DEDITEC GMBH

USB-RS232-TTL-STICK

Hardware-Beschreibung

2010 September

INDEX

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.

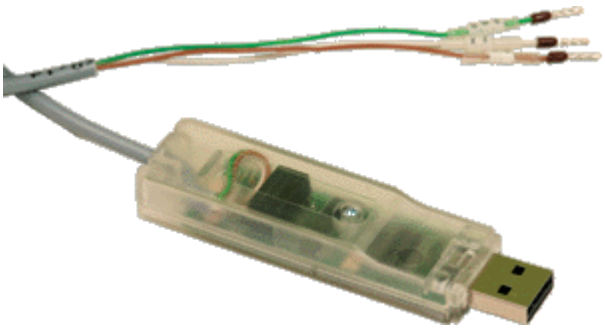
Hardware Beschreibung

2. Hardware Beschreibung

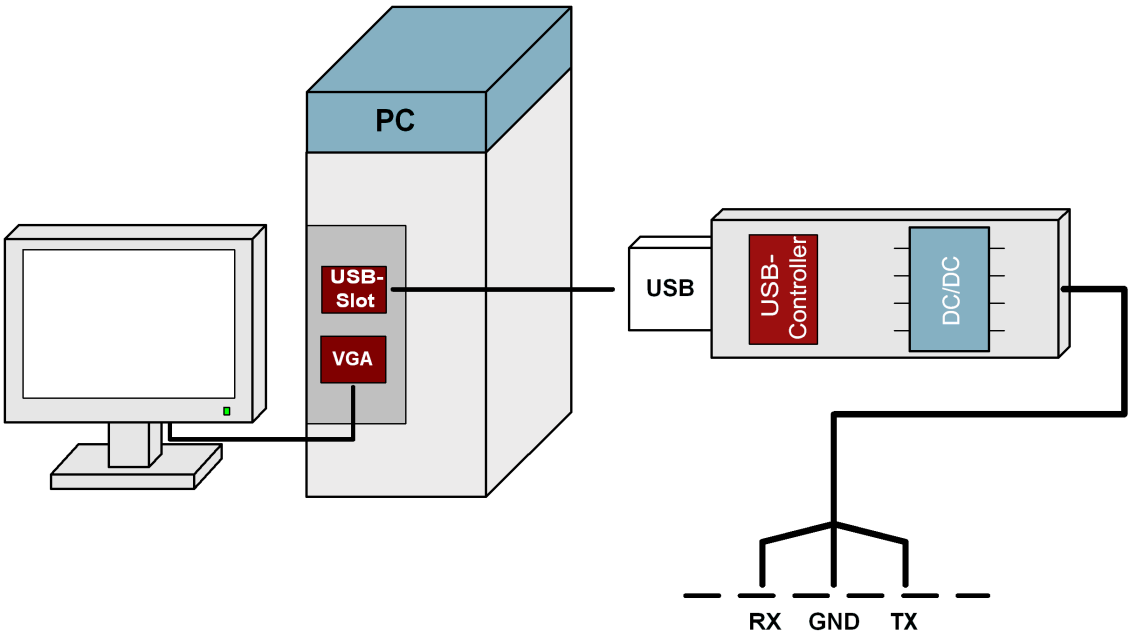
2.1. Einführung

Dieser USB Umsetzer auf RS-232 mit TTL-Pegel ist ideal für z.B. Debugging Zwecke an Microcontroller Schaltungen. Durch die galvanische Trennung wird zudem Ihr Prozessor-Board in hohem Maße geschützt.

Dieser kleine USB-Stick überträgt serielle Daten mit TTL-Pegel (z.B. Debugging-Informationen) über die USB-Schnittstelle. Auf Steuerungen stehen meist nur TTL-Signale an den seriellen Schnittstellen von Prozessoren zur Verfügung. Mit diesem Tool können Sie sich diese Signal jetzt bequem auf einem PC oder Notebook anzeigen lassen. Und die "empfindlichen TTL-Signale" werden durch die galvanische Trennung sicher mit dem PC verbunden.



2.2. Technische Daten



- +5V Spannungsversorgung (über USB-BUS)
- USB auf RS-232 mit TTL Pegel
- Je eine LED für TX und RX
- Galvanisch getrennt
- 50 Baud ..3MBaud (per Software konfigurierbar)
- Windows VCP (Virtueller COM Port)
- Linux Treiber inklusive
- Abmessungen: 84,5 x 21 x 12,5/9,5 mm (ohne Kabel)

2.3. Pinbelegung Steckverbinder (3pol. Anschlussleitung)

Der Anschluss an den Stick erfolgt auf der RS-232 Seite mittels einer 3pol. Anschlussleitung. Die Aderenden sind beschriftet und mit Aderendhülsen versehen.

