli _eth DO1 3 0
-> schaltet das 4. digitale Relais eines RO-ETH-Moduls aus

Digitale Eingänge

Windows

delib_cli DI1 3
Beispiel eines Rückgabewertes: **1**
-> lese den Zustand des 4. digitalen Eingangs und gebe ihn zurück

delib_cli DI8 0 hex
Beispiel eines Rückgabewertes: **0xFF**
-> lese den Wert von digitalen Eingang 1-8 als hexadezimalzahl

delib_cli DI16 0 hex
Beispiel eines Rückgabewertes: **0xFFFF**
-> lese den Wert von digitalen Eingang 1-16 als hexadezimalzahl

delib_cli DI32 0 hex
Beispiel eines Rückgabewertes: **0xFFFFFFFF**
-> lese den Wert von digitalen Eingang 1-32 als hexadezimalzahl

Alternativ kann das Argument "nounit" an alle zu formatierenden Ausgabeanfragen wie folgendermaßen angehangen werden:
delib_cli DI8 0 hex nounit
Beispiel eines Rückgabewertes: **FF**
-> lese den Wert von digitalen Eingang 1-8 als hexadezimalzahl

Linux

sudo delib_cli _usb DI1 3
Beispiel eines Rückgabewertes: **1**
-> lese den Zustand des 4. digitalen Eingangs eines USB-Moduls und gebe ihn zurück

delib_cli_eth DI8 0 hex
Beispiel eines Rückgabewertes: **0xFF**
-> lese den Wert von digitalen Eingang 1-8 eines RO-ETH-Moduls als hexadezimalzahl

Analoge Ausgänge

Windows

delib_cli AO 7 4711
-> setzt den dezimalen Wert 4711 auf den 8. analogen Ausgang

delib_cli AO 6 0x4711
-> setzt den hexadezimalen Wert 0x4AF1 auf den 7. analogen Ausgang

delib_cli AO 7 3.7V
-> setzt die Spannung des 8. analogen Ausgangs auf 3,7 Volt
(sowohl Komma "," als auch Punkt "." können zur Kommatrennung verwendet werden)

delib_cli AO 7 13.3mA
-> setzt den Strom des 8. analogen Ausgangs auf 13,3 milli Ampere
(sowohl Komma "," als auch Punkt "." können zur Kommatrennung verwendet werden)

Linux

sudo delib_cli_usb AO 7 4711
-> setzt den dezimalen Wert 4711 auf den 8. analogen Ausgang eines USB-Moduls

delib_cli_eth AO 6 0x4711
-> setzt den hexadezimalen Wert 0x4AF1 auf den 7. analogen Ausgang eines RO-ETH-Moduls

Analoge Eingänge

Windows

delib_cli AI 2
Beispiel eines Rückgabewertes: **1234**
-> liest den Wert des 3. analogen Eingangs als dezimalzahl

delib_cli AI 2 hex
Beispiel eines Rückgabewertes: **0x1FA**
-> liest den Wert des 3. analogen Eingangs als hexadezimalzahl

delib_cli AI 2 V
Beispiel eines Rückgabewertes: **12.500000V**
-> liest die Spannung des 3. analogen Eingangs als kommazahl

delib_cli AI 2 mA
Beispiel eines Rückgabewertes: **20.551600mA**
-> liest den Strom des 3. analogen Eingangs als kommazahl

Alternativ kann auch hier das Argument "nounit" an alle zu formatierenden Ausgabeanfragen wie folgendermaßen angehangen werden:

delib_cli AI 3 hex nounit
Beispiel eines Rückgabewertes: **1FA**
-> liest den Wert des 4. analogen Eingangs als hexadezimalzahl

delib_cli AI 3 V nounit
Beispiel eines Rückgabewertes: **12.500000**
-> liest die Spannung des 4. analogen Eingangs als kommazahl

delib_cli AI 3 mA nounit
Beispiel eines Rückgabewertes: **20.551600**
-> liest den Strom des 4. analogen Eingangs als kommazahl

Linux

sudo delib_cli_usb AI 2
Beispiel eines Rückgabewertes: **1234**
-> liest den Wert des 3. analogen Eingangs als dezimalzahl eines USB-Moduls

delib_cli_eth AI 2 hex
Beispiel eines Rückgabewertes: **0x1FA**
-> liest den Wert des 3. analogen Eingangs als hexadezimalzahl eines RO-ETH-Moduls

DELIB API Referenz

V

5. DELIB API Referenz

5.1. Verwaltungsfunktionen

5.1.1. DapiOpenModule

Beschreibung
Diese Funktion öffnet ein bestimmtes Modul.

Definition
ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

Parameter
moduleID=Gibt das Modul an, welches geöffnet werden soll (siehe delib.h)
nr=Gibt an, welches (bei mehreren Modulen) geöffnet werden soll.
nr=0 -> 1. Modul
nr=1 -> 2. Modul

Return-Wert
handle=Entsprechender Handle für das Modul
handle=0 -> Modul wurde nicht gefunden

Bemerkung
Der von dieser Funktion zurückgegebene Handle wird zur Identifikation des Moduls für alle anderen Funktionen benötigt.

Programmierbeispiel

```
// USB-Modul öffnen
handle = DapiOpenModule(RO_USB1, 0);
printf("handle = %x\n", handle);
if (handle==0)
{
// USB Modul wurde nicht gefunden
printf("Modul konnte nicht geöffnet werden\n");
return;
}
```

5.1.2. DapiCloseModule

Beschreibung
Dieser Befehl schliesst ein geöffnetes Modul.

Definition
ULONG DapiCloseModule(ULONG handle);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert
Keiner

Programmierbeispiel

```
// Modul schliessen
DapiCloseModule(handle);
```

5.1.3. DapiGetDELIBVersion

Beschreibung
Diese Funktion gibt die installierte DELIB-Version zurück.

Definition
ULONG DapiGetDELIBVersion(ULONG mode, ULONG par);

Parameter
mode=Modus, mit dem die Version ausgelesen wird (muss immer 0 sein).
par=Dieser Parameter ist nicht definiert (muss immer 0 sein).

Return-Wert
version=Versionsnummer der installierten DELIB-Version [hex]

Programmierbeispiel

```
version = DapiGetDELIBVersion(0, 0);
//Bei installierter Version 1.32 ist version = 132(hex)
```

5.1.4. DapiSpecialCMDGetModuleConfig

Beschreibung

Diese Funktion gibt die Hardwareaustattung (Anzahl der Ein- bzw. Ausgangskanäle) des Moduls zurück.

Definition

ULONG DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG, par, 0, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Anzahl der digitalen Eingangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI

Anzahl der digitalen Eingangs Flip-Flops abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI_FF

Anzahl der digitalen Eingangszähler abfragen (16-Bit Zähler)

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI_COUNTER

Anzahl der digitalen Eingangszähler abfragen (48-Bit Zähler)

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_CNT48

Anzahl der digitalen Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO

Anzahl der digitalen Pulsgenerator-Ausgänge abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_PULSE_GEN

Anzahl der digitalen PWM-Ausgänge abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_PWM_OUT

Anzahl der digitalen Ein-/Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX

Anzahl der analogen Eingangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD

Anzahl der analogen Ausgangskanäle abfragen
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA

Anzahl der Temperaturkanäle abfragen
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_TEMP

Anzahl der Stepperkanäle abfragen
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER

Return-Wert

Anzahl der digitalen Eingangskanäle abfragen
return=Anzahl der digitalen Eingangskanäle

Anzahl der digitalen Eingangs Flip-Flops abfragen
return=Anzahl der digitalen Eingangs Flip-Flops

Anzahl der digitalen Eingangszähler abfragen (16-Bit Zähler)
return=Anzahl der digitalen Eingangszähler (16-Bit Zähler)

Anzahl der digitalen Eingangszähler abfragen (48-Bit Zähler)
return=Anzahl der digitalen Eingangszähler (48-Bit Zähler)

Anzahl der digitalen Ausgangskanäle abfragen
return=Anzahl der digitalen Ausgangskanäle

Anzahl der digitalen Pulsgenerator-Ausgänge abfragen
return=Anzahl der digitalen Pulsgenerator-Ausgänge

Anzahl der digitalen PWM-Ausgänge abfragen
return=Anzahl der digitalen PWM-Ausgänge

Anzahl der digitalen Ein-/Ausgangskanäle abfragen
return=Anzahl der digitalen Ein-/Ausgangskanäle

Anzahl der analogen Eingangskanäle abfragen
return=Anzahl der analogen Eingangskanäle

Anzahl der analogen Ausgangskanäle abfragen
return=Anzahl der analogen Ausgangskanäle

Anzahl der Temperaturkanäle abfragen
return=Anzahl der Temperaturkanäle

Anzahl der Stepperkanäle abfragen
return=Anzahl der Stepperkanäle

Programmierbeispiel

```
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI, 0, 0);
//Gibt die Anzahl der digitalen Eingangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO, 0, 0);
//Gibt die Anzahl der digitalen Ausgangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX, 0, 0);
//Gibt die Anzahl der digitalen Ein-/Ausgangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD, 0, 0);
//Gibt die Anzahl der analogen Eingangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA, 0, 0);
//Gibt die Anzahl der analogen Ausgangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER, 0, 0);
//Gibt die Anzahl der Stepperkanäle zurück
```

5.1.5. DapiOpenModuleEx

Beschreibung

Diese Funktion öffnet ein bestimmtes RO-ETH-Modul. Besonderheit hierbei ist, dass Parameter wie IP-Adresse und Portnummer direkt mit angegeben werden.

