maxon motor

maxon motor control | EPOS Positioning Controllers

Command Library | Edition November 2016

EPOS

Positioning Controllers

Command Library

epos.maxonmotor.com

Document ID: rel6806

Table of Contents

Function Group Overview
For a detailed overview on function groups see page 12-161.

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.*

EPOS *positioning controllers are considered as partly completed machinery according to EU Directive 2006/42/EC, Article 2, Clause (g) and* ***are intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment****.*
Therefore, you must not put the device into service,*…*

- *unless you have made completely sure that the other machinery fully complies with the EU directive's requirements!*
- *unless the other machinery fulfills all relevant health and safety aspects!*
- *unless all respective interfaces have been established and fulfill the herein stated requirements!*

••page intentionally left blank••

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 Shared Object Library «libEposCmd.so»

…which can be used for EPOS, EPOS2, and EPOS4 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
EPOS2	stands for "EPOS2 Positioning Controller"
EPOS4	stands for "EPOS4 Positioning Controller"
«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	http://eclipse.org/
FTDI Driver	www.ftdichip.com
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	www.ixxat.de
Kvaser	www.kvaser.com
maxon motor	www.maxonmotor.com
Microsoft Developer Network (MSDN)	http://msdn.microsoft.com/
National Instruments (NI)	www.ni.com/can
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	www.vector-informatik.com

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
Adobe® Reader®	© Adobe Systems Incorporated, USA-San Jose, CA
Borland C++ Builder™ Borland®	© 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

© 2016 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 prosecution under the applicable law.

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

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

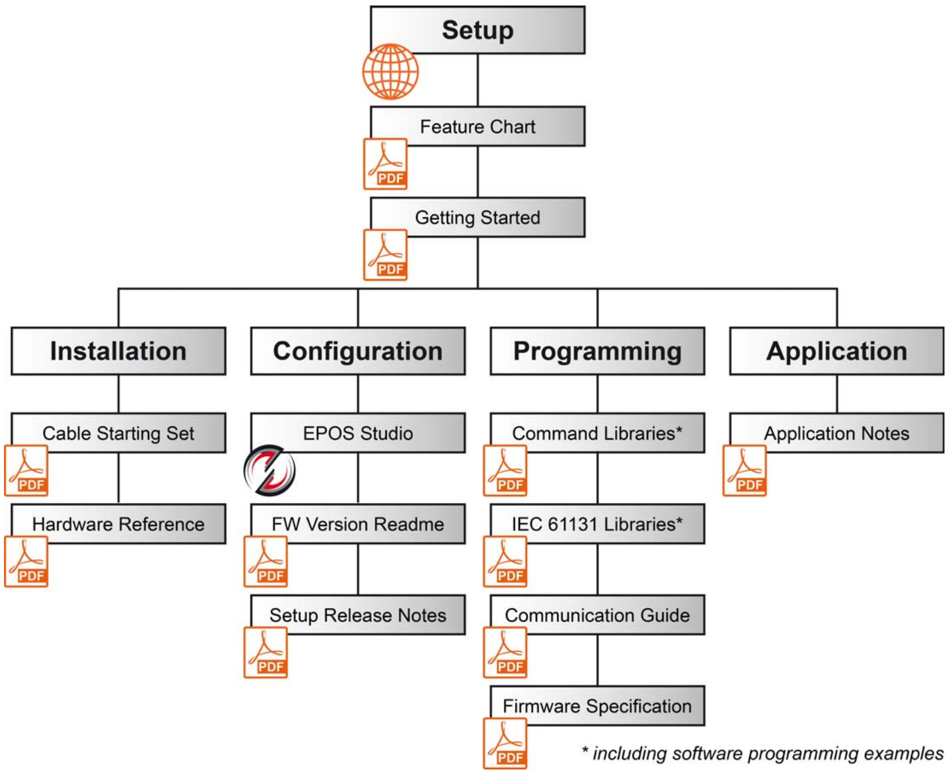


Figure 2-1 EPOS2 documentation structure

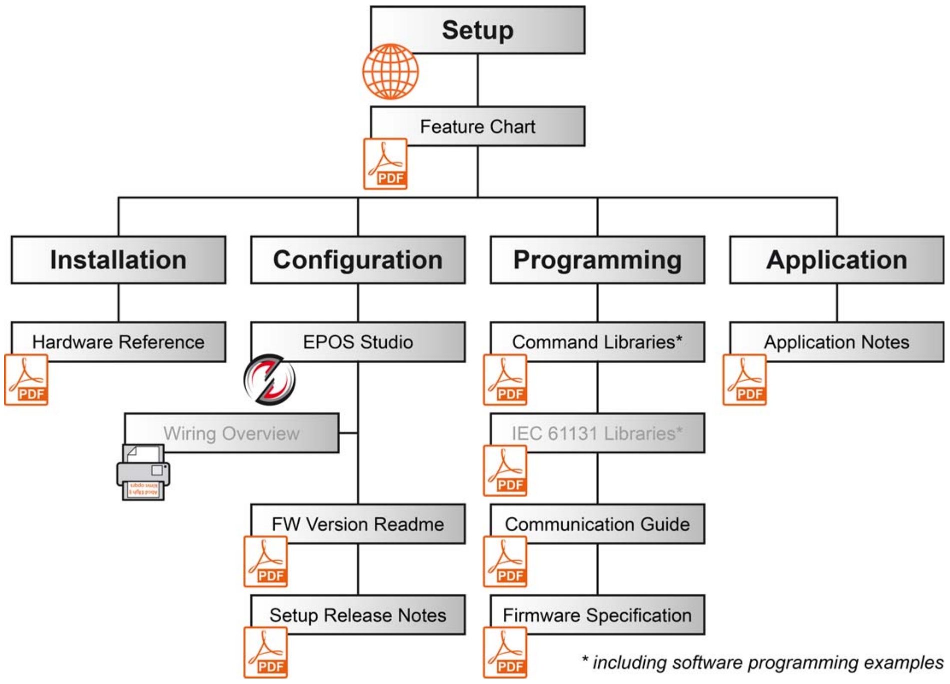


Figure 2-2 EPOS4 documentation structure

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 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 2-11), 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**		
		x86	**x64**	**x86**	**x86_64**	**ARMv7** *1)
RS232		X	X	X	X	X
USB		X	X	X	X	X
CAN Board	IXXAT	X	X	–	–	–
	Kvaser	X	X	–	–	–
	NI	X	X	–	–	–
	Vector	X	X	–	–	–

**1) tested with Raspberry Pi 2/3, Debian Jessie, armhf*

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:

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

2.3 Products by Third Party Suppliers

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

Supplier	Products
IXXAT	IXXAT CANopen interfaces can be operated with the hardware-independent "VCI driver V3" or "VCI driver V4" (Virtual CAN Interface). Check in advanced whether the interface is supported by VCI 3 or VCI 4.
Kvaser	Kvaser CAN interfaces are supported. Thereby, respective driver software and hardware must be installed.
National Instruments	National Instruments CAN interfaces are supported. Thereby, «NI-XNET» or «NI-CAN» software and hardware must be installed.
Vector	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

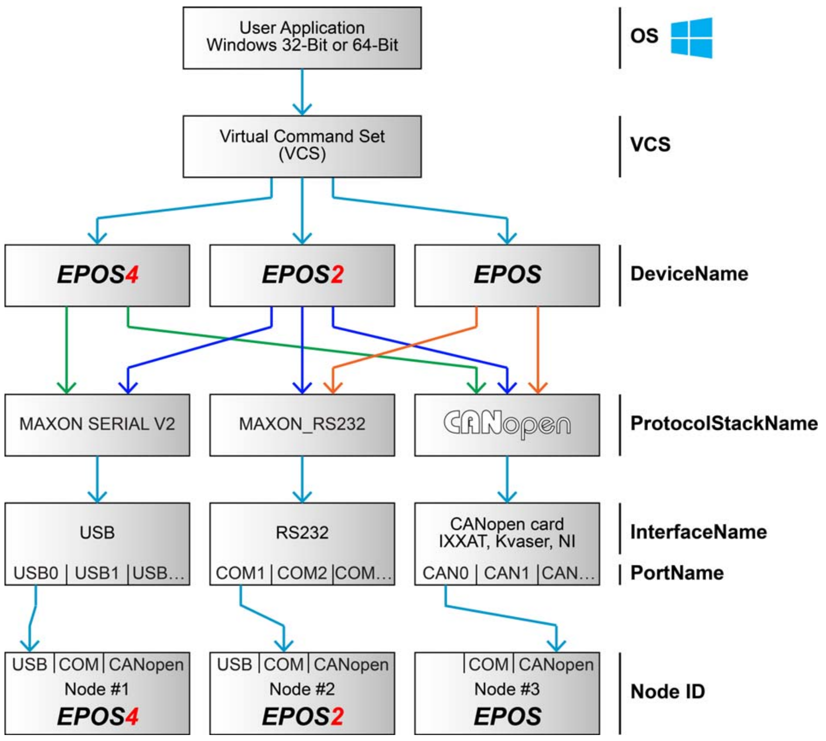


Figure 2-3 Windows – Communication structure (example)

