



RO-STEPPER2

Hardware-Beschreibung

2010 November

INDEX

<u>1. Einleitung</u>	6
<u>1.1. Vorwort</u>	6
<u>1.2. Kundenzufriedenheit</u>	6
<u>1.3. Kundenresonanz</u>	6
<u>2. Hardware Beschreibung</u>	8
<u>2.1. Übersichtsbild</u>	8
<u>2.2. Technische Daten</u>	9
<u>2.3. Ansteuerung</u>	9
<u>2.4. Pinbelegung</u>	10
<u>2.4.1. Stepper Steckverbinder (10pol)</u>	10
<u>3. Software</u>	12
<u>3.1. Benutzung unserer Produkte</u>	12
<u>3.1.1. Ansteuerung über grafische Anwendungen</u>	12
<u>3.1.2. Ansteuerung über unsere DELIB Treiberbibliothek</u>	12
<u>3.1.3. Ansteuerung auf Protokollebene</u>	12
<u>3.1.4. Ansteuerung über mitgelieferte Testprogramme</u>	13
<u>3.2. DELIB Treiberbibliothek</u>	14
<u>3.2.1. Übersicht</u>	14
<u>3.2.2. Unterstützte Betriebssysteme</u>	16
<u>3.2.3. Unterstützte Programmiersprachen</u>	16
<u>3.2.4. Installation DELIB-Treiberbibliothek</u>	17
<u>3.2.5. DELIB Configuration Utility</u>	19
<u>3.3. Testprogramme</u>	20
<u>3.3.1. Stepper Demo</u>	20
<u>4. DELIB API Referenz</u>	23
<u>4.1. Verwaltungsfunktionen</u>	23
<u>4.1.1. DapiOpenModule</u>	23

INDEX

<u>4.1.2. DapiCloseModule</u>	24
<u>4.2. Fehlerbehandlung</u>	25
<u>4.2.1. DapiGetLastError</u>	25
<u>4.2.2. DapiGetLastErrorText</u>	26
<u>4.3. Schrittmotoren Funktionen</u>	27
<u>4.3.1. Befehle mit DapiStepperCommand</u>	27
<u>4.3.1.1. DAPI_STEPPER_CMD_GO_POSITION</u>	27
4.3.1.2.	
<u>DAPI_STEPPER_CMD_GO_POSITION_RELATIVE</u>	28
4.3.1.3. <u>DAPI_STEPPER_CMD_SET_POSITION</u>	29
4.3.1.4. <u>DAPI_STEPPER_CMD_SET_FREQUENCY</u>	30
4.3.1.5. <u>DAPI_STEPPER_CMD_GET_FREQUENCY</u>	31
4.3.1.6.	
<u>DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY</u>	32
4.3.1.7. <u>DAPI_STEPPER_CMD_STOP</u>	33
4.3.1.8. <u>DAPI_STEPPER_CMD_FULLSTOP</u>	34
4.3.1.9. <u>DAPI_STEPPER_CMD_DISABLE</u>	35
4.3.1.10.	
<u>DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC</u>	36
4.3.1.11.	
<u>DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC</u>	41
4.3.1.12.	
<u>DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEP</u> <u>ROM_SAVE</u>	49
4.3.1.13.	
<u>DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEP</u> <u>ROM_LOAD</u>	50
4.3.1.14.	
<u>DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOA</u> <u>D_DEFAULT</u>	51
4.3.1.15. <u>DAPI_STEPPER_CMD_GO_REFSWITCH</u>	52
4.3.1.16. <u>DAPI_STEPPER_CMD_GET_CPU_TEMP</u>	54
4.3.1.17.	
<u>DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAG</u> <u>E</u>	55
<u>4.3.2. Status abfragen mit DapiStepperGetStatus</u>	56
4.3.2.1. <u>DAPI_STEPPER_STATUS_GET_ACTIVITY</u>	56
4.3.2.2. <u>DAPI_STEPPER_STATUS_GET_POSITION</u>	57
4.3.2.3. <u>DAPI_STEPPER_STATUS_GET_SWITCH</u>	58
<u>4.3.3. DapiStepperCommandEx</u>	59

INDEX

<u>4.4. Programmier-Beispiel</u>	60
<u>5. Anhang</u>	63
<u>5.1. Revisionen</u>	63
<u>5.2. Urheberrechte und Marken</u>	64

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.