RS-232 Pinbelegung

Pin	
Braun	RX
Weiß	TX
Grün	GND

2.4. Kontroll LED's

Zwei LED's signalisieren Sende- und Empfangsereignisse.

1*TX (senden)
1*RX (empfangen)

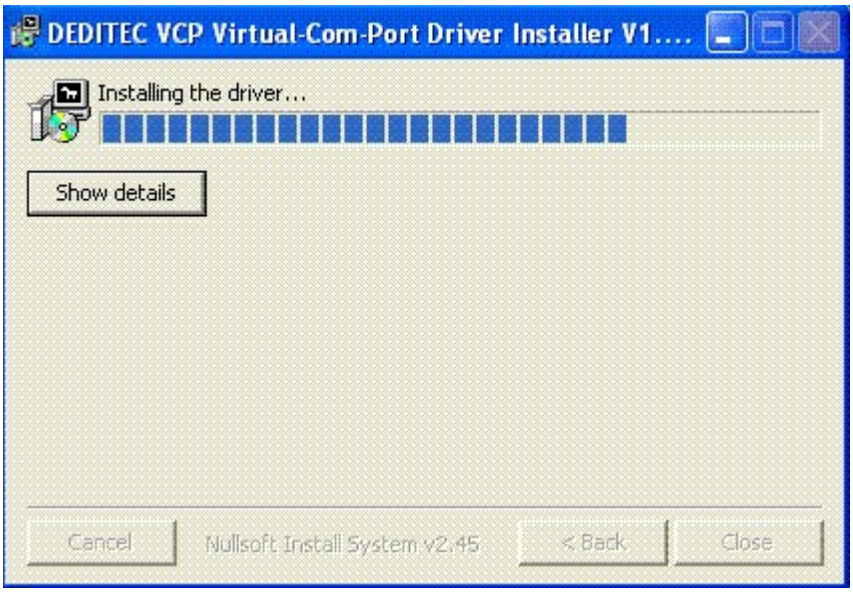
Software

3. Software

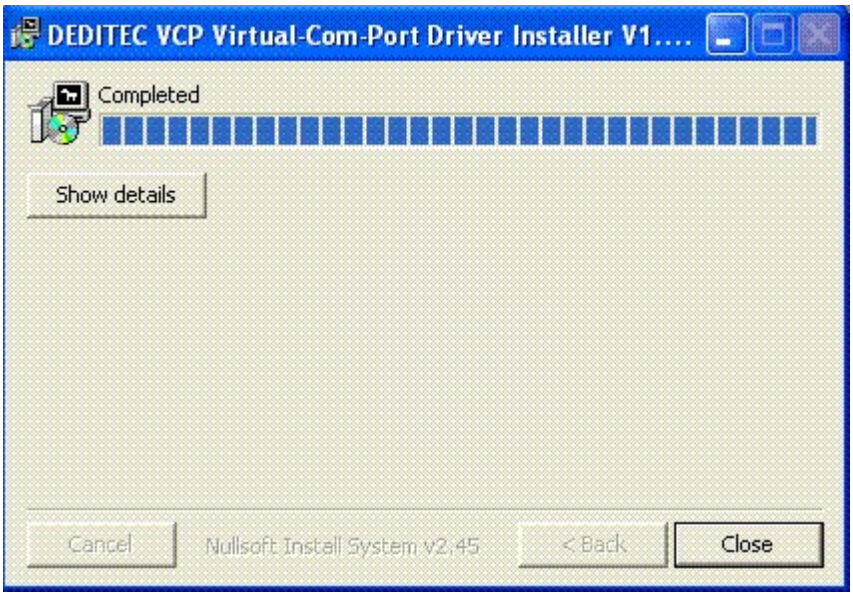
3.1. Installation "VCP Treiber (Virtueller COM-Port)"

Legen Sie die DEDITEC driver CD in das Laufwerk und starten Sie “vcp_driver_install.exe”. Die “VCP Treiber (Virtueller COM-Port)” Software ist auch unter http://www.deditec.de/de/entwicklungstools/download.html erhältlich.

Drücken Sie auf “Install”.



Die Treiber werden nun installiert.



Die “VCP Treiber (Virtueller COM-Port)” Software wurde nun installiert. Drücken sie auf “Close” um die Installation zu beenden.

Anhang

4. Anhang

4.1. Produktinformation

Best.Nr.: USB-RS232-TTL-STICK

Produkt: USB-RS-232(TTL-Pegel)-Konverter für den Anschluss von seriellen Schnittstellen (TTL-Pegel) an den PC über USB

Produktlink: http://www.deditec.de/de/entwicklungstools/prod/usb-rs232-ttl-stick.html

4.2. Revisionen

Rev 1.00	Erste DEDITEC Anleitung
Rev 2.00	Designänderung

4.3. 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.

