

***EPOS***  
***Positioning Controllers***  
***Command Library***

*Document ID: rel4795*

## PLEASE READ THIS FIRST



***These instructions are intended for qualified technical personnel. Prior commencing with any activities ...***

- you must carefully read and understand this manual and
- you must follow the instructions given therein.

We have tried to provide you with all information necessary to install and commission the equipment in a **secure, safe, and time-saving** manner. Our main focus is ...

- to familiarize you with all relevant technical aspects,
- to let you know the easiest way of doing,
- to alert you of any possibly dangerous situation you might encounter or that you might cause if you do not follow the description,
- to **write as little** and to **say as much** as possible and
- not to bore you with things you already know.

Likewise, we tried to skip repetitive information! Thus, you will find things **mentioned just once**. If, for example, an earlier mentioned action fits other occasions you then will be directed to that text passage with a respective reference.



***Follow any stated reference – observe respective information – then go back and continue with the task!***

## PREREQUISITES FOR PERMISSION TO COMMENCE INSTALLATION

The **EPOS** is considered as partly completed machinery according to EU directive 2006/42/EC, Article 2, Clause (g) and therefore **is intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment**.



***You must not put the device into service, ...***

- unless you have made completely sure that the other machinery – the surrounding system the device is intended to be incorporated to – fully complies with the requirements stated in EU directive 2006/42/EC!
- unless the surrounding system fulfills all relevant health and safety aspects!
- unless all respective interfaces have been established and fulfill the stated requirements!

---

**TABLE OF CONTENTS**

<b>1</b>	<b>About this Document</b>	<b>5</b>
<b>2</b>	<b>Introduction</b>	<b>9</b>
<b>3</b>	<b>Initialization Functions</b>	<b>13</b>
3.1	Communication .....	13
3.2	Info.....	17
3.3	Advanced Functions .....	19
<b>4</b>	<b>Configuration Functions</b>	<b>25</b>
4.1	General .....	25
4.2	Advanced Functions .....	28
<b>5</b>	<b>Operation Functions</b>	<b>47</b>
5.1	Operation Mode.....	47
5.2	State Machine .....	49
5.3	Error Handling .....	54
5.4	Motion Info.....	56
5.5	Profile Position Mode (PPM) .....	60
5.6	Profile Velocity Mode (PVM) .....	64
5.7	Homing Mode (HM) .....	68
5.8	Interpolated Position Mode (IPM) .....	73
5.9	Position Mode (PM).....	78
5.10	Velocity Mode (VM) .....	82
5.11	Current Mode (CM) .....	86
5.12	Master Encoder Mode (MEM) .....	90
5.13	Step Direction Mode (SDM).....	92
5.14	Inputs & Outputs .....	94
<b>6</b>	<b>Data Recording Functions</b>	<b>105</b>
6.1	Operation Mode.....	105
6.2	Data Recorder Status .....	108
6.3	Data Recorder Data.....	110
6.4	Advanced Functions .....	112
<b>7</b>	<b>Low Layer Functions</b>	<b>115</b>
<b>8</b>	<b>Error Overview</b>	<b>117</b>
8.1	Communication Errors .....	117
8.2	EPOS Command Library-specified Errors .....	118

---

<b>9    Integration</b>	<b>121</b>
9.1 Windows Operating Systems .....	121
9.1.1 Library Hierarchy .....	121
9.1.2 Integration into Programming Environment .....	121
9.1.3 Programming .....	130
9.2 Linux Operating Systems .....	133
9.2.1 Library Hierarchy .....	133
9.2.2 Framework Conditions .....	133
9.2.3 Integration into Programming Environment .....	133
9.2.4 Installation .....	134
9.2.5 HelloEposCmd .....	135
<b>10   Version History</b>	<b>137</b>

# 1 About this Document

**We strongly stress the following facts:**

- The present document does not replace any other documentation covering the basic installation and/or parameterization described therein!
- Also, any aspect in regard to health and safety as well as to secure and safe operation are not covered in the present document – it is intended and must be understood as complimenting addition to those documents!

## 1.1 Intended Purpose

The present document provides instructions on the implemented functions of the...

- Windows Dynamic-Link Libraries «EposCmd.dll» and «EposCmd64.dll», as well as the
- Linux 32-Bit Shared Object Library «libEposCmd.so»

...which can be used for EPOS and EPOS2 devices.

In addition, the document explains on how to integrate the DLLs into a variety of common programming environments.

## 1.2 Target Audience

This document is meant for trained and skilled personnel working with the equipment described. It conveys information on how to understand and fulfill the respective work and duties.

This document is a reference book. It does require particular knowledge and expertise specific to the equipment described.

## 1.3 How to use

Take note of the following notations and codes which will be used throughout the document.

Notation	Explanation
«Abcd»	indicating a title or a name (such as of document, product, mode, etc.)
¤Abcd¤	indicating an action to be performed using a software control element (such as folder, menu, drop-down menu, button, check box, etc.) or a hardware element (such as switch, DIP switch, etc.)
(n)	referring to an item (such as order number, list item, etc.)
➔	denotes “see”, “see also”, “take note of” or “go to”

Table 1-1      Notations used in this Document

## 1.4 Symbols and Signs



### Requirement / Note / Remark

Indicates an action you must perform prior continuing or refers to information on a particular item.



### Best Practice

Gives advice on the easiest and best way to proceed.



### Material Damage

Points out information particular to potential damage of equipment.

## 1.5 Sources for additional Information

For further details and additional information, please refer to below listed sources:

Topic	Reference
Eclipse	<a href="http://eclipse.org/">http://eclipse.org/</a>
FTDI Driver	<a href="http://www.ftdichip.com">www.ftdichip.com</a>
Functions	Not all functions are supported by all devices as they are dependent on the device version and the firmware version. For details → separate documents «Firmware Specification» and «Hardware Reference» of the respective positioning controller.
Index / Subindex	For detailed descriptions on used objects → separate document «Firmware Specification».
IXXAT	<a href="http://www.ixxat.de">www.ixxat.de</a>
Kvaser	<a href="http://www.kvaser.com">www.kvaser.com</a>
maxon motor	<a href="http://www.maxonmotor.com">www.maxonmotor.com</a>
Microsoft Developer Network (MSDN)	<a href="http://msdn.microsoft.com/">http://msdn.microsoft.com/</a>
National Instruments (NI)	<a href="http://www.ni.com/can">www.ni.com/can</a>
Objects	Not all objects are supported by all devices as they are dependent on the device version and the firmware version. For details → separate documents «Firmware Specification» and «Hardware Reference» of the respective positioning controller.
Vector	<a href="http://www.vector-informatik.com">www.vector-informatik.com</a>

Table 1-2      Sources for additional Information

## 1.6 Trademarks and Brand Names

For easier legibility, registered brand names are listed below and will not be further tagged with their respective trademark. It must be understood that the brands (the below list is not necessarily concluding) are protected by copyright and/or other intellectual property rights even if their legal trademarks are omitted in the later course of this document.

Brand Name	Trademark Owner
Borland® Borland C++ Builder™	© Borland Software Corporation, USA-Rockville MD
CANopen® CiA®	© CiA CAN in Automation e.V, DE-Nuremberg
Eclipse™	© Eclipse Foundation, Inc., CDN-Ottawa ON
LabVIEW™ LabWindows™	© National Instruments Corporation, USA-Austin TX
Linux®	© Linus Torvalds (The Linux Foundation, USA-San Francisco CA)
NI-CAN™ NI-XNET™	© National Instruments Corporation, USA-Austin TX
Ubuntu	© Canonical Group Limited, UK-London
Visual Basic® Visual C++® Visual C#®	© Microsoft Corporation, USA-Redmond WA
Windows®	© Microsoft Corporation, USA-Redmond WA

Table 1-3 Brand Names and Trademark Owners

## 1.7 Legal Notice

The present document is based on maxon motor's experience. maxon motor explicitly states that its content is true and correct as to maxon motor's best knowledge.

Note that all legal aspects, such as terms of use, property rights, warranty, applicable law, and others are covered and valid as stated in the maxon motor's «EPOS Studio» End User License Agreement (EULA) which you have agreed to upon initial installation and prior employment of the «EPOS Studio».

## 1.8 Copyright

© 2014 maxon motor. All rights reserved.

The present document – including all parts thereof – is protected by copyright. Any use (including reproduction, translation, microfilming and other means of electronic data processing) beyond the narrow restrictions of the copyright law without the prior approval of maxon motor ag, is not permitted and subject to persecution under the applicable law.

**maxon motor ag**  
Brünigstrasse 220  
P.O.Box 263  
CH-6072 Sachseln  
Switzerland

Phone +41 41 666 15 00  
Fax +41 41 666 16 50

[www.maxonmotor.com](http://www.maxonmotor.com)

---

**••page intentionally left blank••**

## 2 Introduction

### 2.1 Documentation Structure

The present document is part of a documentation set. Find below an overview on the documentation hierarchy and the interrelationship of its individual parts:

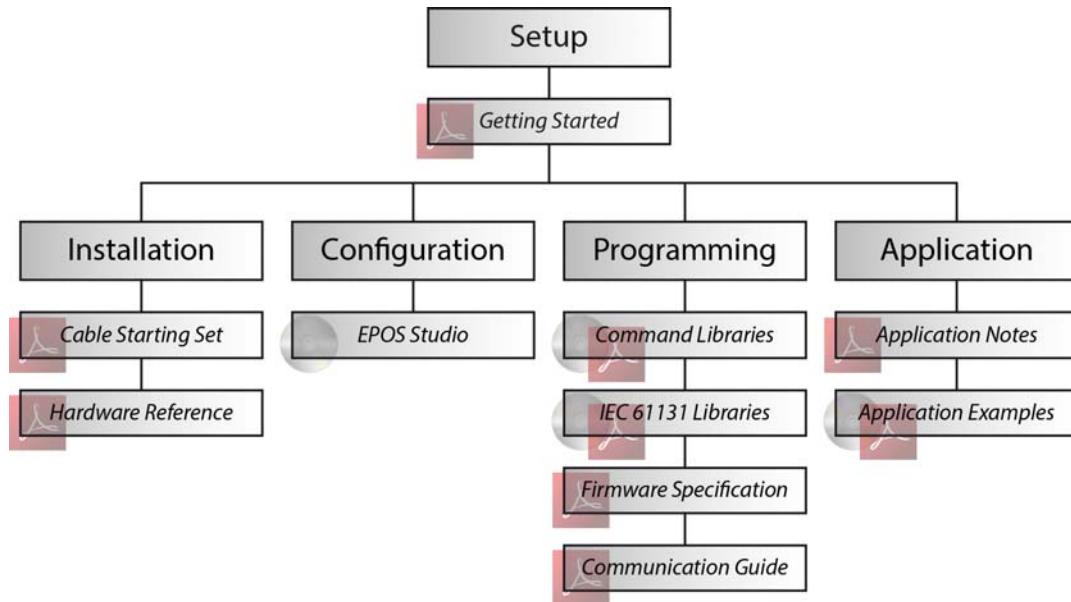


Figure 2-1 Documentation Structure

### 2.2 General Information

The «EPOS Command Libraries» are arranged in groups of functions and are intended to assist you in programming the control software based on Microsoft Windows 32-Bit and 64-Bit as well as Linux 32-Bit operating systems.

The document describes the interfaces between the control software and the libraries. They support maxon motor's EPOS devices, which are connected to a serial RS232 interface or USB port (Windows and Linux) or to a CAN board (Windows) (→Table 2-4).



#### CANopen Hardware

Use one of the listed CANopen interface cards (for details →page 2-10), for which respective motion control libraries are available. All other CANopen products may also be used, however, maxon motor does not provide respective libraries.

Interface		Platform/Architecture					
		Windows		Linux		ARMv7	
	x86	x64	x86	x86_64	SF	HF	
RS232	X	X	X	X	X	X	
USB	X	X	X	X	X	X	
CAN Board	IXXAT	X	X	—	—	—	—
	Kvaser	X	X	—	—	—	—
	NI	X	X	—	—	—	—
	Vector	X	X	—	—	—	—

Table 2-4 Supported Platforms, Architectures, and Interfaces

The parameters for 32-Bit and 64-Bit interfaces are identical. The libraries support the CANopen SDO protocol but are not suitable for real-time communication.

Refer to these chapters for in detail information on library functions and integration into your programming environment:

3 Initialization Functions . . . . .	3-13
4 Configuration Functions . . . . .	4-25
5 Operation Functions . . . . .	5-47
6 Data Recording Functions . . . . .	6-105
7 Low Layer Functions . . . . .	7-115
9 Integration . . . . .	9-121

Find the latest edition of the present document, as well as additional documentation and software to the EPOS Positioning Controllers also on the Internet: → [www.maxonmotor.com](http://www.maxonmotor.com)

## 2.3 Products by Third Party Suppliers

For manufacturers' contact information → "Sources for additional Information" on page 1-6.

Supplier	Products
<b>IXXAT</b>	All IXXAT CANopen interfaces can be operated with the hardware-independent "VCI driver V3" (Virtual CAN Interface). The older version "VCI driver V2" (2.16 and higher) is still being supported but should not be used because of lower performance.
<b>Kvaser</b>	Kvaser CAN interfaces are supported. Thereby, respective driver software and hardware must be installed.
<b>National Instruments</b>	National Instruments CAN interfaces are supported. Thereby, «NI-XNET» or «NI-CAN» software and hardware must be installed.
<b>Vector</b>	For Vector CANopen cards, the "XL-Driver-Library" will be required. The library must be manually installed in the appropriate working directory (or system directory). With this library, you may write your own CANopen applications based on Vector's CAN hardware.

Table 2-5 Third Party Supplier Products

## 2.4 Communication Structure

### 2.4.1 Windows

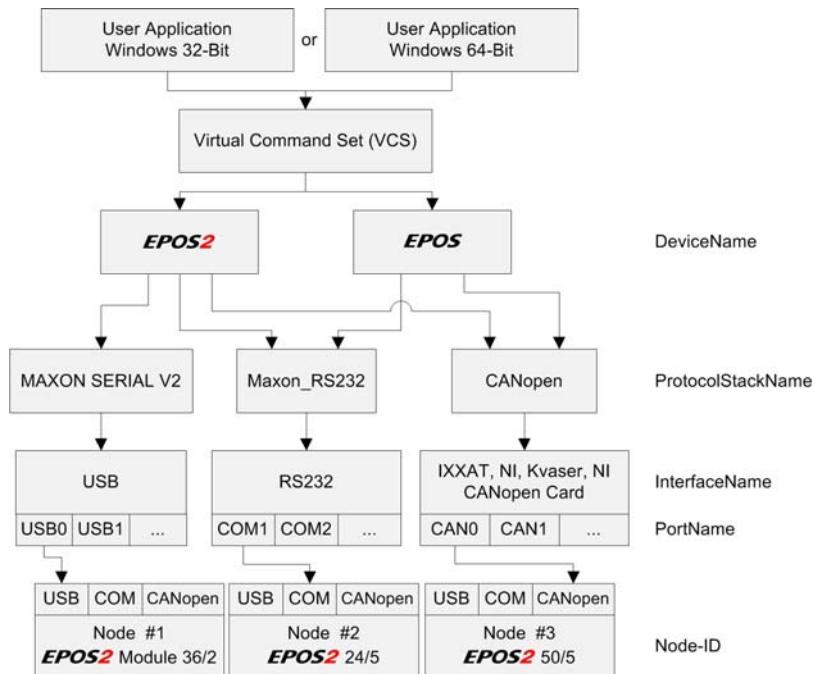


Figure 2-2 Windows – Communication Structure (Example)

### 2.4.2 Linux

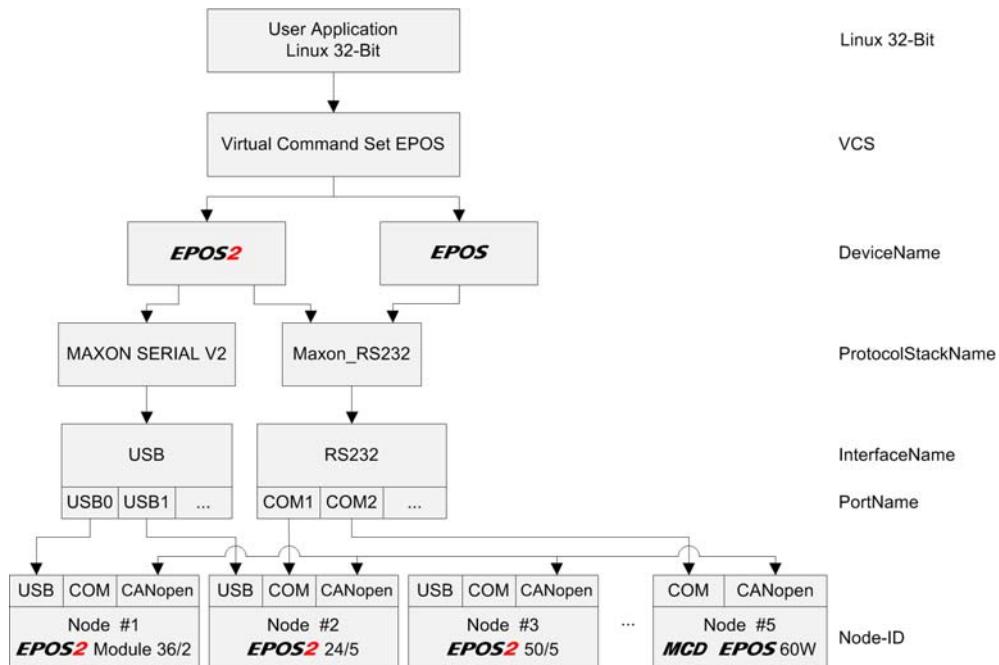


Figure 2-3 Linux – Communication Structure (Example)

## 2.5 Data Type Definitions

Name	Data Types	Size Bits	Size Bytes	Range	Comment
char, __int8	signed integer	8	1	-128...127	
BYTE	unsigned integer	8	1	0...256	
short	signed integer	16	2	-32'768...32'767	
WORD	unsigned integer	16	2	0...65'535	
long	signed integer	32	4	-2'147'483'648...2'147'483'647	
DWORD	unsigned integer	32	4	0...4'294'967'295	
BOOL	signed integer	32	4	TRUE = 1 FALSE = 0	
HANDLE	pointer to an object	32	4	0...4'294'967'295	Depending on OS
		64	8	0...18'446'744'073'709'551'615	

Table 2-6 Data Type Definitions

### 3 Initialization Functions

#### 3.1 Communication

##### 3.1.1 Open Device

###### FUNCTION

```
HANDLE VCS_OpenDevice(char* DeviceName, char* ProtocolStackName, char* InterfaceName,
char* PortName, DWORD* pErrorCode)
```

###### DESCRIPTION

VCS\_OpenDevice opens the port to send and receive commands. Ports can be RS232, USB, and CANopen interfaces.

For correct designations on DeviceName, ProtocolStackName, InterfaceName, and PortName, use the functions → *Get Device Name Selection*, → *Get Protocol Stack Name Selection*, → *Get Interface Name Selection*, and → *Get Port Name Selection*.

###### PARAMETERS

DeviceName	char*	Name of connected device: <ul style="list-style-type: none"> <li>• EPOS</li> <li>• EPOS2</li> </ul>
ProtocolStackName	char*	Name of used communication protocol: <ul style="list-style-type: none"> <li>• MAXON_RS232</li> <li>• MAXON SERIAL V2</li> <li>• CANopen</li> </ul>
InterfaceName	char*	Name of interface: <ul style="list-style-type: none"> <li>• RS232</li> <li>• USB</li> <li>• IXXAT_&lt;&lt;BoardName&gt;&gt; &lt;&lt;DeviceNumber&gt;&gt;</li> <li>• Kvaser_&lt;&lt;BoardName&gt;&gt; &lt;&lt;DeviceNumber&gt;&gt;</li> <li>• NI_&lt;&lt;BoardName&gt;&gt; &lt;&lt;DeviceNumber&gt;&gt;</li> <li>• Vector_&lt;&lt;BoardName&gt;&gt; &lt;&lt;DeviceNumber&gt;&gt;</li> </ul> <p><i>Remark:</i> Use “VCS_OpenDeviceDlg” or “VCS_GetInterfaceNameSel” to identify the exact name</p>
PortName	char*	Name of port: <ul style="list-style-type: none"> <li>• COM1, COM2, ...</li> <li>• USB0, USB1, ...</li> <li>• CAN0, CAN1, ...</li> </ul>

###### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	HANDLE	Handle for communication port access. Nonzero if successful; otherwise “0”.
--------------	--------	--

### 3.1.2 Open Device Dialog

#### FUNCTION

HANDLE VCS\_OpenDeviceDlg(DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_OpenDeviceDlg recognizes available interfaces capable to operate with EPOS and opens the selected interface for communication (not available with Linux).