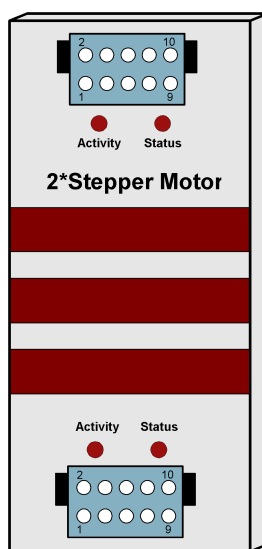
Hardware Beschreibung



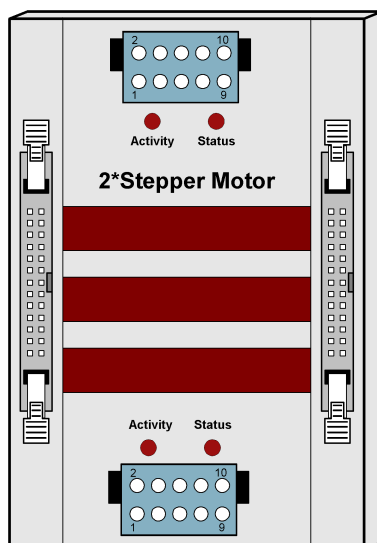
2. Hardware Beschreibung

2.1. 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.



2.2. 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

2.3. Ansteuerung

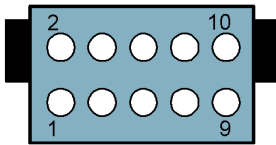
Alle Parameter lassen sich bequem über die DELIB Library einstellen.

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.

2.4. Pinbelegung

2.4.1. 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.

Software



3. Software

3.1. Benutzung unserer Produkte

3.1.1. 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!

3.1.2. 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

3.1.3. 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!

3.1.4. Ansteuerung über mitgelieferte Testprogramme

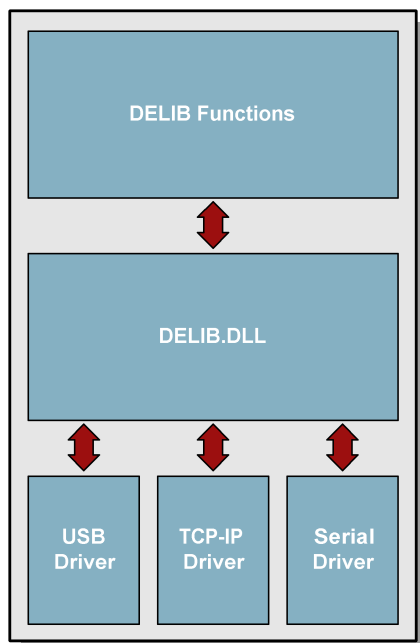
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

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

3.2. DELIB Treiberbibliothek

3.2.1. 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 Installationskripte, die es ermöglichen, die DELIB Installation in Ihre Anwendung mit einzubinden.