DELIB

DEDITEC Treiber Bibliothek

2010 Dezember

INDEX

INDEX

INDEX

INDEX

Software

1. Software

1.1. Benutzung unserer Produkte

1.1.1. 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!

1.1.2. Ansteuerung über unsere DELIB Treiberbibliothek

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

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

1.1.3. 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!

1.1.4. Ansteuerung über mitgelieferte Testprogramme

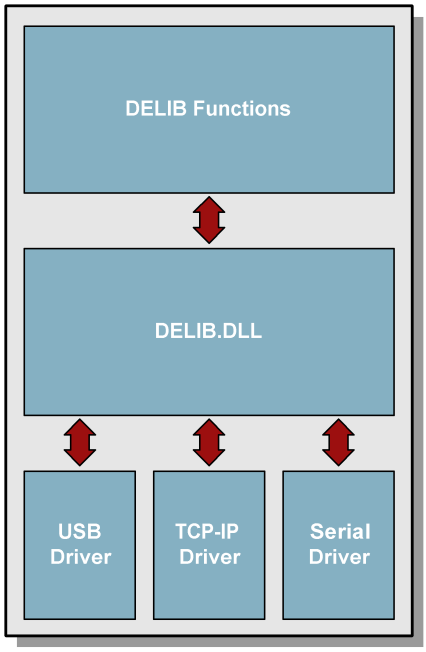
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

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

1.2. DELIB Treiberbibliothek

1.2.1. 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

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 !

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.

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 !

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.

1.2.2. Unterstützte Betriebssysteme

Unsere Produkte unterstützen folgende Betriebssysteme:

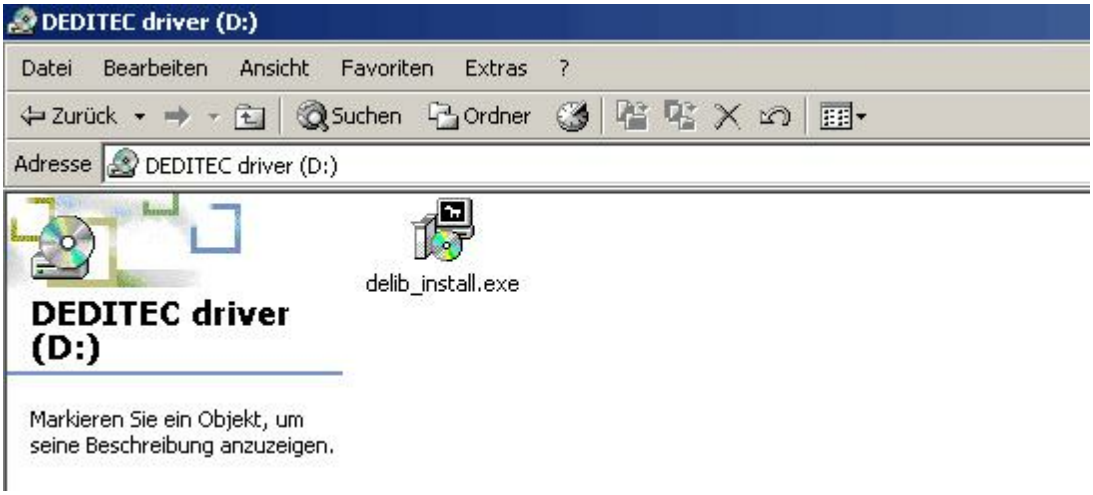
- Windows 2000
- Windows XP
- Windows Vista
- Windows 7
- Linux

1.2.3. Unterstützte Programmiersprachen

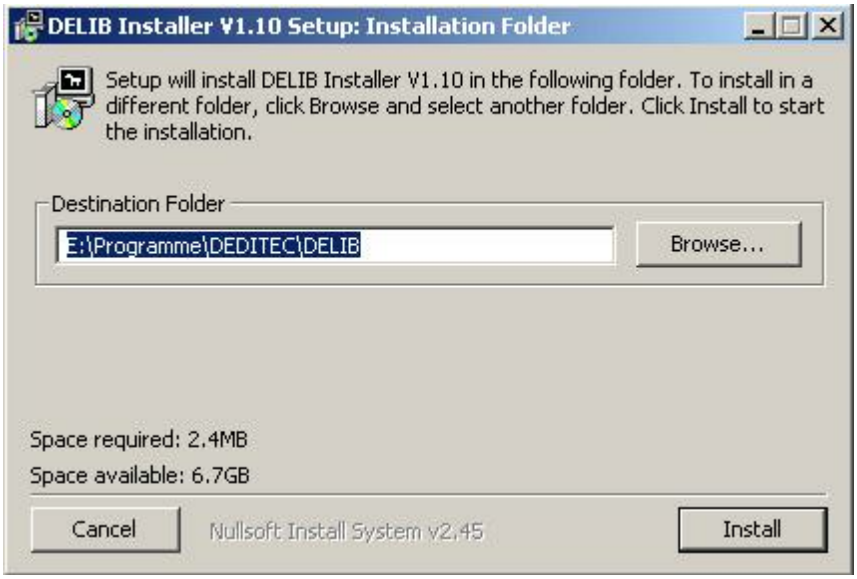
Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