2.4.2 Linux

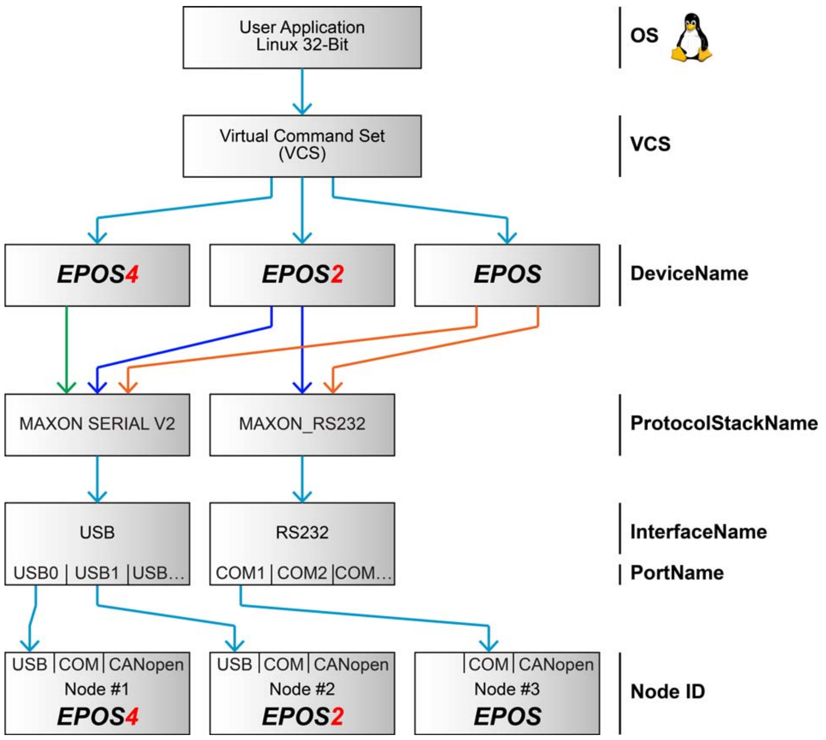


Figure 2-4 Linux – Communication structure (example)

2.4.3 Gateway

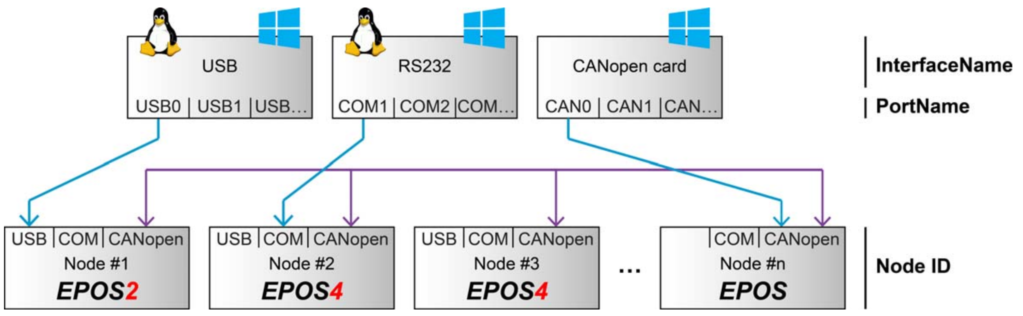


Figure 2-5 Gateway – Communication structure (example)

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

Availability of functions
The availability of certain functions depends on the used hardware. For an overview ➔“Appendix A — Hardware vs. Functions” on page 11-155.

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

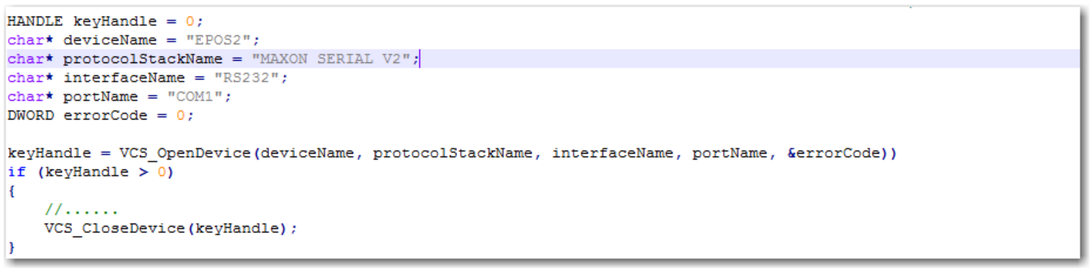
DeviceName	char*	Name of connected device: • EPOS • EPOS2 • EPOS4
ProtocolStackName	char*	Name of used communication protocol: • MAXON_RS232 • MAXON SERIAL V2 • CANopen
InterfaceName	char*	Name of interface: • RS232 • USB • IXXAT_<<BoardName>> <<DeviceNumber>> • Kvaser_<<BoardName>> <<DeviceNumber>> • NI_<<BoardName>> <<DeviceNumber>> • Vector_<<BoardName>> <<DeviceNumber>> *Remark:* Use “VCS_OpenDeviceDlg” or “VCS_GetInterfaceNameSel” to identify the exact name
PortName	char*	Name of port: • COM1, COM2, … • USB0, USB1, … • CAN0, CAN1, …

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

PROGRAMMING EXAMPLE



```
HANDLE keyHandle = 0;
char* deviceName = "EPOS2";
char* protocolStackName = "MAXON SERIAL V2";
char* interfaceName = "RS232";
char* portName = "COM1";
DWORD errorCode = 0;

keyHandle = VCS_OpenDevice(deviceName, protocolStackName, interfaceName, portName, &errorCode))
if (keyHandle > 0)
{
    //......
    VCS_CloseDevice(keyHandle);
}
```

Figure 3-6 OpenDevice (programming example)

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

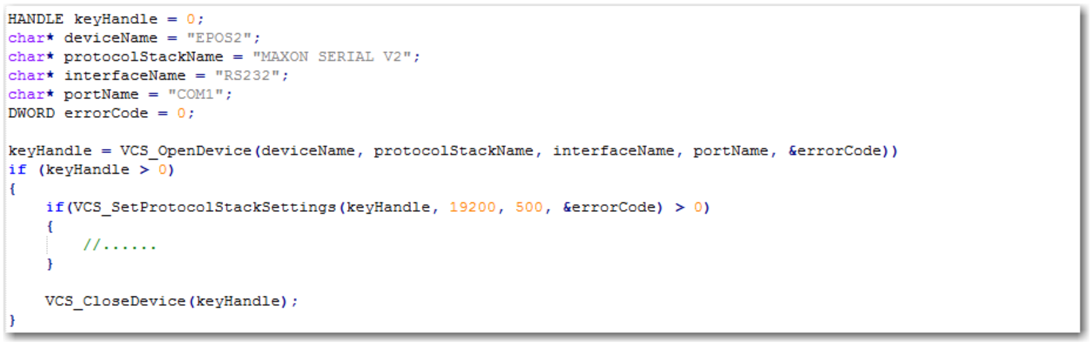
KeyHandle	HANDLE	Handle for port access
Baudrate	DWORD	Actual baud rate from opened port [Bit/s]
Timeout	DWORD	Actual timeout from opened port [ms]

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
HANDLE keyHandle = 0;
char* deviceName = "EPOS2";
char* protocolStackName = "MAXON SERIAL V2";
char* interfaceName = "RS232";
char* portName = "COM1";
DWORD errorCode = 0;

keyHandle = VCS_OpenDevice(deviceName, protocolStackName, interfaceName, portName, &errorCode))
if (keyHandle > 0)
{
    if(VCS_SetProtocolStackSettings(keyHandle, 19200, 500, &errorCode) > 0)
    {
        //......
    }

    VCS_CloseDevice(keyHandle);
}
```

Figure 3-7 SetProtocolStackSettings (programming example)

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. The search will be terminated as the first device is found. 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

Return Value	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. If no opened ports are available, the function returns “0”.

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. If no opened ports are available, the function returns “0”.

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 8-131.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	char*	Error string

Return Value	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.2.3 Get Version

FUNCTION

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

DESCRIPTION

VCS_GetVersion returns the firmware version.

PARAMETERS

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

RETURN PARAMETERS

pHardwareVersion	WORD*	Hardware version
pSoftwareVersion	WORD*	Software version
pApplicationNumber	WORD*	Application number
pApplicationVersion	WORD*	Application version
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.

PARAMETERS

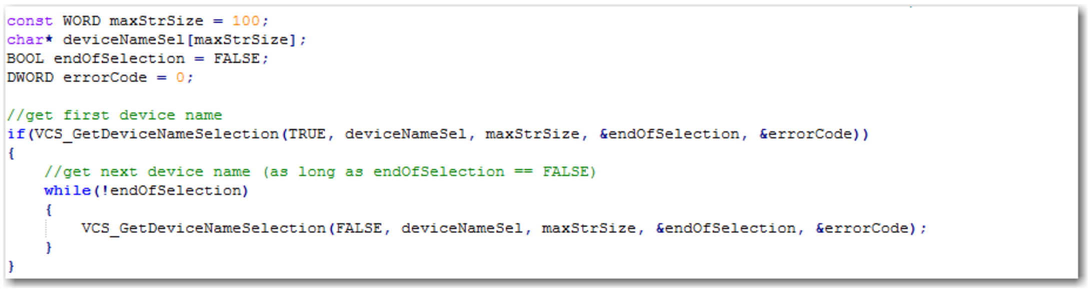
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the device name

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
const WORD maxStrSize = 100;
char* deviceNameSel[maxStrSize];
BOOL endOfSelection = FALSE;
DWORD errorCode = 0;

//get first device name
if(VCS_GetDeviceNameSelection(TRUE, deviceNameSel, maxStrSize, &endOfSelection, &errorCode))
{
    //get next device name (as long as endOfSelection == FALSE)
    while(!endOfSelection)
    {
        VCS_GetDeviceNameSelection(FALSE, deviceNameSel, maxStrSize, &endOfSelection, &errorCode);
    }
}
```

Figure 3-8 GetDeviceNameSelection (programming example)

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.