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	HANDLE	Handle for communication port access. Nonzero if successful; otherwise "0".
--------------	--------	--

### 3.1.3 Set Protocol Stack Settings

#### FUNCTION

BOOL VCS\_SetProtocolStackSettings(HANDLE KeyHandle, DWORD Baudrate, DWORD Timeout, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetProtocolStackSettings writes the communication parameters. For exact values on available baud rates, use function → *Get Baud Rate Selection*.

For correct communication, use the same baud rate as the connected device.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Baudrate	DWORD	Actual baud rate from opened port [Bit/s]
Timeout	DWORD	Actual timeout from opened port [ms]

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 3.1.4 Get Protocol Stack Settings

#### FUNCTION

BOOL VCS\_GetProtocolStackSettings(HANDLE KeyHandle, DWORD\* pBaudrate, DWORD\* pTimeout, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_GetProtocolStackSettings returns the baud rate and timeout communication parameters.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
-----------	--------	------------------------

#### RETURN PARAMETERS

pBaudrate	DWORD*	Actual baud rate from opened port [Bit/s]
pTimeout	DWORD*	Actual timeout from opened port [ms]
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**3.1.5 Find Device Communication Settings****FUNCTION**

```
BOOL VCS_FindDeviceCommunicationSettings(HANDLE KeyHandle, char* pDeviceName, char*
pProtocolStackName, char* pInterfaceName, char* pPortName, WORD SizeName, DWORD*
pBaudrate, DWORD* pTimeout, WORD* pNodeId, int DialogMode, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_FindDeviceCommunicationSettings searches the communication setting parameters. Parameters can be defined to accelerate the process. Not available with Linux.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
pDeviceName	char*	Device name
pProtocolStackName	char*	Protocol stack name
pInterfaceName	char*	Interface name
pPortName	char*	Port name
SizeName	WORD	Reserved memory size for return parameters
DialogMode	int	0: Show progress dialog 1: Show progress and confirmation dialog 2: Show confirmation dialog 3: Do not show any dialog

**RETURN PARAMETERS**

pDeviceName	char*	Device name
pProtocolStackName	char*	Protocol stack name
pInterfaceName	char*	Interface name
pPortName	char*	Port name
pBaudrate	DWORD*	Baud rate
pTimeout	DWORD*	Timeout
pNodeId	WORD*	Node ID
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

### 3.1.6 Close All Devices

#### FUNCTION

BOOL VCS\_CloseAllDevices(DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_CloseAllDevices closes all opened ports and releases them for other applications.

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 3.1.7 Close Device

#### FUNCTION

BOOL VCS\_CloseDevice(HANDLE KeyHandle, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_CloseDevice closes the port and releases it for other applications.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
-----------	--------	------------------------

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 3.2 Info

### 3.2.1 Get Error Info

**FUNCTION**

BOOL VCS\_GetErrorInfo(DWORD ErrorCodeValue, char\* pErrorInfo, WORD MaxStrSize)

**DESCRIPTION**

VCS\_GetErrorInfo returns the error information on the executed function from a received error code. It returns communication and library errors (but not device error descriptions). For error codes → chapter “8 Error Overview” on page 8-117.

**PARAMETERS**

ErrorCodeValue	DWORD	Received error code
MaxStrSize	WORD	Max. length of error string

**RETURN PARAMETERS**

pErrorCode	char*	Error string
------------	-------	--------------

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 3.2.2 Get Driver Info

**FUNCTION**

BOOL VCS\_GetDriverInfo(char\* pLibraryName, WORD MaxStrNameSize, char\* pLibraryVersion, WORD MaxStrVersionSize, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_GetDriverInfo returns the name and version from the «EPOS Command Library». Not available with Linux.

**PARAMETERS**

MaxStrNameSize	WORD	Reserved memory size for the name
MaxStrVersionSize	WORD	Reserved memory size for the version

**RETURN PARAMETERS**

pLibraryName	char*	Name from the library
pLibraryVersion	char*	Version from the library
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 3.2.3 Get Version

#### FUNCTION

```
BOOL VCS_GetVersion(HANDLE KeyHandle, WORD Nodeld, WORD* pHardwareVersion, WORD*  
pSoftwareVersion, WORD* pApplicationNumber, WORD* pApplicationVersion, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetVersion returns the firmware version.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pHardwareVersion	WORD*	Hardware version	0x2003-01
pSoftwareVersion	WORD*	Software version	0x2003-02
pApplicationNumber	WORD*	Application number	0x2003-03
pApplicationVersion	WORD*	Application version	0x2003-04
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 3.3 Advanced Functions

#### 3.3.1 Get Device Name Selection

**FUNCTION**

```
BOOL VCS_GetDeviceNameSelection(BOOL StartOfSelection, char* pDeviceNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetDeviceNameSelection returns all available device names. For programming example →page 3-22.

**PARAMETERS**

StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the device name

**RETURN PARAMETERS**

pDeviceNameSel	char*	Device name
pEndOfSelection	BOOL*	True: No more selection string available False: More string available
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

#### 3.3.2 Get Protocol Stack Name Selection

**FUNCTION**

```
BOOL VCS_GetProtocolStackNameSelection(char* DeviceName, BOOL StartOfSelection, char* pProtocolStackNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetProtocolStackNameSelection returns all available protocol stack names. For programming example →page 3-22.

**PARAMETERS**

DeviceName	char*	Device name
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the name

**RETURN PARAMETERS**

pProtocolStackNameSel	char*	Pointer to available protocol stack name
pEndOfSelection	BOOL	True: No more string available False: More string available
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

### 3.3.3 Get Interface Name Selection

#### FUNCTION

```
BOOL VCS_GetInterfaceNameSelection(char* DeviceName, char* ProtocolStackName, BOOL StartOfSelection, char* pInterfaceNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetInterfaceNameSelection returns all available interface names. For programming example →page 3-22.

#### PARAMETERS

DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the interface name

#### RETURN PARAMETERS

pInterfaceNameSel	char*	Name of interface
pEndOfSelection	BOOL*	True: No more string available False: More string available
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 3.3.4 Get Port Name Selection

#### FUNCTION

```
BOOL VCS_GetPortNameSelection(char* DeviceName, char* ProtocolStackName, char* InterfaceName, BOOL StartOfSelection, char* pPortSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetPortNameSelection returns all available port names. For programming example →page 3-22.

#### PARAMETERS

DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the port name

#### RETURN PARAMETERS

pPortSel	char*	Pointer to port name
pEndOfSelection	BOOL*	True: No more string available False: More string available
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 3.3.5 Get Baud Rate Selection

#### FUNCTION

```
BOOL VCS_GetBaudrateSelection(char* DeviceName, char* ProtocolStackName, char*
InterfaceName, char* PortName, BOOL StartOfSelection, DWORD* pBaudrateSel, BOOL*
pEndOfSelection, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetBaudrateSelection returns all available baud rates for the connected port. For programming example →page 3-22.

#### PARAMETERS

DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
PortName	char*	Port name
StartOfSelection	BOOL	True: Get first selection value False: Get next selection value

#### RETURN PARAMETERS

pBaudrateSel	DWORD*	Pointer to baud rate [Bit/s]
pEndOfSelection	BOOL*	True: No more value available False: More value available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 3.3.6 Programming Example

The example shows how to read the protocol stack names of all available interfaces. Programming language is C++.



#### Note

Analogously, the example can also be used to spot device names, interface names, port names, and available baud rates.

A combination of these names is used to open the communication interface (use function → “Open Device” on page 3-13).

```
-----  
const WORD maxStrSize = 100;  
  
char* strDeviceName = "EPOS2";  
char* strProtocolStackName [maxStrSize];  
BOOL endOfSel;  
DWORD errorCode;  
  
//get first protocol stack name  
if(VCS_GetProtocolStackNameSelection(strDeviceName, TRUE, strProtocolStackName,  
maxStrSize, &endOfSel, &errorCode))  
{  
    //get next protocol stack name (as long as endOfSel == FALSE)  
    while(!endOfSel)  
    {  
        VCS_GetProtocolStackNameSelection (strDeviceName, FALSE,  
        strProtocolStackName, maxStrSize, &endOfSel, &errorCode);  
    }  
}  
}  
-----
```

### 3.3.7 Get Key Handle

#### FUNCTION

BOOL VCS\_GetKeyHandle(char\* DeviceName, char\* ProtocolStackName, char\* InterfaceName, char\* PortName, HANDLE\* pKeyHandle, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_GetKeyHandle returns the key handle from the opened interface.

#### PARAMETERS

DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
PortName	char*	Port name

#### RETURN PARAMETERS

pKeyHandle	HANDLE*	Handle for port access, if parameters are correct; otherwise 0
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 3.3.8 Get Device Name

**FUNCTION**

```
BOOL VCS_GetDeviceName(HANDLE KeyHandle, char* pDeviceName, WORD MaxStrSize,
DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetDeviceName returns the device name to corresponding handle.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
MaxStrSize	WORD	Reserved memory size for the device name

**RETURN PARAMETERS**

pDeviceName	char*	Device name
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 3.3.9 Get Protocol Stack Name

**FUNCTION**

```
BOOL VCS_GetProtocolStackName(HANDLE KeyHandle, char* pProtocolStackName, WORD
MaxStrSize, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetProtocolStackName returns the protocol stack name to corresponding handle.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
MaxStrSize	WORD	Reserved memory size for the protocol stack name

**RETURN PARAMETERS**

pProtocolStackName	char*	Pointer to the protocol stack name
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 3.3.10 Get Interface Name

#### FUNCTION

```
BOOL VCS_GetInterfaceName(HANDLE KeyHandle, char* pInterfaceName, WORD MaxStrSize,  
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetInterfaceName returns the interface name to corresponding handle.

#### PARAMETERS

KeyHandle	char*	Handle for port access
MaxStrSize	DWORD*	Reserved memory size for the interface name

#### RETURN PARAMETERS

pInterfaceName	char*	Name of interface
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 3.3.11 Get Port Name

#### FUNCTION

```
BOOL VCS_GetPortName(HANDLE KeyHandle, char* pPortName, WORD MaxStrSize, DWORD*  
pErrorCode)
```

#### DESCRIPTION

VCS\_GetPortName returns the port name to corresponding handle.

#### PARAMETERS

KeyHandle	char*	Handle for port access
MaxStrSize	DWORD*	Reserved memory size for the port name

#### RETURN PARAMETERS

pPortName	char*	Port name
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 4 Configuration Functions

For detailed information on the objects →separate document «Firmware Specification».

### 4.1 General

#### 4.1.1 Import Parameter

##### FUNCTION

```
BOOL VCS_ImportParameter(HANDLE KeyHandle, WORD Nodeld, char* pParameterFileName,
BOOL ShowDlg, BOOL ShowMsg, DWORD* pErrorCode)
```

##### DESCRIPTION

VCS\_ImportParameter writes parameters from a file to the device. Not available with Linux.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
pParameterFileName	char*	Full path of parameter file for import
ShowDlg	BOOL	Dialog is shown
ShowMsg	BOOL	Message box are activated

##### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise "0"

#### 4.1.2 Export Parameter

##### FUNCTION

```
BOOL VCS_ExportParameter(HANDLE KeyHandle, WORD Nodeld, char* pParameterFileName, char*
pFirmwareFileName, char* pUserID, char* pComment, BOOL ShowDlg, BOOL ShowMsg, DWORD*
pErrorCode)
```

##### DESCRIPTION

VCS\_ExportParameter reads all device parameters and writes them to the file. Not available with Linux.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
pParameterFileName	char*	Full path of parameter file for export
pFirmwareFileName	char*	Full path of firmware file of connected device
pUserID	char*	User name
pComment	char*	Comment
ShowDlg	BOOL	Dialog is shown
ShowMsg	BOOL	Message box are activated

##### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise "0"

### 4.1.3 Set Object

#### FUNCTION

```
BOOL VCS_SetObject(HANDLE KeyHandle, WORD Nodeld, WORD ObjectIndex, BYTE
ObjectSubIndex, void* pData, DWORD NbOfBytesToWrite, DWORD* pNbOfBytesWritten, DWORD*
pErrorCode)
```

#### DESCRIPTION

VCS\_SetObject writes an object value at the given index and subindex.

For information on object index, object subindex, and object length →separate document «Firmware Specification».

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ObjectIndex	WORD	Object index
ObjectSubIndex	BYTE	Object subindex
pData	void*	Object data
NbOfBytesToWrite	DWORD	Object length to write (number of bytes)

#### RETURN PARAMETERS

pNbOfBytesWritten	DWORD*	Object length written (number of bytes)
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 4.1.4 Get Object

#### FUNCTION

```
BOOL VCS_GetObject(HANDLE KeyHandle, WORD Nodeld, WORD ObjectIndex, BYTE
ObjectSubIndex, void* pData, DWORD NbOfBytesToRead, DWORD* pNbOfBytesRead, DWORD*
pErrorCode)
```

#### DESCRIPTION

VCS\_GetObject reads an object value at the given index and subindex.

For information on object index, object subindex, and object length →separate document «Firmware Specification».

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ObjectIndex	WORD	Object index
ObjectSubIndex	BYTE	Object subindex
NbOfBytesToRead	DWORD	Object length to read (number of bytes)

#### RETURN PARAMETERS

pData	void*	Object data
pNbOfBytesRead	DWORD*	Object length read (number of bytes)
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

#### 4.1.5 Restore

**FUNCTION**

BOOL VCS\_Restore(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_Restore restores all default parameters.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

#### 4.1.6 Store

**FUNCTION**

BOOL VCS\_Store(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_Store stores all parameters.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 4.2 Advanced Functions

### 4.2.1 Motor

#### 4.2.1.1 Set Motor Type

##### FUNCTION

BOOL VCS\_SetMotorType(HANDLE KeyHandle, WORD Nodeld, WORD MotorType, DWORD\* pErrorCode)

##### DESCRIPTION

VCS\_SetMotorType writes the motor type.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
MotorType	WORD	Type of motor (→ Table 4-7) 0x6402-00

##### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

Description	Value	Name
brushed DC motor	1	MT_DC_MOTOR
EC motor sinus commutated	10	MT_EC_SINUS_COMMUTATED_MOTOR
EC motor block commutated	11	MT_EC_BLOCK_COMMUTATED_MOTOR

Table 4-7 Motor Types

#### 4.2.1.2 Set DC Motor Parameter

##### FUNCTION

BOOL VCS\_SetDcMotorParameter(HANDLE KeyHandle, WORD Nodeld, WORD NominalCurrent, WORD MaxOutputCurrent, WORD ThermalTimeConstant, DWORD\* pErrorCode)

##### DESCRIPTION

VCS\_SetDcMotorParameter writes all DC motor parameters.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
NominalCurrent	WORD	Maximal continuous current 0x6410-01
MaxOutputCurrent	WORD	Maximal peak current 0x6410-02
ThermalTimeConstant	WORD	Thermal time constant winding 0x6410-05

##### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**4.2.1.3 Set EC Motor Parameter****FUNCTION**

```
BOOL VCS_SetEcMotorParameter(HANDLE KeyHandle, WORD Nodeld, WORD NominalCurrent,
WORD MaxOutputCurrent, WORD ThermalTimeConstant, BYTE NbOfPolePairs, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_SetEcMotorParameter writes all EC motor parameters.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
NominalCurrent	WORD	Maximal continuous current 0x6410-01
MaxOutputCurrent	WORD	Maximal peak current 0x6410-02
ThermalTimeConstant	WORD	Thermal time constant winding 0x6410-05
NbOfPolePairs	BYTE	Number of pole pairs 0x6410-03

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**4.2.1.4 Get Motor Type****FUNCTION**

```
BOOL VCS_GetMotorType(HANDLE KeyHandle, WORD Nodeld, WORD* pMotorType, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetMotorType reads the motor type.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

MotorType	WORD	Type of motor (→Table 4-7) 0x6402-00
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 4.2.1.5 Get DC Motor Parameter

### FUNCTION

```
BOOL VCS_GetDcMotorParameter(HANDLE KeyHandle, WORD Nodeld, WORD* pNominalCurrent,  
WORD* pMaxOutputCurrent, WORD* pThermalTimeConstant, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetDcMotorParameter reads all DC motor parameters.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	

### RETURN PARAMETERS

pNominalCurrent	WORD*	Maximal continuous current	0x6410-01
pMaxOutputCurrent	WORD*	Maximal peak current	0x6410-02
pThermalTimeConstant	WORD*	Thermal time constant winding	0x6410-05
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"	
--------------	------	--------------------------------------	--

## 4.2.1.6 Get EC Motor Parameter

### FUNCTION

```
BOOL VCS_GetEcMotorParameter(HANDLE KeyHandle, WORD Nodeld, WORD* pNominalCurrent,  
WORD* pMaxOutputCurrent, WORD* pThermalTimeConstant, BYTE* pNbOfPolePairs, DWORD*  
pErrorCode)
```

### DESCRIPTION

VCS\_GetEcMotorParameter reads all EC motor parameters.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	

### RETURN PARAMETERS

pNominalCurrent	WORD*	Maximal continuous current	0x6410-01
pMaxOutputCurrent	WORD*	Maximal peak current	0x6410-02
pThermalTimeConstant	WORD*	Thermal time constant winding	0x6410-05
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"	
--------------	------	--------------------------------------	--

## 4.2.2 Sensor

## 4.2.2.1 Set Sensor Type

## FUNCTION

BOOL VCS\_SetSensorType(HANDLE KeyHandle, WORD NodId, WORD SensorType, DWORD\* pErrorCode)

## DESCRIPTION

VCS\_SetSensorType writes the sensor type.

## PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
SensorType	WORD	Position Sensor Type (→Table 4-8) 0x2210-02

## RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

Description	Value	Name
Unknown sensor (undefined)	0	ST_UNKNOWN
Incremental Encoder 1 with index (3-channel)	1	ST_INC_ENCODER_3CHANNEL
Incremental Encoder 1 without index (2-channel)	2	ST_INC_ENCODER_2CHANNEL
Hall Sensors	3	ST_HALL_SENSORS
SSI Encoder binary coded	4	ST_SSI_ABS_ENCODER_BINARY
SSI Encoder Grey coded	5	ST_SSI_ABS_ENCODER_GREY

Table 4-8 Position Sensor Types

## 4.2.2.2 Set Incremental Encoder Parameter

## FUNCTION

BOOL VCS\_SetIncEncoderParameter(HANDLE KeyHandle, WORD NodId, DWORD EncoderResolution, BOOL InvertedPolarity, DWORD\* pErrorCode)

## DESCRIPTION

VCS\_SetIncEncoderParameter writes the incremental encoder parameters.

## PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
EncoderResolution	DWORD	Encoder pulse number [pulse per turn] 0x2210-01
InvertedPolarity	BOOL	Position sensor polarity 0x2210-04

## RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 4.2.2.3 Set Hall Sensor Parameter

### FUNCTION

BOOL VCS\_SetHallSensorParameter(HANDLE KeyHandle, WORD Nodeld, BOOL InvertedPolarity, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_SetHallSensorParameter writes the Hall sensor parameter.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
InvertedPolarity	BOOL	Position sensor polarity 0x2210-04

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 4.2.2.4 Set SSI Absolute Encoder Parameter

### FUNCTION

BOOL VCS\_SetSsiAbsEncoderParameter(HANDLE KeyHandle, WORD Nodeld, WORD DataRate, WORD NbOfMultiTurnDataBits, WORD NbOfSingleTurnDataBits, BOOL InvertedPolarity, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_SetSsiAbsEncoderParameter writes all parameters for SSI absolute encoder.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
DataRate	WORD	SSI encoder data rate 0x2211-01
NbOfMultiTurnDataBits	WORD	Number of bits multi turn
NbOfSingleTurnDataBits	WORD	Number of bits single turn 0x2211-02
InvertedPolarity	BOOL	Position sensor polarity 0x2210-04

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**4.2.2.5 Get Sensor Type****FUNCTION**

```
BOOL VCS_GetSensorType(HANDLE KeyHandle, WORD NodId, WORD* pSensorType, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetSensorType reads the sensor type.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pSensorType	WORD*	Position sensor type (→Table 4-8)	0x2210-02
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

**4.2.2.6 Get Incremental Encoder Parameter****FUNCTION**

```
BOOL VCS_GetIncEncoderParameter(HANDLE KeyHandle, WORD NodId, DWORD* pEncoderResolution, BOOL* plnvertedPolarity, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetIncEncoderParameter reads the incremental encoder parameters.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pEncoderResolution	DWORD	Encoder pulse number [pulse per turn]	0x2210-01
plnvertedPolarity	BOOL	Position sensor polarity	0x2210-04
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

## 4.2.2.7 Get Hall Sensor Parameter

### FUNCTION

```
BOOL VCS_GetHallSensorParameter(HANDLE KeyHandle, WORD Nodeld, BOOL* pInvertedPolarity,  
DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetHallSensorParameter reads the Hall sensor parameters.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	

### RETURN PARAMETERS

pInvertedPolarity	BOOL	Position sensor polarity	0x2210-04
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"	
--------------	------	--------------------------------------	--