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

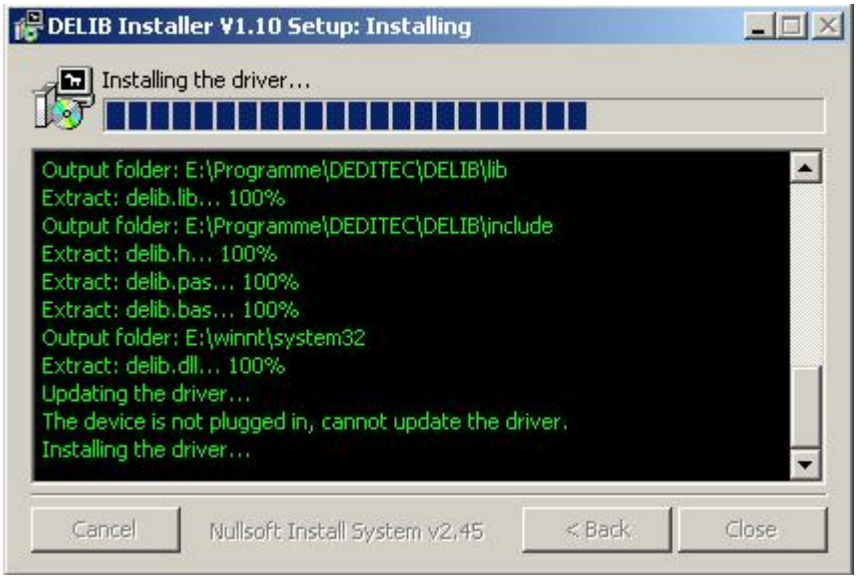
1.2.4. Installation DELIB-Treiberbibliothek



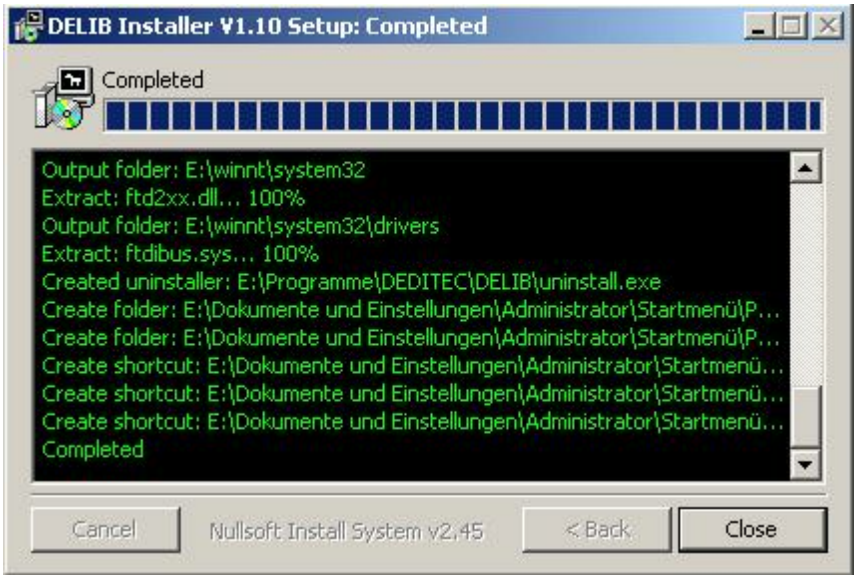
Legen Sie die DEDITEC driver CD in das Laufwerk und starten Sie **“delib_install.exe”**. Die DELIB-Treiberbibliothek ist auch unter http://www.deditec.de/delib erhältlich.



Drücken Sie auf **“Install”**.