PARAMETERS

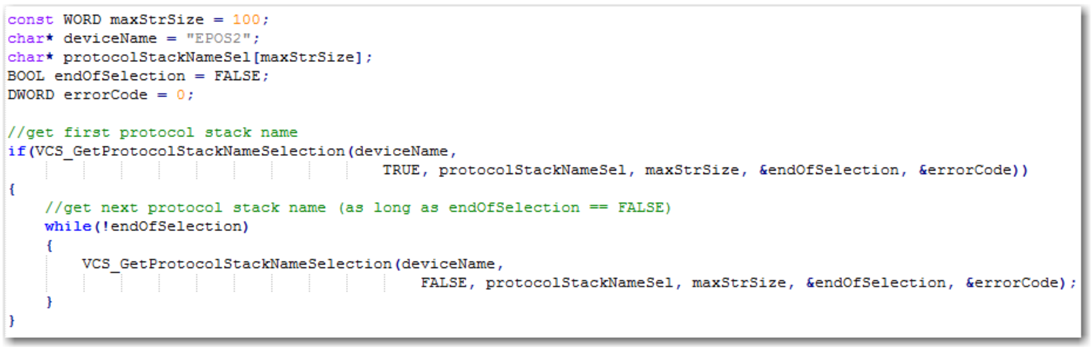
DeviceName	char*	Device name
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the name

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
const WORD maxStrSize = 100;
char* deviceName = "EPOS2";
char* protocolStackNameSel[maxStrSize];
BOOL endOfSelection = FALSE;
DWORD errorCode = 0;

//get first protocol stack name
if(VCS_GetProtocolStackNameSelection(deviceName,
                                      TRUE, protocolStackNameSel, maxStrSize, &endOfSelection, &errorCode))
{
    //get next protocol stack name (as long as endOfSelection == FALSE)
    while(!endOfSelection)
    {
        VCS_GetProtocolStackNameSelection(deviceName,
                                          FALSE, protocolStackNameSel, maxStrSize, &endOfSelection, &errorCode);
    }
}
```

Figure 3-9 GetProtocolStackNameSelection (programming example)

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.

Parameters

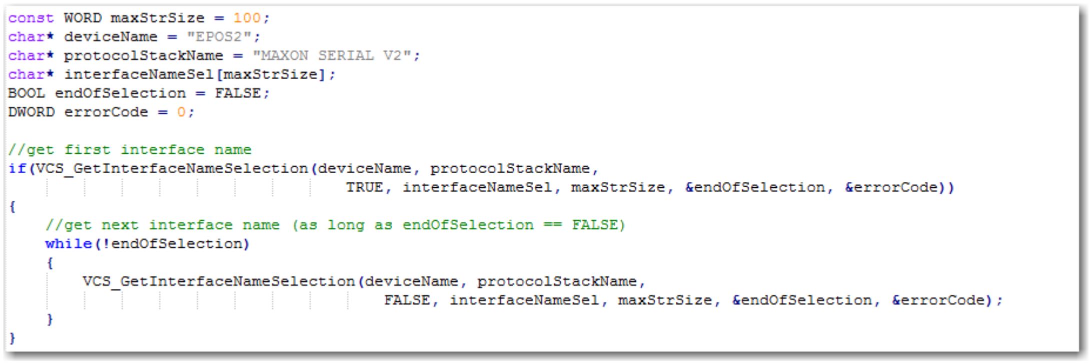
DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the interface name

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

Programming Example



```
const WORD maxStrSize = 100;
char* deviceName = "EPOS2";
char* protocolStackName = "MAXON SERIAL V2";
char* interfaceNameSel[maxStrSize];
BOOL endOfSelection = FALSE;
DWORD errorCode = 0;

//get first interface name
if(VCS_GetInterfaceNameSelection(deviceName, protocolStackName,
                                  TRUE, interfaceNameSel, maxStrSize, &endOfSelection, &errorCode))
{
    //get next interface name (as long as endOfSelection == FALSE)
    while(!endOfSelection)
    {
        VCS_GetInterfaceNameSelection(deviceName, protocolStackName,
                                  FALSE, interfaceNameSel, maxStrSize, &endOfSelection, &errorCode);
    }
}
```

Figure 3-10 GetInterfaceNameSelection (programming example)

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.

PARAMETERS

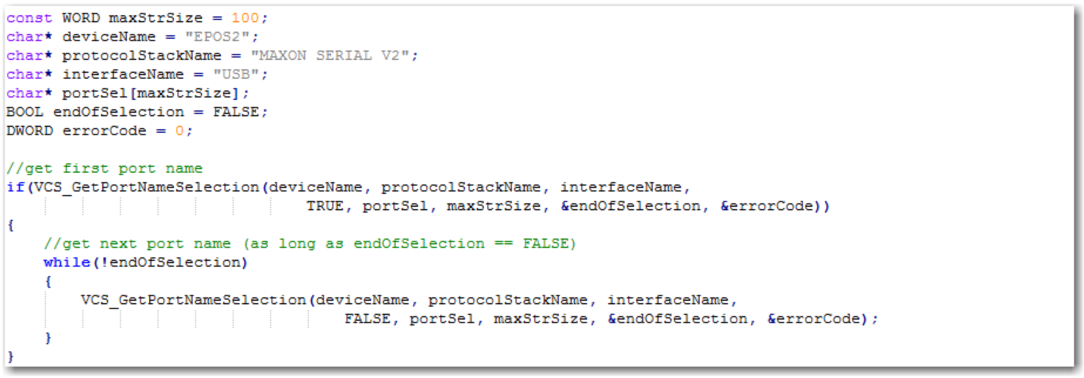
DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
StartOfSelection	BOOL	True: Get first selection string False: Get next selection string
MaxStrSize	WORD	Reserved memory size for the port name

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
const WORD maxStrSize = 100;
char* deviceName = "EPOS2";
char* protocolStackName = "MAXON SERIAL V2";
char* interfaceName = "USB";
char* portSel[maxStrSize];
BOOL endOfSelection = FALSE;
DWORD errorCode = 0;

//get first port name
if(VCS_GetPortNameSelection(deviceName, protocolStackName, interfaceName,
                            TRUE, portSel, maxStrSize, &endOfSelection, &errorCode))
{
    //get next port name (as long as endOfSelection == FALSE)
    while(!endOfSelection)
    {
        VCS_GetPortNameSelection(deviceName, protocolStackName, interfaceName,
                                FALSE, portSel, maxStrSize, &endOfSelection, &errorCode);
    }
}
```

Figure 3-11 GetPortNameSelection (programming example)

3.3.5 Reset Port Name Selection

FUNCTION

BOOL VCS_ResetPortNameSelection(char* DeviceName, char* ProtocolStackName, char* InterfaceName, DWORD* pErrorCode)

DESCRIPTION

VCS_ResetPortNameSelection reinitializes the port enumeration.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.6 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.

PARAMETERS

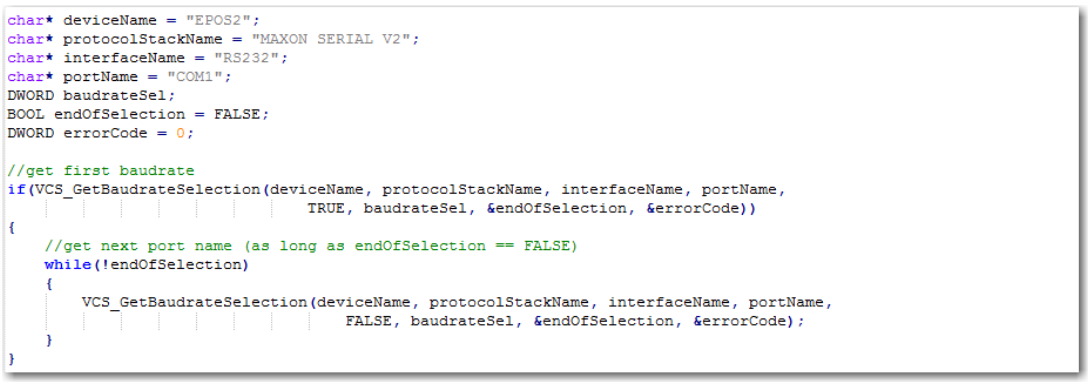
DeviceName	char*	Device name
ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
PortName	char*	Port name
StartOfSelection	BOOL	True: Get first selection value False: Get next selection value

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”

PROGRAMMING EXAMPLE



```
char* deviceName = "EPOS2";
char* protocolStackName = "MAXON SERIAL V2";
char* interfaceName = "RS232";
char* portName = "COM1";
DWORD baudrateSel;
BOOL endOfSelection = FALSE;
DWORD errorCode = 0;

//get first baudrate
if(VCS_GetBaudrateSelection(deviceName, protocolStackName, interfaceName, portName,
                            TRUE, baudrateSel, &endOfSelection, &errorCode))
{
    //get next port name (as long as endOfSelection == FALSE)
    while(!endOfSelection)
    {
        VCS_GetBaudrateSelection(deviceName, protocolStackName, interfaceName, portName,
                                 FALSE, baudrateSel, &endOfSelection, &errorCode);
    }
}
```

Figure 3-12 GetBaudrateSelection (programming example)

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

Return Value	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

Return Value	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”

••page intentionally left blank••

4 Configuration Functions

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

Availability of functions
The availability of certain functions depends on the used hardware. For an overview ➔“Appendix A — Hardware vs. Functions” on page 11-155.

4.1 General

4.1.1 Import Parameter

FUNCTION

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

DESCRIPTION

VCS_ImportParameter writes parameters from a file to the device. Not available with Linux.