3.2.2. Unterstützte Betriebssysteme

Unsere Produkte unterstützen folgende Betriebssysteme:

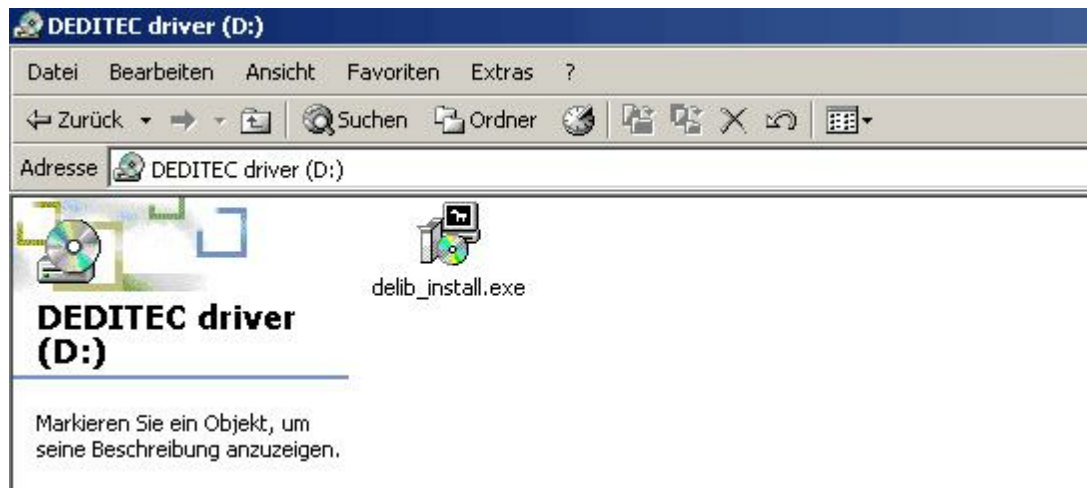
- Windows 2000
- Windows XP
- Windows Vista
- Windows 7
- Linux

3.2.3. Unterstützte Programmiersprachen

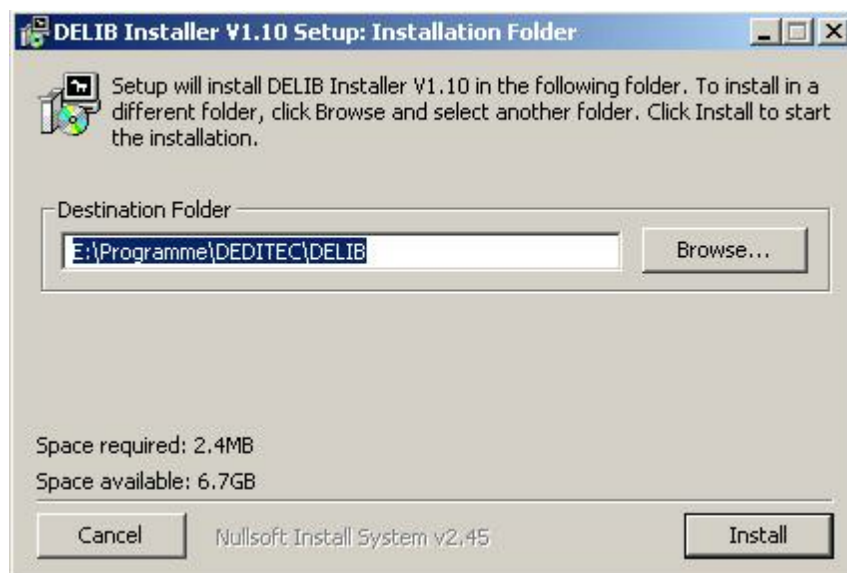
Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