Definition

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

Parameter

moduleID=Gibt das Modul an, welches geöffnet werden soll (siehe delib.h)
nr=Gibt an, welches (bei mehreren Modulen) geöffnet werden soll.
nr=0 -> 1. Modul
nr=1 -> 2. Modul
exbuffer=Buffer für IP-Adresse und Port

Return-Wert

handle=Entsprechender Handle für das Modul
handle=0 -> Modul wurde nicht gefunden

Bemerkung

Der von dieser Funktion zurückgegebene Handle wird zur Identifikation des Moduls für alle anderen Funktionen benötigt.
Dieser Befehl wird derzeit nur von Modulen der RO-ETH-Serie unterstützt.

Programmierbeispiel

```
// Open ETH-Module with parameter

DAPI_OPENMODULEEX_STRUCT open_buffer;

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

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

5.2. Fehlerbehandlung

5.2.1. DapiGetLastError

Beschreibung
Diese Funktion liefert den letzten erfassten Fehler.

Definition
ULONG DapiGetLastError();

Parameter
Keine

Return-Wert
Fehler Code
0=kein Fehler. (siehe delib.h)

Programmierbeispiel

```
ULONG error;
error=DapiGetLastError();
if(error==0) return FALSE;
printf("ERROR = %d”, error);
```

5.2.2. DapiGetLastErrorText

Beschreibung

Diese Funktion liest den Text des letzten erfassten Fehlers.

Definition

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

Parameter

msg = Buffer für den zu empfangenden Text

msg_length = Länge des Text Buffers

Programmierbeispiel

```
BOOL IsError ()
{
  if (DapiGetLastError () != DAPI_ERR_NONE)
       {
         unsigned char msg[500];

         DapiGetLastErrorText((unsigned char*) msg, sizeof(msg));
         printf ("Error Code = %x * Message = %s\n", 0, msg);
         return TRUE;
       }
  return FALSE;
}
```

5.3. Digitale Eingänge lesen

5.3.1. DapiDIGet1

Beschreibung
Dieser Befehl liest einen einzelnen digitalen Eingang.

Definition
ULONG DapiDIGet1(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, der gelesen werden soll (0, 1, 2, 3, ..)

Return-Wert
Zustand des Eingangs (0/1)

5.3.2. DapiDIGet8

Beschreibung
Dieser Befehl liest gleichzeitig 8 digitale Eingänge.

Definition
ULONG DapiDIGet8(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 8, 16, 24, ..)

Return-Wert
Zustand der gelesen Eingänge

5.3.3. DapiDIGet16

Beschreibung
Dieser Befehl liest gleichzeitig 16 digitale Eingänge.

Definition
ULONG DapiDIGet16(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 16, 32, ...)

Return-Wert
Zustand der gelesen Eingänge

5.3.4. DapiDIGet32

Beschreibung
Dieser Befehl liest gleichzeitig 32 digitale Eingänge.

Definition
ULONG DapiDIGet32(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 32, 64, ..)

Return-Wert
Zustand der gelesen Eingänge

Programmierbeispiel

```
unsigned long data;
// ----------------------------------------------------
// Einen Wert von den Eingängen lesen (Eingang 1-31)
data = (unsigned long) DapiDIGet32(handle, 0);
// Chan Start = 0
printf("Eingang 0-31 : 0x%x\n", data);
printf("Taste für weiter\n");
getch();
// ----------------------------------------------------
// Einen Wert von den Eingängen lesen (Eingang 32-64)
data = (unsigned long) DapiDIGet32(handle, 32);
// Chan Start = 32
printf("Eingang 32-64 : 0x%x\n", data);
printf("Taste für weiter\n");
getch();
```

5.3.5. DapiDIGet64

Beschreibung
Dieser Befehl liest gleichzeitig 64 digitale Eingänge.

Definition
ULONGLONG DapiDIGet64(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 64, ..)

Return-Wert
Zustand der gelesen Eingänge

5.3.6. DapiDIGetFF32

Beschreibung

Dieser Befehl liest die Flip-Flops der Eingänge aus und setzt diese zurück (Eingangszustands-Änderung).

Definition

ULONG DapiDIGetFF32(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 32, 64, ..)

Return-Wert

Zustand von 32 Eingangszustandsänderungen

5.3.7. DapiDIGetCounter

Beschreibung

Dieser Befehl liest den Eingangszähler eines digitalen Eingangs.

Definition

ULONG DapiDIGetCounter(ULONG handle, ULONG ch, ULONG mode);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll

mode=0 (Normale Zählfunktion)

mode=DAPI_CNT_MODE_READ_WITH_RESET (Zähler auslesen und direktes Counter resetten)

mode=DAPI_CNT_MODE_READ_LATCHED (Auslesen des gespeicherten Zählerstandes)

Return-Wert

Angabe des Zählerwertes

Programmierbeispiel

```
value = DapiDIGetCounter(handle, 0 , 0);
// Zähler von DI Chan 0 wird gelesen

value = DapiDIGetCounter(handle, 1 , 0);
// Zähler von DI Chan 1 wird gelesen

value = DapiDIGetCounter(handle, 8 ,0);
// Zähler von DI Chan 8 wird gelesen

value = DapiDIGetCounter(handle, 0 , DAPI_CNT_MODE_READ_WITH_RESET);
// Zähler von DI Chan 0 wird gelesen UND resettet

value = DapiDIGetCounter(handle, 1 , DAPI_CNT_MODE_READ_LATCHED);
// Auslesen des gespeicherten Zählerstandes von DI Chan 1
```

5.3.8. DapiSpecialCounterLatchAll

Beschreibung

Dieser Befehl speichert die Zählerstände aller Eingangszähler gleichzeitig in ein Zwischenspeicher (Latch).

So können anschließend alle Zählerstände des Latches nacheinander ausgelesen werden.

Besonderheit hierbei ist, dass ein gleichzeitiges "Einfrieren" der Zählerstände möglich ist und die Eingefrorenen Stände (Latch) dann einzeln nacheinander ausgelesen werden können.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_COUNTER, DAPI_SPECIAL_COUNTER_LATCH_ALL, 0, 0);

Parameter

Keine

Return-Wert

Keiner

Programmierbeispiel

```
DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_COUNTER,
DAPI_SPECIAL_COUNTER_LATCH_ALL, 0, 0);
```

5.3.9. DapiSpecialCounterLatchAllWithReset

Beschreibung

Dieser Befehl speichert die Zählerstände aller Eingangszähler gleichzeitig in ein Zwischenspeicher (Latch).

Zusätzlich werden die Zählerstände der Eingangszähler im Anschluß resettet.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_COUNTER, DAPI_SPECIAL_COUNTER_LATCH_ALL_WITH_RESET, 0, 0);

Parameter

Keine

Return-Wert

Keiner

Programmierbeispiel

```
DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_COUNTER,
DAPI_SPECIAL_COUNTER_LATCH_ALL_WITH_RESET, 0, 0);
```

5.3.10. Dapi_Special_DI_FF_Filter_Value_Get

Beschreibung

Dieser Befehl gibt den Filter [ms] zurück, in welchem Zeitintervall digitale Eingangskanäle abgetastet werden.

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI, DAPI_SPECIAL_DI_FF_FILTER_VALUE_GET, 0, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert

Zeit [ms]

Programmierbeispiel

```
value = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FF_FILTER_VALUE_GET, 0, 0);
//Gibt das Zeitintervall zum Abtasten der digitalen Eingangskanäle zurück.
```

5.3.11. Dapi_Special_DI_FF_Filter_Value_Set

Beschreibung

Dieser Befehl setzt einen Filter [ms], in welchem Zeitintervall digitale Eingangskanäle abgetastet werden.

Definition

void DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI, DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET, ULONG time_ms, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

time_ms=Zeitintervall [ms], indem digitale Eingangskanäle abgetastet werden.

Return-Wert

Keiner.

Bemerkung

Dieser Befehl unterstützt nur Impulszeiten zwischen 5ms und 255ms.

Wird keine Zeit gesetzt, ist der Default-Wert 100ms.

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET, 5, 0);
// Setzt das Zeitintervall auf 5ms
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET, 150, 0);
// Setzt das Zeitintervall auf 150ms
```

5.4. Digitale Ausgänge verwalten

5.4.1. DapiDOSet1

Beschreibung
Dieser Befehl setzt einen einzelnen Ausgang.

Definition
void DapiDOSet1(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des zu setzenden Ausgangs an (0 ..)
data=Gibt den Datenwert an, der geschrieben wird (0 / 1)

Return-Wert
Keiner

5.4.2. DapiDOSet8

Beschreibung
Dieser Befehl setzt gleichzeitig 8 digitale Ausgänge.