PARAMETERS

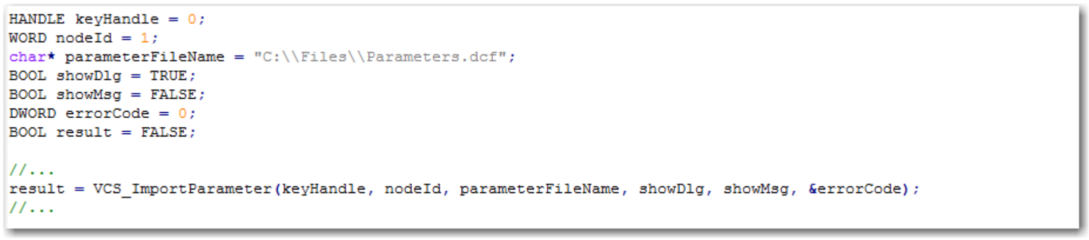
KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
pParameterFileName	char*	Full path of parameter file for import
ShowDlg	BOOL	Dialog is shown
ShowMsg	BOOL	Message box are activated

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
HANDLE keyHandle = 0;
WORD nodeId = 1;
char* parameterFileName = "C:\\Files\\Parameters.dcf";
BOOL showDlg = TRUE;
BOOL showMsg = FALSE;
DWORD errorCode = 0;
BOOL result = FALSE;

//...
result = VCS_ImportParameter(keyHandle, nodeId, parameterFileName, showDlg, showMsg, &errorCode);
//...
```

Figure 4-13 ImportParameter (programming example)

4.1.2 Export Parameter

FUNCTION

BOOL VCS_ExportParameter(HANDLE KeyHandle, WORD NodeId, 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

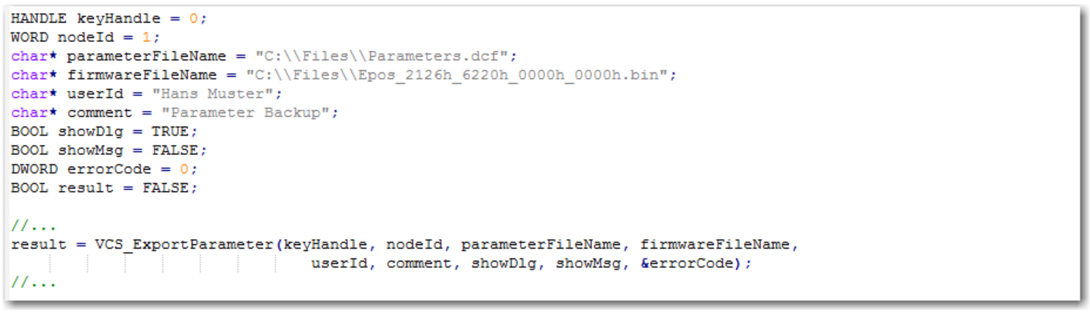
KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
pParameterFileName	char*	Full path of parameter file for export
pFirmwareFileName	char*	Full path of firmware file of connected device
pUserID	char*	User name
pComment	char*	Comment
ShowDlg	BOOL	Dialog is shown
ShowMsg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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
HANDLE keyHandle = 0;
WORD nodeId = 1;
char* parameterFileName = "C:\\Files\\Parameters.dcf";
char* firmwareFileName = "C:\\Files\\Epos_2126h_6220h_0000h_0000h.bin";
char* userId = "Hans Muster";
char* comment = "Parameter Backup";
BOOL showDlg = TRUE;
BOOL showMsg = FALSE;
DWORD errorCode = 0;
BOOL result = FALSE;

//...
result = VCS_ExportParameter(keyHandle, nodeId, parameterFileName, firmwareFileName,
                                userId, comment, showDlg, showMsg, &errorCode);
//...
```

Figure 4-14 ExportParameter (programming example)

4.1.3 Set Object

FUNCTION

BOOL VCS_SetObject(HANDLE KeyHandle, WORD NodeId, 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
NodeId	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 NodeId, 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
NodeId	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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_Restore restores all default parameters.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.1.6 Store

FUNCTION

BOOL VCS_Store(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_Store stores all parameters.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.1.7 Update Firmware

Function

BOOL VCS_UpdateFirmware (HANDLE KeyHandle, WORD NodeId, char *pBinaryFile, BOOL ShowDlg, BOOL ShowHistory, BOOL ShowMsg, DWORD *pErrorCode)

Description

VCS_UpdateFirmware is used to update the binary code for the controller firmware. Not available with Linux.

Parameters

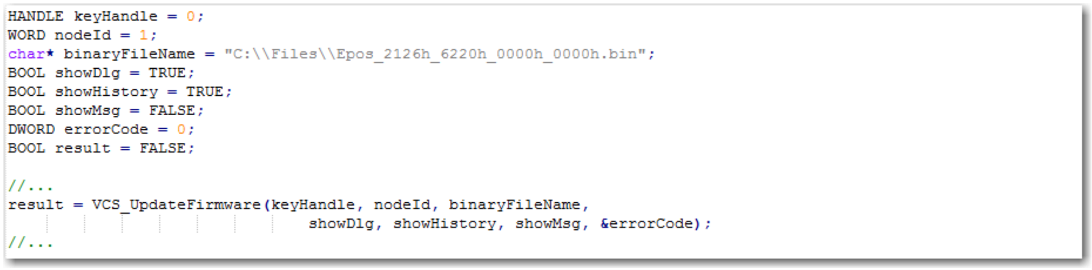
KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Identification ID of the addressed device
pBinaryFile	char*	Full path of firmware file
ShowDlg	BOOL	Progress dialog is shown
ShowHistory	BOOL	History list is shown in the progress dialog
ShowMsg	BOOL	Message boxes are shown during download (for example if an error occurs)

Return Parameters

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Programming Example



```
HANDLE keyHandle = 0;
WORD nodeId = 1;
char* binaryFileName = "C:\\Files\\Epos_2126h_6220h_0000h_0000h.bin";
BOOL showDlg = TRUE;
BOOL showHistory = TRUE;
BOOL showMsg = FALSE;
DWORD errorCode = 0;
BOOL result = FALSE;

//...
result = VCS_UpdateFirmware(keyHandle, nodeId, binaryFileName,
                                showDlg, showHistory, showMsg, &errorCode);
//...
```

Figure 4-15 UpdateFirmware (programming example)

4.2 Advanced Functions

4.2.1 Motor

4.2.1.1 Set Motor Type

FUNCTION

BOOL VCS_SetMotorType(HANDLE KeyHandle, WORD NodeId, WORD MotorType, DWORD* pErrorCode)

DESCRIPTION

VCS_SetMotorType writes the motor type.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
MotorType	WORD	Type of motor (➔Table 4-7)

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 NodeId, WORD NominalCurrent, WORD MaxOutputCurrent, WORD ThermalTimeConstant, DWORD* pErrorCode)

DESCRIPTION

VCS_SetDcMotorParameter writes all DC motor parameters.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
NominalCurrent	WORD	Maximal continuous current
MaxOutputCurrent	WORD	Maximal peak current
ThermalTimeConstant	WORD	Thermal time constant winding

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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
NominalCurrent	WORD	Maximal continuous current
MaxOutputCurrent	WORD	Maximal peak current
ThermalTimeConstant	WORD	Thermal time constant winding
NbOfPolePairs	BYTE	Number of pole pairs

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.1.4 Get Motor Type

FUNCTION

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

DESCRIPTION

VCS_GetMotorType reads the motor type.

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.1.5 Get DC Motor Parameter

FUNCTION

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

DESCRIPTION

VCS_GetDcMotorParameter reads all DC motor parameters.

PARAMETERS

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

RETURN PARAMETERS

pNominalCurrent	WORD*	Maximal continuous current
pMaxOutputCurrent	WORD*	Maximal peak current
pThermalTimeConstant	WORD*	Thermal time constant winding
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 NodeId, 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
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

pNominalCurrent	WORD*	Maximal continuous current
pMaxOutputCurrent	WORD*	Maximal peak current
pThermalTimeConstant	WORD*	Thermal time constant winding
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 NodeId, WORD SensorType, DWORD* pErrorCode)

DESCRIPTION

VCS_SetSensorType writes the sensor type.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
SensorType	WORD	Position Sensor Type (➔Table 4-8)

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Unknown / No sensor	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 NodeId, DWORD EncoderResolution, BOOL InvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCS_SetIncEncoderParameter writes the incremental encoder parameters.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
EncoderResolution	DWORD	Encoder pulse number [pulse per turn]
InvertedPolarity	BOOL	Position sensor polarity

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 NodeId, BOOL InvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCS_SetHallSensorParameter writes the Hall sensor parameter.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
InvertedPolarity	BOOL	Position sensor polarity

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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
DataRate	WORD	SSI encoder data rate
NbOfMultiTurnDataBits	WORD	Number of bits multi turn
NbOfSingleTurnDataBits	WORD	Number of bits single turn
InvertedPolarity	BOOL	Position sensor polarity

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 NodeId, WORD* pSensorType, DWORD* pErrorCode)

DESCRIPTION

VCS_GetSensorType reads the sensor type.

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.2.6 Get Incremental Encoder Parameter

FUNCTION

BOOL VCS_GetIncEncoderParameter(HANDLE KeyHandle, WORD NodeId, DWORD* pEncoderResolution, BOOL* pInvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCS_GetIncEncoderParameter reads the incremental encoder parameters.

PARAMETERS

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

RETURN PARAMETERS

pEncoderResolution	DWORD	Encoder pulse number [pulse per turn]
pInvertedPolarity	BOOL	Position sensor polarity
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.2.7 Get Hall Sensor Parameter

FUNCTION

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

DESCRIPTION

VCS_GetHallSensorParameter reads the Hall sensor parameters.

