



# **RO-STEPPER2**

## **Hardware Description**

2010 November

# INDEX

<b><u>1. Introduction</u></b>	<b>6</b>
<u>1.1. General remarks</u>	6
<u>1.2. Customer satisfaction</u>	6
<u>1.3. Customer response</u>	6
<b><u>2. Hardware description</u></b>	<b>8</b>
<u>2.1. Overview screen</u>	8
<u>2.2. Technical data</u>	9
<u>2.3. Stepping motor control</u>	9
<u>2.4. Stepper connection wiring (10pol) - pinout</u>	10
<b><u>3. Software</u></b>	<b>12</b>
<u>3.1. Using our products</u>	12
<u>3.1.1. Access via graphical applications</u>	12
<u>3.1.2. Access via the DELIB driver library</u>	12
<u>3.1.3. Access via protocol</u>	12
<u>3.1.4. Access via provided test programs</u>	13
<u>3.2. DELIB driver library</u>	14
<u>3.2.1. Overview</u>	14
<u>3.2.2. Supported operating systems</u>	16
<u>3.2.3. Supported programming languages</u>	16
<u>3.2.4. Installation DELIB driver library</u>	17
<u>3.2.5. DELIB Configuration Utility</u>	19
<u>3.3. Test programs</u>	20
<u>3.3.1. Stepper Demo</u>	20
<b><u>4. DELIB API reference</u></b>	<b>23</b>
<u>4.1. Management functions</u>	23
<u>4.1.1. DapiOpenModule</u>	23
<u>4.1.2. DapiCloseModule</u>	24

# INDEX

<b><u>4.2. Error handling</u></b>	<b>25</b>
<b><u>4.2.1. DapiGetLastError</u></b>	<b>25</b>
<b><u>4.2.2. DapiGetLastErrorText</u></b>	<b>26</b>
<b><u>4.3. Stepper motor functions</u></b>	<b>27</b>
<b><u>4.3.1. DapiStepperCommands</u></b>	<b>27</b>
<b><u>4.3.1.1. DAPI_STEPPER_CMD_GO_POSITION</u></b>	<b>27</b>
<b><u>4.3.1.2. DAPI_STEPPER_CMD_GO_POSITION_RELATIVE</u></b>	<b>28</b>
<b><u>4.3.1.3. DAPI_STEPPER_CMD_SET_POSITION</u></b>	<b>29</b>
<b><u>4.3.1.4. DAPI_STEPPER_CMD_SET_FREQUENCY</u></b>	<b>30</b>
<b><u>4.3.1.5. DAPI_STEPPER_CMD_GET_FREQUENCY</u></b>	<b>31</b>
<b><u>4.3.1.6. DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY</u></b>	<b>32</b>
<b><u>4.3.1.7. DAPI_STEPPER_CMD_STOP</u></b>	<b>33</b>
<b><u>4.3.1.8. DAPI_STEPPER_CMD_FULLSTOP</u></b>	<b>34</b>
<b><u>4.3.1.9. DAPI_STEPPER_CMD_DISABLE</u></b>	<b>35</b>
<b><u>4.3.1.10. DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC</u></b>	<b>36</b>
<b><u>4.3.1.11. DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC</u></b>	<b>41</b>
<b><u>4.3.1.12. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEP_ROM_SAVE</u></b>	<b>49</b>
<b><u>4.3.1.13. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEP_ROM_LOAD</u></b>	<b>50</b>
<b><u>4.3.1.14. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT</u></b>	<b>51</b>
<b><u>4.3.1.15. DAPI_STEPPER_CMD_GO_REFSWITCH</u></b>	<b>52</b>
<b><u>4.3.1.16. DAPI_STEPPER_CMD_GET_CPU_TEMP</u></b>	<b>53</b>
<b><u>4.3.1.17. DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE</u></b>	<b>54</b>
<b><u>4.3.2. DapiStepperGetStatus</u></b>	<b>55</b>
<b><u>4.3.2.1. DAPI_STEPPER_STATUS_GET_ACTIVITY</u></b>	<b>55</b>
<b><u>4.3.2.2. DAPI_STEPPER_STATUS_GET_POSITION</u></b>	<b>56</b>
<b><u>4.3.2.3. DAPI_STEPPER_STATUS_GET_SWITCH</u></b>	<b>57</b>
<b><u>4.3.3. DapiStepperCommandEx</u></b>	<b>58</b>

# INDEX

<u>4.4. Example program</u>	59
<u>5. Appendix</u>	62
<u>5.1. Revisions</u>	62
<u>5.2. Copyrights and trademarks</u>	63



# Introduction

---



# 1. Introduction

## 1.1. General remarks

First of all, we would like to congratulate you to the purchase of a high quality DEDITEC product.

Our products are being developed by our engineers according to quality requirements of high standard. Already during design and development we take care that our products have -besides quality- a long availability and an optimal flexibility.

### **Modular design**

The modular design of our products reduces the time and the cost of development. Therefore we can offer you high quality products at a competitive price.

### **Availability**

Because of the modular design of our products, we have to redesign only a module instead of the whole product, in case a specific component is no longer available.

## 1.2. Customer satisfaction

Our philosophy: a content customer will come again. Therefore customer satisfaction is in first place for us.

If by any chance, you are not content with the performance of our product, please contact us by phone or mail immediately.