Definition
void DapiDOSet8(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 8, 16, 24, 32, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

5.4.3. DapiDOSet16

Beschreibung
Dieser Befehl setzt gleichzeitig 16 digitale Ausgänge.

Definition
void DapiDOSet16(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 16, 32, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

5.4.4. DapiDOSet32

Beschreibung

Dieser Befehl setzt gleichzeitig 32 digitale Ausgänge.

Definition

void DapiDOSet32(ULONG handle, ULONG ch, ULONG data);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 32, 64, ..)

data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert

Keiner

Programmierbeispiel

```
// Einen Wert auf die Ausgänge schreiben
data = 0x0000ff00; // Ausgänge 9-16 werden auf 1 gesetzt
DapiDOSet32(handle, 0, data); // Chan Start = 0
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
// ----------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 0x80000000; // Ausgang 32 wird auf 1 gesetzt
DapiDOSet32(handle, 0, data); // Chan Start = 0
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
// ----------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 0x80000000; // Ausgang 64 wird auf 1 gesetzt
DapiDOSet32(handle, 32, data); // Chan Start = 32
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
```

5.4.5. DapiDOSet64

Beschreibung
Dieser Befehl setzt gleichzeitig 64 digitale Ausgänge.

Definition
void DapiDOSet64(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 64, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

5.4.6. DapiDOSet1_WithTimer

Beschreibung

Diese Funktion setzt einen Digitalausgang (ch) auf einen Wert (data - 0 oder 1) für eine bestimmte Zeit in ms.

Definition

void DapiDOSet1_WithTimer(ULONG handle, ULONG ch, ULONG data, ULONG time_ms);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 32, 64, ..)

data=Gibt die Datenwerte an, die geschrieben werden

time_ms=Gibt die Zeit an, in der der Ausgang gesetzt wird [ms]

Return-Wert

Keiner

Bemerkung:

Dieser Befehl wird nur von unserem RO-O8-R8 Modul unterstützt.

Dieser Befehl verliert seine Gültigkeit, sofern er mit anderen Werten überschrieben wird.

Möchte man den Befehl deaktivieren, dann muss er mit time_ms=0 überschrieben werden.

Programmierbeispiel

```
DapiDOSet1_WithTimer(handle, 2, 1, 1000);
//Setting channel 2 for 1000msec to 1
```

5.4.7. DapiDOReadback32

Beschreibung
Dieser Befehl liest die 32 digitalen Ausgänge zurück.

Definition
ULONG DapiDOReadback32(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem zurückgelesen werden soll (0, 32, 64, ..)

Return-Wert
Zustand von 32 Ausgängen.

5.4.8. DapiDOReadback64

Beschreibung
Dieser Befehl liest die 64 digitalen Ausgänge zurück.

Definition
ULONGLONG DapiDOReadback64(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem zurückgelesen werden soll (0, 64, ..)

Return-Wert
Zustand von 64 Ausgängen.

5.5. Digitale Zähler Funktionen

5.5.1. DapiCnt48ModeSet

Beschreibung

Dieser Befehl setzt einen Zählmodus (optional auch Submodus) und Eingangsfilter für einen bestimmten Eingangszählerkanal.

Definition

void DapiCnt48ModeSet(ULONG handle, ULONG ch, ULONG mode);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Nummer des Eingangszählerkanals, dessen Modus gesetzt werden soll (0, 1, 2, 3, ..)

mode=Gibt den Modus an

Mögliche Werte für mode

mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE | DAPI_CNT48_SUBMODE_NO_RESET

In diesem Modus wird bei steigender Flanke gezählt.

mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE | DAPI_CNT48_SUBMODE_RESET_WITH_READ

In diesem Modus wird bei steigender Flanke gezählt. Zusätzlich wird bei jedem Lesevorgang der Zähler resettet.

mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE | DAPI_CNT48_SUBMODE_RESET_ON_CH_7

In diesem Modus wird bei steigender Flanke gezählt. Zusätzlich kann der Zähler über ein externes Signal (letzer Kanal des Moduls = 1) resettet werden.

mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE | DAPI_CNT48_SUBMODE_LATCH_COMMON

Mit dem Befehl "DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_LATCH_GROUP8, 0, 0)" werden alle 8 Zählerstände der Eingangszähler gleichzeitig in einen Zwischenspeicher (Latch) geschrieben. Mit diesem Modus kann dann der gelatchte Zählerstand ausgelesen werden.

mode=DAPI_CNT48_MODE_T

Mit diesem Modus wird die Periodendauer T gemessen. Als Basis hierbei dient ein 100 MHz Zähler.

mode=DAPI_CNT48_MODE_FREQUENCY

Bei diesem Modus lässt sich die Anzahl der steigenden Flanken innerhalb einer Sekunde (=Frequenz) messen.

mode=DAPI_CNT48_MODE_PWM

Mit diesem Modus werden die "high" und "low" Zeit eines Signals gemessen. Dadurch kann anschließend das Verhältnis bestimmt werden (PWM).

Zusätzlich können alle Eingangszähler mit einem Eingangsfilter (mit einer oder-Verknüpfung) kombiniert werden. Hierzu stehen folgende Eingangsfilter zur Verfügung:

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-Wert

Keiner

Bemerkung

Dieser Befehl wird nur von unserem Modul RO-CNT8 unterstützt.

Programmierbeispiel

```
DapiCnt48ModeSet(handle, 0, DAPI_CNT48_MODE_COUNT_RISING_EDGE |
DAPI_CNT48_SUBMODE_RESET_WITH_READ | DAPI_CNT48_FILTER_20ns);
//Eingangszählerkanal 0 zählt alle Impulse <= 20ns bei steigender Flanke.
Zusätzlich wird der Zähler nach Abfrage resettet.
DapiCnt48ModeSet(handle, 1, DAPI_CNT48_MODE_COUNT_RISING_EDGE |
DAPI_CNT48_SUBMODE_RESET_ON_CH_7 | DAPI_CNT48_FILTER_500us);
//Eingangszählerkanal 1 zählt alle Impulse <= 500us bei steigender Flanke.
Dieser Zähler kann mit einem externen Signal (ch7 = 1) resettet werden.
DapiCnt48ModeSet(handle, 2, DAPI_CNT48_MODE_PWM | DAPI_CNT48_FILTER_5ms);
//Eingangszählerkanal 2 misst alle low-/high Zeiten <= 5ms. Anschließend wird
das Verhältnis bestimmt (PWM).
```

5.5.2. DapiCnt48ModeGet

Beschreibung

Dieser Befehl liest den Zählmodus eines bestimmten Eingangszählerkanals zurück.

Definition

ULONG DapiCnt48ModeGet(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Nummer des Eingangszählerkanals, dessen Modus ausgegeben werden soll (0, 1, 2, 3, ..)

Return-Wert

Zählmodus des Eingangszählerkanals.

(Nähere Informationen / Beschreibung der Bits -> siehe delib.h oder Manual "RO-Registerbelegung")

Bemerkung

Dieser Befehl wird nur von unserem Modul RO-CNT8 unterstützt.

Programmierbeispiel

```
value = DapiCnt48ModeGet(handle, 0)
//Gibt den Zählmodus von Eingangszählerkanal 0 zurück
value = DapiCnt48ModeGet(handle, 3)
//Gibt den Zählmodus von Eingangszählerkanal 3 zurück
```

5.5.3. DapiCnt48CounterGet32

Beschreibung

Dieser Befehl liest die ersten 32 Bit eines 48 Bit Eingangszählers.

Definition

ULONG DapiCnt48CounterGet32(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangszählerkanals an, der gelesen werden soll (0, 1, 2, 3, ..)

Return-Wert

Ausgabe des Zählerwertes.

Bemerkung

Dieser Befehl wird nur von unseren Modulen RO-CNT8 und RO-CNT/IGR unterstützt.

Programmierbeispiel

```
value = DapiCnt48CounterGet32(handle, 0);
//gibt den Wert von Eingangszählerkanal 0 aus
value = DapiCnt48CounterGet32(handle, 3);
//gibt den Wert von Eingangszählerkanal 3 aus
```

5.5.4. DapiCnt48CounterGet48

Beschreibung

Dieser Befehl liest einen 48 Bit Zähler eines Eingangszählerkanals.

Definition

ULONGLONG DapiCnt48CounterGet48(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangszählerkanals an, der gelesen werden soll (0, 1, 2, 3, ..)

Return-Wert

Ausgabe des Zählerwertes.

Bemerkung

Dieser Befehl wird nur von unseren Modulen RO-CNT8 und RO-CNT/IGR unterstützt.

Programmierbeispiel

```
value = DapiCnt48CounterGet48(handle, 0);
//gibt den Wert von Eingangszählerkanal 0 aus
value = DapiCnt48CounterGet48(handle, 3);
//gibt den Wert von Eingangszählerkanal 3 aus
```

5.5.5. DapiSpecialCNT48ResetSingle

Beschreibung

Dieser Befehl resettet den Zählerstand eines einzelnen Eingangszählers.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_RESET_SINGLE, ULONG ch, 0)

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangszählers an, dessen Zählerstand resettet werden soll (0, 1, 2, ..)