PARAMETERS

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

RETURN PARAMETERS

pInvertedPolarity	BOOL	Position sensor polarity
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 NodeId, 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
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

pDataRate	WORD*	SSI encoder data rate
pNbOfMultiTurnDataBits	WORD*	Number of bits multi turn
pNbOfSingleTurnDataBits	WORD*	Number of bits single turn
pInvertedPolarity	BOOL*	Position sensor polarity
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 NodeId, DWORD MaxFollowingError, DWORD* pErrorCode)

DESCRIPTION

VCS_SetMaxFollowingError writes the maximal allowed following error parameter.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
MaxFollowingError	DWORD	Maximal allowed difference of position actual value to position demand value

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 NodeId, DWORD* pMaxFollowingError, DWORD* pErrorCode)

DESCRIPTION

VCS_GetMaxFollowingError reads the maximal allowed following error parameter.

PARAMETERS

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

RETURN PARAMETERS

pMaxFollowingError	DWORD*	Maximal allowed difference of position actual value to position demand value
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 NodeId, DWORD MaxProfileVelocity, DWORD* pErrorCode)

DESCRIPTION

VCS_SetMaxProfileVelocity writes the maximal allowed velocity.

PARAMETERS

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

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 NodeId, DWORD* pMaxProfileVelocity, DWORD* pErrorCode)

DESCRIPTION

VCS_GetMaxProfileVelocity reads the maximal allowed velocity.

PARAMETERS

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

RETURN PARAMETERS

pMaxProfileVelocity	DWORD*	Used as velocity limit in a position (or velocity) move
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 NodeId, DWORD MaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetMaxAcceleration writes the maximal allowed acceleration/deceleration.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
MaxAcceleration	DWORD	Limiter of the other acceleration/ deceleration objects

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.3.6 Get Maximal Acceleration

FUNCTION

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

DESCRIPTION

VCS_GetMaxAcceleration reads the maximal allowed acceleration/deceleration.

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
P	WORD	Position regulator P-Gain
I	WORD	Position regulator I-Gain
D	WORD	Position regulator D-Gain

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 NodeId, WORD VelocityFeedForward, WORD AccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionRegulatorFeedForward writes parameters for position regulation with feed forward.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
VelocityFeedForward	WORD	Velocity feed forward factor
AccelerationFeedForward	WORD	Acceleration feed forward factor

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 NodeId, WORD* pP, WORD* pI, WORD* pD, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionRegulatorGain reads all position regulator gains.

PARAMETERS

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

RETURN PARAMETERS

pP	WORD	Position regulator P-Gain
pI	WORD	Position regulator I-Gain
pD	WORD	Position regulator D-Gain
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.4.4 Get Position Regulator Feed Forward

FUNCTION

BOOL VCS_GetPositionRegulatorFeedForward(HANDLE KeyHandle, WORD NodeId, WORD* pVelocityFeedForward, WORD* pAccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionRegulatorFeedForward reads parameters for position regulation feed forward.

PARAMETERS

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

RETURN PARAMETERS

pVelocityFeedForward	WORD*	Velocity feed forward factor
pAccelerationFeedFor-ward	WORD*	Acceleration feed forward factor
pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
P	WORD	Velocity regulator P-Gain
I	WORD	Velocity regulator I-Gain

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 NodeId, WORD VelocityFeedForward, WORD AccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCS_SetVelocityRegulatorFeedForward writes parameters for velocity regulation with feed forward.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
VelocityFeedForward	WORD	Velocity feed forward factor
AccelerationFeedForward	WORD	Acceleration feed forward factor

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 NodeId, WORD* pP, WORD* pI, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityRegulatorGain reads all velocity regulator gains.

PARAMETERS

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

RETURN PARAMETERS

pP	WORD*	Velocity regulator P-Gain
pI	WORD*	Velocity regulator I-Gain
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.5.4 Get Velocity Regulator Feed Forward

FUNCTION

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

DESCRIPTION

VCS_GetVelocityRegulatorFeedForward reads parameters for velocity regulation feed forward.

PARAMETERS

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

RETURN PARAMETERS

pVelocityFeedForward	WORD*	Velocity feed forward factor
pAccelerationFeedForward	WORD*	Acceleration feed forward factor
pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
P	WORD	Current regulator P-Gain
I	WORD	Current regulator I-Gain

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 NodeId, WORD* pP, WORD* pI, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentRegulatorGain enables reading all current regulator gains.

PARAMETERS

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

RETURN PARAMETERS

P	WORD*	Current regulator P-Gain
I	WORD*	Current regulator I-Gain
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)
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 (not supported for EPOS4)
Polarity	BOOL	1: Low active 0: High active
ExecutionMask	BOOL	1: Set the error routine Only for positive and negative switch

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
None	255	DIC_NO_FUNCTIONALITY
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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
DigitalOutputNb	WORD	Number of digital output (object subindex)
Configuration	WORD	Configures the functionality assigned to the digital output (bit number) (➔Table 4-10)
State	BOOL	State of digital output
Mask	BOOL	1: Functionality state will be set 0: not set (not supported for EPOS4)
Polarity	BOOL	1: Low active 0: High active

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
None	255	DOC_NO_FUNCTIONALITY
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 NodeId, WORD AnalogInputNb, WORD Configuration, BOOL Mask, ExecutionMask, DWORD* pErrorCode)

DESCRIPTION

VCS_AnalogInputConfiguration sets the configuration parameter for one analog input.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Analog current set point	0	AIC_ANALOG_CURRENT_SETPOINT
Analog position set point	2	AIC_ANALOG_POSITION_SETPOINT
Analog velocity set point	1	AIC_ANALOG_VELOCITY_SETPOINT
General purpose A	15	AIC_GENERAL_PURPOSE_A
General purpose B	14	AIC_GENERAL_PURPOSE_B

Table 4-11 Analog input configuration

4.2.8 Units

4.2.8.1 Set Velocity Units

FUNCTION

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

DESCRIPTION

VCS_SetVelocityUnits writes velocity unit parameters.

PARAMETERS

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

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^{-1})	−1	VN_DECI
Centi (10^{-2})	−2	VN_CENTI
Milli (10^{-3})	−3	VN_MILLI

Table 4-12 Velocity notation index

4.2.8.2 Get Velocity Units

FUNCTION

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

DESCRIPTION

VCS_GetVelocityUnits reads velocity unit parameters.

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

••page intentionally left blank••

5 Operation Functions

Availability of functions
The availability of certain functions depends on the used hardware. For an overview ➔“Appendix A — Hardware vs. Functions” on page 11-155.

5.1 Operation Mode

5.1.1 Set Operation Mode

FUNCTION

BOOL VCS_SetOperationMode(HANDLE KeyHandle, WORD NodeId, __int8 Mode, DWORD* pErrorCode)

DESCRIPTION

VCS_SetOperationMode sets the operation mode.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
Mode	__int8	Operation mode (➔Table 5-13)

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 NodeId, __int8* pMode, DWORD* pErrorCode)

DESCRIPTION

VCS_GetOperationMode returns the activated operation mode.

PARAMETERS

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

RETURN PARAMETERS

pMode	__int8*	Operation mode (➔Table 5-13)
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 NodeId, 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
NodeId	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 NodeId, WORD State, DWORD* pErrorCode)

DESCRIPTION

VCS_SetState reads the actual state machine state.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
State	WORD	Value of state machine (➔Table 5-14)

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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_SetEnableState changes the device state to “enable”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.4 Set Disable State

FUNCTION

BOOL VCS_SetDisableState(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_SetDisableState changes the device state to “disable”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.5 Set Quick Stop State

FUNCTION

BOOL VCS_SetQuickStopState(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_SetQuickStopState changes the device state to “quick stop”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.6 Clear Fault

FUNCTION

BOOL VCS_ClearFault(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ClearFault changes the device state from “fault” to “disable”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.7 Get State

FUNCTION

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

DESCRIPTION

VCS_GetState reads the new state of the state machine.

PARAMETERS

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

RETURN PARAMETERS

pState	WORD*	Statusword value (➔Table 5-14)
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.8 Get Enable State

FUNCTION

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

DESCRIPTION

VCS_GetEnableState checks if the device is enabled.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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 NodeId, BOOL* pIsDisabled, DWORD* pErrorCode)

DESCRIPTION

VCS_GetDisableState checks if the device is disabled.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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 NodeId, BOOL* pIsQuickStopped, DWORD* pErrorCode)

DESCRIPTION

VCS_GetQuickStopState returns the device state quick stop.

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.11 Get Fault State

FUNCTION

BOOL VCS_GetFaultState(HANDLE KeyHandle, WORD NodeId, BOOL* pIsInFault, DWORD* pErrorCode)

DESCRIPTION

VCS_GetFaultState returns the device state fault (pIsInFault = TRUE). Get error information if the device is in fault state (➔“Error Handling” on page 5-66).

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	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 NodeId, BYTE* pNbDeviceError, DWORD* pErrorCode)

DESCRIPTION

VCS_GetNbOfDeviceError returns the number of actual errors that are recorded.