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

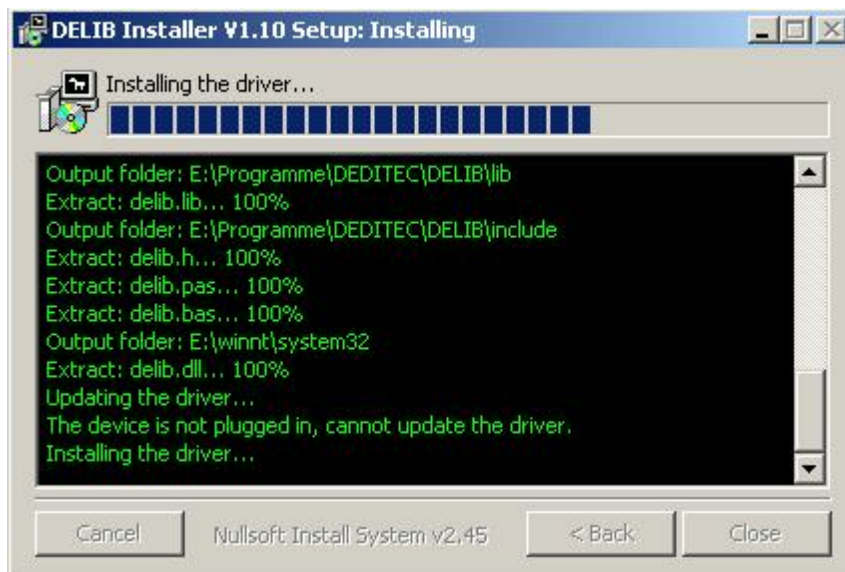
3.2.4. Installation DELIB-Treiberbibliothek



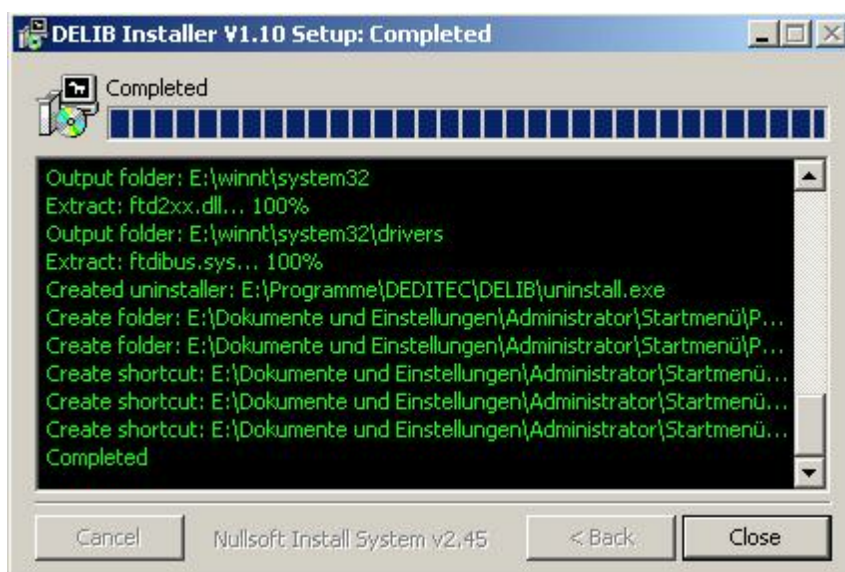
Legen Sie die DEDITEC driver CD in das Laufwerk und starten Sie "delib_install.exe". Die DELIB-Treiberbibliothek ist auch unter <http://www.deditec.de/delib> erhältlich.



Drücken Sie auf "Install".



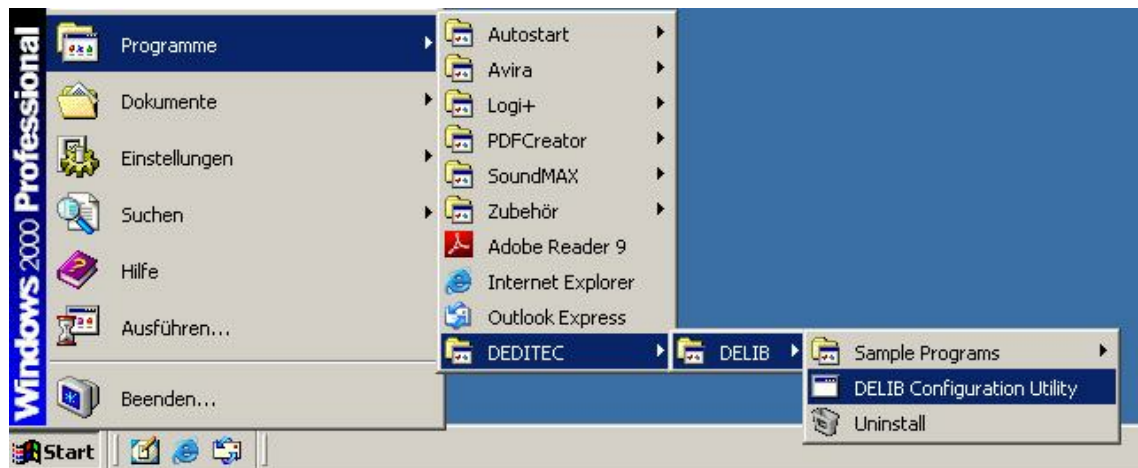
Die Treiber werden nun installiert.