Return-Wert

Keiner

Bemerkung

Dieser Befehl wird nur von unserem Modul RO-CNT8 unterstützt.

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_RESET_SINGLE, 0, 0)
// Zählerstand von Eingangszähler 0 wird resettet
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_RESET_SINGLE, 1, 0)
// Zählerstand von Eingangszähler 1 wird resettet
```

5.5.6. DapiSpecialCNT48ResetGroup8

Beschreibung

Dieser Befehl resettet gleichzeitig die Zählerstände von 8 Eingangszählern.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_RESET_GROUP8, ULONG ch, 0)

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangszählers an, ab dem die Zählerstande von 8 Eingangszählern resettet werden (0, 8, 16, ...)

Return-Wert

Keiner

Bemerkung

Dieser Befehl wird nur von unserem Modul RO-CNT8 unterstützt.

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_RESET_GROUP8, 0, 0)
// Zählerstände der Eingangszähler 0-7 werden resettet
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_RESET_GROUP8, 8, 0)
// Zählerstände der Eingangszähler 8-15 werden resettet
```

5.5.7. DapiSpecialCNT48LatchGroup8

Beschreibung

Dieser Befehl speichert die Zählerstände von 8 Eingangszähler gleichzeitig in ein Zwischenspeicher (Latch). So können anschließend alle Zählerstände des Latches nacheinander ausgelesen werden.

Besonderheit hierbei ist, dass ein gleichzeitiges "Einfrieren" der Zählerstände möglich ist und die eingefrorenen Stände (Latch) dann (langsam) einzeln nacheinander ausgelesen werden können.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_LATCH_GROUP8, ULONG ch, 0)

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangszählers an, ab dem die Zählerstande von 8 Eingangszählern gelatched werden (0, 8, 16, ...)

Return-Wert

Keiner

Bemerkung

Dieser Befehl wird nur von unserem Modul RO-CNT8 unterstützt.

Bitte beachten Sie, dass nur die Zählerstände der Eingangszähler gelatched werden, bei denen zuvor der Modus "DAPI_CNT48_SUBMODE_LATCH_COMMON" gesetzt wurde. (-> DapiCnt48ModeSet)

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_LATCH_GROUP8, 0, 0)
// Zählerstände der Eingangszähler 0-7 werden gelatched
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_LATCH_GROUP8, 8, 0)
// Zählerstände der Eingangszähler 8-15 werden gelatched
```

5.6. Pulsgenerator Ausgänge verwalten

5.6.1. DapiPulseGenSet

Beschreibung

Dieser Befehl generiert eine gewisse Anzahl an Impulsen mit vorgegebenen low- und high-Zeiten.

Definition

void DapiPulseGenSet(ULONG handle, ULONG ch, ULONG mode, ULONG par0, ULONG par1, UNLONG par2);

Parameter

handle = Dies ist das Handle eines geöffneten Moduls
ch = Gibt die Nummer des zu setzenden Ausgangs an (0, 1, 2, ..)
mode = Modus, mit dem Impulse generiert werden (muss immer 0 sein).
par0 = Anzahl der zu erzeugenden Impulse (par0 = 0 -> unendlich viele Impulse)
par1 = Low-Zeit des Impulses (t[ns] / 10 - 1)
par2 = High-Zeit des Impulses (t[ns] / 10 - 1)

Beispiel für das Einstellen der low-/high-Zeit (Par1/Par2)

500ns -> 500 /10 - 1 = 49(dez)
7µ -> 7000 / 10 - 1 = 699(dez)
2,5ms -> 2500000 / 10 - 1 = 249,999(dez)

Return-Wert

keiner

Programmierbeispiel

```
DapiPulseGenSet(handle, 0, 0, 10, 29, 69);
// generiert 10 Impulse an Pulsgenerator-Ausgang 0 mit einer low-Zeit von 300ns
und einer high-Zeit von 700ns.
DapiPulseGenSet(handle, 3, 0, 100, 7999, 9999);
// generiert 100 Impulse an Pulsgenerator-Ausgang 3 mit einer low-Zeit von 80µs
und einer high-Zeit von 100µs.
DapiPulseGenSet(handle, 10, 0, 0, 249999, 299999);
// generiert unendlich viele Impulse an Pulsgenerator-Ausgang 10 mit einer low-
Zeit von 2,5ms und einer high-Zeit von 3ms.
```

5.7. A/D Wandler Funktionen

5.7.1. DapiADSetMode

Beschreibung
Dieser Befehl konfiguriert den Spannungsbereich für einen A/D Wandler.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des A/D Wandlers an (0 ..)
mode=Gibt den Modus für den Kanal an

Return-Wert
keiner

Bemerkung
Folgende Modi werden unterstützt:
(diese sind abhängig von dem verwendeten A/D-Modul)

Unipolare Spannungen:
ADDA_MODE_UNIPOL_10V
ADDA_MODE_UNIPOL_5V
ADDA_MODE_UNIPOL_2V5

Bipolare Spannungen:
ADDA_MODE_BIPOL_10V
ADDA_MODE_BIPOL_5V
ADDA_MODE_BIPOL_2V5

Ströme:

ADDA_MODE_0_20mA
ADDA_MODE_4_20mA
ADDA_MODE_0_24mA
ADDA_MODE_0_25mA
ADDA_MODE_0_50mA

5.7.2. DapiADGetMode

Beschreibung

Dieser Befehl liest den eingestellten Modus eines A/D Wandlers zurück. Modus-Beschreibung siehe DapiADSetMode.

Definition

ULONG DapiADGetMode(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des A/D Wandlers an (0 ..)

Return-Wert

Modus des A/D Wandlers

5.7.3. DapiADGet

Beschreibung
Dieser Befehl liest einen Datenwert von einen Kanal eines A/D Wandlers.

Definition
ULONG DapiADGet(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des A/D Wandlers an (0 ..)

Return-Wert
Wert vom A/D Wandler in Digits

5.7.4. DapiADGetVolt

Beschreibung
Dieser Befehl liest einen Datenwert von einen Kanal eines A/D Wandlers in Volt.

Definition
float DapiADGetVolt(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des A/D Wandlers an (0 ..)

Return-Wert
Wert vom A/D Wandler in Volt

5.7.5. DapiADGetmA

Beschreibung
Dieser Befehl liest einen Datenwert von einen Kanal eines A/D Wandlers in mA.

Definition
float DapiADGetmA(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des A/D Wandlers an (0 ..)

Return-Wert
Wert vom A/D Wandler in mA.

Bemerkung
Dieser Befehl ist Modul abhängig. Er funktioniert natürlich nur, wenn das Modul auch den Strom-Modus unterstützt.

5.8. D/A Ausgänge verwalten

5.8.1. DapiDASetMode

Beschreibung

Dieser Befehl übergibt ein Datenwert an einen Kanal eines 16-Bit D/A Wandlers.

Definition

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

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt den Kanal des D/A Wandlers an (0 ..)

data=Gibt den Datenwert an, der geschrieben wird (16-Bit Datenwert -> Datenwertebereich: 0-65535)

Return-Wert

Keiner

Programmierbeispiel

```
DapiDASet(handle, 0, 65535);
// Setzt den 1.Ausgang des D/A Wandlers auf maximalen Wert des gewählten Modus.
//(bei ausgewähltem Modus ADDA_MODE_UNIPOL_10V wird der 1. Ausgang des D/A
Wandlers auf 10V gesetzt)
```

5.8.2. DapiDAGetMode

Beschreibung
Dieser Befehl liest den eingestellten Modus eines D/A Wandlers zurück.

Definition
ULONG DapiDAGetMode(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des D/A Wandlers an (0 ..)

Return-Wert
Modus des D/A Wandlers

5.8.3. DapiDASet

Beschreibung

Dieser Befehl übergibt ein Datenwert an einen Kanal eines 16-Bit D/A Wandlers.

Definition

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

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt den Kanal des D/A Wandlers an (0 ..)

data=Gibt den Datenwert an, der geschrieben wird (16-Bit Datenwert -> Datenwertebereich: 0-65535)

Return-Wert

Keiner

Programmierbeispiel

```
DapiDASet(handle, 0, 65535);
// Setzt den 1.Ausgang des D/A Wandlers auf maximalen Wert des gewählten Modus.
//(bei ausgewähltem Modus ADDA_MODE_UNIPOL_10V wird der 1. Ausgang des D/A
Wandlers auf 10V gesetzt)
```

5.8.4. DapiDASetVolt

Beschreibung
Dieser Befehl setzt eine Spannung an einen Kanal eines D/A Wandlers.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des D/A Wandlers an (0 ..)
data=Gibt die Spannung an, die eingestellt werden soll [V]

Return-Wert
Keiner

Programmierbeispiel