PARAMETERS

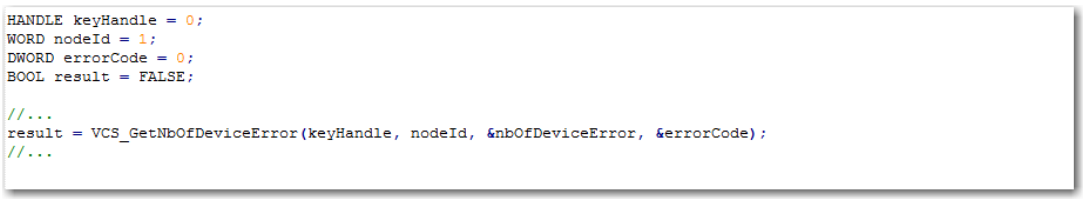
KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
HANDLE keyHandle = 0;
WORD nodeId = 1;
DWORD errorCode = 0;
BOOL result = FALSE;

//...
result = VCS_GetNbOfDeviceError(keyHandle, nodeId, &nbOfDeviceError, &errorCode);
//...
```

Figure 5-16 GetNbOfDeviceError (programming example)

5.3.2 Get Device Error Code

FUNCTION

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

DESCRIPTION

VCS_GetDeviceErrorCode returns the error code of the selected error number.

PARAMETERS

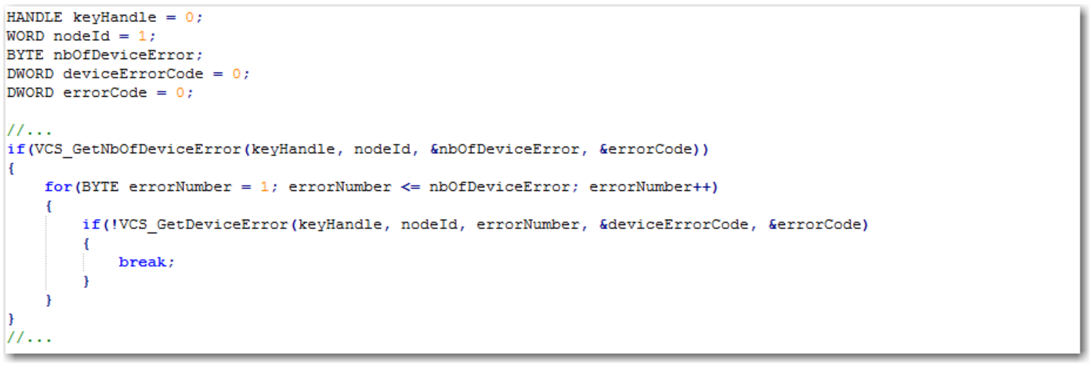
KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
ErrorNumber	BYTE	Number (object subindex) of device error (≥1)

RETURN PARAMETERS

pDeviceErrorCode	DWORD*	Actual error code from error history
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

PROGRAMMING EXAMPLE



```
HANDLE keyHandle = 0;
WORD nodeId = 1;
BYTE nbOfDeviceError;
DWORD deviceErrorCode = 0;
DWORD errorCode = 0;