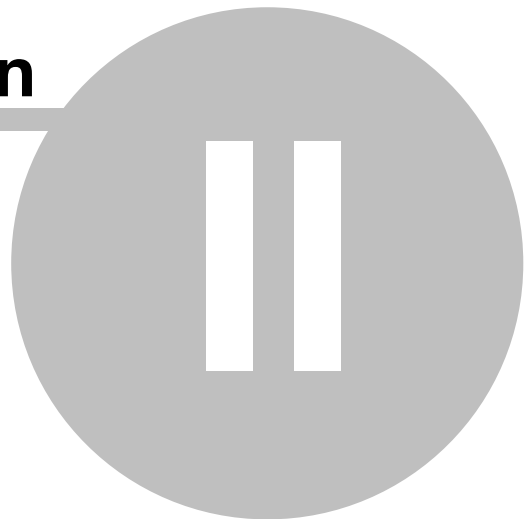
We take care of the problem.

## 1.3. Customer response

Our best products are co-developments together with our customers. Therefore we are thankful for comments and suggestions.

# Hardware description

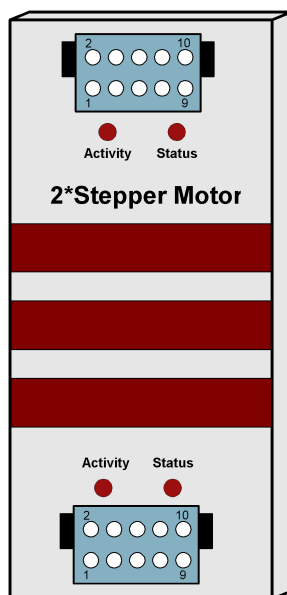
---



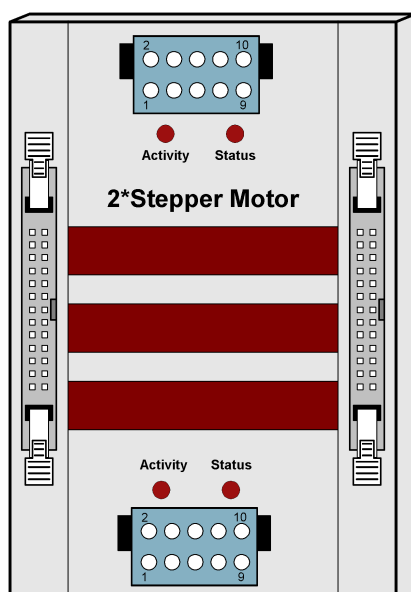
## 2. Hardware description

### 2.1. Overview screen

The lower figure shows a module with a terminal block and corresponding numbered connection ports.



The following figure shows a flexible stepper module with a terminal block and corresponding numbered connection ports.



## 2.2. Technical data

### Common:

- Module for two stepper motors
- Comfortable connector system with ejection mechanism
- Can be combined without any problem to other modules of the RO series

### Configurable parameters

- Start-/ stop frequency
- Maximum stepping frequency
- Acceleration slope
- Deceleration slope
- Phase current
- Hold current
- Hold time

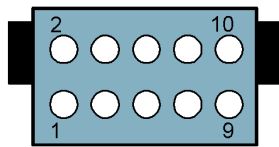
## 2.3. Stepping motor control

Every parameter can be conveniently set using the DELIB library.

Two reference switches are used to reach a reference position. Two additional end switches provide a safe stopping. If they are operated, the motors may exclusively be driven back in the opposite direction.

## 2.4. Stepper connection wiring (10pol) - pinout

Pinout of a socket connector and also of a stepper motor:



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

\*) The switches must be connected towards 24 V.

# Software



## 3. Software

### 3.1. Using our products

#### 3.1.1. Access via graphical applications

We provide driverinterfaces e.g. for LabVIEW and ProfiLab. The DELIB driver library is the basis, which can be directly activated by ProfiLAB.

For LabVIEW, we provide a simple driver connection with examples!

#### 3.1.2. Access via the DELIB driver library

In the appendix, you can find the complete function reference for the integration of our API-functions in your software. In addition we provide examples for the following programming languages:

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

#### 3.1.3. Access via protocol

The protocol for the activation of our products is open source. So you are able to use our products on systems without Windows or Linux.

### **3.1.4. Access via provided test programs**

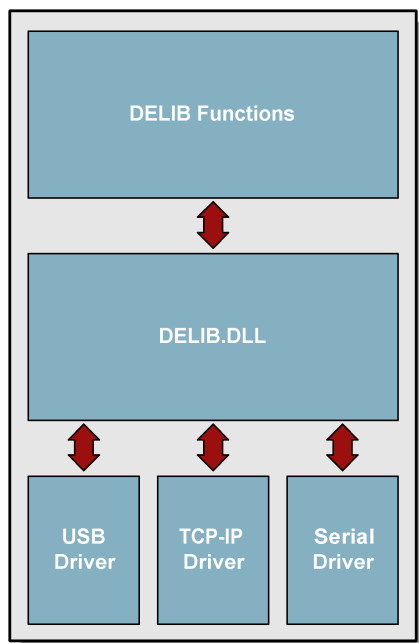
We provide simple handling test programs for the most important functions of our products. These will be installed automatically by the installation of the DELIB driver library.

So you can test directly e.g. relays or you can check the voltage of an A/D converter.

## 3.2. DELIB driver library

### 3.2.1. Overview

The following figure explains the structure of the DELIB driver library



The DELIB driver library allows an uniform response of DEDITEC hardware with particular consideration of the following viewpoints:

- Independent of operating system
- Independent of programming language
- Independent of the product

#### **Program under diverse operating systems**

The DELIB driver library allows an uniform response of our products on diverse operating systems.

We has made sure, that all of our products can be responded by a few commands.

Whatever which operating system you use. - Therefore the DELIB cares!

### **Program with diverse programming languages**

We provide uniform commands to create own applications. This will be solved by the DELIB driver library.

### **You choose the programming language!**

It can be simply developed applications under C++, C, Visual Basic, Delphi or LabVIEW® .

### **Program independent of the interface**

Write your application independent of the interface !

Program an application for an USB product of us. - Also, it will work with an ethernet or RS-232 product of us !