```
DapiDASetVolt(handle, 0, 5,4321);
// Setzt den 1. Ausgang des D/A Wandlers auf 5,4321 V
```

5.8.5. DapiDASetmA

Beschreibung
Dieser Befehl setzt einen Strom an einen Kanal eines D/A Wandlers.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des D/A Wandlers an (0 ..)
data=Gibt den Strom an, der geschrieben wird [mA]

Return-Wert
Keiner

Bemerkung
Dieser Befehl ist Modul abhängig. Er funktioniert natürlich nur, wenn das Modul auch den Strom-Modus unterstützt.

5.8.6. DapiSpecialCmd_DA

Beschreibung

Dieser Befehl setzt die Spannungswerte bei einem Kanal beim Einschalten bzw. nach einem Timeout eines D/A Wandlers (EEPROM-Konfiguration).

Definition

void DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA, cmd, ch, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt den Kanal des D/A Wandlers an (0, 1, 2, ..)

Zurücksetzen der Einstellungen auf Default Konfiguration

cmd=DAPI_SPECIAL_DA_PAR_DA_LOAD_DEFAULT

Speichern der Konfiguration in das EEPROM

cmd=DAPI_SPECIAL_DA_PAR_DA_SAVE_EEPROM_CONFIG

Laden der Konfiguration aus dem EEPROM

cmd=DAPI_SPECIAL_DA_PAR_DA_LOAD_EEPROM_CONFIG

Return-Wert

Keiner

Bemerkung

DAPI_SPECIAL_CMD_DA_PAR_DA_LOAD_DEFAULT

Mit diesem Befehl wird die Default Konfiguration eines D/A Wandlers geladen. Der D/A Wandler hat jetzt als Ausgabespannung 0V.

DAPI_SPECIAL_DA_PAR_DA_SAVE_EEPROM_CONFIG

Mit diesem Befehl wird die aktuelle D/A Wandler Einstellung (Spannung/Strom-Wert, Enable/Disable und D/A Wandler Modus) in das EEPROM gespeichert.

DAPI_SPECIAL_DA_PAR_DA_LOAD_EEPROM_CONFIG

Mit diesem Befehl wird der D/A Wandler, mit der im EEPROM gespeicherten Konfiguration, gesetzt.

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA,
DAPI_SPECIAL_DA_PAR_DA_LOAD_DEFAULT, 1, 0);
```

```
//Zurücksetzen der EEPROM-Konfiguration auf Default Konfiguration bei Kanal 1.

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA,
DAPI_SPECIAL_DA_PAR_DA_SAVE_EEPROM_CONFIG, 3, 0);
//Speichern der D/A Wandler Einstellungen in das EEPROM bei Kanal 3.

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA,
DAPI_SPECIAL_DA_PAR_DA_LOAD_EEPROM_CONFIG, 2, 0);
//Setzen des D/A Wandlers, mit der im EEPROM gespeicherten Konfiguration bei
Kanal 2.
```

5.9. PT100 Funktionen

5.9.1. DapiTempGet

Beschreibung
Dieser Befehl liest einen Temperatur Eingang.

Definition
float DapiTempGet(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, der gelesen werden soll (0, 1, 2, 3, ..)

Return-Wert
Temperatur [°C]

Programmierbeispiel

```
ret=DapiTempGet(handle, 0)
//gibt die Temperatur von Kanal 0 zurück
```

5.10. Schrittmotoren Funktionen

5.10.1. Befehle mit DapiStepperCommand

5.10.1.1. DAPI_STEPPER_CMD_GO_POSITION

Beschreibung

Hiermit wird eine bestimmte Position angefahren. Dieses Kommando darf nur ausgeführt werden, wenn der Motor nicht "disabled" ist und kein Go_Position oder Go_Referenz ausgeführt wird.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_POSITION, position, 0, 0, 0);

Programmierbeispiel

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_POSITION, go_pos_par, 0,
0, 0);
```

5.10.1.2. DAPI_STEPPER_CMD_GO_POSITION_RELATIVE

Beschreibung

Hiermit wird eine relative Position angefahren. Im Gegensatz zum Befehl GO_POSITION, der eine absolute Position anfährt, wird hier die momentane Position berücksichtigt. Dieses Kommando darf nur ausgeführt werden, wenn der Motor nicht "disabled" ist und kein Go_Position oder Go_Referenz ausgeführt wird.

Definition

void DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_POSITION_RELATIVE, go_pos_rel_par, 0, 0, 0);

Programmierbeispiel

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_POSITION_RELATIVE, 100,
0, 0, 0);
//Motor fährt, von der aktuellen Position aus gesehen, 100 Schritte nach
rechts.
```

5.10.1.3. DAPI_STEPPER_CMD_SET_POSITION

Beschreibung

Dieses Kommando dient zum setzten der Motorposition. Die Auflösung beträgt 1/16 Vollschritt. Dieses Kommando darf nur bei angehaltenem Motor verwendet werden.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_POSITION, par1, 0, 0, 0);

Parameter

par1 = Motorposition

5.10.1.4. DAPI_STEPPER_CMD_SET_FREQUENCY

Beschreibung

Dieses Kommando dient zur Einstellung der Motorsollfrequenz. Die Motorfrequenzregelung übernimmt dabei die Einhaltung der Beschleunigungs- / Bremsrampe. Schrittverluste treten nicht auf. Die Motorsollfrequenz ist bezogen auf Vollschrittbetrieb. Über das Vorzeichen wird die Richtung ausgewählt. Die Motorsollfrequenz darf nicht über der Maxfrequenz liegen, ansonsten wird das Kommando abgelehnt.

Bei geschlossenem Endschalter1 läßt sich nur in positive Richtung verfahren, bei geschlossenem Endschalter2 läßt sich nur in negative Richtung verfahren, ansonsten wird das Kommando abgelehnt.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_FREQUENCY, par1, 0, 0, 0);

Parameter

par1 = Motorsollfrequenz [Hz]

5.10.1.5. DAPI_STEPPER_CMD_GET_FREQUENCY

Beschreibung

Dieses Kommando dient zum Abfragen der Motorfrequenz. Dieses Kommando darf immer verwendet werden.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GET_FREQUENCY, par1, 0 ,0 ,0);

Return-Wert

Motorfrequenz [Hz]

5.10.1.6. DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY

Beschreibung

Dieses Kommando dient zur Einstellung der Motorfrequenz. Die Motorfrequenzregelung übernimmt dabei keine Funktion. Für die Einhaltung der Beschleunigungs- / Bremsrampe ist der Anwender verantwortlich. Schrittverluste können bei Nichteinhaltung auftreten.

Die Motorfrequenz ist bezogen auf Vollschrittbetrieb. Über das Vorzeichen wird die Richtung ausgewählt.

Die Frequenz darf nicht über der Maxfrequenz liegen.

Definition

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY, par1, 0 ,0 ,0);

Parameter

par1 = Motorfrequenz [Hz]

5.10.1.7. DAPI_STEPPER_CMD_STOP

Beschreibung

Dieses Kommando dient zum Anhalten des Motors, die Bremsrampe wird dabei eingehalten.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_STOP, 0, 0, 0, 0);

5.10.1.8. DAPI_STEPPER_CMD_FULLSTOP

Beschreibung

Dieses Kommando dient zum sofortigen Anhalten des Motors, die Bremsrampe wird dabei nicht eingehalten. Die Motorposition kann vielleicht danach nicht mehr stimmen, da der Motor unkontrolliert angehalten wird.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_FULLSTOP, 0, 0, 0, 0);