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).

3.2.5. DELIB Configuration Utility



“**DELIB Configuration Utility**” wird auf dem folgendem Weg gestartet:

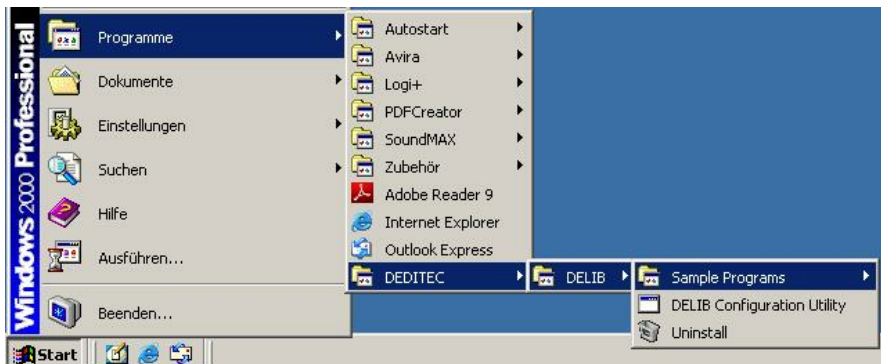
Start → Programme → DEDITEC → DELIB → DELIB Configuration Utility.

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**”

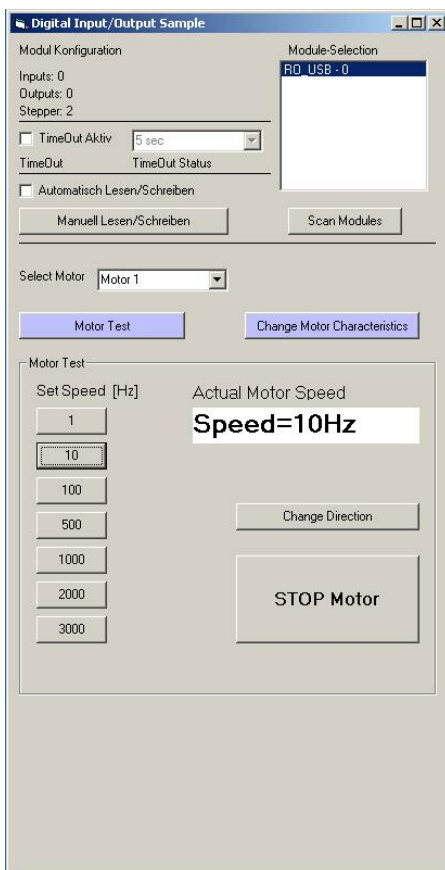
3.3. Testprogramme

3.3.1. 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).