### **SDK-Kit for Programmer**

Integrate the DELIB in your application. On demand you receive an installation script for free, which allows you, to integrate the DELIB installation in your application.

### **3.2.2. Supported operating systems**

Our products support the following operating systems:

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

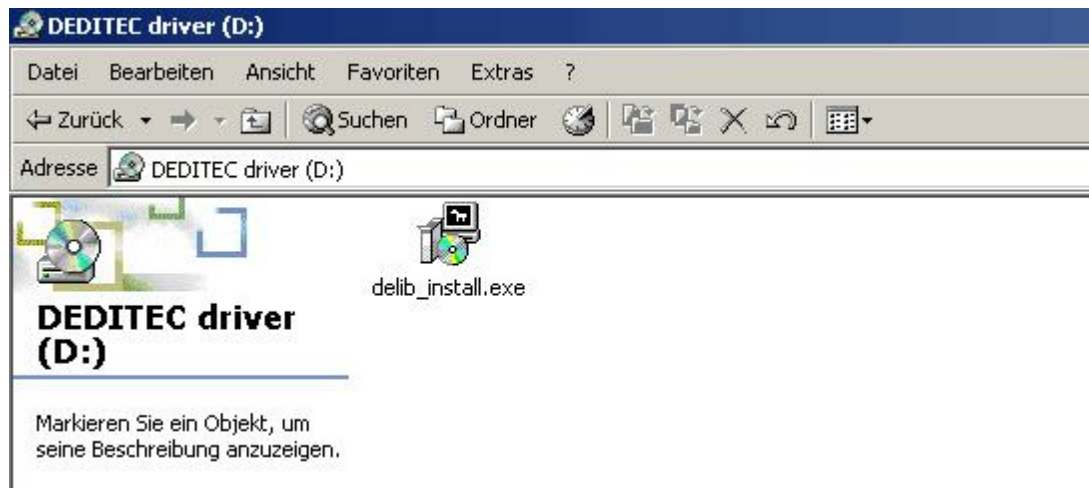
### **3.2.3. Supported programming languages**

Our products are responsive via the following programming languages:

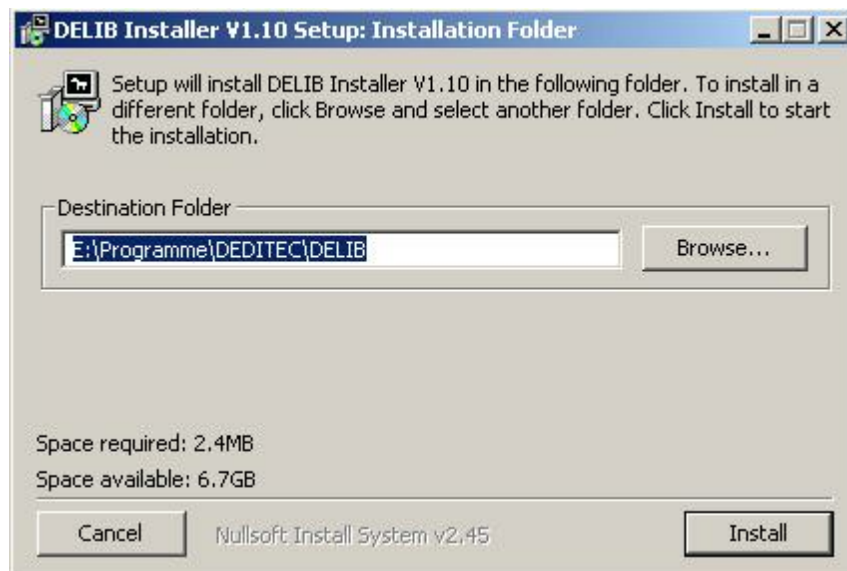
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

### 3.2.4. Installation DELIB driver library

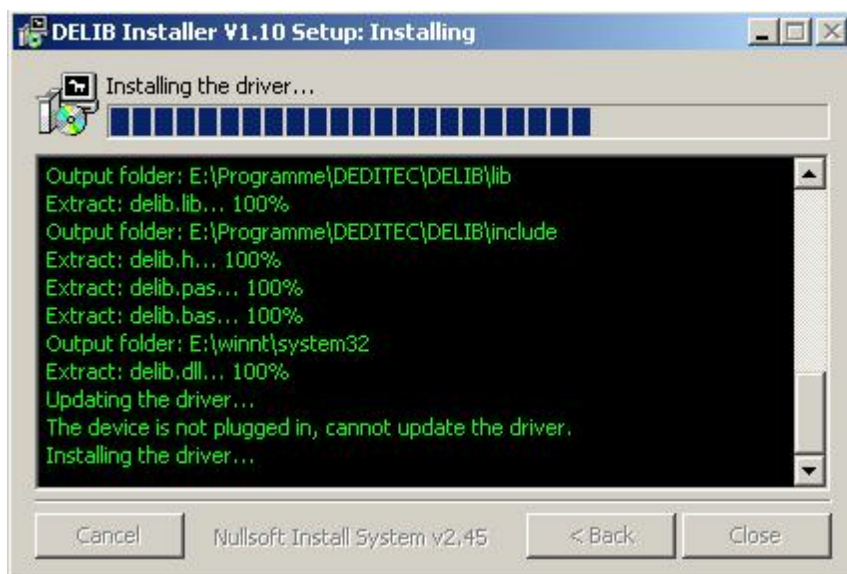
DELIB stands for DEDITEC Library and contains the necessary libraries for the modules in the programming languages C, Delphi and Visual Basic.