Programmierbeispiel

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_FULLSTOP, 0, 0, 0, 0);
```

5.10.1.9. DAPI_STEPPER_CMD_DISABLE

Beschreibung

Dieses Kommando dient zum disablen/enabeln des Motors, der Motor verfährt dann nicht mehr/oder wieder. Dieses Kommando darf nur bei Motorstillstand benutzt werden.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_DISABLE, par1, 0, 0, 0);

Parameter

par1 = Disablemode (0=Normale Funktion / 1=Disable)

5.10.1.10. DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC

Beschreibung

Hiermit werden neue Motor Konfigurationen gesetzt.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC, par1, par2, 0, 0);

Parameter

Parameter-Stepmode setzen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE
par2=0 (Vollschrittbetrieb)
par2=1 (Halbschrittbetrieb)
par2=2 (Viertelschrittbetrieb)
par2=3 (Achtelschrittbetrieb)
par2=4 (Sechzehntelschrittbetrieb)

Parameter-GO-Frequency setzen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY
par2=Geschwindigkeit [Vollschritt / s] - bezogen auf Vollschritt Frequenz - (Maximalwert=5000)

Parameter-Start-Frequency setzen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY
par2=Startfrequenz [Vollschritt / s] - bezogen auf Vollschritt Frequenz - (Maximalwert=5000)

Parameter-Stop-Frequency setzen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY
par2=Stopfrequenz [Vollschritt / s] - bezogen auf Vollschritt Frequenz - (Maximalwert=5000)

Parameter-Max-Frequency setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY
par2=Maximale Frequenz [Vollschritt / s] - bezogen auf Vollschritt Frequenz - (Maximalwert=5000)

Parameter-Accelerationslope setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE
par2=Beschleunigungsrampe [Vollschritt / 10ms] - (Maximalwert=1000)

Parameter-Decelerationslope setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE
par2= Bremsrampe [Vollschritt / 10ms] - (Maximalwert=1000)

Parameter-Phasecurrent setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT
par2=Phasenstrom [mA] - (Maximalwert = 1500)

Parameter-Hold-Phasecurrent setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT
par2=Phasenstrom bei Motorstillstand [mA] - (Maximalwert=1500)

Parameter-Hold-Time setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME
par2=Zeit in der der Haltestrom fließt nach Motorstop [ms]
par2=-1 / FFFF hex / 65535 dez (Zeit unendlich)

Parameter-Status-LED-Mode setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE
par2=Betriebsart der Status-LED
par2=0 = (MOVE - LED leuchtet bei Motorbewegung)
par2=1 = (HALT - LED leuchtet bei Motorstillstand)
par2=2 = (ENDSW1 - LED leuchtet bei geschlossenen Endschalter1)
par2=3 = (ENDSW2 - LED leuchtet bei geschlossenen Endschalter2)
par2=4 = (REFSW1 - LED leuchtet bei geschlossenen Referenzschalterschalter1)
par2=5 = (REFSW2 - LED leuchtet bei geschlossenen Referenzschalterschalter2)

Parameter-Invert-END-Switch1 setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1
par2=Invertiere Funktion des Endschalter1 (0=normal / 1=invertieren)

Parameter-Invert-END-Switch2 setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2
par2=Invertiere Funktion des Endschalter2 (0=normal / 1=invertieren)

Parameter-Invert-Ref-Switch1 setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1
par2=Invertiere Funktion des Referenzschalterschalter1 (0=normal / 1=invertieren)

Parameter-Invert-Ref-Switch2 setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2
par2=Invertiere Funktion des Referenzschalterschalter2 (0=normal / 1=invertieren)

Parameter-Invert-direction setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION
par2=Invertiere alle Richtungsangaben (0=normal / 1=invertieren)

Parameter-Endswitch-Stopmode setzen
par1= DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE
par2=Einstellen des Stopverhaltens (0=Fullstop / 1=Stop)

Parameter-GoReferenceFrequency setzen (ACHTUNG: Dieser Parameter wird nicht mehr unterstützt!)
par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY
Bemerkung: Dieser Parameter wird durch die nachfolgenden drei Parametern vollständig ersetzt.

Parameter-GoReferenceFrequnecyToEndSwitch setzen
par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH
par2=Geschwindigkeit, mit der der Enschalter angefahren wird (Frequenz [Vollschritt / s] - (Maximalwert=5000))

Parameter GoReferenceFrequencyAfterEndSwitch setzen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERENDSWITCH

par2=Geschwindigkeit, mit der vom Enschalter abgefahren wird (Frequenz [Vollschritt / s] - (Maximalwert=5000))

Parameter GoReferenceFrequencyToOffset setzen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSET

par2=Geschwindigkeit, mit der der optionale Offset angefahren wird (Frequenz [Vollschritt / s] - (Maximalwert=5000))

Programmierbeispiel

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE, 4,0,0);
// Schrittmode (Voll-, Halb-, Viertel-, Achtel-, Sechszehntelschritt)

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY, 1000,0,0);
// Schrittmode (Voll-, Halb-, Viertel-, Achtel-, Sechszehntelschritt)

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY, 100,0,0);
// Startfrequenz [Vollschritt / s]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY, 100,0,0);
// Stopfrequenz [Vollschritt / s]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY, 3500,0,0);
// maximale Frequenz [Vollschritt / s]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE, 20,0,0);
// Beschleunigun in [Vollschritten / ms]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE, 20,0,0);
// Bremsung in [Vollschritten / ms]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT, 750,0,0);
// Phasenstrom [mA]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT, 500,0,0);
// Phasenstrom bei Motorstillstand [mA]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME, 15000,0,0);
```

```
// Zeit in der der Haltestrom fließt nach Motorstop [s]

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE, 0,0,0);
// Betriebsart der Status-LED

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1, 0,0,0);
// invertiere Funktion des Endschalter1

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2, 0,0,0);
// invertiere Funktion des Endschalter2

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1, 0,0,0);
// invertiere Funktion des Referenzschalterschalter1

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2, 0,0,0);
// invertiere Funktion des Referenzschalterschalter2

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION, 0,0,0);
// invertiere alle Richtungsangaben

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE, 0,0,0);
// einstellen des Stopverhaltens

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH, 100,0,0);
// Einstellung der Geschwindigkeit, mit der zum Endschalter angefahren wird.

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERENDSWITCH , 200,0,0);
// Einstellung der Geschwindigkeit, mit der vom Endschalter abgefahren wird.

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSET, 300,0,0);
// Einstellung der Geschwindigkeit, mit der zum optionalen Offset angefahren
wird.
```

5.10.1.11. DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC

Beschreibung

Hiermit wird der Motorspezifische Parameter ausgelesen. Dieses Kommando darf immer benutzt werden. Es teilt sich in Unterkommandos auf, die analog den Parametern von DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC sind.

Definition

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC, par1, 0, 0, 0);

Parameter

Parameter-Stepmode abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE

Parameter-GO-Frequency abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY

Parameter-Start-Frequency abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY

Parameter-Stop-Frequency abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY

Parameter-Max-Frequency abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY

Parameter-Accelerationslope abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE

Parameter-Decelerationslope abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE

Parameter-Phasecurrent abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT

Parameter-Hold-Phasecurrent abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT

Parameter-Hold-Time abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME

Parameter-Status-LED-Mode abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE

Parameter-Invert-END-Switch1 abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1

Parameter-Invert-END-Switch2 abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2

Parameter-Invert-Ref-Switch1 abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1

Parameter-Invert-Ref-Switch2 abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2

Parameter-Invert-direction abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION

Parameter-Endswitch-Stopmode abfragen
par1= DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE

Parameter-GoReferenceFrequency abfragen (ACHTUNG: Dieser Parameter wird nicht mehr unterstützt!)
par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY
Bemerkung: Dieser Parameter wird durch die nachfolgenden drei Parametern vollständig ersetzt.

Parameter-GoReferenceFrequnecyToEndSwitch abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH

Parameter GoReferenceFrequencyAfterEndSwitch abfragen
par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERENDSWITCH

Parameter GoReferenceFrequencyToOffSet abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSET

Return-Wert

Parameter-Stepmode ablesen

par1=DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE
return=0 (Vollschrittbetrieb)
return=1 (Halbschrittbetrieb)
return=2 (Viertelschrittbetrieb)
return=3 (Achtelschrittbetrieb)
return=4 (Sechzehntelschrittbetrieb)

Parameter-GO-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY
return=Geschwindigkeit [Vollschritt / s] - bezogen auf Vollschritt

Parameter-Start-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY
return=Startfrequenz [Vollschritt / s]

Parameter-Stop-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY
return=Stopfrequenz [Vollschritt / s]

Parameter-Max-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY
return=maximale Frequenz [Vollschritt / s]

Parameter-Accelerationslope

par1=DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE
return=Beschleunigungsrampe [Vollschritten / ms]

Parameter-Decelerationslope

par1=DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE
return= Bremsrampe [Vollschritten / ms]

Parameter-Phasecurrent
par1=DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT
return=Phasenstrom [mA]

Parameter-Hold-Phasecurrent
par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT
return= Phasenstrom bei Motorstillstand [mA]

Parameter-Hold-Time
par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME
return=Zeit in der der Haltestrom fließt nach Motorstop [ms]
return=-1 / FFFF hex / 65535 dez (Zeit unendlich)

Parameter-Status-LED-Mode
par1=DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE
return=Betriebsart der Status-LED
return=0 (MOVE - LED leuchtet bei Motorbewegung)
return=1 (HALT - LED leuchtet bei Motorstillstand)
return=2 (ENDSW1 - LED leuchtet bei geschlossenen Endschalter1)
return=3 (ENDSW2 - LED leuchtet bei geschlossenen Endschalter2)
return=4 (REFSW1 - LED leuchtet bei geschlossenen Referenzschalterschalter1)
return=5 (REFSW2 - LED leuchtet bei geschlossenen Referenzschalterschalter2)

Parameter-Invert-END-Switch1
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1
return=Endschalter1 wird invertiert (0=normal / 1=invertieren)

Parameter-Invert-END-Switch2
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2
return=Endschalter2 wird invertiert (0=normal / 1=invertieren)

Parameter-Invert-Ref-Switch1
par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1
return=Referenzschalterschalter1 wird invertiert (0=normal / 1=invertieren)

Parameter-Invert-Ref-Switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2
return=Referenzschalterschalter2 wird invertiert (0=normal / 1=invertieren)

Parameter-Invert-direction

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION
return=Richtungsangaben werden invertiert (0=normal / 1=invertieren)

Parameter-Endswitch-Stopmode

par1= DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE
return=Einstellung des Stopverhaltens (0=Fullstop / 1=Stop)

Parameter-GoReferenceFrequnecyToEndSwitch

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH
return=Frequenz [Vollschritt / s]

Parameter GoReferenceFrequencyAfterEndSwitch abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERENDSWITCH
return=Frequenz [Vollschritt / s]

Parameter GoReferenceFrequencyToOffset abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSET
return=Frequenz [Vollschritt / s]