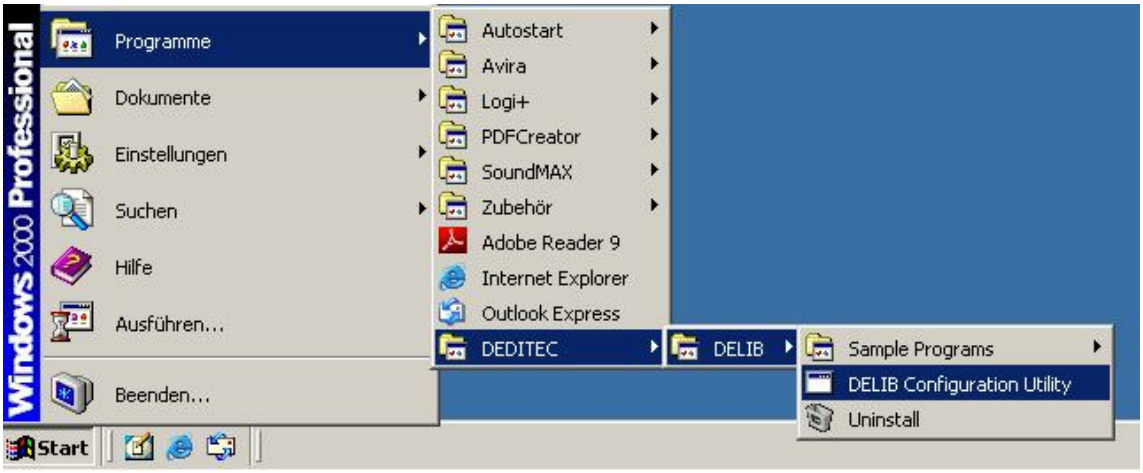
Die Treiber werden nun installiert.



Die DELIB Treiberbibliothek wurde nun Installiert. Drücken sie auf **“Close”** um die Installation zu beenden.

Mit dem **“DELIB Configuration Utility”** (nächstes Kapitel) können Sie Ihr Modul konfigurieren (dies ist nur nötig, wenn Sie mehr als ein Modul ansprechen möchten).

1.2.5. DELIB Configuration Utility



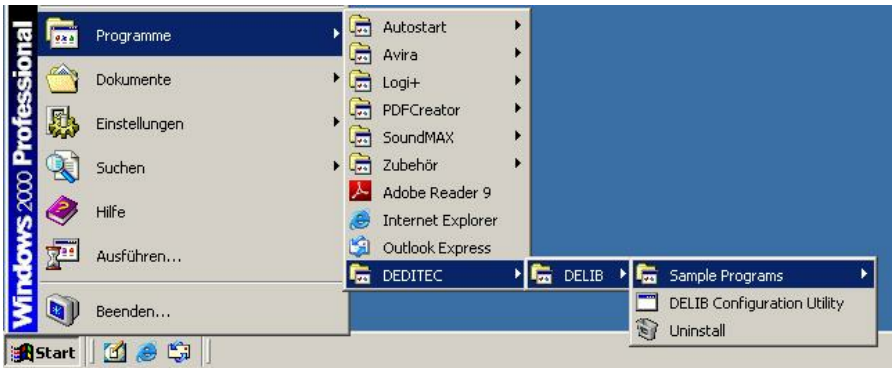
“DELIB Configuration Utility” wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → DELIB Configuration Utility.

Das **“DELIB Configuration Utility”** ist ein Programm zur Konfiguration und Unterteilung Identischer USB-Module im System. Dies ist aber nicht nötig falls nur ein Modul vorhanden ist.

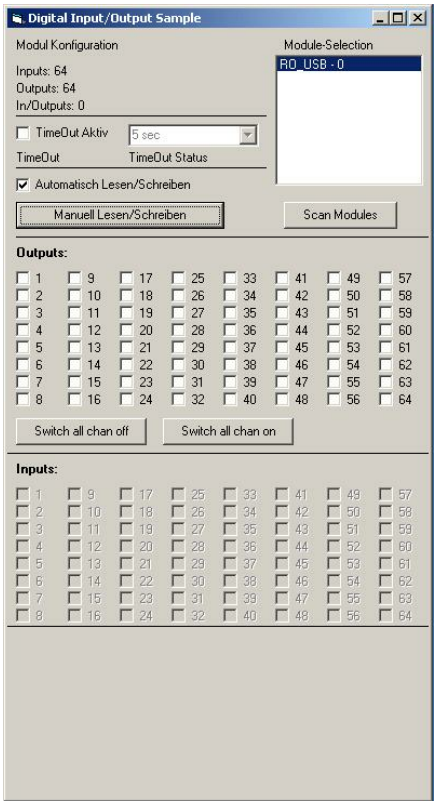
Weiteres zum Inhalt der **“DELIB Installation”**, siehe **“Manual für DELIB Treiberbibliothek”**