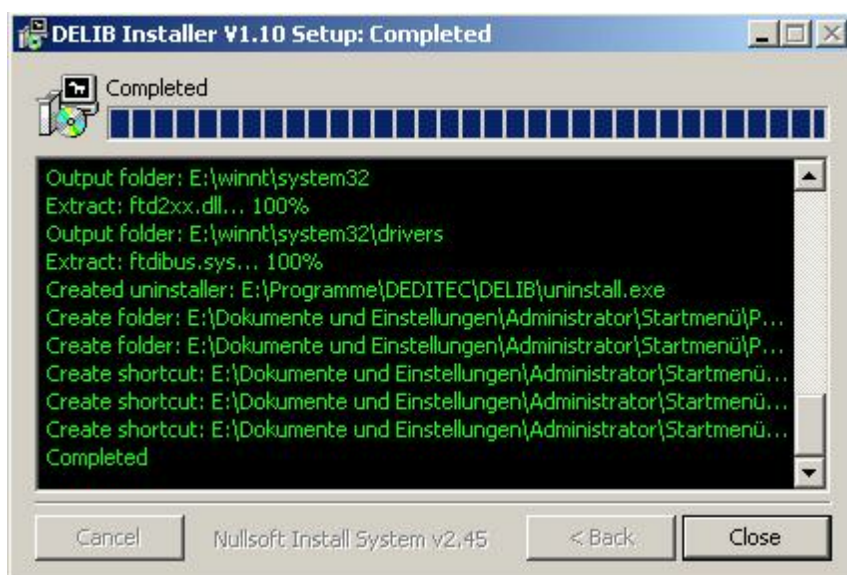
Insert the DEDITEC driver CD into the drive and start „**delib\_install.exe**“. The DELIB driver library is also available on <http://www.deditec.eu/delib>



Click on „**Install**“.



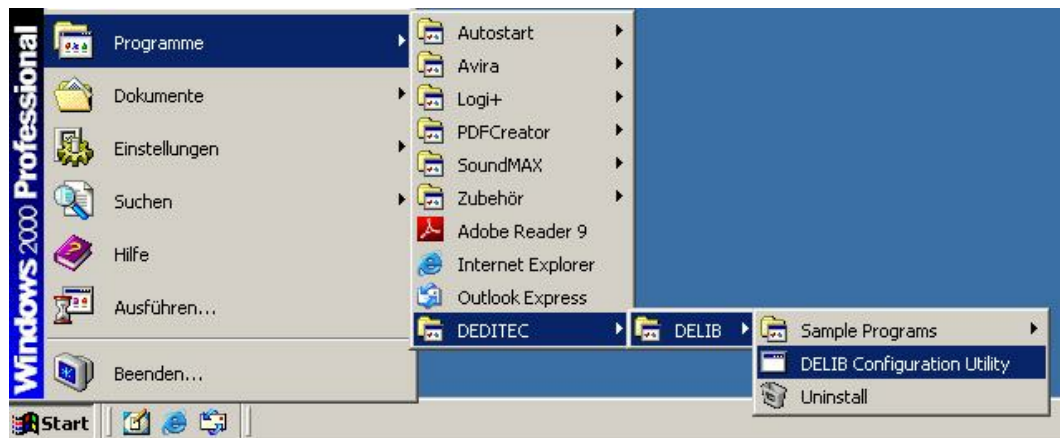
The drivers will be installed.



The DELIB driver library is now installed. Press „**Close**“ to finish the installation.

You can configure your module with the „**DELIB Configuration Utility**“ (see next chapter). This is only necessary, if more than one module is present.

### 3.2.5. DELIB Configuration Utility



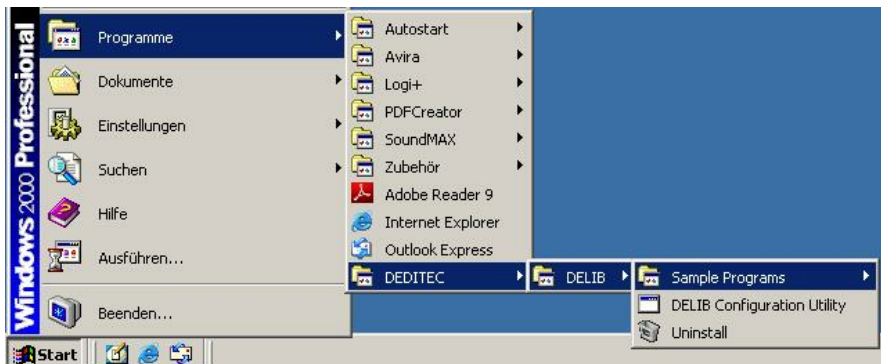
Start the “**DELIB Configuration Utility**” as follows:

**Start → Programs → DEDITEC → DELIB → DELIB Configuration Utility.**

The „**DELIB Configuration Utility**“ is a program to configure and subdivide identical USB-modules in the system. This is only necessary if more than one module is present.

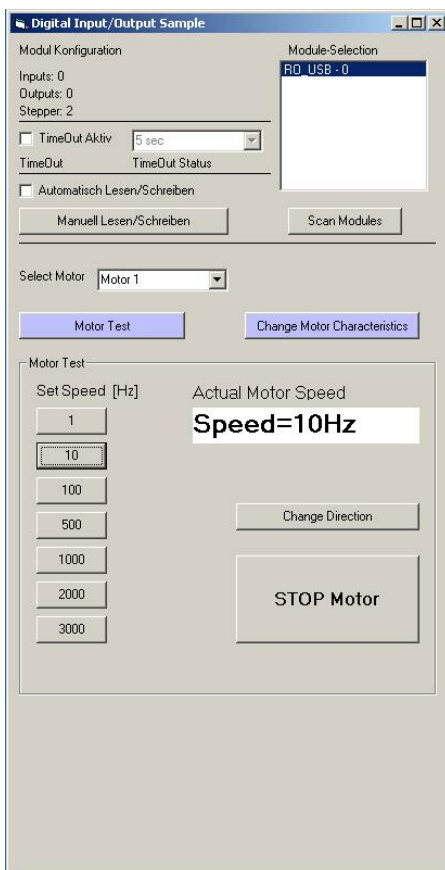
### 3.3. Test programs

#### 3.3.1. Stepper Demo



Start “**Stepper Demo**” as follows:

**Start → Programme → DEDITEC → DELIB → Stepper Demo.**



The screenshot shows a test of the RO-USB-STEPPER2. The configuration of the module (2 Stepper) is shown on the upper left side.