Programmierbeispiel

```
value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC, DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE,
0, 0, 0);
// Schrittmode (Voll-, Halb-, Viertel-, Achtel-, Sechszehntelschritt)

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY, 0, 0, 0);
// Schrittmode bei Motorstop (Voll-, Halb-, Viertel-, Achtel-,
Sechszehntelschritt)

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY, 0, 0, 0);
// Startfrequenz [Vollschritt / s]

value = DapiStepperCommand(handle, motor,
```

```
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY, 0, 0, 0);
// Stopfrequenz [Vollschritt / s]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY, 0, 0, 0);
// maximale Frequenz [Vollschritt / s]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE, 0, 0, 0);
// Beschleunigun in [Vollschritten / ms]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE, 0, 0, 0);
// Bremsung in [Vollschritten / ms]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT, 0, 0, 0);
// Phasenstrom [mA]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT, 0, 0, 0);
// Phasenstrom bei Motorstillstand [mA]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC, DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME,
0, 0, 0);
// Zeit in der der Haltestrom fließt nach Motorstop [s]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE, 0, 0, 0);
// Betriebsart der Status-LED

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1, 0, 0, 0);
// invertiere Funktion des Endschalter1

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2, 0, 0, 0);
// invertiere Funktion des Endschalter12

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1, 0, 0, 0);
// invertiere Funktion des Referenzschalterschalter1

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
```

```
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2, 0, 0, 0);
// invertiere Funktion des Referenzschalterschalter2

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION, 0, 0, 0);
// invertiere alle Richtungsangaben

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE, 0, 0, 0);
// einstellen des Stopverhaltens

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH, 0,0,0);
// Abfrage der Geschwindigkeit, mit der der Endschalter angefahren wird.

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERENDSWITCH, 0,0,0);
// Abfrage der Geschwindigkeit, mit der vom Endschalter abgefahren wird.

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSET, 0,0,0);
// Abfrage der Geschwindigkeit, mit der der optionale Offset angefahren wird.
```

5.10.1.12. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_SAVE

Beschreibung

Es wird die aktuelle Motorcharakteristik des Motors ins EEPROM abgespeichert.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_SAVE, 0, 0, 0, 0);

5.10.1.13. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_LOAD

Beschreibung

Es wird die Motorcharakteristik des Motors aus dem EEPROM geladen.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_LOAD, 0, 0, 0, 0);

5.10.1.14. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT

Beschreibung

Es wird die Motorcharakteristik des Motors auf Defaultwerte zurück gesetzt.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT, 0, 0, 0, 0);

Bemerkung

Die Defaultwerte sind folgende:

- Stepmode Vollschritt
- Schrittfrequenz bei GoPosition [Vollschritt / s]: 1000 Hz
- Startfrequenz [Vollschritt / s]: 200Hz
- Stopfrequenz [Vollschritt / s]: 200Hz
- Maximale Schrittfrequenz [Vollschritt / s]: 3000Hz
- Beschleunigungsrampe [Hz/10ms]: 10Hz/10ms
- Bremsrampe [Hz/10ms]: 10Hz/10ms
- Phasenstrom 0..1,5A [1mA]: 750mA
- Haltestorm 0..1,5A [1mA]: 500mA
- Haltezeit 0..unendlich [ms]: 15000ms
- Status_LEDfunktion: Move
- Funktion des Endschalter1: nicht invertiert
- Funktion des Endschalter2: nicht invertiert
- Funktion des Referenzschalter1: nicht invertiert
- Funktion des Referenzschalter2: nicht invertiert
- Funktion aller Richtungsangaben: nicht invertiert
- Endschaltermode: Fullstop
- Schrittfrequenz bei GoReferenz [Vollschritt / s]: 1000 Hz

5.10.1.15. DAPI_STEPPER_CMD_GO_REFSWITCH

Beschreibung

Der Motor fährt zur Referenzposition.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_REFSWITCH, par1, par2, par3, 0);

Parameter

Mögliche Werte für par1: (werden mehrere benötigt, müssen die einzelnen addiert werden)

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=Motorpositionsoffset (1/16 Vollschritt)

par3=Timeoutzeit [ms]

Bemerkung

Anfahren des Referenzschalters

Zunächst fährt der Motor zur Referenzposition 1 oder 2 (siehe par1).

Hierbei kann angegeben werden, ob der Referenzschalter 1 (DAPI_STEPPER_GO_REFSWITCH_PAR_REF1) oder der Referenzschalter 2 (DAPI_STEPPER_GO_REFSWITCH_PAR_REF2) angefahren wird. Dabei läßt sich die Richtung wählen in die der Motor startet. Mit dem Parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_NEGATIVE wird nach links und mit dem Parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_POSITIVE wird nach rechts gestartet.

Hierbei wird die Geschwindigkeit GOREFERENCEFREQUENCY_TOENDSWITCH benutzt (siehe DapiStepperCommand_SetMotorcharacteristic).

Herausfahren aus dem Referenzschalter

Danach fährt der Motor mit der Geschwindigkeit GOREFERENCEFREQUENCY_AFTERENDSWITCH aus der Referenzposition heraus. Dabei läßt sich wählen, ob der Motor die rechte oder linke Seite des Referenzschalters anfährt. Mit dem Parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_LEFT wird die linke Kante angefahren und mit dem Parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_RIGHT wird die rechte Kante angefahren.

Optionales Anfahren eines Offsets

Nach dem Herausfahren aus dem Referenzschalter kann noch ein Offset angefahren werden. Falls dieser Parameter nicht = 0 ist (par2), fährt der Motor zu diesem Offset mit der Geschwindigkeit GOREFERENCEFREQUENCY_TOOFFSET.

Nullen der Position des Motors

Mit dem Parameter DAPI_STEPPER_GO_REFSWITCH_PAR_SET_POS_0 kann zusätzlich eingestellt werden, ob der Motor jetzt die Position 0 bekommt.

Programmierbeispiel

```
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);
```

5.10.1.16. DAPI_STEPPER_CMD_GET_CPU_TEMP

Beschreibung

Die Temperatur des CPU wird abgefragt.

Definition

ULONG DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GET_CPU_TEMP, 0, 0, 0, 0);

Parameter

cmd=DAPI_STEPPER_CMD_GET_CPU_TEMP

Return-Wert

Temperatur [°C]

5.10.1.17. DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Beschreibung

Hiermit wird die Versorgungsspannung des Motors abgefragt.

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-Wert

Motorversorgungsspannung in [mV]

5.10.2. Status abfragen mit DapiStepperGetStatus

5.10.2.1. DAPI_STEPPER_STATUS_GET_ACTIVITY

Beschreibung

Hiermit werden verschiedene Statusinformationen (z.B. die Aktivität des Motorstroms, etc.) abgefragt.

Definition

ULONG DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_ACTIVITY);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

motor=Nummer des anzusprechenden Motors

Return-Wert

Bit	Command	Beschreibung
0	DISABLE	Motor darf nicht verfahren
1	MOTORSTROMACTIV	Motorstrom ist aktiv
2	HALTESTROMACTIV	Haltestrom ist aktiv
3	GOPOSITIONACTIV	GoPosition ist akitv
4	GOPOSITIONBREMSEN	GoPosition Bremsung ist aktiv
5	GOREFERENZACTIV	GoReference ist aktiv

Programmierbeispiel

```
ret = DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_ACTIVITY);
```

5.10.2.2. DAPI_STEPPER_STATUS_GET_POSITION

Beschreibung
Hiermit wird eine bestimmte Position abgelesen.

Definition
ULONG DapiStepperGetStatus(handle, motor, cmd);

Parameter
cmd=DAPI_STEPPER_STATUS_GET_POSITION

Return-Wert
Es wird die aktuelle Motorposition in 1/16 Schritteinheiten zurückgegeben

Programmierbeispiel

```
value = DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_POSITION);
```

5.10.2.3. DAPI_STEPPER_STATUS_GET_SWITCH

Beschreibung
Hiermit wird der Zustand der Schalter abgefragt.

Definition
ULONG DapiStepperGetStatus(handle, motor, cmd);

Parameter
cmd=DAPI_STEPPER_STATUS_GET_SWITCH

Return-Wert
Es wird der Zustand der Schalter zurückgeliefert:
Bit0: ENDSCHALTER1; 1 = Endschalter1 ist geschlossen
Bit1: ENDSCHALTER2; 1 = Endschalter2 ist geschlossen
Bit2: REFSCHALTER1; 1 = Referenzschalter1 ist geschlossen
Bit3: REFSCHALTER2; 1 = Referenzschalter2 ist geschlossen

Programmierbeispiel

```
pos = DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_SWITCH);
```

5.10.3. DapiStepperCommandEx

Beschreibung

Dieser erweiterte Befehl steuert Schrittmotoren an.

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=Dies ist das Handle eines geöffneten Moduls
motor=Nummer des anzusprechenden Motors
cmd=Erweitertes Kommando
par1..7=Erweiterte kommandoabhängige Parameter (s. Bemerkung)

Bemerkung

Siehe delib.h für die erweiterten Kommandos und den zugehörigen Parametern.