## 4.2.2.8 Get SSI Absolute Encoder Parameter

### FUNCTION

```
BOOL VCS_GetSsiAbsEncoderParameter(HANDLE KeyHandle, WORD Nodeld, WORD* pDataRate,  
WORD* pNbOfMultiTurnDataBits, WORD* pNbOfSingleTurnDataBits, BOOL* pInvertedPolarity,  
DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetSsiAbsEncoderParameter reads all parameters from SSI absolute encoder.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	

### RETURN PARAMETERS

pDataRate	WORD*	SSI encoder data rate	0x2211-01
pNbOfMultiTurnDataBits	WORD*	Number of bits multi turn	
pNbOfSingleTurnDataBits	WORD*	Number of bits single turn	0x2211-02
pInvertedPolarity	BOOL*	Position sensor polarity	0x2210-04
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"	
--------------	------	--------------------------------------	--

## 4.2.3 Safety

## 4.2.3.1 Set Maximal Following Error

## FUNCTION

BOOL VCS\_SetMaxFollowingError(HANDLE KeyHandle, WORD Nodeld, DWORD MaxFollowingError, DWORD\* pErrorCode)

## DESCRIPTION

VCS\_SetMaxFollowingError writes the maximal allowed following error parameter.

## PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	
MaxFollowingError	DWORD	Maximal allowed difference of position actual value to position demand value	0x6065-00

## RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 4.2.3.2 Get Maximal Following Error

## FUNCTION

BOOL VCS\_GetMaxFollowingError(HANDLE KeyHandle, WORD Nodeld, DWORD\* pMaxFollowingError, DWORD\* pErrorCode)

## DESCRIPTION

VCS\_GetMaxFollowingError reads the maximal allowed following error parameter.

## PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	

## RETURN PARAMETERS

pMaxFollowingError	DWORD*	Maximal allowed difference of position actual value to position demand value	0x6065-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 4.2.3.3 Set Maximal Profile Velocity

#### FUNCTION

```
BOOL VCS_SetMaxProfileVelocity(HANDLE KeyHandle, WORD Nodeld, DWORD MaxProfileVelocity,  
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_SetMaxProfileVelocity writes the maximal allowed velocity.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
MaxProfileVelocity	DWORD	Used as velocity limit in a position (or velocity) move 0x607F-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 4.2.3.4 Get Maximal Profile Velocity

#### FUNCTION

```
BOOL VCS_GetMaxProfileVelocity(HANDLE KeyHandle, WORD Nodeld, DWORD*  
pMaxProfileVelocity, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetMaxProfileVelocity reads the maximal allowed velocity.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pMaxProfileVelocity	DWORD*	Used as velocity limit in a position (or velocity) move 0x607F-00
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**4.2.3.5 Set Maximal Acceleration****FUNCTION**

```
BOOL VCS_SetMaxAcceleration(HANDLE KeyHandle, WORD Nodeld, DWORD MaxAcceleration,
                           DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_SetMaxAcceleration writes the maximal allowed acceleration/deceleration.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
MaxAcceleration	DWORD	Limiter of the other acceleration/ deceleration objects 0x60C5-00

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**4.2.3.6 Get Maximal Acceleration****FUNCTION**

```
BOOL VCS_GetMaxAcceleration(HANDLE KeyHandle, WORD Nodeld, DWORD* pMaxAcceleration,
                           DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetMaxAcceleration reads the maximal allowed acceleration/deceleration.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pMaxAcceleration	DWORD*	Limiter of the other acceleration/ deceleration objects 0x60C5-00
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 4.2.4 Position Regulator

### 4.2.4.1 Set Position Regulator Gain

#### FUNCTION

BOOL VCS\_SetPositionRegulatorGain(HANDLE KeyHandle, WORD Nodeld, WORD P, WORD I, WORD D, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetPositionRegulatorGain writes all position regulator gains. Determine the optimal parameters using "Regulation Tuning" in «EPOS Studio».

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
P	WORD	Position regulator P-Gain 0x60FB-01
I	WORD	Position regulator I-Gain 0x60FB-02
D	WORD	Position regulator D-Gain 0x60FB-03

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 4.2.4.2 Set Position Regulator Feed Forward

#### FUNCTION

BOOL VCS\_SetPositionRegulatorFeedForward(HANDLE KeyHandle, WORD Nodeld, WORD VelocityFeedForward, WORD AccelerationFeedForward, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetPositionRegulatorFeedForward writes parameters for position regulation with feed forward.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
VelocityFeedForward	WORD	Velocity feed forward factor 0x60FB-04
AccelerationFeedForward	WORD	Acceleration feed forward factor 0x60FB-05

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**4.2.4.3 Get Position Regulator Gain****FUNCTION**

```
BOOL VCS_GetPositionRegulatorGain(HANDLE KeyHandle, WORD Nodeld, WORD* pP, WORD* pl,
WORD* pD, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetPositionRegulatorGain reads all position regulator gains.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pP	WORD	Position regulator P-Gain	0x60FB-01
pl	WORD	Position regulator I-Gain	0x60FB-02
pD	WORD	Position regulator D-Gain	0x60FB-03
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**4.2.4.4 Get Position Regulator Feed Forward****FUNCTION**

```
BOOL VCSGetPositionRegulatorFeedForward(HANDLE KeyHandle, WORD Nodeld, WORD* pVelocityFeedForward, WORD* pAccelerationFeedForward, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetPositionRegulatorFeedForward reads parameters for position regulation feed forward.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pVelocityFeedForward	WORD*	Velocity feed forward factor	0x60FB-04
pAccelerationFeedFor-	WORD*	Acceleration feed forward factor	0x60FB-05
ward	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 4.2.5 Velocity Regulator

### 4.2.5.1 Set Velocity Regulator Gain

#### FUNCTION

BOOL VCS\_SetVelocityRegulatorGain(HANDLE KeyHandle, WORD Nodeld, WORD P, WORD I, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetVelocityRegulatorGain writes all velocity regulator gains. Determine the optimal parameters using "Regulation Tuning" in «EPOS Studio».

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
P	WORD	Velocity regulator P-Gain 0x60F9-01
I	WORD	Velocity regulator I-Gain 0x60F9-02

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 4.2.5.2 Set Velocity Regulator Feed Forward

#### FUNCTION

BOOL VCS\_SetVelocityRegulatorFeedForward(HANDLE KeyHandle, WORD Nodeld, WORD VelocityFeedForward, WORD AccelerationFeedForward, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetVelocityRegulatorFeedForward writes parameters for velocity regulation with feed forward.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
VelocityFeedForward	WORD	Velocity feed forward factor 0x60F9-04
AccelerationFeedForward	WORD	Acceleration feed forward factor 0x60F9-05

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**4.2.5.3 Get Velocity Regulator Gain****FUNCTION**

```
BOOL VCS_GetVelocityRegulatorGain(HANDLE KeyHandle, WORD NodId, WORD* pP, WORD* pl,
DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetVelocityRegulatorGain reads all velocity regulator gains.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pP	WORD*	Velocity regulator P-Gain	0x60F9-01
pl	WORD*	Velocity regulator I-Gain	0x60F9-02
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**4.2.5.4 Get Velocity Regulator Feed Forward****FUNCTION**

```
BOOL VCS_GetVelocityRegulatorFeedForward(HANDLE KeyHandle, WORD NodId, WORD* pVelocityFeedForward,
WORD* pAccelerationFeedForward, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetVelocityRegulatorFeedForward reads parameters for velocity regulation feed forward.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pVelocityFeedForward	WORD*	Velocity feed forward factor	0x60F9-04
pAccelerationFeedFor-	WORD*	Acceleration feed forward factor	0x60F9-05
ward	WORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 4.2.6 Current Regulator

### 4.2.6.1 Set Current Regulator Gain

#### FUNCTION

BOOL VCS\_SetCurrentRegulatorGain(HANDLE KeyHandle, WORD Nodeld, WORD P, WORD I, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetCurrentRegulatorGain writes all current regulator gains. Determine the optimal parameters using "Regulation Tuning" in «EPOS Studio».

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
P	WORD	Current regulator P-Gain 0x60F6-01
I	WORD	Current regulator I-Gain 0x60F6-02

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 4.2.6.2 Get Current Regulator Gain

#### FUNCTION

BOOL VCS\_GetCurrentRegulatorGain(HANDLE KeyHandle, WORD Nodeld, WORD\* pP, WORD\* pl, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_GetCurrentRegulatorGain enables reading all current regulator gains.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

P	WORD*	Current regulator P-Gain 0x60F6-01
I	WORD*	Current regulator I-Gain 0x60F6-02
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 4.2.7 Inputs/Outputs

## 4.2.7.1 Digital Input Configuration

**FUNCTION**

BOOL VCS\_DigitalInputConfiguration(HANDLE KeyHandle, WORD NodeId, WORD DigitalInputNb,  
WORD Configuration, BOOL Mask, BOOL Polarity, BOOL ExecutionMask, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_DigitalInputConfiguration sets the parameter for one digital input.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
DigitalInputNb	WORD	Number of digital input (object subindex) 0x2070-0x
Configuration	WORD	Configures the functionality assigned to the digital input (bit number) (→Table 4-9)
Mask	BOOL	1: Functionality state will be displayed 0: not displayed 0x2071-02
Polarity	BOOL	1: Low active 0: High active 0x2071-03
ExecutionMask	BOOL	1: Set the error routine Only for positive and negative switch 0x2071-04

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

Description	Value	Name
General purpose A	15	DIC_GENERAL_PURPOSE_A
General purpose B	14	DIC_GENERAL_PURPOSE_B
General purpose C	13	DIC_GENERAL_PURPOSE_C
General purpose D	12	DIC_GENERAL_PURPOSE_D
General purpose E	11	DIC_GENERAL_PURPOSE_E
General purpose F	10	DIC_GENERAL_PURPOSE_F
General purpose G	9	DIC_GENERAL_PURPOSE_G
General purpose H	8	DIC_GENERAL_PURPOSE_H
General purpose I	7	DIC_GENERAL_PURPOSE_I
General purpose J	6	DIC_GENERAL_PURPOSE_J
Quick stop	5	DIC_QUICK_STOP
Device enable	4	DIC_DRIVE_ENABLE
Position marker	3	DIC_POSITION_MARKER
Home switch	2	DIC_HOME_SWITCH
Positive limit switch	1	DIC_POSITIVE_LIMIT_SWITCH
Negative limit switch	0	DIC_NEGATIVE_LIMIT_SWITCH

Table 4-9 Digital Input Configuration

## 4.2.7.2 Digital Output Configuration

### FUNCTION

```
BOOL VCS_DigitalOutputConfiguration(HANDLE KeyHandle, WORD Nodeld, WORD DigitalOutputNb,  
WORD Configuration, BOOL State, BOOL Mask, BOOL Polarity, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_DigitalOutputConfiguration sets parameter for one digital output.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
DigitalOutputNb	WORD	Number of digital output (object subindex) 0x2079-0x
Configuration	WORD	Configures the functionality assigned to the digital output (bit number) (→Table 4-10)
State	BOOL	State of digital output 0x2078-01
Mask	BOOL	1: Register will be modified 0x2078-02
Polarity	BOOL	1: Output will be inverted 0x2078-03

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

Description	Value	Name
General purpose A	15	DIC_GENERAL_PURPOSE_A
General purpose B	14	DIC_GENERAL_PURPOSE_B
General purpose C	13	DIC_GENERAL_PURPOSE_C
General purpose D	12	DIC_GENERAL_PURPOSE_D
General purpose E	11	DIC_GENERAL_PURPOSE_E
Position compare	1	DOC_POSITION_COMPARE
Ready / Fault	0	DOC_READY_FAULT

Table 4-10     Digital Output Configuration

**4.2.7.3 Analog Input Configuration****FUNCTION**

```
BOOL VCS_AnalogInputConfiguration(HANDLE KeyHandle, WORD Nodeld, WORD AnalogInputNb,
WORD Configuration, BOOL Mask, ExecutionMask, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_DigitalOutputConfiguration sets parameter for one digital output.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNb	WORD	Number of analog input (object subindex) 0x207B-0x
Configuration	WORD	Configures the functionality assigned to the analog input (bit number) (→Table 4-11)
ExecutionMask	BOOL	1: Register will be modified 0x207D-00

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

Description	Value	Name
Analog position set point	2	AIC_ANALOG_POSITION_SETPOINT
Analog velocity set point	1	AIC_ANALOG_VELOCITY_SETPOINT
Analog current set point	0	AIC_ANALOG_CURRENT_SETPOINT

Table 4-11 Analog Input Configuration

## 4.2.8 Units

### 4.2.8.1 Set Velocity Units

#### FUNCTION

```
BOOL VCS_SetVelocityUnits(HANDLE KeyHandle, WORD Nodeld, BYTE VelDimension, char
VelNotation, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_SetVelocityUnits writes velocity unit parameters.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
VelDimension	BYTE	Velocity dimension index VD_RPM = 0xA4 0x608C-00
VelNotation	char	Velocity notation index (→Table 4-12) 0x608B-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

Description	Value	Name
Standard	0	VN_STANDARD
Deci (10 <sup>-1</sup> )	-1	VN_DECI
Centi (10 <sup>-2</sup> )	-2	VN_CENTI
Milli (10 <sup>-3</sup> )	-3	VN_MILLI

Table 4-12 Velocity Notation Index

### 4.2.8.2 Get Velocity Units

#### FUNCTION

```
BOOL VCS_GetVelocityUnits(HANDLE KeyHandle, WORD Nodeld, BYTE* pVelDimension, char*
pVelNotation, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetVelocityUnits reads velocity unit parameters.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pVelDimension	BYTE*	Velocity dimension index VD_RPM = 0xA4 0x608C-00
pVelNotation	char*	Velocity notation index (→Table 4-12) 0x608B-00
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5 Operation Functions

### 5.1 Operation Mode

#### 5.1.1 Set Operation Mode

##### FUNCTION

BOOL VCS\_SetOperationMode(HANDLE KeyHandle, WORD Nodeld, \_\_int8 Mode, DWORD\* pErrorCode)

##### DESCRIPTION

VCS\_SetOperationMode sets the operation mode.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
Mode	__int8	Operation mode (→Table 5-13) 0x6060-00

##### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

Description	Value	Name
Position Profile Mode	1	OMD_PROFILE_POSITION_MODE
Position Velocity Mode	3	OMD_PROFILE_VELOCITY_MODE
Homing Mode	6	OMD_HOMING_MODE
Interpolated Position Mode	7	OMD_INTERPOLATED_POSITION_MODE
Position Mode	-1	OMD_POSITION_MODE
Velocity Mode	-2	OMD_VELOCITY_MODE
Current Mode	-3	OMD_CURRENT_MODE
Master Encoder Mode	-5	OMD_MASTER_ENCODER_MODE
Step Direction Mode	-6	OMD_STEP_DIRECTION_MODE

Table 5-13 Operation Modes

## 5.1.2 Get Operation Mode

### FUNCTION

```
BOOL VCS_GetOperationMode(HANDLE KeyHandle, WORD Nodeld, __int8* pMode, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetOperationMode returns the activated operation mode.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pMode	__int8*	Operation mode (→Table 5-13)	0x6061-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.2 State Machine

For detailed information on the state machine →separate document «Firmware Specification».

### 5.2.1 Reset Device

#### FUNCTION

BOOL VCS\_ResetDevice(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ResetDevice is used to send the NMT service “Reset Node”. Command is without acknowledge.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.2.2 Set State

#### FUNCTION

BOOL VCS\_SetState(HANDLE KeyHandle, WORD Nodeld, WORD State, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetState reads the actual state machine state.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
State	WORD	Value of state machine (→Table 5-14) 0x6040-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

Description	Value	Name
Get/Set Disable State	0x0000	ST_DISABLED
Get/Set Enable State	0x0001	ST_ENABLED
Get/Set Quickstop State	0x0002	ST_QUICKSTOP
Get Fault State	0x0003	ST_FAULT

Table 5-14 State Modes

## 5.2.3 Set Enable State

### FUNCTION

BOOL VCS\_SetEnableState(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_SetEnableState changes the device state to “enable”.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.2.4 Set Disable State

### FUNCTION

BOOL VCS\_SetDisableState(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_SetDisableState changes the device state to “disable”.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.2.5 Set Quick Stop State

### FUNCTION

BOOL VCS\_SetQuickStopState(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_SetQuickStopState changes the device state to “quick stop”.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.2.6 Clear Fault****FUNCTION**

```
BOOL VCS_ClearFault(HANDLE KeyHandle, WORD NodId, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ClearFault changes the device state from “fault” to “disable”.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

**5.2.7 Get State****FUNCTION**

```
BOOL VCS_GetState(HANDLE KeyHandle, WORD NodId, WORD* pState, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetState reads the new state of the state machine.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pState	WORD*	Control word value (→Table 5-14)	0x6040-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

## 5.2.8 Get Enable State

### FUNCTION

```
BOOL VCS_GetEnableState(HANDLE KeyHandle, WORD Nodeld, BOOL* pIsEnabled, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetEnableState checks if the device is enabled.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pIsEnabled	BOOL*	1: Device enabled 0: Device not enabled
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.2.9 Get Disable State

### FUNCTION

```
BOOL VCS_GetDisableState(HANDLE KeyHandle, WORD Nodeld, BOOL* pIsDisabled, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetDisableState checks if the device is disabled.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pIsDisabled	BOOL*	1: Device disabled 0: Device not disabled
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.2.10 Get Quick Stop State****FUNCTION**

`BOOL VCS_GetQuickStopState(HANDLE KeyHandle, WORD Nodeld, BOOL* plsQuickStopped, DWORD* pErrorCode)`

**DESCRIPTION**

VCS\_GetQuickStopState returns the device state quick stop.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

plsQuickStopped	BOOL*	1: Device is in quick stop state 0: Device is not in quick stop state
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.2.11 Get Fault State****FUNCTION**

`BOOL VCS_GetFaultState(HANDLE KeyHandle, WORD Nodeld, BOOL* plsInFault, DWORD* pErrorCode)`

**DESCRIPTION**

VCS\_GetFaultState returns the device state fault (plsInFault = TRUE). Get error information if the device is in fault state (→ “Error Handling” on page 5-54).