## **DELIB API reference**

---



**IV**

## 4. DELIB API reference

### 4.1. Management functions

#### 4.1.1. DapiOpenModule

##### Description

This function opens a particular module.

##### Definition

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

##### Parameters

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

nr=Indicates No of module which is to be opened.

nr=0 -> 1. module

nr=1 -> 2. module

##### Return value

handle=handle to the corresponding module

handle=0 -> Module was not found

##### Remarks

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

##### Example program

```
// 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

### Description

This command closes an opened module.

### Definition

*ULONG DapiCloseModule(ULONG handle);*

### Parameters

handle=This is the handle of an opened module

### Return value

none

### Example program

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

## 4.2. Error handling

### 4.2.1. DapiGetLastError

#### Description

This function returns the last registered error.

#### Definition

```
ULONG DapiGetLastError();
```

#### Parameters

None

#### Return value

Error code

0=no error. (see delib.h)

#### Example program

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

## 4.2.2. DapiGetLastErrorText

### Description

This function reads the text of the last registered error.

### Definition

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

### Parameters

msg = text buffer

msg\_length = length of the buffer

### Example program

```
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. Stepper motor functions

### 4.3.1. DapiStepperCommands

#### 4.3.1.1. DAPI\_STEPPEER\_CMD\_GO\_POSITION

##### Description

With this command the motor will drive to a position. This command can only be used when the motor is not disabled and Go\_Position or Go\_Reference are not executed.

##### Definition

*DapiStepperCommand(handle, motor, DAPI\_STEPPEER\_CMD\_GO\_POSITION, position, 0, 0, 0);*

##### Example program

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

#### 4.3.1.2. DAPI\_STEPPEER\_CMD\_GO\_POSITION\_RELATIVE

##### Description

With this command the motor will go to a relative position. In contrast to the command GO\_POSITION, which goes to an absolute position, this command considers the current position. This command can only be used when the motor is not disabled and Go\_Position or Go\_Reference are not executed.

##### Definition

```
void DapiStepperCommand(handle, motor,  
DAPI_STEPPEER_CMD_GO_POSITION_RELATIVE, go_pos_rel_par, 0, 0, 0);
```