5.11. Ausgabe-Timeout verwalten

5.11.1. DapiSpecialCMDTimeout

Beschreibung
Dieser Befehl dient zum Setzen der Timeout-Zeit

Definition
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, par1, par2);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls

Timeout-Zeit setzen
cmd=DAPI_SPECIAL_CMD_TIMEOUT_SET_VALUE_SEC
par1=Sekunden [s]
par2=Millisekunden [100ms] (Wert 6 bedeutet 600ms)

Timeout aktivieren setzen
cmd=DAPI_SPECIAL_CMD_TIMEOUT_ACTIVATE

Timeout deaktivieren setzen
cmd=DAPI_SPECIAL_CMD_TIMEOUT_DEACTIVATE

Return-Wert
Keiner

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_SET_VALUE_SEC, 3, 7);
//Die Zeit des Timeouts wird auf 3,7sek gesetzt.

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_ACTIVATE, 0, 0);
//Der Timeout wird aktiviert.

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DEACTIVATE, 0, 0);
//Der Timeout wird deaktiviert.
```

5.11.2. DapiSpecialCMDTimeoutGetStatus

Beschreibung

Dieser Befehl dient zum Auslesen des Timeout-Status.

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, DAPI_SPECIAL_TIMEOUT_GET_STATUS, 0, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert

Return=0 (Timeout ist deaktivert)
Return=1 (Timeout ist aktiviert)
Return=2 (Timeout hat stattgefunden)

Programmierbeispiel

```
status = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_GET_STATUS, 0, 0);
//Abfrage des Timeout-Status.
```

5.12. Register Schreib-Befehle

5.12.1. DapiWriteByte

Beschreibung
Dieser Befehl führt einen direkten Register Schreibbefehl auf das Modul aus.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll
value=Gibt den Datenwert an, der geschrieben wird (8 Bit)

Return-Wert
Keiner

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.12.2. DapiWriteWord

Beschreibung
Dieser Befehl führt einen direkten Register Schreibbefehl auf das Modul aus.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll
value=Gibt den Datenwert an, der geschrieben wird (16 Bit)

Return-Wert
Keiner

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.12.3. DapiWriteLong

Beschreibung
Dieser Befehl führt einen direkten Register Schreibbefehl auf das Modul aus.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll
value=Gibt den Datenwert an, der geschrieben wird (32 Bit)

Return-Wert
Keiner

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.12.4. DapiWriteLongLong

Beschreibung
Dieser Befehl führt einen direkten Register Schreibbefehl auf das Modul aus.

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

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll
value=Gibt den Datenwert an, der geschrieben wird (64 Bit)

Return-Wert
Keiner

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.13. Register Lese-Befehle

5.13.1. DapiReadByte

Beschreibung
Dieser Befehl führt einen direkten Register Lese-Befehl auf das Modul aus.

Definition
ULONG DapiReadByte(ULONG handle, ULONG adress);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll

Return-Wert
Inhalt des zu lesenden Registers (8 Bit)

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.13.2. DapiReadWord

Beschreibung
Dieser Befehl führt einen direkten Register Lese-Befehl auf das Modul aus.

Definition
ULONG DapiReadWord(ULONG handle, ULONG adress);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll

Return-Wert
Inhalt des zu lesenden Registers (16 Bit)

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.13.3. DapiReadLong

Beschreibung
Dieser Befehl führt einen direkten Register Lese-Befehl auf das Modul aus.

Definition
ULONG DapiReadLong(ULONG handle, ULONG adress);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll

Return-Wert
Inhalt des zu lesenden Registers (32 Bit)

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.13.4. DapiReadLongLong

Beschreibung
Dieser Befehl führt einen direkten Register Lese-Befehl auf das Modul aus.

Definition
ULONGLONG DapiReadLongLong(ULONG handle, ULONG adress);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
adress=Adresse, auf die zugegriffen werden soll

Return-Wert
Inhalt des zu lesenden Registers (64 Bit)

Bemerkung
Dies sollte nur von erfahrenen Programmieren benutzt werden. So kann auf alle zur Verfügung stehenden Register direkt zugegriffen werden.

5.14. Programmier-Beispiel

```
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
//
// (c) DEDITEC GmbH, 2009
//
// web: http://www.deditec.de
//
// mail: vertrieb@deditec.de
//
//
//
// dtapi_prog_beispiel_input_output.cpp
//
//
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
//
//
// Folgende Bibliotheken beim Linken mit einbinden: delib.lib
// Dies bitte in den Projekteinstellungen (Projekt/Einstellungen/Linker(Objekt-
Bibliothek-Module) .. letzter Eintrag konfigurieren
#include <windows.h>
#include <stdio.h>
#include "conio.h"
#include "delib.h"
// ------------------------------------------------------------------------------
// ------------------------------------------------------------------------------
// ------------------------------------------------------------------------------
// ------------------------------------------------------------------------------
// ------------------------------------------------------------------------------

void main(void)
{
unsigned long handle;
unsigned long data;
unsigned long anz;
unsigned long i;
unsigned long chan;
// ----------------------------------------------------
// USB-Modul öffnen
handle = DapiOpenModule(USB_Interface8,0);
printf("USB_Interface8 handle = %x\n", handle);
if (handle==0)
{
// USB Modul wurde nicht gefunden
printf("Modul konnte nicht geöffnet werden\n");
printf("TASTE für weiter\n");
getch();
```

```
return;
}
// Zum Testen - ein Ping senden
// ----------------------------------------------------
printf("PING\n");
anz=10;
for(i=0;i!=anz;++i)
{
data=DapiPing(handle, i);
if(i==data)
{
// OK
printf(".");
}
else
{
// No answer
printf("E");
}
}
printf("\n");

// ----------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 0, data);
printf("Schreibe auf Adresse=0 daten=0x%x\n", data);
// ----------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 1, data);
printf("Schreibe auf Adresse=0 daten=0x%x\n", data);
// ----------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 2, data);
printf("Schreibe auf Adresse=2 daten=0x%x\n", data);
// ----------------------------------------------------
// Einen Wert von den Eingängen lesen
data = (unsigned long) DapiReadByte(handle, 0);
printf("Gelesene Daten = 0x%x\n", data);
// ----------------------------------------------------
// Einen A/D Wert lesen
chan=11; // read chan. 11
data = DapiReadWord(handle, 0xff010000 + chan*2);
printf("Adress=%x, ret=%x volt=%f\n", chan, data, ((float) data) / 1024*5);//
Bei 5 Volt Ref
// ----------------------------------------------------
// Modul wieder schliessen
DapiCloseModule(handle);
printf("TASTE für weiter\n");
getch();
return ;
}
```

Anhang

VI

6. Anhang

6.1. Revisionen

Rev 1.00	Erste Anleitung
Rev 1.01	Erweiterung flexibles Steckverbindersystem Überarbeitung der Blockschaltbilder
Rev 1.1	Software Installation
Rev 1.2	Ergänzung RO-AD16_ISO
Rev 1.3	Änderung Konfiguration seriell / CAN
Rev 1.4	Erweiterung Kapitel Stepper
Rev 1.5	Änderung Abschnitt CAN-Interface Einstellungen
Rev 1.6	Kapitel 2.2.2.3 um CAN/SER Adapter erweitert
Rev 1.7	Manual wurde um Ethernet Interface erweitert
Rev 2.00	Designänderung
Rev 2.01	Neuer Stepper Befehl "DAPI_STEPPER_CMD_GO_POSITION_RELATIVE" Ergänzung des Return-Werts bei Befehl "DAPI_STEPPER_STATUS_GET_ACTIVITY" Ergänzung des Parameters Hold-Time (Zeit unendlich) bei Befehl "DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC"
Rev 2.02	Erweiterung um Kapitel "RO-O8-R8" und neuer DO-Befehl "DapiDOSet1_WithTimer"
Rev 2.03	Neue Verwaltungsfunktion "DapiGetDELIBVersion", neuer DI Befehl "DapiSpecialCounterLatchAll", "DapiSpecialCounterLatchAllWithReset" und Ergänzung der Modi bei "DapiDIGetCounter"
Rev 2.04	Kapitel Abmessung der RO-Serie hinzugefügt
Rev 2.05	PT100 Modul ins Manual aufgenommen
Rev 2.06	Kapitel "Firmware Update" hinzugefügt, Kapitel "Software" überarbeitet

6.2. Urheberrechte und Marken

Linux ist eine registrierte Marke von Linus Torvalds.

Windows CE ist eine registrierte Marke von Microsoft Corporation.

USB ist eine registrierte Marke von USB Implementers Forum Inc.

LabVIEW ist eine registrierte Marke von National Instruments.

Intel ist eine registrierte Marke von Intel Corporation.

AMD ist eine registrierte Marke von Advanced Micro Devices, Inc.

ProfiLab ist eine registrierte Marke von ABACOM Ingenieurbüro GbR.

ispVM System ist eine registrierte Marke von Lattice Semiconductor Corporation

Visual-C/C++, -C#, -Basic, -Basic.NET sind registrierte Marken von Microsoft Corporation.

Delphi ist eine registrierte Marke von Borland Software Corporation.