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

plsInFault	BOOL*	1: Device is in fault state 0: Device is not in fault state
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.3 Error Handling

### 5.3.1 Get Number of Device Error

#### FUNCTION

```
BOOL VCS_GetNbOfDeviceError(HANDLE KeyHandle, WORD Nodeld, BYTE* pNbDeviceError,  
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetNbOfDeviceError returns the number of actual errors that are recorded (→“Programming Example” on page 5-55).

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pNbDeviceError	BYTE*	Number of occurred device errors	0x1003-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.3.2 Get Device Error Code

#### FUNCTION

```
BOOL VCS_GetDeviceErrorCode(HANDLE KeyHandle, WORD Nodeld, BYTE ErrorNumber, DWORD*  
pDeviceErrorCode, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetDeviceErrorCode returns the error code of the selected error number.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ErrorNumber	BYTE	Number (object subindex) of device error (≥1) 0x1003-0x

#### RETURN PARAMETERS

pDeviceErrorCode	DWORD*	Actual error code from error history	0x1003-0x
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.3.3 Programming Example

The example shows how to read the error history from a device. Programming language is C++.

```
--  
//Global parameters  
HANDLE KeyHandle = 1; //handle from opened interface  
WORD NodeId = 1; //node ID from connected device  
  
//Functional parameters  
BYTE nbOfDeviceError = 0; //number of actual errors  
DWORD functionErrorCode = 0; //error code from function  
DWORD deviceErrorCode = 0; //error code from device  
  
//get number of device errors  
if(VCS_GetNbOfDeviceError (KeyHandle, NodeId, &nbOfDeviceError,  
&functionErrorCode))  
{  
    //read device error code  
    for(BYTE errorNumber = 1; subIndex <= nbOfDeviceError; errorNumber++)  
    {  
        if(!VCS_GetDeviceErrorCode(KeyHandle, NodeId, errorNumber,  
        &deviceErrorCode, &functionErrorCode))  
        {  
            break;  
        }  
    }  
--
```

## 5.4 Motion Info

### 5.4.1 Get Movement State

#### FUNCTION

```
BOOL VCS_GetMovementState(HANDLE KeyHandle, WORD Nodeld, BOOL* pTargetReached,  
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetMovementState checks if the drive has reached target.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pTargetReached	BOOL*	Drive has reached the target. Function reads actual state of bit 10 from the status word.	0x6041-00 Bit 10
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 5.4.2 Get Position Is

#### FUNCTION

```
BOOL VCSGetPositionIs(HANDLE KeyHandle, WORD Nodeld, long* pPositionIs, DWORD*  
pErrorCode)
```

#### DESCRIPTION

VCSGetPositionIs returns the position actual value.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pPositionIs	long*	Position actual value	0x6064-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.4.3 Get Velocity Is****FUNCTION**

```
BOOL VCS_GetVelocityIs(HANDLE KeyHandle, WORD Nodeld, long* pVelocityIs, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetVelocityIs reads the velocity actual value.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pVelocityIs	long*	Velocity actual value	0x606C-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.4.4 Get Velocity Is Averaged****FUNCTION**

```
BOOL VCS_GetVelocityIsAveraged(HANDLE KeyHandle, WORD Nodeld, long* pVelocityIsAveraged, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetVelocityIsAveraged reads the velocity actual averaged value.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pVelocityIsAveraged	long*	Velocity actual value averaged	0x2028-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.4.5 Get Current Is

### FUNCTION

```
BOOL VCS_GetCurrentIs(HANDLE KeyHandle, WORD Nodeld, short* pCurrentIs, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetCurrentIs returns the current actual value.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pCurrentIs	short*	Current actual value	0x6078-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.4.6 Get Current Is Averaged

### FUNCTION

```
BOOL VCS_GetCurrentIsAveraged(HANDLE KeyHandle, WORD Nodeld, short* pCurrentIsAveraged, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetCurrentIsAveraged returns the current actual averaged value.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pCurrentIsAveraged	short*	Current actual value averaged	0x2027-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.4.7 Wait For Target Reached****FUNCTION**

```
BOOL VCS_WaitForTargetReached(HANDLE KeyHandle, WORD Nodeld, DWORD Timeout,  
DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_WaitForTargetReached waits until the state is changed to target reached or until the time is up.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
Timeout	DWORD	Max. wait time [ms] until target reached

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.5 Profile Position Mode (PPM)

### 5.5.1 Activate Profile Position Mode

#### FUNCTION

BOOL VCS\_ActivateProfilePositionMode(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ActivateProfilePositionMode changes the operational mode to “profile position mode”.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.5.2 Set Position Profile

#### FUNCTION

BOOL VCS\_SetPositionProfile(HANDLE KeyHandle, WORD Nodeld, DWORD ProfileVelocity, DWORD ProfileAcceleration, DWORD ProfileDeceleration, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetPositionProfile sets the position profile parameters.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ProfileVelocity	DWORD	Position profile velocity 0x6081-00
ProfileAcceleration	DWORD	Position profile acceleration 0x6083-00
ProfileDeceleration	DWORD	Position profile deceleration 0x6084-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.5.3 Get Position Profile****FUNCTION**

```
BOOL VCS_GetPositionProfile(HANDLE KeyHandle, WORD Nodeld, DWORD* pProfileVelocity,
                           DWORD* pProfileAcceleration, DWORD* pProfileDeceleration, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetPositionProfile returns the position profile parameters.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pProfileVelocity	DWORD*	Position profile velocity	0x6081-00
pProfileAcceleration	DWORD*	Position profile acceleration	0x6083-00
pProfileDeceleration	DWORD*	Position profile deceleration	0x6084-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.5.4 Move To Position****FUNCTION**

```
BOOL VCS_MoveToPosition(HANDLE KeyHandle, WORD Nodeld, long TargetPosition, BOOL
                        Absolute, BOOL Immediately, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_MoveToPosition starts movement with position profile to target position.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	
TargetPosition	long	Target position	0x607A-00
Absolute	BOOL	TRUE starts an absolute FALSE a relative movement	0x6040-00 Bit 6
Immediately	BOOL	TRUE starts immediately FALSE waits to end of last positioning	0x6040-00 Bit 5

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.5.5 Get Target Position

### FUNCTION

```
BOOL VCS_GetTargetPosition(HANDLE KeyHandle, WORD Nodeld, long* pTargetPosition, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetTargetPosition returns the profile position mode target value.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pTargetPosition	long*	Target position	0x607A-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.5.6 Halt Position Movement

### FUNCTION

```
BOOL VCS_HaltPositionMovement(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_HaltPositionMovement stops the movement with profile deceleration.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.5.7 Advanced Functions

### 5.5.7.1 Enable Position Window

**FUNCTION**

BOOL VCS\_EnablePositionWindow(HANDLE KeyHandle, WORD Nodeld, DWORD PositionWindow, WORD PositionWindowTime, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_EnablePositionWindow activates the position window.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
PositionWindow	DWORD	Position window value
PositionWindowTime	WORD	Position window time value

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

### 5.5.7.2 Disable Position Window

**FUNCTION**

BOOL VCS\_DisablePositionWindow(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_DisablePositionWindow deactivates the position window.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.6 Profile Velocity Mode (PVM)

### 5.6.1 Activate Profile Velocity Mode

#### FUNCTION

BOOL VCS\_ActivateProfileVelocityMode(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ActivateProfileVelocityMode changes the operational mode to “profile velocity mode”.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.6.2 Set Velocity Profile

#### FUNCTION

BOOL VCS\_SetVelocityProfile(HANDLE KeyHandle, WORD Nodeld, DWORD ProfileAcceleration,  
DWORD ProfileDeceleration, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetVelocityProfile sets the velocity profile parameters.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ProfileAcceleration	WORD	Velocity profile acceleration 0x6083-00
ProfileDeceleration	WORD	Velocity profile deceleration 0x6084-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.6.3 Get Velocity Profile****FUNCTION**

```
BOOL VCS_GetVelocityProfile(HANDLE KeyHandle, WORD Nodeld, DWORD* pProfileAcceleration,
                           DWORD* pProfileDeceleration, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetVelocityProfile returns the velocity profile parameters.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pProfileAcceleration	DWORD*	Velocity profile acceleration	0x6083-00
pProfileDeceleration	DWORD*	Velocity profile deceleration	0x6084-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.6.4 Move With Velocity****FUNCTION**

```
BOOL VCS_MoveWithVelocity(HANDLE KeyHandle, WORD Nodeld, long TargetVelocity, DWORD*
                           pErrorCode)
```

**DESCRIPTION**

VCS\_MoveWithVelocity starts the movement with velocity profile to target velocity.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
TargetVelocity	long	Target velocity

**RETURN PARAMETERS**

<b>pErrorCode</b>	DWORD*	Error information on the executed function
-------------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.6.5 Get Target Velocity

### FUNCTION

```
BOOL VCS_GetTargetVelocity(HANDLE KeyHandle, WORD Nodeld, long* pTargetVelocity, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetTargetVelocity returns the profile velocity mode target value.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pTargetVelocity	long*	Target velocity	0x60FF-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.6.6 Halt Velocity Movement

### FUNCTION

```
BOOL VCS_HaltVelocityMovement(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_HaltVelocityMovement stops the movement with profile deceleration.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.6.7 Advanced Functions

## 5.6.7.1 Enable Velocity Window

**FUNCTION**

BOOL VCS\_EnableVelocityWindow(HANDLE KeyHandle, WORD NodId, DWORD VelocityWindow, WORD VelocityWindowTime, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_EnableVelocityWindow activates the velocity window.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
VelocityWindow	DWORD	Velocity window value
VelocityWindowTime	WORD	Velocity window time value

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.6.7.2 Disable Velocity Window

**FUNCTION**

BOOL VCS\_DisableVelocityWindow(HANDLE KeyHandle, WORD NodId, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_DisableVelocityWindow deactivates the velocity window.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.7 Homing Mode (HM)

### 5.7.1 Activate Homing Mode

**FUNCTION**

BOOL VCS\_ActivateHomingMode(HANDLE KeyHandle, WORD NodId, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_ActivateHomingMode changes the operational mode to “homing mode”.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 5.7.2 Set Homing Parameter

**FUNCTION**

BOOL VCS\_SetHomingParameter(HANDLE KeyHandle, WORD NodId, DWORD HomingAcceleration, DWORD SpeedSwitch, DWORD SpeedIndex, long HomeOffset, WORD CurrentThreshold, long HomePosition, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_SetHomingParameter writes all homing parameters. The parameter units depend on (position, velocity, acceleration) notation index.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
HomingAcceleration	DWORD	Acceleration for homing profile 0x609A-00
SpeedSwitch	DWORD	Speed during search for switch 0x6099-01
SpeedIndex	DWORD	Speed during search for index signal 0x6099-02
HomeOffset	long	Home offset after homing 0x607C-00
CurrentThreshold	DWORD	Current threshold for homing methods -1, -2, -3, and -4 0x2080-00
HomePosition	long	Used to assign the present position as homing position 0x2081-00

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

**5.7.3 Get Homing Parameter****FUNCTION**

```
BOOL VCS_GetHomingParameter(HANDLE KeyHandle, WORD Nodeld, DWORD*  
pHomingAcceleration, DWORD* pSpeedSwitch, DWORD* pSpeedIndex, long* pHomOffset, WORD*  
pCurrentThreshold, long* pHomPosition, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetHomingParameter reads all homing parameters. The parameter units depend on (position, velocity, acceleration) notation index.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pHomingAcceleration	DWORD*	Acceleration for homing profile	0x609A-00
pSpeedSwitch	DWORD*	Speed during search for switch	0x6099-01
pSpeedIndex	DWORD*	Speed during search for index signal	0x6099-02
pHomOffset	long*	Home offset after homing	0x607C-00
pCurrentThreshold	DWORD*	Current threshold for homing methods -1, -2, -3, and -4	0x2080-00
pHomPosition	long*	Home position value	0x2081-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.7.4 Find Home****FUNCTION**

```
BOOL VCS_FindHome(HANDLE KeyHandle, WORD Nodeld, __int8 HomingMethod, DWORD*  
pErrorCode)
```

**DESCRIPTION**

VCS\_FindHome and HomingMethod permit to find the system home (for example, a home switch).

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
HomingMethod	__int8	Homing method (→Table 5-15)

**RETURN PARAMETERS**

<b>pErrorCode</b>	DWORD*	Error information on the executed function
-------------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**HOMING METHODS**

Description	Method	Name
Actual Position	35	HM_ACTUAL_POSITION
Index Positive Speed	34	HM_INDEX_POSITIVE_SPEED
Index Negative Speed	33	HM_INDEX_NEGATIVE_SPEED
Home Switch Negative Speed	27	HM_HOME_SWITCH_NEGATIVE_SPEED
Home Switch Positive Speed	23	HM_HOME_SWITCH_POSITIVE_SPEED
Positive Limit Switch	18	HM_POSITIVE_LIMIT_SWITCH
Negative Limit Switch	17	HM_NEGATIVE_LIMIT_SWITCH
Home Switch Negative Speed & Index	11	HM_HOME_SWITCH_NEGATIVE_SPEED_AND_INDEX
Home Switch Positive Speed & Index	7	HM_HOME_SWITCH_POSITIVE_SPEED_AND_INDEX
Positive Limit Switch & Index	2	HM_POSITIVE_LIMIT_SWITCH_AND_INDEX
Negative Limit Switch & Index	1	HM_NEGATIVE_LIMIT_SWITCH_AND_INDEX
No homing operation required	0	-
Current Threshold Positive Speed & Index	-1	HM_CURRENT_THRESHOLD_NEGATIVE_SPEED_AND_INDEX
Current Threshold Negative Speed & Index	-2	HM_CURRENT_THRESHOLD_NEGATIVE_SPEED_AND_INDEX
Current Threshold Positive Speed	-3	HM_CURRENT_THRESHOLD_POSITIVE_SPEED
Current Threshold Negative Speed	-4	HM_CURRENT_THRESHOLD_NEGATIVE_SPEED

Table 5-15 Homing Methods

**5.7.5 Stop Homing****FUNCTION**

BOOL VCS\_StopHoming(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_StopHoming interrupts homing.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.7.6 Define Position****FUNCTION**

```
BOOL VCS_DefinePosition(HANDLE KeyHandle, WORD Nodeld, long HomePosition, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_DefinePosition uses homing method 35 (Actual Position) to set a new home position.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
HomePosition	long	Used to assign the present position as homing position 0x2081-00

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.7.7 Get Homing State****FUNCTION**

```
BOOL VCS_GetHomingState(HANDLE KeyHandle, WORD Nodeld, BOOL* pHomingAttained, BOOL* pHomingError, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetHomingState returns the states if the homing position is attained and if an homing error has occurred.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pHomingAttained	BOOL*	0: Homing mode not yet completed 1: Homing mode successfully terminated
pHomingError	BOOL*	0: No homing error 1: Homing error occurred
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.7.8 Wait For Homing Attained

### FUNCTION

```
BOOL VCS_WaitForHomingAttained(HANDLE KeyHandle, WORD Nodeld, DWORD Timeout,  
DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_WaitForHomingAttained waits until the homing mode is successfully terminated or until the time has elapsed.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
Timeout	DWORD	Max. wait time [ms] until target reached

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.8 Interpolated Position Mode (IPM)

### 5.8.1 Activate Interpolated Position Mode

#### FUNCTION

```
BOOL VCS_ActivateInterpolatedPositionMode(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_ActivateInterpolatedPositionMode changes the operational mode to “interpolated position mode”.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 5.8.2 Set IPM Buffer Parameter

#### FUNCTION

```
BOOL VCS_SetlpmBufferParameter(HANDLE KeyHandle, WORD Nodeld, WORD UnderflowWarningLimit, WORD OverflowWarningLimit, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_SetlpmBufferParameter sets warning borders of the data input.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
UnderflowWarningLimit	WORD	Gives lower signalization level of the data input FIFO 0x20C4-02
OverflowWarningLimit	WORD	Gives the higher signalization level of the data input FIFO 0x20C4-03

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

## 5.8.3 Get IPM Buffer Parameter

### FUNCTION

```
BOOL VCS_GetIpmBufferParameter(HANDLE KeyHandle, WORD Nodeld, WORD*  
pUnderflowWarningLimit, WORD* pOverflowWarningLimit, DWORD* pMaxBufferSize, DWORD* pEr-  
rorCode)
```

### DESCRIPTION

VCS\_GetIpmBufferParameter reads warning borders and the max. buffer size of the data input.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pUnderflowWarningLimit	WORD*	Gives lower signalization level of the data input FIFO	0x20C4-02
pOverflowWarningLimit	WORD*	Gives the higher signalization level of the data input FIFO	0x20C4-03
pMaxBufferSize	DWORD*	Provides the maximal buffer size	0x60C4-01
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.8.4 Clear IPM Buffer

### FUNCTION

```
BOOL VCS_ClearIpmBuffer(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_ClearIpmBuffer clears the input buffer and enables access to the input buffer for drive functions.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.8.5 Get Free IPM Buffer Size****FUNCTION**

```
BOOL VCS_GetFreeIpmbufferSize(HANDLE KeyHandle, WORD Nodeld, DWORD* pBufferSize,
                               DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetFreeIpmbufferSize reads the available buffer size.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pBufferSize	DWORD	Actual free buffer size	0x60C4-02
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.8.6 Add PVT Value To IPM Buffer****FUNCTION**

```
BOOL VCS_AddPvtValueTolpmBuffer(HANDLE KeyHandle, WORD Nodeld, long Position,
                                 long Velocity, BYTE Time, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_AddPvtValueTolpmBuffer adds a new PVT reference point to the device.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
Position	long	Position of the reference point
Velocity	long	Velocity of the reference point
Time	BYTE	Time of the reference point

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.8.7 Start IPM Trajectory

### FUNCTION

BOOL VCS\_StartIpmpTrajectory(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_StartIpmpTrajectory starts the IPM trajectory.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.8.8 Stop IPM Trajectory

### FUNCTION

BOOL VCS\_StopIpmpTrajectory(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_StopIpmpTrajectory stops the IPM trajectory.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.8.9 Get IPM Status****FUNCTION**

```
BOOL VCS_GetIpmsStatus(HANDLE KeyHandle, WORD Nodeld, BOOL* pTrajectoryRunning, BOOL*
plsUnderflowWarning, BOOL* plsOverflowWarning, BOOL* plsVelocityWarning, BOOL*
plsAccelerationWarning, BOOL* plsUnderflowError, BOOL* plsOverflowError, BOOL*
plsVelocityError, BOOL* plsAccelerationError, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetIpmsStatus returns different warning and error states.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pTrajectoryRunning	BOOL*	State if IPM active
plsUnderflowWarning	BOOL*	State if buffer underflow level is reached
plsOverflowWarning	BOOL*	State if buffer overflow level is reached
plsVelocityWarning	BOOL*	State if IPM velocity greater than profile velocity
plsAccelerationWarning	BOOL*	State if IPM acceleration greater than profile acceleration
plsUnderflowError	BOOL*	State of underflow error
plsOverflowError	BOOL*	State of overflow error
plsVelocityError	BOOL*	State if IPM velocity greater than max. profile velocity
plsAccelerationError	BOOL*	State if IPM acceleration greater than max. profile acceleration
pErrorCode	DWORD*	Error information on the executed function

0x20C4-01

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.9 Position Mode (PM)

### 5.9.1 Activate Position Mode

**FUNCTION**

```
BOOL VCS_ActivatePositionMode(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ActivatePositionMode changes the operational mode to “position mode”.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 5.9.2 Set Position Must

**FUNCTION**

```
BOOL VCS_SetPositionMust(HANDLE KeyHandle, WORD Nodeld, long PositionMust, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_SetPositionMust sets the position mode setting value.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
PositionMust	long	Position mode setting value 0x2062-00

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

**5.9.3 Get Position Must****FUNCTION**

```
BOOL VCS_GetPositionMust(HANDLE KeyHandle, WORD Nodeld, long* pPositionMust, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetPositionMust reads the position mode setting value.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pPositionMust	long*	Position mode setting value	0x2062-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.9.4 Advanced Functions****5.9.4.1 Activate Analog Position Setpoint****FUNCTION**

```
BOOL VCS_ActivateAnalogPositionSetpoint(HANDLE KeyHandle, WORD Nodeld, WORD
AnalogInputNumber, float Scaling, long Offset, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ActivateAnalogPositionSetpoint configures the selected analog input for analog position setpoint.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input 0x207B-01 or 0x207B-02
Scaling	float	Scaling factor for analog position setpoint functionality 0x2303-01
Offset	long	Offset for analog position setpoint functionality 0x2303-02

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.9.4.2 Deactivate Analog Position Setpoint

### FUNCTION

```
BOOL VCS_DeactivateAnalogPositionSetpoint(HANDLE KeyHandle, WORD Nodeld, WORD  
AnalogInputNumber, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_DeactivateAnalogPositionSetpoint disables the selected analog input for analog position setpoint.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input 0x207B-01 or 0x207B-02

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.9.4.3 Enable Analog Position Setpoint

### FUNCTION

```
BOOL VCS_EnableAnalogPositionSetpoint(HANDLE KeyHandle, WORD Nodeld, DWORD* pEr-  
rorCode)
```

### DESCRIPTION

VCS\_EnableAnalogPositionSetpoint enables the execution mask for analog position setpoint.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.9.4.4 Disable Analog Position Setpoint****FUNCTION**

```
BOOL VCS_DisableAnalogPositionSetpoint(HANDLE KeyHandle, WORD Nodeld, DWORD*  
pErrorCode)
```

**DESCRIPTION**

VCS\_DisableAnalogPositionSetpoint disables the execution mask for analog position setpoint.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.10 Velocity Mode (VM)

### 5.10.1 Activate Velocity Mode

#### FUNCTION

BOOL VCS\_ActivateVelocityMode(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ActivateVelocityMode changes the operational mode to “velocity mode”.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.10.2 Set Velocity Must

#### FUNCTION

BOOL VCS\_SetVelocityMust(HANDLE KeyHandle, WORD Nodeld, long VelocityMust, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetVelocityMust sets the velocity mode setting value.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
VelocityMust	long	Velocity mode setting value 0x206B-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.10.3 Get Velocity Must****FUNCTION**

```
BOOL VCS_GetVelocityMust(HANDLE KeyHandle, WORD Nodeld, long* pVelocityMust, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetVelocityMust returns the velocity mode setting value.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pVelocityMust	long*	Velocity mode setting value	0x206B-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.10.4 Advanced Functions****5.10.4.1 Activate Analog Velocity Setpoint****FUNCTION**

```
BOOL VCS_ActivateAnalogVelocitySetpoint(HANDLE KeyHandle, WORD Nodeld, WORD
AnalogInputNumber, float Scaling, long Offset, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ActivateAnalogVelocitySetpoint configures the selected analog input for analog velocity setpoint.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input 0x207B-01 or 0x207B-02
Scaling	float	Scaling factor for analog velocity setpoint functionality 0x2302-01
Offset	long	Offset for analog velocity setpoint functionality 0x2302-02

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.10.4.2 Deactivate Analog Velocity Setpoint

### FUNCTION

```
BOOL VCS_DeactivateAnalogVelocitySetpoint(HANDLE KeyHandle, WORD Nodeld, WORD  
AnalogInputNumber, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_DeactivateAnalogVelocitySetpoint disables the selected analog input for analog velocity setpoint.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input 0x207B-01 or 0x207B-02

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.10.4.3 Enable Analog Velocity Setpoint

### FUNCTION

```
BOOL VCS_EnableAnalogVelocitySetpoint(HANDLE KeyHandle, WORD Nodeld, DWORD*  
pErrorCode)
```

### DESCRIPTION

VCS\_EnableAnalogVelocitySetpoint enables the execution mask for analog velocity setpoint.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.10.4.4 Disable Analog Velocity Setpoint****FUNCTION**