##### Parameters

go\_pos\_rel\_par=the relative position, to which will be gone

##### Example program

```
DapiStepperCommand(handle, motor, DAPI_STEPPEER_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

#### Description

This command ist used to set the motor position. The resolution ist 1/16 Full-step. This command may be used anytime.

#### Definition

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

#### Parameters

par1=Motor position

#### 4.3.1.4. DAPI\_STEPPER\_CMD\_SET\_FREQUENCY

##### **Description**

This command is used to set the motor reference frequency. The motor frequency regulation takes on the compliance of the acceleration and deceleration slope. Step losses do not occur. The motor reference frequency is related to the full-step-mode.

The direction will be defined by the prefix. If the motor reference frequency is higher than the maximum frequency, the command is ignored.

With closed Endswitch1 the motor can only drive in positive direction, with closed Endswitch2 the motor can only drive in negative direction, otherwise the command is ignored.

##### **Definition**

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

##### **Parameters**

par1=Motor reference frequency [Hz]

#### 4.3.1.5. DAPI\_STEPPEER\_CMD\_GET\_FREQUENCY

##### **Description**

This command is used to read the motor frequency. This command can be used everytime.

##### **Definition**

*DapiStepperCommand(handle, motor, DAPI\_STEPPEER\_CMD\_GET\_FREQUENCY, par1, 0,0,0);*

##### **Return value**

Motor frequency [Hz]

#### 4.3.1.6. DAPI\_STEPPER\_CMD\_SET\_FREQUENCY\_DIRECTLY

##### **Description**

This command is used to set the motor frequency. The motor frequency regulation takes no function on the compliance of the acceleration and deceleration slope. The user is responsible. Step losses can occur. The motor reference frequency is related to the full-step. The direction can be defined by the prefix.

The motor frequency can't exceed the maximum frequency.

##### **Definition**

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

##### **Parameters**

par1=Motor frequency [Hz]

#### 4.3.1.7. DAPI\_STEPPEER\_CMD\_STOP

##### **Description**

This command is used to stop the motor, the deceleration slope will be used.

##### **Definition**

*DapiStepperCommand(handle, motor, DAPI\_STEPPEER\_CMD\_STOP, 0, 0, 0, 0);*

#### 4.3.1.8. DAPI\_STEPPEER\_CMD\_FULLSTOP

##### **Description**

This command is used to stop the motors immediately without using the the deceleration slope. After this command the motor position might be ignored because the motor has been stopped uncontrolled.

##### **Definition**

*DapiStepperCommand(handle, motor, DAPI\_STEPPEER\_CMD\_FULLSTOP, 0, 0, 0, 0);*

##### **Example program**

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

#### 4.3.1.9. DAPI\_STEPPEER\_CMD\_DISABLE

##### **Description**

This command is used to disable/enable the motor. The motor stops or starts driving. This command can be only used when the motor stopped.

##### **Definition**

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

##### **Parameters**

par1 = Disablemode (0=Normal function / 1=Disable)

#### 4.3.1.10. DAPI\_STEPPER\_CMD\_SET\_MOTORCHARACTERISTIC

##### Description

With this command, you can set motor characteristics.

##### Definition

*DapiStepperCommand(handle, motor,  
DAPI\_STEPPER\_CMD\_SET\_MOTORCHARACTERISTIC, par1, par2, 0, 0);*

##### Parameters

###### Set Parameter-Stepmode

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STEPMODE

par2=0 (Full-step)

par2=1 (1/2-step)

par2=2 (1/4-step)

par2=3 (1/8-step)

par2=4 (1/16-step)

###### Set Parameter-GO-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOFREQUENCY

par2=Speed [Full-step / s] - related to full-step frequency - (maximum value=5000)

###### Set Parameter-Start-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STARTFREQUENCY

par2=Startfrequency [Full-step / s] - related to full-step frequency - (maximum value=5000)

###### Set Parameter-Stop-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STOPFREQUENCY

par2=Stopfrequency [Full-step / s] - related to full-step frequency - (maximum value=5000)

###### Set Parameter-Max-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_MAXFREQUENCY

par2=maximum frequency [Full-step / s] - related to full-step frequency - (maximum value=5000)

### **Set Parameter-Accelerationslope**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_ACCELERATIONSLOPE

par2=Acceleration slope [Full-step / 10ms] - (maximum value=1000)

### **Set Parameter-Decelerationslope**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_DECELERATIONSLOPE

par2=Deceleration slope [Full-step / 10ms] - (maximum value=1000)

### **Set Parameter-Phasecurrent**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_PHASECURRENT

par2=Phase current [mA] - (maximum value=1500)

### **Set Parameter-Hold-Phasecurrent**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_HOLDPHASECURRENT

par2=Phase current at motor hold [mA] - (maximum value=1500)

### **Set Parameter-Hold-Time**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_HOLDTIME

par2=Time in that the hold goes to motorstop [ms]

par2=-1 / FFFF hex / 65535 dez (endless time)

### **Set Parameter-Status-LED-Mode**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STATUSLEDMODE

par2=Mode of the Status-LED

par2=0 (MOVE - LED is on if the stepper moves)

par2=1 (HALT - LED is on if the stepper stands still)

par2=2 (ENDSW1 - LED is on if the end switch1 is closed)

par2=3 (ENDSW2 - LED is on if the end switch2 is closed)

par2=4 (REFSW1 - LED is on if the Reference switch1 is closed)

par2=5 (REFSW2 - LED is on if the Reference switch2 is closed)

### **Set Parameter-Invert-END-Switch1**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_ENDSW1

par2=Endswitch1 is being inverted (0=normal / 1=inverted)

### **Set Parameter-Invert-END-Switch2**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_ENDSW2

par2=Endswitch2 is being inverted (0=normal / 1=inverted)

### **Set Parameter-Invert-Ref-Switch1**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_REFSW1

par2=Referenzswitch1 is being inverted (0=normal / 1=inverted)

### **Set Parameter-Invert-Ref-Switch2**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_REFSW2

par2=Referenzswitch2 is being inverted (0=normal / 1=inverted)

### **Set Parameter-Invert-direction**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_DIRECTION

par2=invert all direction details (0=normal / 1=inverted)

### **Set Parameter-Endswitch-Stopmode**

par1= DAPI\_STEPPER\_MOTORCHAR\_PAR\_ENDSWITCH\_STOPMODE

par2=setting of the stop behaviour (0=Fullstop / 1=Stop)

### **Set Parameter-GoReferenceFrequency (WARNING: This parameter will not be supported anymore)**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY

Remark: This parameter is replaced completely by the following three parameters.

### **Set Parameter-GoReferenceFrequencyToEndSwitch**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_TOE  
NDSWITCH

par2=frequency [Full-step / s] - (maximum value=5000)

### **Set Parameter-GoReferenceFrequencyAfterEndSwitch**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_AFT  
ERENDSWITCH

par2=frequency [Full-step / s] - (maximum value=5000)

## Set Parameter-GoReferenceFrequencyToOffset

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_TO  
OFFSET

par2=frequency [Full-step / s] - (maximum value=5000)

### Example program

```
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 Endschaltes2

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1, 0,0,0);
// invertiere Funktion des Referenzschalterschaltes1

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2, 0,0,0);
// invertiere Funktion des Referenzschalterschaltes2

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 Endschaltes 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 Endschaltes 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

##### Description

With this command, motor specific parameter can be read. This command may be used everytime. It is splitted in sub commands, that are analog to the parameters of the DAPI\_STEPPER\_CMD\_SET\_MOTORCHARATERISTIC.

##### Definition

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

##### Parameters

###### Get Parameter-Stepmode

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STEPMODE

###### Get Parameter-GO-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOFREQUENCY

###### Get Parameter-Start-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STARTFREQUENCY

###### Get Parameter-Stop-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STOPFREQUENCY

###### Get Parameter-Max-Frequency

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_MAXFREQUENCY

###### Get Parameter-Accelerationslope

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_ACCELERATIONSLOPE

###### Get Parameter-Decelerationslope

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_DECELERATIONSLOPE

###### Get Parameter-Phasecurrent

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_PHASECURRENT

### **Get Parameter-Hold-Phasecurrent**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_HOLDPHASECURRENT

### **Get Parameter-Hold-Time**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_HOLDTIME

### **Get Parameter-Status-LED-Mode**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STATUSLEDMODE

### **Get Parameter-Invert-END-Switch1**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_ENDSW1

### **Get Parameter-Invert-END-Switch2**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_ENDSW2

### **Get Parameter-Invert-Ref-Switch1**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_REFSW1

### **Get Parameter-Invert-Ref-Switch2**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_REFSW2

### **Get Parameter-Invert-direction**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_DIRECTION

### **Get Parameter-Endswitch-Stopmode**

par1= DAPI\_STEPPER\_MOTORCHAR\_PAR\_ENDSWITCH\_STOPMODE

### **Get Parameter-GoReferenceFrequency (WARNING: This parameter will not be supported anymore)**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY -  
(maximum value=5000)

Remark: This parameter is replaced completely by the following three parameters.