1.3. Testprogramme

1.3.1. Digital Input-Output Demo

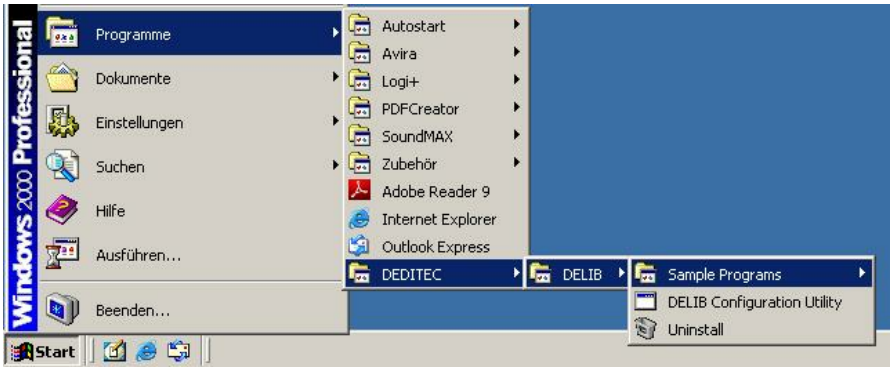


“Digital Input-Output Demo” wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → Digital Input-Output Demo.

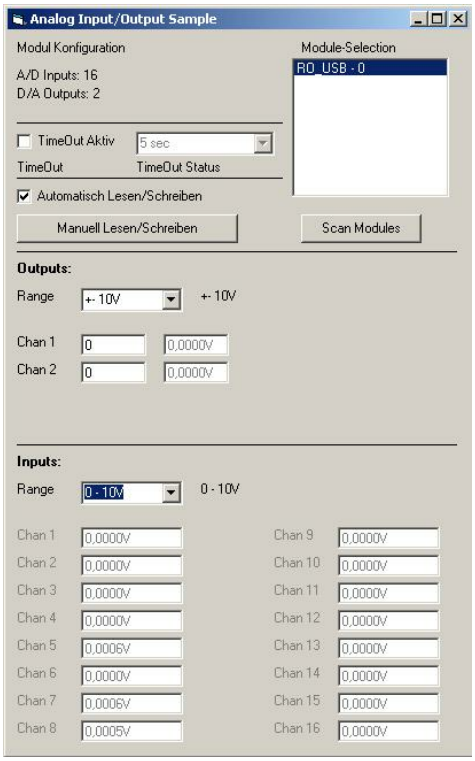


Diese Grafik zeigt einen Test des RO-USB-O64-R64. Oben links kann man die Konfiguration des Moduls ablesen (64 Eingänge und 64 Ausgänge).

1.3.2. Analog Input-Output Demo

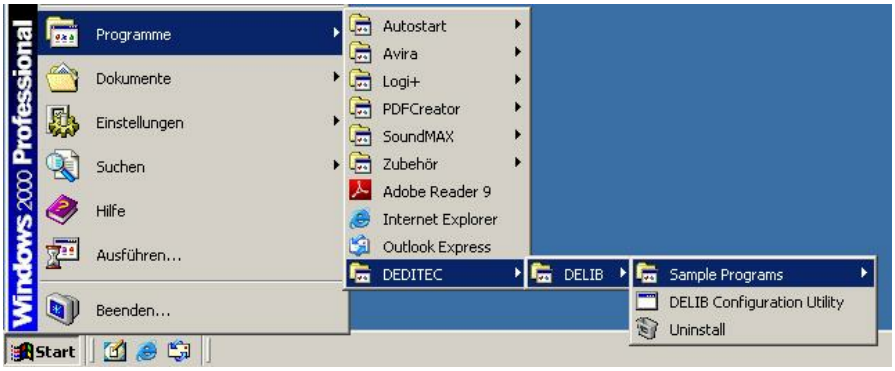


"Analog Input-Output Demo" wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → Analog Input-Output Demo.

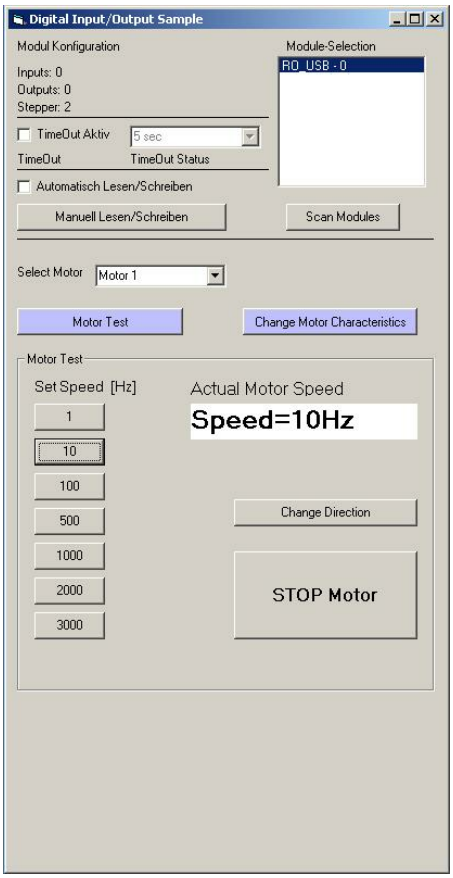


Diese Grafik zeigt einen Test des RO-USB-AD16-DA4. Oben links kann man die Konfiguration des Moduls ablesen (16 A/D-Eingänge und 4 D/A-Ausgänge).