```
BOOL VCS_DisableAnalogVelocitySetpoint(HANDLE KeyHandle, WORD NodeId, DWORD*  
pErrorCode)
```

**DESCRIPTION**

VCS\_DisableAnalogVelocitySetpoint disables the execution mask for analog velocity setpoint.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.11 Current Mode (CM)

### 5.11.1 Activate Current Mode

#### FUNCTION

BOOL VCS\_ActivateCurrentMode(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ActivateCurrentMode changes the operational mode to “current mode”.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

### 5.11.2 Get Current Must

#### FUNCTION

BOOL VCS\_GetCurrentMust(HANDLE KeyHandle, WORD Nodeld, short\* pCurrentMust, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_GetCurrentMust reads the current mode setting value.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pCurrentMust	short*	Current mode setting value	0x2030-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.11.3 Set Current Must****FUNCTION**

`BOOL VCS_SetCurrentMust(HANDLE KeyHandle, WORD Nodeld, short CurrentMust, DWORD* pErrorCode)`

**DESCRIPTION**

VCS\_SetCurrentMust writes current mode setting value.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
CurrentMust	short	Current mode setting value 0x2030-00

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.11.4 Advanced Functions****5.11.4.1 Activate Analog Current Setpoint****FUNCTION**

`BOOL VCS_ActivateAnalogCurrentSetpoint(HANDLE KeyHandle, WORD Nodeld, WORD AnalogInputNumber, float Scaling, short Offset, DWORD* pErrorCode)`

**DESCRIPTION**

VCS\_ActivateAnalogCurrentSetpoint configures the selected analog input for analog current setpoint.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input 0x207B-01 or 0x207B-02
Scaling	float	Scaling factor for analog current setpoint functionality 0x2301-01
Offset	short	Offset for analog current setpoint functionality 0x2301-02

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.11.4.2 Deactivate Analog Current Setpoint

### FUNCTION

BOOL VCS\_DeactivateAnalogCurrentSetpoint(HANDLE KeyHandle, WORD Nodeld, WORD AnalogInputNumber, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_DeactivateAnalogCurrentSetpoint disables the selected analog input for analog current setpoint.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input 0x207B-01 or 0x207B-02

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.11.4.3 Enable Analog Current Setpoint

### FUNCTION

BOOL VCS\_EnableAnalogCurrentSetpoint(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

### DESCRIPTION

VCS\_EnableAnalogCurrentSetpoint enables the execution mask for analog current setpoint.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.11.4.4 Disable Analog Current Setpoint****FUNCTION**

```
BOOL VCS_DisableAnalogCurrentSetpoint(HANDLE KeyHandle, WORD Nodeld, DWORD*  
pErrorCode)
```

**DESCRIPTION**

VCS\_DisableAnalogCurrentSetpoint disables the execution mask for analog current setpoint.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.12 Master Encoder Mode (MEM)

### 5.12.1 Activate Master Encoder Mode

**FUNCTION**

```
BOOL VCS_ActivateMasterEncoderMode(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ActivateMasterEncoderMode changes the operational mode to “master encoder mode”.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 5.12.2 Set Master Encoder Parameter

**FUNCTION**

```
BOOL VCS_SetMasterEncoderParameter(HANDLE KeyHandle, WORD Nodeld, WORD ScalingNumerator, WORD ScalingDenominator, BYTE Polarity, DWORD MaxVelocity, DWORD MaxAcceleration, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_SetMasterEncoderParameter writes all parameters for master encoder mode.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ScalingNumerator	WORD	Scaling numerator for position calculation 0x2300-02
ScalingDenominator	WORD	Scaling denominator for position calculation 0x2300-03
Polarity	BYTE	Polarity of the direction input. 0: Positive 0x2300-04 1: Negative
MaxVelocity	DWORD	Maximal allowed speed during a profiled move 0x607F-01
MaxAcceleration	DWORD	Defines the maximal allowed acceleration 0x60C5-01

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

**5.12.3 Get Master Encoder Parameter****FUNCTION**

```
BOOL VCS_GetMasterEncoderParameter(HANDLE KeyHandle, WORD Nodeld, WORD*
pScalingNumerator, WORD* pScalingDenominator, BYTE* pPolarity, DWORD* pMaxVelocity,
DWORD* pMaxAcceleration, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetMasterEncoderParameter reads all parameters for master encoder mode.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pScalingNumerator	WORD*	Scaling numerator for position calculation	0x2300-02
pScalingDenominator	WORD*	Scaling denominator for position calculation	0x2300-03
pPolarity	BYTE*	Polarity of the direction input. 0: Positive 1: Negative	0x2300-04
pMaxVelocity	DWORD*	Maximal allowed speed during a profiled move	0x607F-01
pMaxAcceleration	DWORD*	Defines the maximal allowed acceleration	0x60C5-01
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.13 Step Direction Mode (SDM)

### 5.13.1 Activate Step Direction Mode

#### FUNCTION

BOOL VCS\_ActivateStepDirectionMode(HANDLE KeyHandle, WORD NodId, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ActivateStepDirectionMode changes the operational mode to "step direction mode".

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 5.13.2 Set Step Direction Parameter

#### FUNCTION

BOOL VCS\_SetStepDirectionParameter(HANDLE KeyHandle, WORD NodId, WORD ScalingNumerator, WORD ScalingDenominator, BYTE Polarity, DWORD MaxVelocity, DWORD MaxAcceleration, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_SetStepDirectionParameter writes all parameters for step direction mode.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
ScalingNumerator	WORD	Scaling numerator for position calculation 0x2300-02
ScalingDenominator	WORD	Scaling denominator for position calculation 0x2300-03
Polarity	BYTE	Polarity of the direction input. 0: Positive 1: Negative 0x2300-04
MaxVelocity	DWORD	Maximal allowed speed during a profiled move 0x607F-01
MaxAcceleration	DWORD	Defines the maximal allowed acceleration 0x60C5-01

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.13.3 Get Step Direction Parameter****FUNCTION**

```
BOOL VCS_GetStepDirectionParameter(HANDLE KeyHandle, WORD Nodeld, WORD*
pScalingNumerator, WORD* pScalingDenominator, BYTE* pPolarity, DWORD* pMaxVelocity,
DWORD* pMaxAcceleration, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetStepDirectionParameter reads all parameters for step direction mode.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pScalingNumerator	WORD*	Scaling numerator for position calculation	0x2300-02
pScalingDenominator	WORD*	Scaling denominator for position calculation	0x2300-03
pPolarity	BYTE*	Polarity of the direction input. 0: Positive 1: Negative	0x2300-04
pMaxVelocity	DWORD*	Maximal allowed speed during a profiled move	0x607F-01
pMaxAcceleration	DWORD*	Defines the maximal allowed acceleration	0x60C5-01
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.14 Inputs & Outputs

For details →separate document «Firmware Specification».

### 5.14.1 Get All Digital Inputs

#### FUNCTION

```
BOOL VCS_GetAllDigitalInputs(HANDLE KeyHandle, WORD Nodeld, WORD* pInputs, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetAllDigitalInputs returns state of all digital inputs.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pInputs	WORD*	Displays the state of the digital input functionalities. Activated if a bit is read as “1”. 0x2071-01
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 5.14.2 Get All Digital Outputs

#### FUNCTION

```
BOOL VCS_GetAllDigitalOutputs(HANDLE KeyHandle, WORD Nodeld, WORD* pOutputs, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_GetAllDigitalOutputs returns state of all digital outputs.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pOutputs	WORD*	State of all digital outputs. Activated if a bit is read as “1”. 0x2078-01
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

**5.14.3 Set All Digital Outputs****FUNCTION**

```
BOOL VCS_SetAllDigitalOutputs(HANDLE KeyHandle, WORD Nodeld, WORD Outputs, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_SetAllDigitalOutputs sets the state of all digital outputs.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
Outputs	WORD	State of all digital outputs. Activated if a bit is written as "1".

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.14.4 Get Analog Input****FUNCTION**

```
BOOL VCS_GetAnalogInput(HANDLE KeyHandle, WORD Nodeld, WORD InputNumber, WORD* pAnalogValue, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_GetAnalogInput returns the value from an analog input.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
InputNumber	WORD	Analog input number

**RETURN PARAMETERS**

pAnalogValue	WORD*	Analog value from input
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.14.5 Set Analog Output

### FUNCTION

```
BOOL VCS_SetAnalogOutput(HANDLE KeyHandle, WORD Nodeld, WORD OutputNumber, WORD  
AnalogValue, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_SetAnalogOutput sets the voltage level of an analog output.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
OutputNumber	WORD	Analog output number
pAnalogValue	WORD*	Analog value for output 0x207E-00

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

## 5.14.6 Position Compare

### 5.14.6.1 Set Position Compare Parameter

#### FUNCTION

```
BOOL VCS_SetPositionCompareParameter(HANDLE KeyHandle, WORD Nodeld, BYTE  
OperationalMode, BYTE IntervalMode, BYTE DirectionDependency, WORD IntervalWidth, WORD  
IntervalRepetitions, WORD PulseWidth, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_SetPositionCompareParameter writes all parameters for position compare.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
OperationalMode	BYTE	Used operational mode in position sequence mode (→Table 5-16)
IntervalMode	BYTE	Used interval mode in position sequence mode (→Table 5-17) 0x207A-01
DirectionDependency	BYTE	Used direction dependency in position sequence mode (→Table 5-18)
IntervalWidth	WORD	Holds the width of the position intervals 0x207A-03
IntervalRepetitions	WORD	Allows to configure the number of position intervals to be considered by position compare 0x207A-04
PulseWidth	WORD	Configures the pulse width of the trigger output 0x207A-05

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**OPERATIONAL MODE**

Description	Value	Name
Single position mode	0	PCO_SINGLE_POSITION_MODE
Position sequence mode	1	PCO_POSITION_SEQUENCE_MODE

Table 5-16 Position Compare – Operational Modes

**INTERVAL MODE**

Description	Value	Name
Interval positions are set in negative direction relative to the position compare reference position	0	PCI_NEGATIVE_DIR_TO_REFPOS
Interval positions are set in positive direction relative to the position compare reference position	1	PCI_POSITIVE_DIR_TO_REFPOS
Interval positions are set in positive and negative direction relative to the position compare reference position	2	PCI_BOTH_DIR_TO_REFPOS

Table 5-17 Position Compare – Interval Modes

**DIRECTION DEPENDENCY**

Description	Value	Name
Positions are compared only if actual motor direction is negative	0	PCD_MOTOR_DIRECTION_NEGATIVE
Positions are compared only if actual motor direction is positive	1	PCD_MOTOR_DIRECTION_POSITIVE
Positions are compared regardless of the actual motor direction	2	PCD_MOTOR_DIRECTION_BOTH

Table 5-18 Position Compare – Direction Dependency

## 5.14.6.2 Get Position Compare Parameter

### FUNCTION

```
BOOL VCS_GetPositionCompareParameter(HANDLE KeyHandle, WORD Nodeld, BYTE* pOperationalMode, BYTE* pIntervalMode, BYTE* pDirectionDependency, WORD* pIntervalWidth, WORD* pIntervalRepetitions, WORD* pPulseWidth, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetPositionCompareParameter reads all parameters for position compare.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pOperationalMode	BYTE*	Used operational mode in position sequence mode (→Table 5-16)	
pIntervalMode	BYTE*	Used interval mode in position sequence mode (→Table 5-17)	0x207A-01
pDirectionDependency	BYTE*	Used direction dependency in position sequence mode (→Table 5-18)	
pIntervalWidth	WORD*	Holds the width of the position intervals	0x207A-03
pIntervalRepetitions	WORD*	Allows to configure the number of position intervals to be considered by position compare	0x207A-04
pPulseWidth	WORD*	Configures the pulse width of the trigger output	0x207A-05
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.14.6.3 Activate Position Compare

### FUNCTION

```
BOOL VCS_ActivatePositionCompare(HANDLE KeyHandle, WORD Nodeld, WORD DigitalOutputNumber, BOOL Polarity, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_ActivatePositionCompare enables the output to position compare method.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
Nodeld	WORD	Node ID of the addressed device	
DigitalOutputNumber	WORD	Selected digital output for position compare	0x2079-0x
Polarity	BOOL	Polarity of the selected output	0x2078-03

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.14.6.4 Deactivate Position Compare****FUNCTION**

```
BOOL VCS_DeactivatePositionCompare(HANDLE KeyHandle, WORD NodId, WORD
DigitalOutputNumber, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_DeactivatePositionCompare disables the output to position compare method.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
DigitalOutputNumber	WORD	Selected digital output for position compare 0x2079-0x

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.14.6.5 Enable Position Compare****FUNCTION**

```
BOOL VCS_EnablePositionCompare(HANDLE KeyHandle, WORD NodId, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_EnablePositionCompare enables the output mask for position compare method.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.14.6.6 Disable Position Compare****FUNCTION**

```
BOOL VCS_DisablePositionCompare(HANDLE KeyHandle, WORD NodId, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_DisablePositionCompare disables the output mask from position compare method.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.14.6.7 Set Position Compare Reference Position

### FUNCTION

```
BOOL VCS_SetPositionCompareReferencePosition(HANDLE KeyHandle, WORD Nodeld, long ReferencePosition, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_SetPositionCompareReferencePosition writes the reference position for position compare method.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ReferencePosition	long	Holds the position that is compared with the position actual value 0x207A-02

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**5.14.7 Position Marker****5.14.7.1 Set Position Marker Parameter****FUNCTION**

`BOOL VCS_SetPositionMarkerParameter(HANDLE KeyHandle, WORD Nodeld, BYTE PositionMarkerEdgeType, BYTE PositionMarkerMode, DWORD* pErrorCode)`

**DESCRIPTION**

`VCS_SetPositionMarkerParameter` writes all parameters for position marker method.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
PositionMarkerEdgeType	BYTE	Defines the type of edge of the position to be captured (→Table 5-19) 0x2074-02
PositionMarkerMode	BYTE	Defines the position marker capturing mode (→Table 5-20) 0x2074-03

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**POSITIONMARKEREDGE TYPE**

Description	Value	Name
Both edges	0	PET_BOTH_EDGES
Rising edge	1	PET_RISING_EDGE
Falling edge	2	PET_FALLING_EDGE

Table 5-19 Position Marker Edge Types

**POSITIONMARKER MODE**

Description	Value	Name
Continuous	0	PM_CONTINUOUS
Single	1	PM_SINGLE
Multiple	2	PM_MULTIPLE

Table 5-20 Position Marker Modes

## 5.14.7.2 Get Position Marker Parameter

### FUNCTION

```
BOOL VCS_GetPositionMarkerParameter(HANDLE KeyHandle, WORD NodId, BYTE* pPositionMarkerEdgeType, BYTE* pPositionMarkerMode, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_GetPositionMarkerParameter reads all parameters for position marker method.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pPositionMarkerEdgeType	BYTE*	Defines the type of edge of the position to be captured (→Table 5-19)	0x2074-02
pPositionMarkerMode	BYTE*	Defines the position marker capturing mode (→Table 5-20)	0x2074-03
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.14.7.3 Activate Position Marker

### FUNCTION

```
BOOL VCS_ActivatePositionMarker(HANDLE KeyHandle, WORD NodId, WORD DigitalInputNumber, BOOL Polarity, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_ActivatePositionMarker enables the digital input to position marker method.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
DigitalInputNumber	WORD	Selected digital input for position marker
Polarity	BOOL	Polarity of the selected input

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

**5.14.7.4 Deactivate Position Marker****FUNCTION**

```
BOOL VCS_DeactivatePositionMarker(HANDLE KeyHandle, WORD NodId, WORD
DigitalInputNumber, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_DeactivatePositionMarker disables the digital input to position marker method.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device
DigitalInputNumber	WORD	Selected digital input for position marker 0x2070-0x

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**5.14.7.5 Read Position Marker Counter****FUNCTION**

```
BOOL VCS_ReadPositionMarkerCounter(HANDLE KeyHandle, WORD NodId, WORD* pCount,
DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ReadPositionMarkerCounter returns the number of the detected edges.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
NodId	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pCount	WORD*	Counts the number of detected edges 0x2074-04
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 5.14.7.6 Read Position Marker Captured Position

### FUNCTION

```
BOOL VCS_ReadPositionMarkerCapturedPosition(HANDLE KeyHandle, WORD Nodeld, WORD CounterIndex, long* pCapturedPosition, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_ReadPositionMarkerCapturedPosition returns the last captured position or the position from the position marker history.

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
CounterIndex	WORD	0: Read position marker captured position 0x2074-01
		1: Read position marker history 0x2074-05
		2: Read position marker history 0x2074-06

### RETURN PARAMETERS

pCapturedPosition	long*	Contains the captured position or the position marker history 0x2074-01 or 0x2074-05 or 0x2074-06
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 5.14.7.7 Reset Position Marker Counter

### FUNCTION

```
BOOL VCS_ResetPositionMarkerCounter(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

### DESCRIPTION

VCS\_ResetPositionMarkerCounter clears the counter and the captured positions by writing zero to object position marker counter (0x2074-04).

### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise “0”
--------------	------	--------------------------------------

## 6 Data Recording Functions

### 6.1 Operation Mode

#### 6.1.1 Set Recorder Parameter

##### FUNCTION

BOOL VCS\_SetRecorderParameter(HANDLE KeyHandle, WORD NodId, WORD SamplingPeriod, WORD NbOfPrecedingSamples, WORD PulseWidth, DWORD\* pErrorCode)

##### DESCRIPTION

VCS\_SetRecorderParameter writes parameters for data recorder.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
NodId	WORD	Node ID of the addressed device	
SamplingPeriod	WORD	Sampling Period as a multiple of the current regulator cycle (n-times 0.1 ms)	0x2012-00
NbOfPrecedingSamples	WORD	Number of preceding samples (data history)	0x2013-00

##### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

#### 6.1.2 Get Recorder Parameter

##### FUNCTION

BOOL VCS\_GetRecorderParameter(HANDLE KeyHandle, WORD NodId, WORD\* pSamplingPeriod, WORD\* pNbOfPrecedingSamples, WORD PulseWidth, DWORD\* pErrorCode)

##### DESCRIPTION

VCS\_GetRecorderParameter reads parameters for data recorder.

##### PARAMETERS

KeyHandle	HANDLE	Handle for port access	
NodId	WORD	Node ID of the addressed device	

##### RETURN PARAMETERS

pSamplingPeriod	WORD*	Sampling Period as a multiple of the current regulator cycle (n-times 0.1 ms)	0x2012-00
pNbOfPrecedingSamples	WORD*	Number of preceding samples (data history)	0x2013-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

### 6.1.3 Enable Trigger

#### FUNCTION

```
BOOL VCS_EnableTrigger(HANDLE KeyHandle, WORD Nodeld, BYTE TriggerType, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_EnableTrigger connects the trigger(s) for data recording.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
TriggerType	BYTE	Configuration of Auto Trigger functions. Activated if a bit is written as "1" (→Table 6-21). Activation of more than one trigger at the same time is possible. 0x2011-00

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

Description	Value	Name
Trigger movement start	1	DR_MOVEMENT_START_TRIGGER
Error trigger	2	DR_ERROR_TRIGGER
Digital input trigger	4	DR_DIGITAL_INPUT_TRIGGER
Trigger movement end	8	DR_MOVEMENT_END_TRIGGER

Table 6-21 Data Recorder Trigger Types

### 6.1.4 Disable all Triggers

#### FUNCTION

```
BOOL VCS_DisableAllTrigger(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_DisableAllTrigger sets data recorder configuration (0x2011-00) for triggers to zero.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**6.1.5 Activate Channel****FUNCTION**

```
BOOL VCS_ActivateChannel(HANDLE KeyHandle, WORD Nodeld, BYTE ChannelNumber, WORD
ObjectIndex, BYTE ObjectSubIndex, BYTE ObjectSize, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ActivateChannel connects object for data recording.