### **Get Parameter-GoReferenceFrequencyToEndSwitch**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_TOE  
NDSWITCH

### **Get Parameter-GoReferenceFrequencyAfterEndSwitch**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_AFT  
ERENDSWITCH

### **Get Parameter-GoReferenceFrequencyToOffSet**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_TO  
OFFSET

## **Return value**

### **Parameter-Stepmode**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STEPMODE

return=0 (Full-step)

return=1 (1/2-step)

return=2 (1/4-step)

return=3 (1/8-step)

return=4 (1/16-step)

### **Parameter-GO-Frequency**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOFREQUENCY

return=Speed [Full-step / s] - related to full-step

### **Parameter-Start-Frequency**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STARTFREQUENCY

return=Startfrequency [Full-step / s]

### **Parameter-Stop-Frequency**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STOPFREQUENCY

return=Stopfrequency [Full-step / s]

### **Parameter-Max-Frequency**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_MAXFREQUENCY

return=maximum frequency [Full-step / s]

### **Parameter-Accelerationslope**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_ACCELERATIONSLOPE

return=Acceleration slope [Full-step / 10ms]

### **Parameter-Decelerationslope**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_DECELERATIONSLOPE

return=Deceleration slope [Full-step / 10ms]

### **Parameter-Phasecurrent**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_PHASECURRENT  
return=Phase current [mA]

### **Parameter-Hold-Phasecurrent**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_HOLDPHASECURRENT  
return=Phase current for motor hold [mA]

### **Parameter-Hold-Time**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_HOLDTIME  
return=Time in that the hold goes to motorstop [ms]  
return=-1 / FFFF hex / 65535 dez (endless time)

### **Parameter-Status-LED-Mode**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_STATUSLEDMODE  
return=Mode of the Status-LED  
return=0 (MOVE - LED is on if the stepper moves)  
return=1 (HALT - LED is on if the stepper stands still)  
return=2 (ENDSW1 - LED is on if the end switch1 is closed)  
return=3 (ENDSW2 - LED is on if the end switch2 is closed)  
return=4 (REFSW1 - LED is on if the Reference switch1 is closed)  
return=5 (REFSW2 - LED is on if the Reference switch2 is closed)

### **Parameter-Invert-END-Switch1**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_ENDSW1  
return=invert endswitch1 (0=normal / 1=inverted)

### **Parameter-Invert-END-Switch2**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_ENDSW2  
return=invert endswitch2 (0=normal / 1=inverted)

### **Parameter-Invert-Ref-Switch1**

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_REFSW1  
return=invert referenceswitch1 (0=normal / 1=inverted)

### Parameter-Invert-Ref-Switch2

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_REFSW2  
return=invert referenceswitch2 (0=normal / 1=inverted)

### Parameter-Invert-direction

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_INVERT\_DIRECTION  
return=invert all direction details (0=normal / 1=inverted)

### Parameter-Endswitch-Stopmode

par1= DAPI\_STEPPER\_MOTORCHAR\_PAR\_ENDSWITCH\_STOPMODE  
return=setting of the stop behaviour (0=Fullstop / 1=Stop)

### Parameter-GoReferenceFrequencyToEndSwitch

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_TOE  
NDSWITCH  
return=frequency [Full-step / s]

### Parameter-GoReferenceFrequencyAfterEndSwitch

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_AFT  
ERENDSWITCH  
return=frequency [Full-step / s]

### Parameter-GoReferenceFrequencyToOffSet

par1=DAPI\_STEPPER\_MOTORCHAR\_PAR\_GOREFERENCEFREQUENCY\_TO  
OFFSET  
return=frequency [Full-step / s]

### Example program

```
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

#### Description

The current motor characteristic will be stored in the EEPROM.

#### Definition

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

#### 4.3.1.13.

### DAPI\_STEPPER\_CMD\_MOTORCHARACTERISTIC\_EEPROM\_LOAD

#### Description

The motor characteristic can be loaded from the EEPROM.

#### Definition

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

#### 4.3.1.14.

### DAPI\_STEPPER\_CMD\_MOTORCHARACTERISTIC\_LOAD\_DEFAULT

#### Description

The characteristic of the motor is set back to default.

#### Definition

*DapiStepperCommand(handle, motor,*  
*DAPI\_STEPPER\_CMD\_MOTORCHARACTERISTIC\_LOAD\_DEFAULT, 0, 0, 0, 0);*

#### Remarks

The default values are:

- Stepmode: Full-step
- Step frequency at GoPosition [Full-step / s]: 1000 Hz
- Start frequency [Full-step / s]: 200Hz
- Stop frequency [Full-step / s]: 200Hz
- Maximal step frequency [Full-step / s]: 3000Hz
- Acceleration slope [Hz/10ms]: 10Hz/10ms
- Deceleration slope [Hz/10ms]: 10Hz/10ms
- Phase current 0..1,5A [1mA]: 750mA
- Hold current 0..1,5A [1mA]: 500mA
- Hold time 0..infinite [ms]: 15000ms
- Status\_LED-function: Move
- Function of the Endswitch1: not inverted
- Function of the Endswitch2: not inverted
- Function of the Referenceswitch1: not inverted
- Function of the Referenceswitch2: not inverted
- Function of all direction details: not inverted
- Endswitchmode: Fullstop
- Step frequency at GoReference [Full-step / s]: 1000 Hz

#### 4.3.1.15. DAPI\_STEPPER\_CMD\_GO\_REFSWITCH

##### Description

The motor goes to the referece position.

##### Definition

*DapiStepperCommand(handle, motor, DAPI\_STEPPER\_CMD\_GO\_REFSWITCH, par1, par2, par3, 0);*

##### Parameters

Values for par1: (if multiple are needed, the individual must be added)

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_REF1

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_REF2

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_REF\_LEFT

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_REF\_RIGHT

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_REF\_GO\_POSITIVE

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_REF\_GO\_NEGATIVE

DAPI\_STEPPER\_GO\_REFSWITCH\_PAR\_SET\_POS\_0

par2=Motorpositionsoffset (1/16 Full-step)

par3=Timeout [ms]

##### Remarks

At first the motor goes to referenceposition 1 or 2 (see par1). Therefor the speed GOREFERENCFREQUENCY\_TOENDSWITCH is used for (see DapiStepperCommand\_SetMotorcharacteristic). After this the motor goes with the speed GOREFERENCFREQUENCY\_AFTERENDSWITCH out of the Referenceposition.

If there is declaration of an offset as parameter (par2), the motor will go to this offset with the speed GOREFERENCFREQUENCY\_TOOFFSET

##### Example program