//...
if(VCS_GetNbOfDeviceError(keyHandle, nodeId, &nbOfDeviceError, &errorCode))
{
    for(BYTE errorNumber = 1; errorNumber <= nbOfDeviceError; errorNumber++)
    {
        if(!VCS_GetDeviceError(keyHandle, nodeId, errorNumber, &deviceErrorCode, &errorCode)
        {
            break;
        }
    }
}
//...
```

Figure 5-17 GetDeviceErrorCode (programming example)

5.4 Motion Info

5.4.1 Get Movement State

FUNCTION

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

DESCRIPTION

VCS_GetMovementState checks if the drive has reached target.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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 statusword.
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.4.2 Get Position Is

FUNCTION

BOOL VCS_GetPositionIs(HANDLE KeyHandle, WORD NodeId, long* pPositionIs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionIs returns the position actual value.

PARAMETERS

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

RETURN PARAMETERS

pPositionIs	long*	Position actual value
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 NodeId, long* pVelocityIs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityIs reads the velocity actual value.

PARAMETERS

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

RETURN PARAMETERS

pVelocityIs	long*	Velocity actual value
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.4.4 Get Velocity Is Averaged

FUNCTION

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

DESCRIPTION

VCS_GetVelocityIsAveraged reads the velocity actual averaged value.

PARAMETERS

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

RETURN PARAMETERS

pVelocityIsAveraged	long*	Velocity actual value averaged
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.4.5 Get Current Is

FUNCTION

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

DESCRIPTION

VCS_GetCurrentIs returns the current actual value.

PARAMETERS

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

RETURN PARAMETERS

pCurrentIs	short*	Current actual value
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 NodeId, short* pCurrentIsAveraged, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentIsAveraged returns the current actual averaged value.

PARAMETERS

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

RETURN PARAMETERS

pCurrentIsAveraged	short*	Current actual value averaged
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 NodeId, 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
NodeId	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”

5.5 Profile Position Mode (PPM)

5.5.1 Activate Profile Position Mode

FUNCTION

BOOL VCS_ActivateProfilePositionMode(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateProfilePositionMode changes the operational mode to “profile position mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.5.2 Set Position Profile

FUNCTION

BOOL VCS_SetPositionProfile(HANDLE KeyHandle, WORD NodeId, DWORD ProfileVelocity, DWORD ProfileAcceleration, DWORD ProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionProfile sets the position profile parameters.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
ProfileVelocity	DWORD	Position profile velocity
ProfileAcceleration	DWORD	Position profile acceleration
ProfileDeceleration	DWORD	Position profile deceleration

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 NodeId, DWORD* pProfileVelocity, DWORD* pProfileAcceleration, DWORD* pProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionProfile returns the position profile parameters.

PARAMETERS

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

RETURN PARAMETERS

pProfileVelocity	DWORD*	Position profile velocity
pProfileAcceleration	DWORD*	Position profile acceleration
pProfileDeceleration	DWORD*	Position profile deceleration
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.5.4 Move To Position

FUNCTION

BOOL VCS_MoveToPosition(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
TargetPosition	long	Target position
Absolute	BOOL	TRUE starts an absolute FALSE a relative movement
Immediately	BOOL	TRUE starts immediately FALSE waits to end of last positioning

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.5.5 Get Target Position

FUNCTION

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

DESCRIPTION

VCS_GetTargetPosition returns the profile position mode target value.

PARAMETERS

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

RETURN PARAMETERS

pTargetPosition	long*	Target position
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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_HaltPositionMovement stops the movement with profile deceleration.

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

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 NodeId, DWORD PositionWindow, WORD PositionWindowTime, DWORD* pErrorCode)

DESCRIPTION

VCS_EnablePositionWindow activates the position window.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.5.7.2 Disable Position Window

FUNCTION

BOOL VCS_DisablePositionWindow(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DisablePositionWindow deactivates the position window.

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

Return Value	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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateProfileVelocityMode changes the operational mode to “profile velocity mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.6.2 Set Velocity Profile

FUNCTION

BOOL VCS_SetVelocityProfile(HANDLE KeyHandle, WORD NodeId, DWORD ProfileAcceleration, DWORD ProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetVelocityProfile sets the velocity profile parameters.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
ProfileAcceleration	DWORD	Velocity profile acceleration
ProfileDeceleration	DWORD	Velocity profile deceleration

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 NodeId, DWORD* pProfileAcceleration, DWORD* pProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityProfile returns the velocity profile parameters.

PARAMETERS

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

RETURN PARAMETERS

pProfileAcceleration	DWORD*	Velocity profile acceleration
pProfileDeceleration	DWORD*	Velocity profile deceleration
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.6.4 Move With Velocity

FUNCTION

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

DESCRIPTION

VCS_MoveWithVelocity starts the movement with velocity profile to target velocity.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.6.5 Get Target Velocity

FUNCTION

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

DESCRIPTION

VCS_GetTargetVelocity returns the profile velocity mode target value.

PARAMETERS

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

RETURN PARAMETERS

pTargetVelocity	long*	Target velocity
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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_HaltVelocityMovement stops the movement with profile deceleration.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.6.7 Advanced Functions

5.6.7.1 Enable Velocity Window

FUNCTION

BOOL VCS_Enable Velocity Window(HANDLE KeyHandle, WORD NodeId, DWORD VelocityWindow, WORD VelocityWindowTime, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableVelocityWindow activates the velocity window.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.6.7.2 Disable Velocity Window

FUNCTION

BOOL VCS_DisableVelocityWindow(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableVelocityWindow deactivates the velocity window.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7 Homing Mode (HM)

5.7.1 Activate Homing Mode

FUNCTION

BOOL VCS_ActivateHomingMode(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateHomingMode changes the operational mode to “homing mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7.2 Set Homing Parameter

FUNCTION

BOOL VCS_SetHomingParameter(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
HomingAcceleration	DWORD	Acceleration for homing profile
SpeedSwitch	DWORD	Speed during search for switch
SpeedIndex	DWORD	Speed during search for index signal
HomeOffset	long	Home offset after homing
CurrentThreshold	DWORD	Current threshold for homing methods −1, −2, −3, and −4
HomePosition	long	Used to assign the present position as homing position

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7.3 Get Homing Parameter

FUNCTION

BOOL VCS_GetHomingParameter(HANDLE KeyHandle, WORD NodeId, DWORD* pHomingAcceleration, DWORD* pSpeedSwitch, DWORD* pSpeedIndex, long* pHomeOffset, WORD* pCurrentThreshold, long* pHomePosition, 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
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

pHomingAcceleration	DWORD*	Acceleration for homing profile
pSpeedSwitch	DWORD*	Speed during search for switch
pSpeedIndex	DWORD*	Speed during search for index signal
pHomeOffset	long*	Home offset after homing
pCurrentThreshold	DWORD*	Current threshold for homing methods −1, −2, −3, and −4
pHomePosition	long*	Home position value
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7.4 Find Home

FUNCTION

BOOL VCS_FindHome(HANDLE KeyHandle, WORD NodeId, __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
NodeId	WORD	Node ID of the addressed device
HomingMethod	__int8	Homing method (➔Table 5-15)

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, DWORD* pErrorCode)

Description

VCS_StopHoming interrupts homing.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7.6 Define Position

FUNCTION

BOOL VCS_DefinePosition(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
HomePosition	long	Used to assign the present position as homing position

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7.7 Get Homing State

FUNCTION

BOOL VCS_GetHomingState(HANDLE KeyHandle, WORD NodeId, 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
NodeId	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.7.8 Wait For Homing Attained

FUNCTION

BOOL VCS_WaitForHomingAttained(HANDLE KeyHandle, WORD NodeId, 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
NodeId	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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateInterpolatedPositionMode changes the operational mode to “interpolated position mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.8.2 Set IPM Buffer Parameter

FUNCTION

BOOL VCS_SetIpmBufferParameter(HANDLE KeyHandle, WORD NodeId, WORD UnderflowWarningLimit, WORD OverflowWarningLimit, DWORD* pErrorCode)

DESCRIPTION

VCS_SetIpmBufferParameter sets warning borders of the data input.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
UnderflowWarningLimit	WORD	Gives lower signalization level of the data input FIFO
OverflowWarningLimit	WORD	Gives the higher signalization level of the data input FIFO

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.8.3 Get IPM Buffer Parameter

Function

BOOL VCS_GetIpmBufferParameter(HANDLE KeyHandle, WORD NodeId, WORD* pUnderflowWarningLimit, WORD* pOverflowWarningLimit, DWORD* pMaxBufferSize, DWORD* pErrorCode)

Description

VCS_GetIpmBufferParameter reads warning borders and the max. buffer size of the data input.

Parameters

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

Return Parameters

pUnderflowWarningLimit	WORD*	Gives lower signalization level of the data input FIFO
pOverflowWarningLimit	WORD*	Gives the higher signalization level of the data input FIFO
pMaxBufferSize	DWORD*	Provides the maximal buffer size
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 NodeId, 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
NodeId	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 NodeId, DWORD* pBufferSize, DWORD* pErrorCode)

DESCRIPTION

VCS_GetFreeIpmBufferSize reads the available buffer size.

PARAMETERS

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

RETURN PARAMETERS

pBufferSize	DWORD	Actual free buffer size
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.8.6 Add PVT Value To IPM Buffer

FUNCTION

BOOL VCS_AddPvtValueToIpmBuffer(HANDLE KeyHandle, WORD NodeId, long Position, long Velocity, BYTE Time, DWORD* pErrorCode)

DESCRIPTION

VCS_AddPvtValueToIpmBuffer adds a new PVT reference point to the device.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.8.7 Start IPM Trajectory

FUNCTION

BOOL VCS_StartIpmTrajectory(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_StartIpmTrajectory starts the IPM trajectory.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.8.8 Stop IPM Trajectory

FUNCTION

BOOL VCS_StopIpmTrajectory(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_StopIpmTrajectory stops the IPM trajectory.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.8.9 Get IPM Status

FUNCTION

BOOL VCS_GetIpmStatus(HANDLE KeyHandle, WORD NodeId, BOOL* pTrajectoryRunning, BOOL* pIsUnderflowWarning, BOOL* pIsOverflowWarning, BOOL* pIsVelocityWarning, BOOL* pIsAccelerationWarning, BOOL* pIsUnderflowError, BOOL* pIsOverflowError, BOOL* pIsVelocityError, BOOL* pIsAccelerationError, DWORD* pErrorCode)

DESCRIPTION

VCS_GetIpmStatus returns different warning and error states.

PARAMETERS

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

RETURN PARAMETERS

pTrajectoryRunning	BOOL*	State if IPM active
pIsUnderflowWarning	BOOL*	State if buffer underflow level is reached
pIsOverflowWarning	BOOL*	State if buffer overflow level is reached
pIsVelocityWarning	BOOL*	State if IPM velocity greater than profile velocity
pIsAccelerationWarning	BOOL*	State if IPM acceleration greater than profile acceleration
pIsUnderflowError	BOOL*	State of underflow error
pIsOverflowError	BOOL*	State of overflow error
pIsVelocityError	BOOL*	State if IPM velocity greater than max. profile velocity
pIsAccelerationError	BOOL*	State if IPM acceleration greater than max. profile acceleration
pErrorCode	DWORD*	Error information on the executed function

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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivatePositionMode changes the operational mode to “position mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.9.2 Set Position Must

FUNCTION

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

DESCRIPTION

VCS_SetPositionMust sets the position mode setting value.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
PositionMust	long	Position mode setting value

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.9.3 Get Position Must

FUNCTION

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

DESCRIPTION

VCS_GetPositionMust reads the position mode setting value.

PARAMETERS

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

RETURN PARAMETERS

pPositionMust	long*	Position mode setting value
pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input
Scaling	float	Scaling factor for analog position setpoint functionality
Offset	long	Offset for analog position setpoint functionality

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.9.4.2 Deactivate Analog Position Setpoint

FUNCTION

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

DESCRIPTION

VCS_DeactivateAnalogPositionSetpoint disables the selected analog input for analog position setpoint.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input

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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableAnalogPositionSetpoint enables the execution mask for analog position 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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.9.4.4 Disable Analog Position Setpoint

FUNCTION

BOOL VCS_DisableAnalogPositionSetpoint(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableAnalogPositionSetpoint disables the execution mask for analog position 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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.10 Velocity Mode (VM)

5.10.1 Activate Velocity Mode

FUNCTION

BOOL VCS_ActivateVelocityMode(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateVelocityMode changes the operational mode to “velocity mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.10.2 Set Velocity Must

FUNCTION

BOOL VCS_SetVelocityMust(HANDLE KeyHandle, WORD NodeId, long VelocityMust, DWORD* pErrorCode)

DESCRIPTION

VCS_SetVelocityMust sets the velocity mode setting value.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
VelocityMust	long	Velocity mode setting value

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 NodeId, long* pVelocityMust, DWORD* pErrorCode)

Description

VCS_GetVelocityMust returns the velocity mode setting value.

Parameters

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

Return Parameters

pVelocityMust	long*	Velocity mode setting value
pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input
Scaling	float	Scaling factor for analog velocity setpoint functionality
Offset	long	Offset for analog velocity setpoint functionality

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.10.4.2 Deactivate Analog Velocity Setpoint

FUNCTION

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

DESCRIPTION

VCS_DeactivateAnalogVelocitySetpoint disables the selected analog input for analog velocity setpoint.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input

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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableAnalogVelocitySetpoint enables 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

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.11 Current Mode (CM)

5.11.1 Activate Current Mode

FUNCTION

BOOL VCS_ActivateCurrentMode(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateCurrentMode changes the operational mode to "current mode".

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

Return Value	BOOL	Nonzero if successful; otherwise "0"

5.11.2 Get Current Must

FUNCTION

BOOL VCS_GetCurrentMust(HANDLE KeyHandle, WORD NodeId, short* pCurrentMust, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentMust reads the current mode setting value.

PARAMETERS

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

RETURN PARAMETERS

pCurrentMust	short*	Current mode setting value
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 NodeId, short CurrentMust, DWORD* pErrorCode)

DESCRIPTION

VCS_SetCurrentMust writes current mode setting value.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
CurrentMust	short	Current mode setting value

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input
Scaling	float	Scaling factor for analog current setpoint functionality
Offset	short	Offset for analog current setpoint functionality

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.11.4.2 Deactivate Analog Current Setpoint

FUNCTION

BOOL VCS_DeactivateAnalogCurrentSetpoint(HANDLE KeyHandle, WORD NodeId, WORD AnalogInputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivateAnalogCurrentSetpoint disables the selected analog input for analog current setpoint.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
AnalogInputNumber	WORD	Number of the used analog input

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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableAnalogCurrentSetpoint enables the execution mask for analog current 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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.11.4.4 Disable Analog Current Setpoint

FUNCTION

BOOL VCS_DisableAnalogCurrentSetpoint(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableAnalogCurrentSetpoint disables the execution mask for analog current 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

Return Value	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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateMasterEncoderMode changes the operational mode to “master encoder mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.12.2 Set Master Encoder Parameter

FUNCTION

BOOL VCS_SetMasterEncoderParameter(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
ScalingNumerator	WORD	Scaling numerator for position calculation
ScalingDenominator	WORD	Scaling denominator for position calculation
Polarity	BYTE	Polarity of the direction input. 0: Positive 1: Negative
MaxVelocity	DWORD	Maximal allowed speed during a profiled move
MaxAcceleration	DWORD	Defines the maximal allowed acceleration

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.12.3 Get Master Encoder Parameter

FUNCTION

BOOL VCS_GetMasterEncoderParameter(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

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

Return Value	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 NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateStepDirectionMode changes the operational mode to “step direction mode”.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.13.2 Set Step Direction Parameter

FUNCTION

BOOL VCS_SetStepDirectionParameter(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
ScalingNumerator	WORD	Scaling numerator for position calculation
ScalingDenominator	WORD	Scaling denominator for position calculation
Polarity	BYTE	Polarity of the direction input. 0: Positive 1: Negative
MaxVelocity	DWORD	Maximal allowed speed during a profiled move
MaxAcceleration	DWORD	Defines the maximal allowed acceleration

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 NodeId, 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
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

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

Return Value	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 NodeId, WORD* pInputs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetAllDigitalInputs returns state of all digital inputs.

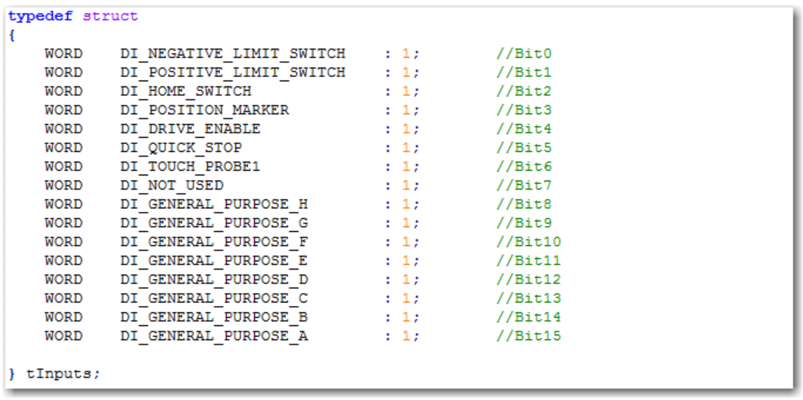
PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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”. ➔Figure 5-18 for “tInputs” structure
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”