Start with channel 1 (one)! Then, for every activated channel, the number of sampling variables (0x2014-00) will be incremented.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ChannelNumber	BYTE	Channel number [1...4]
ObjectIndex	WORD	Object index for data recording 0x2015- Channel Number
ObjectSubIndex	BYTE	Object subindex for data recording 0x2016- Channel Number
ObjectSize	BYTE	Object size in bytes for data recording

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**6.1.6 Deactivate all Channels****FUNCTION**

```
BOOL VCS_DeactivateAllChannel(HANDLE KeyHandle, WORD Nodeld, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_DeactivateAllChannel zeros all data recording objects (0x2014, 0x2015, and 0x2016).

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 6.2 Data Recorder Status

### 6.2.1 Start Recorder

#### FUNCTION

BOOL VCS\_StartRecorder(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_StartRecorder starts data recording. Not available with Linux.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 6.2.2 Stop Recorder

#### FUNCTION

BOOL VCS\_StopRecorder(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_StopRecorder stops data recording. Not available with Linux.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 6.2.3 Force Trigger

#### FUNCTION

BOOL VCS\_ForceTrigger(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

#### DESCRIPTION

VCS\_ForceTrigger forces the data recording triggers. Not available with Linux.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 6.2.4 Is Recorder Running

**FUNCTION**

BOOL VCS\_IsRecorderRunning(HANDLE KeyHandle, WORD Nodeld, BOOL\* pRunning, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_IsRecorderRunning returns the data recorder status “running”. Not available with Linux.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pRunning	BOOL	1: Data recorder running 0: Data recorder stopped (Bit 0)	0x2017-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

### 6.2.5 Is Recorder Triggered

**FUNCTION**

BOOL VCS\_IsRecorderTriggered(HANDLE KeyHandle, WORD Nodeld, BOOL\* pTriggered, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_IsRecorderTriggered returns data recorder status “triggered”. Not available with Linux.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pTriggered	BOOL	1: Data recorder triggered 0: Data recorder not triggered (Bit 1)	0x2017-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”
---------------------	------	--------------------------------------

## 6.3 Data Recorder Data

### 6.3.1 Read Channel Vector Size

#### FUNCTION

```
BOOL VCS_ReadChannelVectorSize(HANDLE KeyHandle, WORD Nodeld, DWORD* pVectorSize,  
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_ReadChannelVectorSize returns the maximal number of samples per variable. It is dynamically calculated by the data recorder. Not available with Linux.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

#### RETURN PARAMETERS

pVectorSize	DWORD*	Maximal number of samples per variable	0x2018-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

### 6.3.2 Read Channel Data Vector

#### FUNCTION

```
BOOL VCS_ReadChannelDataVector(HANDLE KeyHandle, WORD Nodeld, BYTE ChannelNumber,  
BYTE* pDataVector, DWORD VectorSize, DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_ReadChannelDataVector returns the data points of a selected channel. Not available with Linux.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ChannelNumber	BYTE	Selected channel
VectorSize	DWORD	Size of data points

#### RETURN PARAMETERS

pDataVector	BYTE*	Data points of selected channel	0x201B-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"
--------------	------	--------------------------------------

**6.3.3 Show Channel Data Dialog****FUNCTION**

BOOL VCS\_ShowChannelDataDlg(HANDLE KeyHandle, WORD Nodeld, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_ShowChannelDataDlg opens the dialog to show the data channel(s). Not available with Linux.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**6.3.4 Export Channel Data to File****FUNCTION**

BOOL VCS\_ExportChannelDataToFile(HANDLE KeyHandle, WORD Nodeld, char\* FileName, DWORD\* pErrorCode)

**DESCRIPTION**

VCS\_ExportChannelDataToFile saves the data point in a file. Not available with Linux.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
FileName	char*	Path and file name to save data points (*.csv, *.txt, *.rda)

**RETURN PARAMETERS**

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

## 6.4 Advanced Functions

### 6.4.1 Read Data Buffer

**FUNCTION**

```
BOOL VCS_ReadDataBuffer(HANDLE KeyHandle, WORD Nodeld, BYTE* pDataBuffer, DWORD  
BufferSizeToRead, DWORD* pBufferSizeRead, WORD* pVectorStartOffset, WORD*  
pMaxNbOfSamples, WORD* pNbOfRecordedSamples, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ReadDataBuffer returns the buffer data points.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
BufferSizeToRead	DWORD	Buffer size

**RETURN PARAMETERS**

pDataBuffer	BYTE*	Data points	0x201B-00
pBufferSizeRead	DWORD*	Size of read data buffer	
pVectorStartOffset	WORD*	Offset to the start of the recorded data vector within the ring buffer	0x201A-00
pMaxNbOfSamples	WORD*	Maximal number of samples per variable	0x2018-00
pNbOfRecordedSamples	WORD*	Number of recorded samples	0x2019-00
pErrorCode	DWORD*	Error information on the executed function	

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**6.4.2 Extract Channel Data Vector****FUNCTION**

```
BOOL VCS_ExtractChannelDataVector(HANDLE KeyHandle, WORD Nodeld, BYTE
ChannelNumber, BYTE* pDataBuffer, DWORD BufferSize, BYTE* pDataVector, DWORD VectorSize,
WORD VectorStartOffset, WORD MaxNbOfSamples, WORD NbOfRecordedSamples, DWORD* pErrorCode)
```

**DESCRIPTION**

VCS\_ExtractChannelDataVector returns the vector of a data channel.

**PARAMETERS**

KeyHandle	HANDLE	Handle for port access
Nodeld	WORD	Node ID of the addressed device
ChannelNumber	BYTE	Selected channel
pDataBuffer	BYTE	Data points 0x201B-00
BufferSize	DWORD	Buffer size
VectorSize	DWORD	Vector size
VectorStartOffset	WORD	Offset to the start of the recorded data vector within the ring buffer 0x201A-00
MaxNbOfSamples	WORD	Maximal number of samples per variable 0x2018-00
NbOfRecordedSamples	WORD	Number of recorded samples 0x2019-00

**RETURN PARAMETERS**

pDataVector	BYTE*	Data points of the channel
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

**••page intentionally left blank••**

## 7 Low Layer Functions

### 7.1 Send CAN Frame

#### FUNCTION

```
BOOL VCS_SendCANFrame(HANDLE KeyHandle, WORD CobID, WORD Length, void* pData,
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_SendCANFrame sends a general CAN frame to the CAN bus.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
CobID	WORD	CAN frame 11-bit identifier
Length	WORD	CAN frame data length
pData	void*	CAN frame data

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”

### 7.2 Read CAN Frame

#### FUNCTION

```
BOOL VCS_ReadCANFrame(HANDLE KeyHandle, WORD CobID, WORD Length, void* pData,
DWORD Timeout , DWORD* p ErrorCode)
```

#### DESCRIPTION

VCS\_ReadCANFrame reads a general CAN frame from the CAN bus.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
CobID	WORD	CAN frame 11-bit identifier
Length	WORD	CAN frame data length
Timeout	WORD	Maximum waiting period

#### RETURN PARAMETERS

pData	void*	CAN frame data
pErrorCode	DWORD*	Error information on the executed function
<b>Return Value</b>	BOOL	Nonzero if successful; otherwise “0”

### 7.3 Request CAN Frame

#### FUNCTION

```
BOOL VCS_RequestCANFrame(HANDLE KeyHandle, WORD CobID, WORD Length, void* pData,
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_RequestCANFrame requests a general CAN frame from the CAN bus using Remote Transmit Request (RTR).

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
CobID	WORD	CAN frame 11-bit identifier
Length	WORD	CAN frame data length

#### RETURN PARAMETERS

pData	void*	CAN frame data
pErrorCode	DWORD*	Error information on the executed function

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

### 7.4 Send NMT Frame

#### FUNCTION

```
BOOL VCS_SendNMTService(HANDLE KeyHandle, WORD NodId, WORD CommandSpecifier,
DWORD* pErrorCode)
```

#### DESCRIPTION

VCS\_SendNMTService is used to send a NMT protocol from a master to one slave/all slaves in a network. Command is without acknowledge.

#### PARAMETERS

KeyHandle	HANDLE	Handle for port access
CobID	WORD	1...127: NMT slave with given Node ID 0: All NMT slaves
CommandSpecifier	WORD	NMT service (→Table 7-22)

#### RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
------------	--------	--

<b>Return Value</b>	BOOL	Nonzero if successful; otherwise "0"
---------------------	------	--------------------------------------

Description	Value	Name
Start remote node	-1	NCS_START_REMOTE_NODE
Stop remote node	-2	NCS_STOP_REMOTE_NODE
Enter pre-operational	-128	NCS_ENTER_PRE_OPERATIONAL
Reset node	-129	NCS_RESET_NODE
Reset communication	-130	NCS_RESET_COMMUNICATION

Table 7-22      Command Specifier

## 8 Error Overview

### 8.1 Communication Errors

Abort Code	Name	Error Cause
0x0000 0000	No Communication Error	Communication was successful
0x0503 0000	Toggle Error	Toggle bit not alternated
0x0504 0000	SDO Time Out	SDO protocol timed out
0x0504 0001	Client/server specifier Error	Client/server command specifier not valid or unknown
0x0504 0002	Invalid block size	Invalid block size (block mode only)
0x0504 0003	Invalid sequence	Invalid sequence number (block mode only)
0x0504 0004	CrcError	CRC error (block mode only)
0x0504 0005	Out of Memory Error	Out of Memory
0x0601 0000	Access Error	Unsupported access to an object (e.g. write command to a read-only object)
0x0601 0001	Write Only	Read command to a write only object
0x0601 0002	Read Only	Write command to a read only object
0x0602 0000	Object does not exist Error	Last read or write command had a wrong object index or subindex
0x0604 0041	PDO mapping Error	Object cannot be mapped to PDO
0x0604 0042	PDO length Error	Number and length of objects to be mapped would exceed PDO length
0x0604 0043	General parameter Error	General parameter incompatibility
0x0604 0047	General Intern Incompatibility Error	General internal incompatibility in device
0x0606 0000	Hardware Error	Access failed due to an hardware error
0x0607 0010	Service Parameter Error	Data type does not match, length or service parameter does not match
0x0607 0012	Service Parameter Error too High Error	Data type does not match, length or service parameter too high
0x0607 0013	Service Parameter Error too Low Error	Data type does not match, length or service parameter too low
0x0609 0011	Object Subindex Error	Last read or write command had a wrong subindex
0x0609 0030	Value Range Error	Value range of parameter exceeded
0x0609 0031	Value too High Error	Value of parameter written too high
0x0609 0032	Value too Low Error	Value of parameter written too low
0x0609 0036	Maximum less Minimum Error	Maximum value is less than minimum value
0x0800 0000	General Error	General error
0x0800 0020	Transfer or store Error	Data cannot be transferred or stored
0x0800 0021	Local control Error	Data cannot be transferred or stored to application because of local control
0x0800 0022	Wrong Device State	Data cannot be transferred or stored to application because of present device state
0x0F00 FFB9	Error CAN id	Wrong CAN id
0x0F00 FFBC	Error Service Mode	Device is not in service mode
0x0F00 FFBE	Password Error	Password is wrong
0x0F00 FFBF	Illegal Command Error	RS232 command is illegal (does not exist)
0x0F00 FFC0	Wrong NMT State Error	Device is in wrong NMT state

Table 8-23 Communication Errors

## 8.2 EPOS Command Library-specified Errors

### 8.2.1 General Errors

Abort Code	Name	Error Cause
0x0000 0000	No Error	Function was successful
0x1000 0001	Internal Error	Internal error
0x1000 0002	Null Pointer	Null pointer passed to function
0x1000 0003	Handle not Valid	Handle passed to function is not valid
0x1000 0004	Bad Virtual Device Name	Virtual device name is not valid
0x1000 0005	Bad Device Name	Device name is not valid
0x1000 0006	Bad ProtocolStack Name	Protocol stack name is not valid
0x1000 0007	Bad Interface Name	Interface name is not valid
0x1000 0008	Bad Port Name	Port is not valid
0x1000 0009	Library not Loaded	Could not load external library
0x1000 000A	Executing Command	Command failed
0x1000 000B	Timeout	Timeout occurred during execution
0x1000 000C	Bad Parameter	Bad parameter passed to function
0x1000 000D	Command Aborted By User	Command aborted by user
0x1000 000E	Buffer Too Small	Buffer is too small
0x1000 000F	No Communication Found	No communication settings found
0x1000 0010	Function Not Supported	Function not supported
0x1000 0011	Parameter Already Used	Parameter already used
0x1000 0020	Bad Device State	Bad device state
0x1000 0021	Bad File Content	Bad file content
0x1000 0022	Path Does Not Exist	System cannot find specified path

Table 8-24 General Errors

### 8.2.2 Interface Layer Errors

Abort Code	Name	Error Cause
0x2000 0001	Opening Interface	Error opening interface
0x2000 0002	Closing Interface	Error closing interface
0x2000 0003	Interface not Open	Interface is not open
0x2000 0004	Opening Port	Error opening port
0x2000 0005	Closing Port	Error closing port
0x2000 0006	Port not Open	Port is not open
0x2000 0007	Reset Port	Error resetting port
0x2000 0008	Set Port Settings	Error configuring port settings
0x2000 0009	Set Port Mode	Error configuring port mode

Table 8-25 Interface Layer Errors

**8.2.2.1 Interface Layer “RS232” Errors**

<b>Abort Code</b>	<b>Name</b>	<b>Error Cause</b>
0x2100 0001	Write Data	Error writing data
0x2100 0002	Read Data	Error reading data

Table 8-26 Interface Layer “RS232” Errors

**8.2.2.2 Interface Layer “CAN” Errors**

<b>Abort Code</b>	<b>Name</b>	<b>Error Cause</b>
0x2200 0001	Receive CAN Frame	Error receiving CAN frame
0x2200 0002	Transmit CAN Frame	Error transmitting CAN frame

Table 8-27 Interface Layer “CAN” Errors

**8.2.2.3 Interface Layer “USB” Errors**

<b>Abort Code</b>	<b>Name</b>	<b>Error Cause</b>
0x2300 0001	Write Data	Error writing data
0x2300 0002	Read Data	Error reading data

Table 8-28 Interface Layer “USB” Errors

**8.2.3 Protocol Layer Errors****8.2.3.1 Protocol Layer “MaxonRS232” Errors**

<b>Abort Code</b>	<b>Name</b>	<b>Error Cause</b>
0x3100 0001	NegAckReceived	Negative acknowledge received
0x3100 0002	BadCrcReceived	Bad checksum received
0x3100 0003	BadDataSizeReceived	Bad data size received

Table 8-29 Protocol Layer “MaxonRS232” Errors

**8.2.3.2 Protocol Layer “CANopen” Errors**

<b>Abort Code</b>	<b>Name</b>	<b>Error Cause</b>
0x3200 0001	SdoReceiveFrameNotReceived	CAN frame of SDO protocol not received
0x3200 0002	RequestedCanFrameNotReceived	Requested CAN frame not received
0x3200 0003	CanFrameNotReceived	Can frame not received

Table 8-30 Protocol Layer “CANopen” Errors

**8.2.3.3 Protocol Layer “USB” Errors**

Abort Code	Name	Error Cause
0x3400 0001	Stuffing	Failed stuffing data
0x3400 0002	Destuffing	Failed destuffing data
0x3400 0003	BadCrcReceived	Bad CRC received
0x3400 0004	BadDataSizeReceived	Bad data received
0x3400 0005	BadDataSizeWritten	Bad data size written
0x3400 0006	SendFrame	Failed writing data
0x3400 0007	ReceiveFrame	Failed reading data

Table 8-31     Protocol Layer “USB” Errors

## 9 Integration

Consider this chapter as a “How To” on the integration of the library into your programming environment.

The «EPOS Command Library» is an implementation of protocols to communicate between an EPOS Positioning Controller and a PC running Windows 32-Bit and 64-Bit as well as Linux 32-Bit operating systems. All EPOS commands (including generating/sending/receiving data frames) are implemented and they can be called directly from your own program.

Use the library as an easy and simple way to develop your own application. Do not bother about protocol details; the only thing you need to ensure are the correct communication port settings.

The chapter splits into descriptions for Windows (→as of page 9-121) and Linux (→as of page 9-133) operating systems and comprises the following sections:

- a) Library hierarchy
- b) Integration and programming environment-specific information on how to incorporate the library
- c) Programming and a programming environment-specific example on how to configure and establish communication

### 9.1 Windows Operating Systems

#### 9.1.1 Library Hierarchy

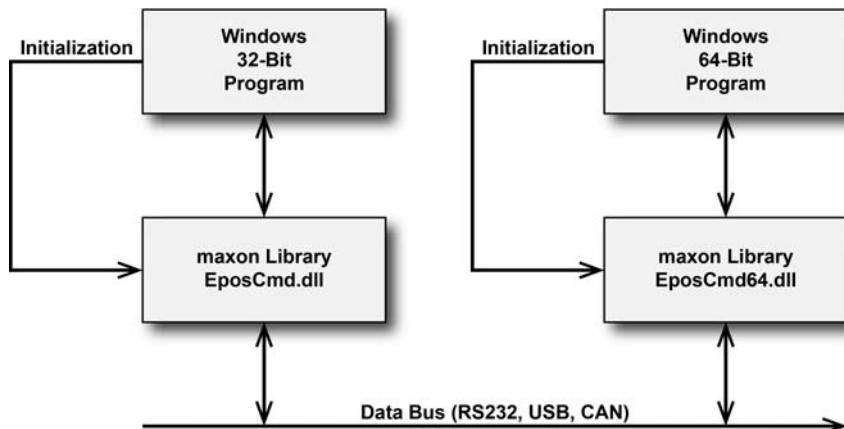


Figure 9-4 Windows – Library Hierarchy (left: 32-Bit OS / right: 64-Bit OS)

#### 9.1.2 Integration into Programming Environment

The way to include the library functions in your own windows program depends on the compiler and the programming language you are using. Subsequently described are the procedures based on the most commonly used programming languages.

To include the library and to establish communication, proceed as follows:

- 1) Copy the library **EposCmd.dll** (for Windows 32-Bit) or **EposCmd64.dll** for Windows 64-Bit) to your working directory.
- 2) Use the function **VCS\_OpenDevice** to configure the library if the settings are known. You also may use the dialog **VCS\_OpenDeviceDlg** to open a port.
- 3) Use the function **VCS\_SetProtocolStackSettings** to select baud rate and timeout.
- 4) Close all opened ports at the end of your program.
- 5) For detailed information on the initialization procedure →chapter “9.1.3 Programming” on page 9-130.

### 9.1.2.1 Borland C++ Builder

You will need to integrate the following files:

- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library
- **EposCmd.lib** – Import library (OMF format)

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Include the file "Definition.h" to your program code using the instruction "#include Definitions.h".
- 3) Add the file "EposCmd.lib" to the project using menu **Project\Add to project**. Select the file and click **Open**.

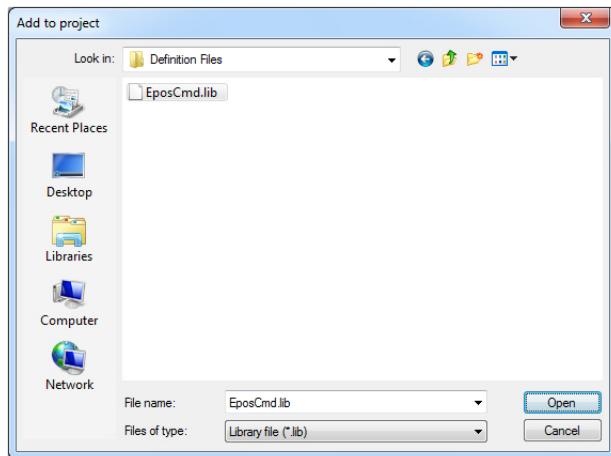


Figure 9-5 Borland C++Builder – Adding Library

- 4) Now, you can execute all library functions in your own code.



#### Best Practice

Use the calling convention `__stdcall`. It will manage how the parameters are put on the stack and how the stack will be cleaned once executed.

### 9.1.2.2 Borland Delphi

You will need to integrate the following files:

- **Definitions.pas** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Write the instruction "Definitions" into the uses clause of your program header.
- 3) Now, you can execute all library functions in your own code.

## 9.1.2.3 Microsoft Visual Basic

**Remark**

The «EPOS Command Library» was developed in programming language Microsoft Visual C++. Take note that data types in Microsoft Visual Basic and Microsoft Visual C++ differ. For more details consult the MSDN library, *Visual Basic Concepts*, →«Converting C Declarations to Visual Basic».

You will need to integrate the following files:

**32-BIT**

- **Definitions.vb** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library

**64-BIT**

- **Definitions.vb** – Constant definitions and declarations of library functions
- **EposCmd64.dll** – Dynamic link library

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Add the file “Definitions.vb” to the project using the project tree in “Solution Explorer”. Click right on **Add**, select **Existing Item**, select the file, and click **Add**.

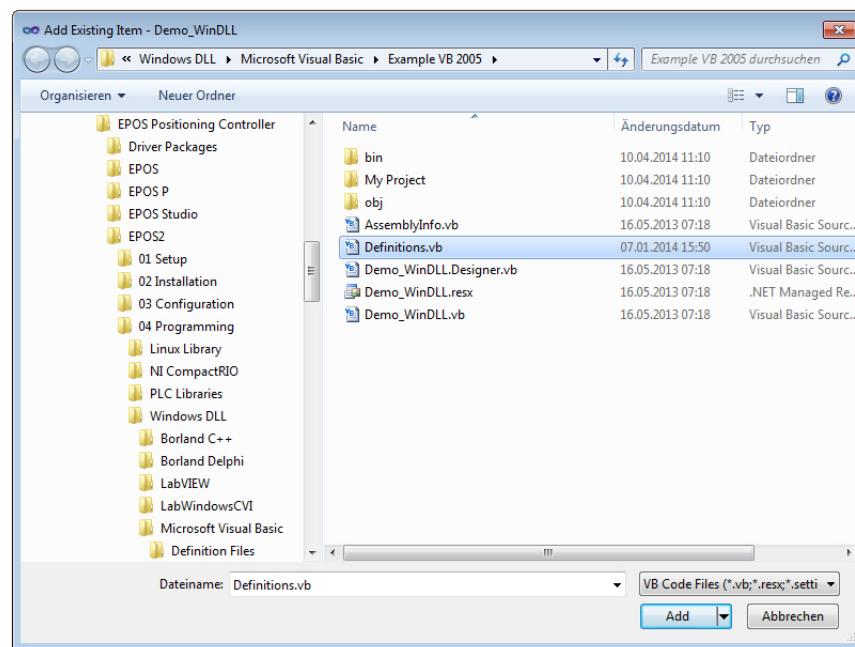


Figure 9-6 Visual Basic – Adding Modules

- 3) Choose one of the two ways:
  - a) Copy the file "EposCmd.dll" (for Windows 32-Bit) or "EposCmd64.dll" for Windows 64-Bit) into the release directory.
  - b) Open menu **Properties**, switch to the **Compile** tab and type ".\\" into the **Build output path** edit line.