```
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

##### **Description**

The temperature of the CPU can be read.

##### **Definition**

```
ULONG DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_CPU_TEMP);
```

##### **Parameters**

cmd=DAPI\_STEPPER\_CMD\_GET\_CPU\_TEMP

##### **Return value**

temperature [°C]

#### 4.3.1.17. DAPI\_STEPPER\_CMD\_GET\_MOTOR\_SUPPLY\_VOLTAGE

##### **Description**

The voltage supply of the CPU can be read.

##### **Definition**

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

##### **Parameters**

cmd=DAPI\_STEPPER\_CMD\_GET\_MOTOR\_SUPPLY\_VOLTAGE

##### **Return value**

Motor voltage supply in [mV]

## 4.3.2. DapiStepperGetStatus

### 4.3.2.1. DAPI\_STEPPER\_STATUS\_GET\_ACTIVITY

#### Description

With this command, some status informations (e.g. activity of the motor phase current) can be read.

#### Definition

```
ULONG DapiStepperGetStatus(handle, motor,  
DAPI_STEPPER_STATUS_GET_ACTIVITY);
```

#### Parameters

handle=This is the handle of an opened module

motor=Number of addressed motor (1,2 ..)

#### Return value

Bit	Command	Description
0	DISABLE	Motor is disabled
1	MOTORSTROMACTIV	Motor phase current is active
2	HALTESTROMACTIV	Hold phase current is active
3	GOPOSITIONACTIV	GoPosition is active
4	GOPOSITIONBREMSSEN	GoPosition deceleration is active
5	GOREFERENZACTIV	GoReference is active

#### Example program

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

#### 4.3.2.2. DAPI\_STEPPER\_STATUS\_GET\_POSITION

##### Description

With this command, the motor position can be read.

##### Definition

*ULONG DapiStepperGetStatus(handle, motor, cmd);*

##### Parameters

cmd=DAPI\_STEPPER\_STATUS\_GET\_POSITION

##### Return value

The current motor position in 1/16 step-units can be read back

##### Example program

```
value = DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_POSITION);
```

### 4.3.2.3. DAPI\_STEPPER\_STATUS\_GET\_SWITCH

#### Description

With this command, the status of the switches can be read.

#### Definition

*ULONG DapiStepperGetStatus(handle, motor, cmd);*

#### Parameters

cmd=DAPI\_STEPPER\_STATUS\_GET\_SWITCH

#### Return value

Status of the switches will be delivered back

Bit0: ENDSWITCH1; 1 = Endswitch1 is closed

Bit1: ENDSWITCH2; 1 = Endswitch2 is closed

Bit2: REFSWITCH1; 1 = Referenceswitch1 is closed

Bit3: REFSWITCH2; 1 = Referenceswitch2 is closed

#### Example program

```
pos = DapiStepperGetStatus(handle, motor, DAPI_STEPPER_STATUS_GET_SWITCH);
```

### 4.3.3. DapiStepperCommandEx

#### Description

This extended command controls stepper motors.

#### Definition

*ULONG DapiStepperCommandEx(ULONG handle, ULONG motor, ULONG cmd, ULONG par1, ULONG par2, ULONG par3, ULONG par4, ULONG par5, ULONG par6, ULONG par7);*

#### Parameters

handle=This is the handle of an opened module

motor=Number of addressed motor (1,2 ..)

cmd=Extended command

par1..7=Extended command-depedent parameter (see remarks)

#### Remarks

See delib.h for the extended commands and parameters.

## 4.4. Example program

```
// *****
// *****
// *****
// *****
// *****
//
// (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 ;
}

```

# Appendix



## 5. Appendix

### 5.1. Revisions

Rev 1.00	First issue
Rev 2.00	Design change
Rev 2.01	Pinout modification Update of the DELIB functions
Rev 2.02	Supplement of DELIB functions "DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC", "DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC" and "DAPI_STEPPER_CMD_GO_REFSWITCH"
Rev 2.03	Supplement of return value for command "DAPI_STEPPER_STATUS_GET_ACTIVITY" Supplement of parameter hold-time (endless time) at command "DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC"

## 5.2. Copyrights and trademarks

Linux is registered trade-mark of Linus Torvalds.

Windows CE is registered trade-mark of Microsoft Corporation.

USB is registered trade-mark of USB Implementers Forum Inc.

LabVIEW is registered trade-mark of National Instruments.

Intel is registered trade-mark of Intel Corporation

AMD is registered trade-mark of Advanced Micro Devices, Inc.