DELIB API Referenz



IV

4. DELIB API Referenz

4.1. Verwaltungsfunktionen

4.1.1. 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;
}
```

4.1.2. 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);
```

4.2. Fehlerbehandlung

4.2.1. 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);
```

4.2.2. 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;
}
```

4.3. Schrittmotoren Funktionen

4.3.1. Befehle mit DapiStepperCommand

4.3.1.1. 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);
```

4.3.1.2. 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.
```

4.3.1.3. DAPI_STEPPEER_CMD_SET_POSITION

Beschreibung

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

Definition

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

Parameter

par1 = Motorposition

4.3.1.4. DAPI_STEPPEER_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_STEPPEER_CMD_SET_FREQUENCY, par1, 0, 0, 0);

Parameter

par1 = Motorsollfrequenz [Hz]

4.3.1.5. 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]

4.3.1.6. 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. Schritverluste 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]

4.3.1.7. DAPI_STEPPEER_CMD_STOP

Beschreibung

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

Definition

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

4.3.1.8. DAPI_STEPPEER_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_STEPPEER_CMD_FULLSTOP, 0, 0, 0, 0);

Programmierbeispiel

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

4.3.1.9. DAPI_STEPPEER_CMD_DISABLE

Beschreibung

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

Definition

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

Parameter

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

4.3.1.10. 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-GoReferenceFrequencyToEndSwitch 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 Enscharter 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);
// Beschleunigung 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.

```

4.3.1.11. 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-GoReferenceFrequencyToEndSwitch abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOE
NDSWITCH

Parameter GoReferenceFrequencyAfterEndSwitch abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFT
ERENDSWITCH

Parameter GoReferenceFrequencyToOffset abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TO
OFFSET

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]

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-GoReferenceFrequencyToEndSwitch

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOE
NDSWITCH

return=Frequenz [Vollschritt / s]

Parameter GoReferenceFrequencyAfterEndSwitch abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFT
ERENDSWITCH

return=Frequenz [Vollschritt / s]

Parameter GoReferenceFrequencyToOffset abfragen

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TO
OFFSET

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);
// Beschleunigung 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.

```

4.3.1.12.

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);
```

4.3.1.13.

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);
```

4.3.1.14.

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
- Haltestrom 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

4.3.1.15. 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 Vollschrift)

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 GOREFERENCFREQUENCY_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 GOREFERENCFREQUENCY_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);
```

4.3.1.16. 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]

4.3.1.17. DAPI_STEPPEER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Beschreibung

Hiermit wird die Versorgungsspannung des CPU abgefragt.

Definition

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

Parameter

cmd=DAPI_STEPPEER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Return-Wert

Motorversorgungsspannung in [mV]

4.3.2. Status abfragen mit DapiStepperGetStatus

4.3.2.1. 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 aktiv
4	GOPOSITIONBREMSSEN	GoPosition Bremsung ist aktiv
5	GOREFERENZACTIV	GoReference ist aktiv

Programmierbeispiel

```
ret = DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_ACTIVITY);
```

4.3.2.2. 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);
```

4.3.2.3. 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);
```

4.3.3. 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.

4.4. 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=1 daten=0x%x\n", data);
// -----
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 2, data);
printf("Schreibe auf Adresse=2 daten=0x%x\n", data);
// -----
// Einen Wert von den Eingängen lesen
data = (unsigned long) DapiReadByte(handle, 0);
printf("Gelesene Daten = 0x%x\n", data);
// -----
// Einen A/D Wert lesen
chan=11; // read chan. 11
data = DapiReadWord(handle, 0xff010000 + chan*2);
printf("Adress=%x, ret=%x volt=%f\n", chan, data, ((float) data) / 1024*5); //
Bei 5 Volt Ref
// -----
// Modul wieder schliessen
DapiCloseModule(handle);
printf("TASTE für weiter\n");
getch();
return ;
}

```

Anhang



5. Anhang

5.1. Revisionen

Rev 1.00	Erste DEDITEC Anleitung
Rev 2.00	Designänderung
Rev 2.01	Neuer Stepper Befehl "DAPI_STEPPER_CMD_GO_POSITION_RELATIVE"
Rev 2.02	Ergänzung des Return-Werts bei Befehl "DAPI_STEPPER_STATUS_GET_ACTIVITY" Ergänzung des Parameters Hold-Time (Zeit unendlich) bei den Befehlen "DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC" und "DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC"

5.2. 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.