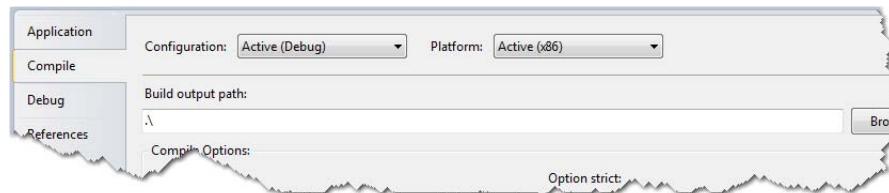


Figure 9-7 Visual Basic – Output Path

- 4) Now, you can execute all library functions in your own code.

#### 9.1.2.4 Microsoft Visual Basic .NET

You will need to integrate the following files:

- **EposCmd.Net.dll** – .Net assembly
- **EposCmd.dll/ EposCmd64.dll** – Dynamic link library

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Add the .NET assembly "EposPCmd.Net.dll" to the project references using the project tree in "Solution Explorer". Click right on **Add**, select **Existing Item**, select the file, and click **Add**.

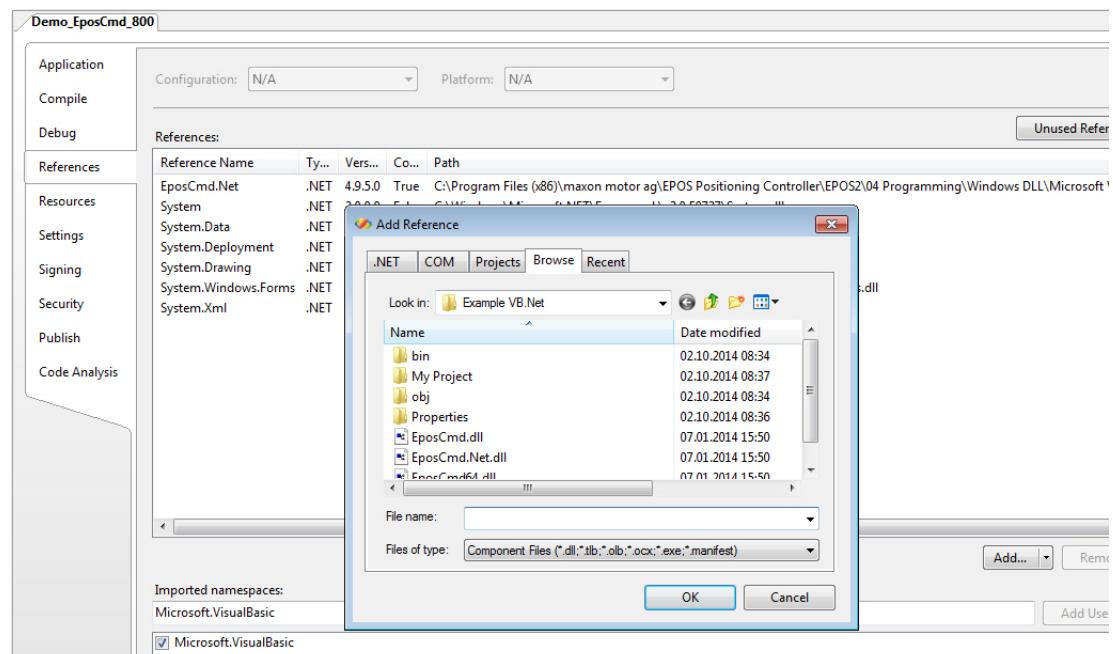


Figure 9-8 Visual Basic .NET – Adding Modules

- 3) Choose one of the two ways:
  - a) Copy the file "EposCmd.dll" (for Windows 32-Bit) or "EposCmd64.dll" (for Windows 64-Bit) into the release directory.
  - b) Open menu **Properties**, switch to the **Compile** tab and type "\." into the **Build output path** edit line.



Figure 9-9 Visual Basic .NET – Output Path

- 4) Now, you can execute all library functions in your own code.



#### Remark

For further details and parameter description of the EposCmd.Net wrapper → separate document «EposCmd.Net.chm».

#### 9.1.2.5 Microsoft Visual C#

You will need to integrate the following files:

- **EposCmd.Net.dll** – .Net assembly
- **EposCmd.dll/ EposCmd64.dll** – Dynamic link library

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Setup the using directory in your program code using the instruction “using EposCmd.Net;”.
- 3) Add the file “EposCmd.Net” to the project using the project tree in “Solution Explorer”. Click right on «References», select «Add Reference», select the file, and click «OK».

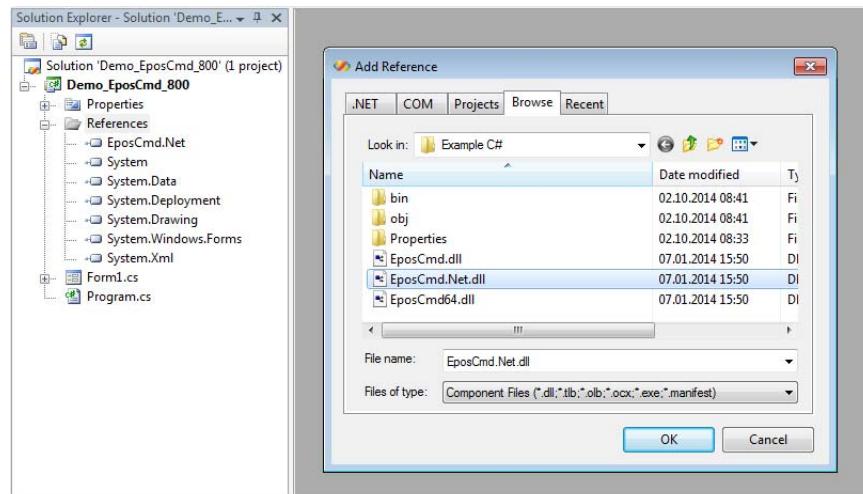


Figure 9-10 Visual C# – Project Settings

- 4) Now, you can execute all library functions in your own code.



#### Remark

For further details and parameter description of the EposCmd.Net wrapper → separate document «EposCmd.Net.chm».

### 9.1.2.6 Microsoft Visual C++

You will need to integrate the following files:

#### 32-BIT

- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library
- **EposCmd.lib** – Import library (COFF format)

#### 64-BIT

- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd64.dll** – Dynamic link library
- **EposCmd64.lib** – Import library (COFF format)

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Include the file "Definition.h" to your program code using the instruction "#include Definitions.h".
- 3) Add the library to your project using menu **Project\Properties**. Select **Linker\Input** from the tree and type the file name "EposCmd.lib" (for Windows 32-Bit) or "EposCmd64.lib" (for Windows 64-Bit) into the **Additional Dependencies** edit line.

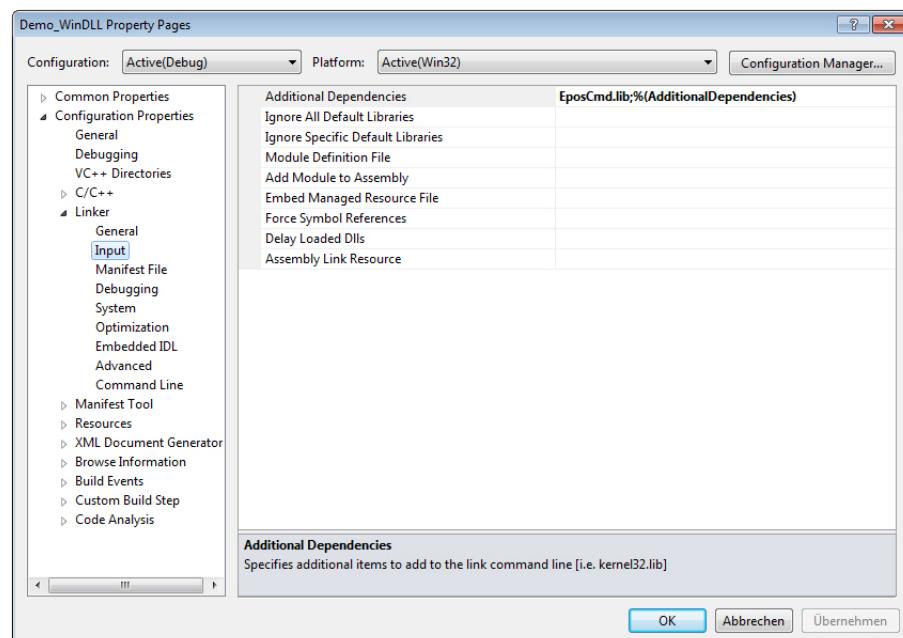


Figure 9-11 Visual C++ – Project Settings

- 4) Now, you can execute all library functions in your own code.



#### Best Practice

Use the calling convention `__stdcall`. It will manage how the parameters are put on the stack and how the stack will be cleaned once executed.

### 9.1.2.7 National Instruments LabVIEW

LabVIEW offers a function block to include external library functions. For each library function of the EPOS library, a VI will be created.

For an easy start with LabVIEW programming, most of the function blocks are already configured in an instrument driver. Each EPOS command has a VI block:

- 32-Bit VIs are supported with LabVIEW 7.1 and higher
- 64-Bit VIs are supported with LabVIEW 2010 64-Bit and higher

You will need to integrate the following files:

#### 32-BIT

- **EPOSLibrary.llb** – LabVIEW Library
- **EposCmd.dll** – Dynamic link library

#### 64-BIT

- **EPOSLibrary.llb** – LabVIEW Library
- **EposCmd64.dll** – Dynamic link library

Proceed as follows:

- 1) Copy the directory "maxon EPOS" to the LabVIEW program directory in path "...\\National Instruments\\LabVIEW X.x\\instr.lib".
- 2) Make the file "EposCmd.dll" (for Windows 32-Bit) or "EposCmd64.dll" (for Windows 64-Bit) available.
- 3) Include (and use) the maxon EPOS Instrument Driver VIs via the functions **maxon EPOS**, **Instrument Drivers**, or **Instrument I/O**.

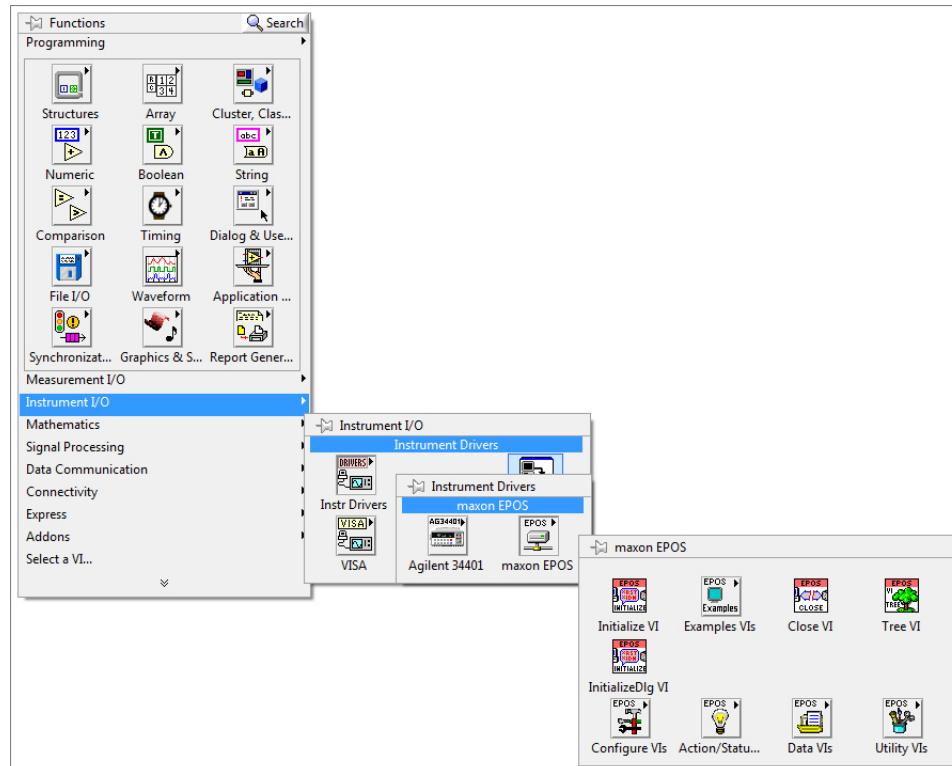


Figure 9-12 LabVIEW – maxon EPOS Instrument Driver

- 4) Now, you can execute all library functions in your own code.

**9.1.2.8 National Instruments LabWindows**

You will need to integrate the following files:

**32-BIT**

- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library
- **EposCmd.lib** – Import library

**64-BIT**

- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd64.dll** – Dynamic link library
- **EposCmd64.lib** – Import library

**Import Library (\*.lib)**

*The import library is dependent on compiler:*

- For Borland compiler use the file from directory "...\\borland".
- For Microsoft Visual C++ compiler use the file from directory "...\\msvc".

Proceed as follows:

- 1) Copy the files to the working directory of your project.
- 2) Include the file "Definition.h" to your program code using the instruction "#include Definitions.h".
- 3) Add the files...
  - "Definitions.h", "EposCmd.dll", "EposCmd.lib" (for Windows 32-Bit) or
  - "Definitions.h", "EposCmd64.dll", EposCmd64.lib (for Windows 64-Bit)
 ... to your project using menu **Edit>Add to project**.  
 Click **All Files...**, select the files, and click **Add**.

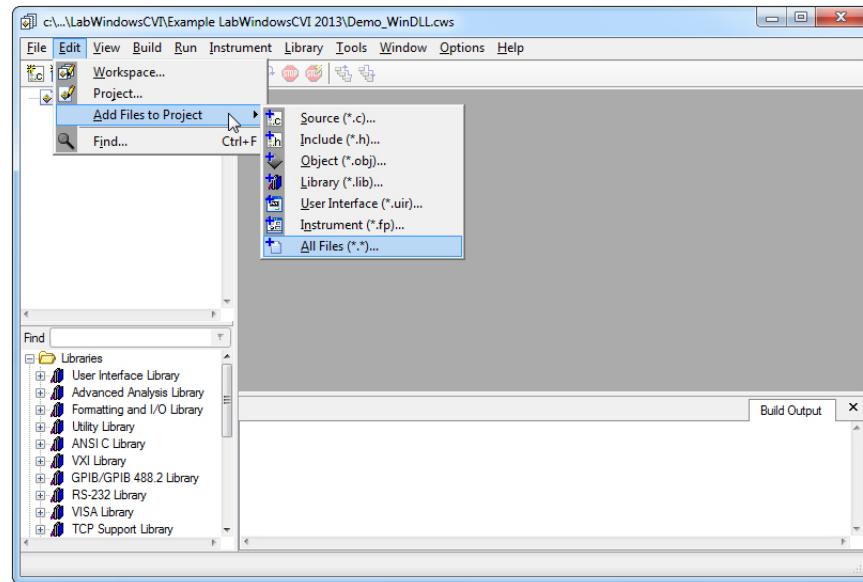


Figure 9-13 LabWindows – add Files to Project

- 4) Now, you can execute all library functions in your own code.

**Best Practice**

Use the calling convention `__stdcall`. It will manage how the parameters are put on the stack and how the stack will be cleaned once executed.

### 9.1.3 Programming

For correct communication with the EPOS, you must execute an initialization function before the first communication command. The fundamental program flow is as follows:

#### INITIALIZATION

Execute the functions at the beginning of the program.

Function	Description
VCS_OpenDevice	Initialization of the port with the user data. Use the help functions for information on the interface settings.
VCS_OpenDeviceDlg	Initialization of the port. The dialog shows all available communication ports.
VCS_SetProtocolStackSettings	Initialization of the new baud rate and timeout
VCS_ClearFault	Deletes possibly existent errors/warnings

#### HELP

Use the functions if you do not exactly know how your interface is configured.

Function	Description
VCS_GetDeviceNameSelection	Returns available DeviceNames for function VCS_OpenDevice
VCS_GetProtocolStackNameSelection	Returns available ProtocolStackNames for function VCS_OpenDevice
VCS_GetInterfaceNameSelection	Returns available InterfaceNames for function VCS_OpenDevice
VCS_GetPortNameSelection	Returns available PortNames for function VCS_OpenDevice

#### COMMUNICATION WITH EPOS

Choose any of the EPOS commands.

Function	Description
VCS_OperationMode	Set the operation mode (Position Mode, Profile Position Mode, Current Mode, ...)
VCS_GetEncoderParameter	Read all encoder parameters
etc.	

#### CLOSING PROCEDURE

Release the port before closing the program.

Function	Description
VCS_CloseDevice	Release the opened port
VCS_CloseAllDevices	Release all opened ports

## 9.1.3.1 Examples



### Applicability

- For an universally valid example applicable for most programming environments → [Demo\\_WinDLL](#).
- For a National Instruments LabView-specific example → [LabVIEW](#).



### Best Practice

Prior starting one of the example programs, set the control parameters (e.g. motor, sensor, and regulator parameters). Use the «EPOS Studio» for configuration.

### DEMO\_WINDLL

The example “Demo\_WinDLL” is a dialog-based application. It demonstrates how to configure communication with the EPOS device.

- 1) A configuration dialog will open as you adjust your communication settings.
- 2) At the beginning, the EPOS is set into “Profile Position Mode”. Initialization is programmed in the member function **Create()** of the class **Demo\_WinDLL**. The opened port is released at the end in the function **Destroy()**.
- 3) You can execute the EPOS commands by clicking the buttons.
  - VCS\_SetEnableState
  - VCS\_SetDisableState
  - VCS\_MoveToPosition
  - VCS\_HaltPositionMovement

The function **VCS\_MoveToPosition** may be used as absolute or relative positioning. Click to change your communication settings.

A timer triggers a periodical update of the state and actual position. The function **UpdateStatus()** will be executed every 100 ms. If an error occurs during the update of the state, the timer is stopped and an error report is displayed.

## LABVIEW

The maxon EPOS instrument driver contains the following example VIs:

### MOVEWITHVELOCITY

Example to perform a velocity movement showing how to...

- initialize and close an interface (e.g. USB)
- start a velocity movement with correct operation mode
- wait until the target velocity is reached (e.g. 5 seconds)

### MOVETORELATIVEPOSITION

Example to do a relative position step showing how to...

- initialize and close an interface (e.g. USB)
- start positioning with correct operation mode
- wait until the target position is reached

### DATARECORDER

Example to configure and use the data recording functions showing how to...

- initialize and close an interface (e.g. USB)
- configure the data recorder
- start relative positioning
- display the recorded data (position, velocity, current)

### GUI DEMO

Example on how to work with maxon EPOS VIs showing how to...

- initialize and close an interface (with a dialog)
- configure parameters and data
- enable/disable a device
- start/stop a relative movement
- configure profile and node settings
- use the data recorder
- update actual values

### MOVEWITHIPM

Example on how to do an IPM trajectory showing how to...

- initialize and close an interface (e.g. USB)
- configure interpolated position mode parameters
- start IPM trajectory
- add PVT reference points
- stop IPM trajectory

## 9.2 Linux Operating Systems

### 9.2.1 Library Hierarchy

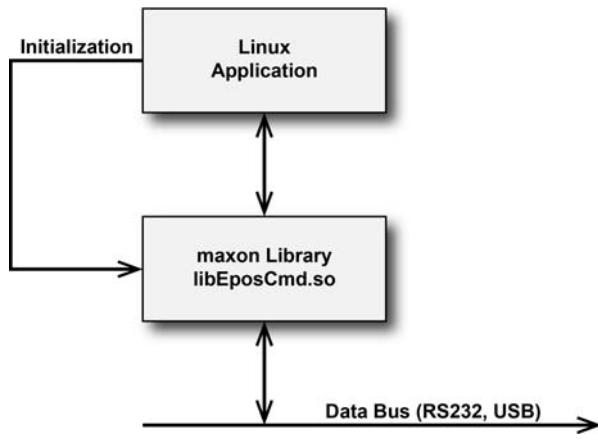


Figure 9-14     Linux – Library Hierarchy

### 9.2.2 Framework Conditions

#### TESTBED

The library has been tested with the following operating systems:

- x86/x64
  - Ubuntu 12.04...14.04
  - Mint 16...17
- arm
  - Ubuntu 13.04

With the library, most EPOS commands are available. However, not supported are...