1.3.3. Stepper Demo



“Stepper Demo” wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → Stepper Demo.



Diese Grafik zeigt einen Test des RO-USB-STEPPER2. Oben links kann man die Konfiguration des Moduls ablesen (2 Stepper).

Verzeichnisstruktur der DELIB

2. Verzeichnisstruktur der DELIB

Nach erfolgreicher Installation liegt folgender Verzeichnissbaum vor:

C:\Programme\DEDITEC\DELIB\

- include Includes für Programmiersprachen (→ Abschnitt 3.1.1)
- lib Library (→ Abschnitt 3.1.2)
- lib\bc Borland Compiler Library (→ Abschnitt 3.1.2)
- programs Modul-Testprogramme (→ Abschnitt 2.3)
- USB-Driver Treiber für USB-Module

Zudem wurde auch in C:\WINNT\system32\ die Datei delib.dll sowie diverse USB Systemtreiber installiert.

2.1. Include Verzeichnis

Das für die DELIB angelegte Include-Verzeichnis enthält die Dateien, welche die entsprechenden Library-Funktionen beschreiben. Diese sind für die Programmiersprachen C (.h), Delphi (.pas) und Visual Basic (.bas) gegeben.

2.2. Library-Verzeichnis

Hierin befindet sich die Datei **“DELIB.lib”**. Sie dient als Bindeglied für das Compilieren von eigenen Programmen, die die **“DELIB.dll”** benutzen.

2.3. Library-Verzeichnis für Borland

Für Borland Compiler gibt es eine separate DELIB.lib, die sich im Unterverzeichnis **“bc”** befindet. Diese dient ebenfalls als Bindeglied für das Compilieren von eigenen Programmen, die die **“DELIB.dll”** benutzen.

2.4. Umgebungsvariablen

Zwei Umgebungsvariablen weisen auf wichtige Verzeichnisse hin, die Dateien für die Programmiersprachen C, Delphi und Visual Basic enthalten.

“DELIB_INCLUDE” zeigt auf das Include-Verzeichnis.

%DELIB_INCLUDE% → c:\Programme\DEDITEC\DELIB\include"

“DELIB_LIB” zeigt auf das Library-Verzeichnis.

%DELIB_LIB% → c:\ Programme\DEDITEC\DELIB\lib

DELIB API Referenz

3. DELIB API Referenz

3.1. Verwaltungsfunktionen

3.1.1. 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;
}
```

3.1.2. 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);
```

3.2. Fehlerbehandlung

3.2.1. 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);
```

3.2.2. 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;
}
```

3.3. Digitale Eingänge lesen

3.3.1. 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)

3.3.2. 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

3.3.3. 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

3.3.4. 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();
```

3.3.5. 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

3.3.6. 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

3.3.7. DapiDIGetCounter

Beschreibung

Dieser Befehl liest den Eingangszähler eines digitalen Eingangs.

Definition

ULONG DapiDIGetCounter(handle, ch, par1);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

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

par1=0 (Normale Zählfunktion)

par1=DAPI_CNT_MODE_READ_WITH_RESET (Zähler auslesen und direktes Counter resetten)

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
```

3.4. Digitale Ausgänge verwalten

3.4.1. 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

3.4.2. 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

3.4.3. 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

3.4.4. 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();
```

3.4.5. 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

3.4.6. 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.

3.4.7. 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.

3.5. A/D Wandler Funktionen

3.5.1. 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

3.5.2. 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

3.5.3. 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

3.5.4. 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

3.5.5. 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.

3.6. D/A Ausgänge verwalten

3.6.1. DapiDASetMode

Beschreibung
Dieser Befehl setzt den Modus für einen D/A Wandler.

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

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

Return-Wert
Keiner

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

3.6.2. 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

3.6.3. DapiDASet

Beschreibung
Dieser Befehl übergibt ein Datenwert an einen Kanal eines 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

Return-Wert
Keiner

3.6.4. 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

3.6.5. 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.

3.6.6. 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.
```

3.7. TTL-Ein-/Ausgangs Richtungen setzen mit DapiSpecialCommand

3.7.1. DAPI_SPECIAL_CMD_SET_DIR_DX_1

Beschreibung

Dieser Befehl setzt die Richtung von TTL-Ein/Ausgängen (1-Bit weise).