- DataRecorder
- Export/Import parameters
- GUI-related functions (i.e. VCS\_OpenDeviceDlg)

#### DEVELOPMENT TOOLS

The following items are required:

- Eclipse IDE for C/C++ Developer or Eclipse with CDT plugin ([→http://eclipse.org/](http://eclipse.org/))
- gcc/g++ GNU compiler and libraries ([→http://gcc.gnu.org/](http://gcc.gnu.org/))

### 9.2.3 Integration into Programming Environment

You will need to integrate the following files:

- **Definitions.h** – Constant definitions and declarations of library functions
- **libEposCmd.so.<major>.<minor>.<rev>.0** – EPOS Linux Shared library
- **libftd2xx.so.<major>.<minor>.<rev>** – FTDI library

## 9.2.4 Installation

### 9.2.4.1 EPOS Command Library

- 1) Download and extract the zip file to a temporary directory.
- 2) Select your target architecture directory (x86, x86\_64, arm hf/sf).
- 3) Take note that the following actions will require root privileges.
- 4) Copy the shared library file "libEposCmd.so.5.0.1.0" to the directory "/usr/local/lib" (or any directory listed in the LD\_LIBRARY\_PATH):
  - a) sudo cp libEposCmd.so.5.0.1.0 /usr/local/lib
  - b) Create a soft link to this library into directory "/usr/lib".
  - c) Execute the following commands from directory "/usr/lib":
    - cd /usr/lib
    - sudo ln -s /usr/local/lib/libEposCmd.so libEposCmd.so
- 5) Optional: Copy the file "Definitions.h" to the directory "/usr/include".

### 9.2.4.2 FTDI Library



#### Best Practice

The «EPOS Command Library» features built-in USB support. Even if you do not plan to use the USB interface in your application, you will need to install the FTD library on the target host.

- 1) Download the latest version from →<http://www.ftdichip.com/Drivers/D2XX.htm> and extract the zip file to a temporary directory.
- 2) Select your target architecture directory (x86, x86\_64, arm hf/sf).
- 3) Take note that the following actions will require root privileges.
- 4) Copy the shared library file "libftd2xx.so.1.1.12" to the directory "/usr/local/lib".
  - a) sudo cp libftd2xx.so.1.1.12 /usr/local/lib
  - b) Create soft links to this library into directory "/usr/local/lib" and as root. Execute the following commands:
    - cd /usr/local/lib
    - sudo ln -s libftd2xx.so.1.1.12 libftd2xx.so
  - c) Execute the following commands from the directory "/usr/lib" as root:
    - cd /usr/lib
    - sudo ln -s /usr/local/lib/libftd2xx.so libftd2xx.so
- 5) To allow the non-root user to work with EPOS controllers over USB, the permissions must be properly set. Do so by configuring a new rule for the linux device manager (the so-called "udev server").  
Copy the file EPOS file "99-ftdi.rules" to the directory "/etc/udev/rules".
- 6) Restart the udev service to adopt the changes:
  - sudo service udev restart
  - or, alternatively,
  - sudo /etc/init.d/udev restart

### 9.2.5 HelloEposCmd

Proceed as follows for the Terminal C++ example project «HelloEposCmd»:

- 1) Find the Eclipse project «HelloEposCmd» in the directory “Epos\EposCmdLinux\Examples”.
- 2) Import the project into your workspace.
- 3) Optional: Include the preconfigured libraries for gcc linker (-L option):
  - EposCmd
  - ftd2xx

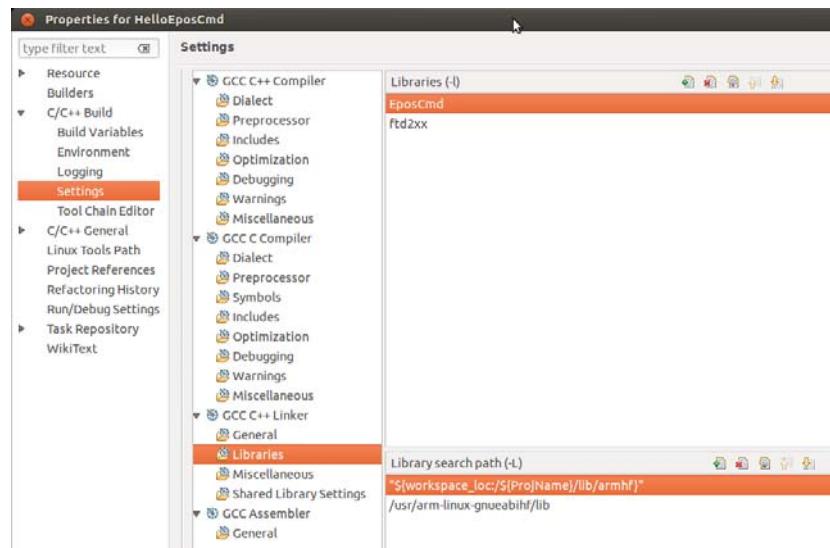


Figure 9-15 Eclipse – Libraries Configuration

- 4) Set the target build configuration and start the build. Thereby, make use of the preconfigured debug and release build configurations for x86, x64, or armhf/sf.

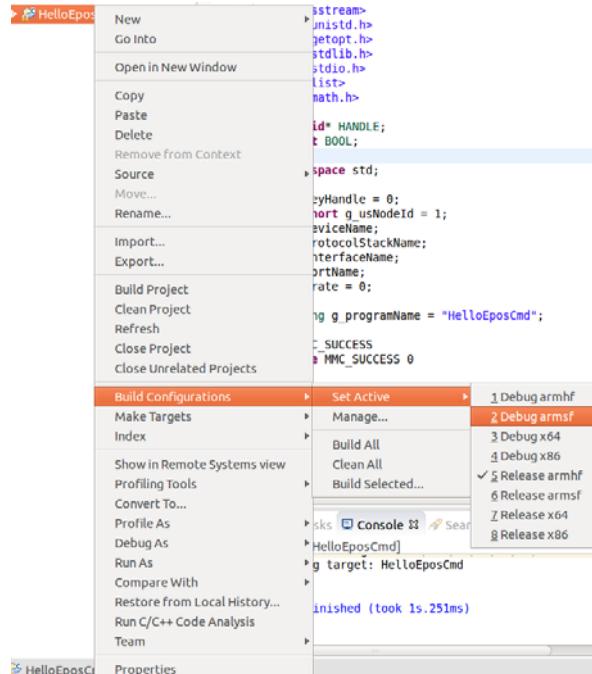


Figure 9-16 Eclipse – Build Configuration

- 5) Optional: Include the file "Definition.h".

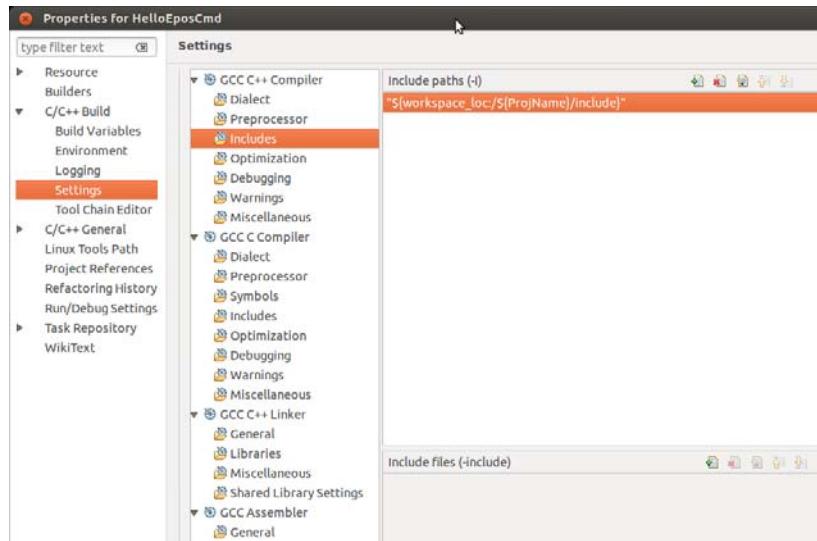


Figure 9-17 Eclipse – Includes Configuration

- 6) Set the program options as to the following scheme:

HelloEposCmd [-h] [-d] device [-s] protocol stack [-i] interface [-p] port [-b] baudrate [-n] node id

- b      baudrate; for example "115200" for RS232 or "1000000" for USB (default)
- d      device name; for example "EPOS2" (default) or "EPOS"
- i      interface name; for example "USB" (default) or "RS232"
- n      node id (the controller node identifier) (default "1" = USB interface)
- p      port name; for example "USB0" (default) or "/dev/ttyS0", "COM1"
- s      protocol stack name; for example "MAXON SERIAL V2" (default) or "MAXON RS232"

```
ubuntu@ubuntu-armhf:~$ ./HelloEposCmd
-----
Epos Command Library Example Program, (c) maxonmotor ag 2014
-----
default settings:
node id          = 1
device name      = 'EPOS2'
protocol stack name = 'MAXON SERIAL V2'
interface name   = 'USB'
port name        = 'USB0'
baudrate         = 1000000
-----
Open device...
set profile velocity mode, node = 1
move with target velocity = 1000 rpm, node = 1
move with target velocity = 300 rpm, node = 1
move with target velocity = 600 rpm, node = 1
move with target velocity = 900 rpm, node = 1
move with target velocity = 200 rpm, node = 1
halt velocity movement
set profile position mode, node = 1
move to position = 5000, node = 1
move to position = -20000, node = 1
move to position = 20000, node = 1
move to position = -10000, node = 1
move to position = 10000, node = 1
move to position = -10000, node = 1
move to position = 5000, node = 1
halt position movement
Close device
ubuntu@ubuntu-armhf:~$
```

Figure 9-18 Eclipse – HelloEposCmd with Default Settings

- 7) A properly configured and tuned controller presumed, you now will get several movements in «Position Mode» and «Velocity Mode».

## 10 Version History

Date [d.m.y]	DLL Version	Documentation Edition	Description
24.10.2014	5.0.1.0	October 2014	Documentation update New: Support for Kvaser CAN interfaces New: Support for NI-XNET driver
17.12.2013	4.9.5.0	December 2013	Documentation update Bugfix: Function VCS_GetDriverInfo 64-Bit variant DataRecorder: Check path (VCS_ExportChannelDataToFile)
22.03.2013	4.9.2.0	March 2013	Function VCS_ExportParamter: Parameters renamed
04.01.2013	4.9.1.0	December 2012	New functions: VCS_GetHomingState, VCS_WaitForHomingAttained, VCS_GetVelocityIsAveraged, VCS_GetCurrentIsAveraged
10.10.2012	4.8.7.0	October 2012	Bugfix: Command Send NMT Service New functions: VCS_SetVelocityRegulatorFeedForward, VCS_SetVelocityRegulatorFeedForward
08.10.2012	4.8.6.0	October 2012	New: CANopen Vector Interface support for VN1600 series
10.04.2012	4.8.5.0	April 2012	Bugfix: Sporadic CAN failure with IXXAT VCI V3.3
02.02.2011	4.8.2.0	February 2011	Bugfix: NI-LIN device
28.01.2011	4.8.1.0	January 2011	New: Expand to 64-Bit Windows OS and 32-Bit Linux OS Bugfix: Segmented Write
28.10.2010	4.7.3.0	November 2010	Bugfix: VCS_CloseDevice, VCS_CloseAllDevices
11.10.2010	4.7.2.0	October 2010	Bugfix: Deadlock when closing Application fixed Bugfix: Communication for IXXAT VCI V3.3 fixed
30.08.2009	4.7.1.0	August 2010	New parameters: DialogMode for Findxxx functions New: ProtocolStack Name "MAXON SERIAL V2" (Library is still compatible with old name "EPOS2_USB") Bugfix: VCS_WaitForTargetReached returns false, if timeout elapses
22.10.2009	4.6.1.3	October 2009	Bugfix: Multithreading
04.09.2009	4.6.0.0	September 2009	New: Support for EPOS2 functionality, data recorder, parameter export and import, VCS_ReadCANFrame
01.05.2008	4.5.0.0	April 2008	New: Functions for read device errors (Get Device Error), adaption for EPOS2
10.08.2007	4.4.0.0	August 2007	New: Support for IXXAT VCI V3
01.02.2007	4.3.0.0	January 2007	New: Support for National Instruments Interfaces
16.10.2006	4.2.1.0	October 2006	Bugfix: VCS_GetDriverInfo, VCS_SetHomingParameter
11.10.2006	4.2.0.0	October 2006	New function: VCS_GetErrorInfo(...)
12.04.2006	4.1.1.0	April 2006	Bugfix: VCS_SendCANFrame
12.04.2006	4.1.0.0	April 2006	New error codes
03.02.2006	4.0.0.0	February 2006	Additional information on error codes
01.10.2005	4.0.0.0	October 2005	Error correction documentation
01.03.2005	3.0.0.0	March 2005	Insert from Vector CAN cards details
16.07.2004	2.0.3.0	July 2004	Documentation update New: Additional information on error codes

## Version History

Date [d.m.y]	DLL Version	Documentation Edition	Description
06.04.2004	2.0.0.0	April 2004	New functions documented: VCS_CloseAllDevices(...), VCS_DigitalInputConfiguration(...), VCS_DigitalOutputConfiguration(...), VCS_GetAllDigitalInputs(...), VCS_GetAllDigitalOutputs(...), VCS_GetAnalogInput(...), VCS_SetAllDigitalOutputs(...), VCS_SendNMTService(...), VCS_OpenDeviceDlg(...) Changed functions: VCS_GetBaudrateSelection(...), VCS_FindHome(...), VCS_GetHomingParameter(...), VCS_SetHomingParameter(...), VCS_MoveToPosition(...), VCS_GetOperationMode(...), VCS_SetOperationMode(...), VCS_GetObject(...), VCS_SetObject(...) Deleted functions: VCS_GetProtocolStackMode(...), VCS_GetProtocolStackModeSelection(...)
05.01.2004	1.02	January 2004	Insert IXXAT details
01.12.2003	1.01	December 2003	Changed functions: VCS_GetBaudrateSelection(...), VCS_GetDeviceName(...), VCS_SetDeviceNameSelection(...), VCS_GetDriverInfo(...), VCS_SetInterfaceName(...), VCS_SetInterfaceNameSelection(...), VCS_SetPortName(...), VCS_SetPortNameSelection(...), VCS_SetProtocolStackModeSelection(...), VCS_SetProtocolStackName(...), VCS_SetProtocolStackNameSelection(...)
11.11.2003	1.00	November 2003	Initial release

Table 10-32 Version History

**LIST OF FIGURES**

Figure 2-1	Documentation Structure .....	9
Figure 2-2	Windows – Communication Structure (Example).....	11
Figure 2-3	Linux – Communication Structure (Example).....	11
Figure 9-4	Windows – Library Hierarchy (left: 32-Bit OS / right: 64-Bit OS) .....	121
Figure 9-5	Borland C++Builder – Adding Library.....	122
Figure 9-6	Visual Basic – Adding Modules .....	123
Figure 9-7	Visual Basic – Output Path.....	124
Figure 9-8	Visual Basic .NET – Adding Modules.....	125
Figure 9-9	Visual Basic .NET – Output Path .....	125
Figure 9-10	Visual C# – Project Settings.....	126
Figure 9-11	Visual C++ – Project Settings.....	127
Figure 9-12	LabVIEW – maxon EPOS Instrument Driver .....	128
Figure 9-13	LabWindows – add Files to Project .....	129
Figure 9-14	Linux – Library Hierarchy .....	133
Figure 9-15	Eclipse – Libraries Configuration .....	135
Figure 9-16	Eclipse – Build Configuration .....	135
Figure 9-17	Eclipse – Includes Configuration .....	136
Figure 9-18	Eclipse – HelloEposCmd with Default Settings .....	136

**LIST OF TABLES**

Table 1-1	Notations used in this Document .....	5
Table 1-2	Sources for additional Information .....	6
Table 1-3	Brand Names and Trademark Owners.....	7
Table 2-4	Supported Platforms, Architectures, and Interfaces.....	10
Table 2-5	Third Party Supplier Products .....	10
Table 2-6	Data Type Definitions.....	12
Table 4-7	Motor Types .....	28
Table 4-8	Position Sensor Types .....	31
Table 4-9	Digital Input Configuration .....	43
Table 4-10	Digital Output Configuration.....	44
Table 4-11	Analog Input Configuration.....	45
Table 4-12	Velocity Notation Index.....	46
Table 5-13	Operation Modes .....	47
Table 5-14	State Modes .....	49
Table 5-15	Homing Methods .....	70
Table 5-16	Position Compare – Operational Modes .....	97
Table 5-17	Position Compare – Interval Modes.....	97
Table 5-18	Position Compare – Direction Dependency .....	97
Table 5-19	Position Marker Edge Types .....	101
Table 5-20	Position Marker Modes.....	101
Table 6-21	Data Recorder Trigger Types.....	106
Table 7-22	Command Specifier .....	116
Table 8-23	Communication Errors .....	117
Table 8-24	General Errors .....	118
Table 8-25	Interface Layer Errors.....	118
Table 8-26	Interface Layer “RS232” Errors .....	119
Table 8-27	Interface Layer “CAN” Errors .....	119
Table 8-28	Interface Layer “USB” Errors .....	119
Table 8-29	Protocol Layer “MaxonRS232” Errors .....	119
Table 8-30	Protocol Layer “CANopen” Errors .....	119
Table 8-31	Protocol Layer “USB” Errors.....	120
Table 10-32	Version History.....	138

## INDEX

### A

applicable EU directive **2**  
architectures (supported) **10**

### B

Borland C++ (integration into) **122**  
Borland Delphi (integration into) **122**

### C

CM (Current Mode) functions **86**  
configuration functions **25**

### D

data recording functions **105**  
data type definitions **12**  
Delphi (integration into) **122**  
development tools **133**  
drivers by 3rd party manufacturers **10**

### E

Eclipse (integration into) **133**  
EPOS Command Library, integration of **121**  
error codes (overview) **117**  
Error Handling functions **54**  
EU directive, applicable **2**

### F

functions  
    configuration **25**  
    data recording **105**  
    initialization **13**  
    low layer **115**  
    operation **47**  
functions for  
    Current Mode **86**  
    Homing Mode **68**  
    inputs & outputs **94**  
    Interpolated Position Mode **73**  
    Master Encoder Mode **90**  
    Position Mode **78**  
    Profile Position Mode **60**  
    Profile Velocity Mode **64**  
    Step Direction Mode **92**  
    Velocity Mode **82**

### H

hierarchy (of library) **121, 133**  
HM (Homing Mode) functions **68**  
homing methods **70**  
how to  
    integrate the «EPOS Command Library» **121**  
    interpret icons (and signs) used in the document **6**  
    read this document **2**

### I

incorporation into surrounding system **2**  
initialization functions **13**  
input/output functions **94**  
interfaces (supported) **10**  
IPM (Interpolated Position Mode) functions **73**  
IXXAT (supported devices) **10**

### K

Kvaser (supported devices) **10**

### L

LabVIEW (integration into) **128**  
LabWindows (integration into) **129**  
legal notice **7**  
Linux  
    communication structure **11**  
    framework conditions **133**  
    library hierarchy **133**  
low layer functions **115**

### M

manufacturers of supported products **10**  
MEM (Master Encoder Mode) functions **90**  
Motion Info functions **56**

### N

National Instruments (supported devices) **10**  
NI-CAN **10**  
NI-XNET **10**  
non-compliance of surrounding system **2**

### O

operating license **2**  
operating systems (supported) **10**  
operation functions **47**  
Operation Mode functions **47**  
other machinery (incorporation into) **2**

## P

platforms (supported) **10**  
PM (Position Mode) functions **78**  
PPM (Profile Position Mode) functions **60**  
prerequisites prior installation **2**  
purpose of this document **5**  
PVM (Profile Velocity Mode) functions **64**

## S

SDM (Step Direction Mode) functions **92**  
signs used **6**  
State Machine functions **49**  
supported interfaces and platforms **10**  
supported products by 3rd party suppliers **10**  
surrounding system (incorporation into) **2**  
symbols used **6**

## V

VCI driver **10**  
Vector (supported devices) **10**  
Visual Basic .NET (integration into) **125**  
Visual Basic (integration into) **123**  
Visual C# (integration into) **126**  
Visual C++ (integration into) **127**  
VM (Velocity Mode) functions **82**

## W

warranty **7**  
Windows  
    communication structure **11**  
    library hierarchies **121**

## X

XL driver **10**

---

*••page intentionally left blank••*

© 2014 maxon motor. All rights reserved.

The present document – including all parts thereof – is protected by copyright. Any use (including reproduction, translation, microfilming and other means of electronic data processing) beyond the narrow restrictions of the copyright law without the prior approval of maxon motor ag, is not permitted and subject to persecution under the applicable law.

**maxon motor ag**

Brünigstrasse 220  
P.O.Box 263  
CH-6072 Sachseln  
Switzerland

Phone +41 41 666 15 00  
Fax +41 41 666 16 50

[www.maxonmotor.com](http://www.maxonmotor.com)