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, ULONG ch, ULONG dir, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem die Richtung gesetzt werden soll (0, 1, 2, 3, 4 ..)

dir=Gibt die Richtung für 8 Kanäle an (1=output / 0=input) / Bit 0 steht für Kanal 0, Bit 1 für Kanal 1 ...

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x01 , 0);
// Set Dir of TTL-I/O CH0 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x02 , 0);
// Set Dir of TTL-I/O CH1 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x04 , 0);
// Set Dir of TTL-I/O CH2 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x08 , 0);
// Set Dir of TTL-I/O CH3 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x10 , 0);
// Set Dir of TTL-I/O CH4 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x20 , 0);
// Set Dir of TTL-I/O CH5 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x40 , 0);
// Set Dir of TTL-I/O CH6 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x80 , 0);
// Set Dir of TTL-I/O CH7 to output, others to input

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0x0f , 0);
// Set Dir of TTL-I/O CH0-3 to output, others to input
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_1, 0, 0xff , 0);
// Set Dir of TTL-I/O CH0-7 to output, others to input
```

3.7.2. DAPI_SPECIAL_CMD_SET_DIR_DX_8

Beschreibung

Dieser Befehl setzt die Richtung von TTL-Ein/Ausgängen (8-Bit weise).

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_SET_DIR_DX_8, ULONG ch, ULONG dir, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem die Richtung gesetzt werden soll (0, 8, 16, 24 ..). Zwischenwerte sind ungültig

dir=(8-Bit) gibt die Richtung für 8 hintereinanderliegende Ein/Ausgänge an. (1=output / 0=input)

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_8, 0, 1, 0);
// Set Dir of TTL-I/O CH0 to out
```

3.8. Schrittmotoren Funktionen

3.8.1. Befehle mit DapiStepperCommand

3.8.1.1. 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);
```

3.8.1.2. 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.
```

3.8.1.3. 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

3.8.1.4. 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]

3.8.1.5. 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]

3.8.1.6. 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]

3.8.1.7. 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);

3.8.1.8. 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);
```

3.8.1.9. 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)

3.8.1.10. 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.
```

3.8.1.11. 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.
```

3.8.1.12. 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);

3.8.1.13. 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);

3.8.1.14. 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

3.8.1.15. 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);
```

3.8.1.16. DAPI_STEPPER_CMD_GET_CPU_TEMP

Beschreibung

Die Temperatur des CPU wird abgefragt.

Definition

ULONG DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GET_CPU_TEMP);

Parameter

cmd=DAPI_STEPPER_CMD_GET_CPU_TEMP

Return-Wert

Temperatur [°C]

3.8.1.17. DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Beschreibung

Hiermit wird die Versorgungsspannung des Motors abgefragt.

Definition

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

Parameter

cmd=DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Return-Wert

Motorversorgungsspannung in [mV]

3.8.2. Status abfragen mit DapiStepperGetStatus

3.8.2.1. 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);
```

3.8.2.2. 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);
```

3.8.2.3. 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);
```

3.8.3. 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.

3.9. Ausgabe-Timeout verwalten

3.9.1. 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.
```

3.9.2. 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.
```

3.10. Testfunktionen

3.10.1. DapiPing

Beschreibung
Dieser Befehl prüft die Verbindung zu einem geöffneten Modul.

Definition
ULONG DapiPing(ULONG handle, ULONG value);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
value=Übergebener Testwert an das Modul

Return-Wert
Hier muß der mit “**value**” übergebene Testwert zurückkommen

3.11. Register Schreib-Befehle

3.11.1. 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.

3.11.2. 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.

3.11.3. 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.

3.11.4. 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.

3.12. Register Lese-Befehle

3.12.1. 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.

3.12.2. 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.

3.12.3. 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.

3.12.4. 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.

3.13. 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

4. Anhang

4.1. Revisionen

Rev 1.00	Erste DEDITEC Anleitung
Rev 1.1	Ergänzung von diversen AD/DA Befehlen
Rev 1.2	Ergänzung des Stepper Motors
Rev 1.3	Softwareinstallation und Verzeichnisstruktur der DELIB
Rev 2.00	Designanpassung
Rev 2.01	Ergänzung der DELIB Befehle "DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC", "DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC" und "DAPI_STEPPER_CMD_GO_REFSWITCH"
Rev 2.02	Neuer D/A Befehl "DAPI_SPECIAL_CMD_DA" und DO Befehl "DAPI_SPECIAL_CMD_TIMEOUT_GET_STATUS"
Rev 2.03	Neuer Stepper Befehl "DAPI_STEPPER_CMD_GO_POSITION_RELATIVE"
Rev 2.04	Neues Programmierbeispiel bei Befehl "DAPI_SPECIAL_CMD_SET_DIR_DX_1" 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"

4.2. 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.