```
typedef struct
{
    WORD    DI_NEGATIVE_LIMIT_SWITCH    : 1;        //Bit0
    WORD    DI_POSITIVE_LIMIT_SWITCH    : 1;        //Bit1
    WORD    DI_HOME_SWITCH              : 1;        //Bit2
    WORD    DI_POSITION_MARKER          : 1;        //Bit3
    WORD    DI_DRIVE_ENABLE             : 1;        //Bit4
    WORD    DI_QUICK_STOP               : 1;        //Bit5
    WORD    DI_TOUCH_PROBE1             : 1;        //Bit6
    WORD    DI_NOT_USED                 : 1;        //Bit7
    WORD    DI_GENERAL_PURPOSE_H        : 1;        //Bit8
    WORD    DI_GENERAL_PURPOSE_G        : 1;        //Bit9
    WORD    DI_GENERAL_PURPOSE_F        : 1;        //Bit10
    WORD    DI_GENERAL_PURPOSE_E        : 1;        //Bit11
    WORD    DI_GENERAL_PURPOSE_D        : 1;        //Bit12
    WORD    DI_GENERAL_PURPOSE_C        : 1;        //Bit13
    WORD    DI_GENERAL_PURPOSE_B        : 1;        //Bit14
    WORD    DI_GENERAL_PURPOSE_A        : 1;        //Bit15

} tInputs;
```

Figure 5-18 GetAllDigitalInputs (tInputs)

5.14.2 Get All Digital Outputs

FUNCTION

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

DESCRIPTION

VCS_GetAllDigitalOutputs returns state of all digital outputs.

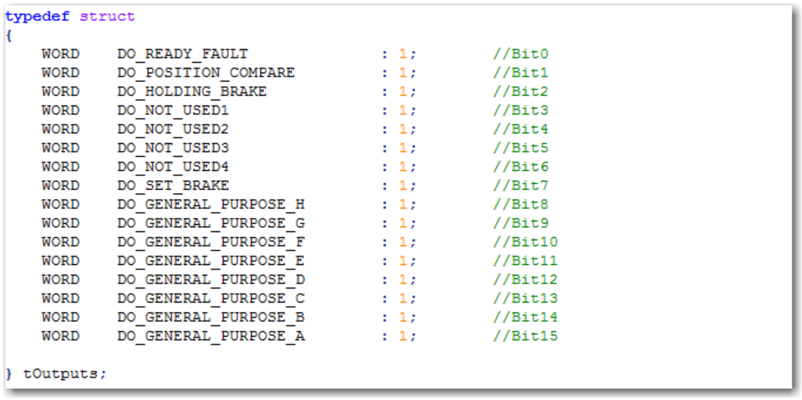
PARAMETERS

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

RETURN PARAMETERS

pOutputs	WORD*	State of all digital outputs. Activated if a bit is read as “1”. ➔Figure 5-19 for “tOutputs” structure
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”



```
typedef struct
{
    WORD    DO_READY_FAULT              : 1;        //Bit0
    WORD    DO_POSITION_COMPARE         : 1;        //Bit1
    WORD    DO_HOLDING_BRAKE            : 1;        //Bit2
    WORD    DO_NOT_USED1                : 1;        //Bit3
    WORD    DO_NOT_USED2                : 1;        //Bit4
    WORD    DO_NOT_USED3                : 1;        //Bit5
    WORD    DO_NOT_USED4                : 1;        //Bit6
    WORD    DO_SET_BRAKE                : 1;        //Bit7
    WORD    DO_GENERAL_PURPOSE_H        : 1;        //Bit8
    WORD    DO_GENERAL_PURPOSE_G        : 1;        //Bit9
    WORD    DO_GENERAL_PURPOSE_F        : 1;        //Bit10
    WORD    DO_GENERAL_PURPOSE_E        : 1;        //Bit11
    WORD    DO_GENERAL_PURPOSE_D        : 1;        //Bit12
    WORD    DO_GENERAL_PURPOSE_C        : 1;        //Bit13
    WORD    DO_GENERAL_PURPOSE_B        : 1;        //Bit14
    WORD    DO_GENERAL_PURPOSE_A        : 1;        //Bit15

} tOutputs;
```

Figure 5-19 GetAllDigitalOutputs (tOutputs)

5.14.3 Set All Digital Outputs

FUNCTION

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

DESCRIPTION

VCS_SetAllDigitalOutputs sets the state of all digital outputs.

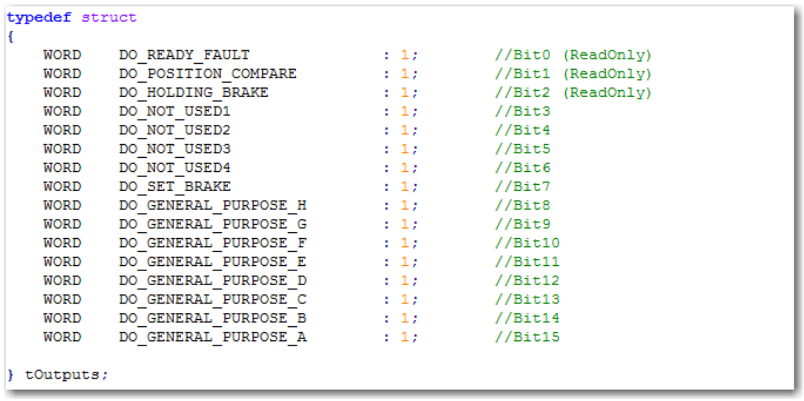
PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
Outputs	WORD	State of all digital outputs. Activated if a bit is written as “1”. ➔Figure 5-20 for “tOutputs” structure

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”



```
typedef struct
{
    WORD    DO_READY_FAULT              : 1;        //Bit0  (ReadOnly)
    WORD    DO_POSITION_COMPARE         : 1;        //Bit1  (ReadOnly)
    WORD    DO_HOLDING_BRAKE            : 1;        //Bit2  (ReadOnly)
    WORD    DO_NOT_USED1                : 1;        //Bit3
    WORD    DO_NOT_USED2                : 1;        //Bit4
    WORD    DO_NOT_USED3                : 1;        //Bit5
    WORD    DO_NOT_USED4                : 1;        //Bit6
    WORD    DO_SET_BRAKE                : 1;        //Bit7
    WORD    DO_GENERAL_PURPOSE_H        : 1;        //Bit8
    WORD    DO_GENERAL_PURPOSE_G        : 1;        //Bit9
    WORD    DO_GENERAL_PURPOSE_F        : 1;        //Bit10
    WORD    DO_GENERAL_PURPOSE_E        : 1;        //Bit11
    WORD    DO_GENERAL_PURPOSE_D        : 1;        //Bit12
    WORD    DO_GENERAL_PURPOSE_C        : 1;        //Bit13
    WORD    DO_GENERAL_PURPOSE_B        : 1;        //Bit14
    WORD    DO_GENERAL_PURPOSE_A        : 1;        //Bit15

} tOutputs;
```

Figure 5-20 SetAllDigitalOutputs (tOutputs)

5.14.4 Get Analog Input

FUNCTION

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

DESCRIPTION

VCS_GetAnalogInput returns the value from an analog input.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.5 Set Analog Output

FUNCTION

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

DESCRIPTION

VCS_SetAnalogOutput sets the voltage level of an analog output.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
OutputNumber	WORD	Analog output number
pAnalogValue	WORD*	Analog value for output

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 NodeId, 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
NodeId	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)
DirectionDependency	BYTE	Used direction dependency in position sequence mode (➔Table 5-18)
IntervalWidth	WORD	Holds the width of the position intervals
IntervalRepetitions	WORD	Allows to configure the number of position intervals to be considered by position compare
PulseWidth	WORD	Configures the pulse width of the trigger output

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

OPERATIONALMODE

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

INTERVALMODE

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

DIRECTIONDEPENDENCY

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 NodeId, 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
NodeId	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)
pDirectionDependency	BYTE*	Used direction dependency in position sequence mode (➔Table 5-18)
pIntervalWidth	WORD*	Holds the width of the position intervals
pIntervalRepetitions	WORD*	Allows to configure the number of position intervals to be considered by position compare
pPulseWidth	WORD*	Configures the pulse width of the trigger output
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 NodeId, WORD DigitalOutputNumber, BOOL Polarity, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivatePositionCompare enables the output to position compare method.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
DigitalOutputNumber	WORD	Selected digital output for position compare
Polarity	BOOL	Polarity of the selected output

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 NodeId, WORD DigitalOutputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivatePositionCompare disables the output to position compare method.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
DigitalOutputNumber	WORD	Selected digital output for position compare

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.6.5 Enable Position Compare

FUNCTION

BOOL VCS_EnablePositionCompare(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_EnablePositionCompare enables the output mask for position compare method.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.6.6 Disable Position Compare

FUNCTION

BOOL VCS_DisablePositionCompare(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DisablePositionCompare disables the output mask from position compare method.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.6.7 Set Position Compare Reference Position

FUNCTION

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

DESCRIPTION

VCS_SetPositionCompareReferencePosition writes the reference position for position compare method.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
ReferencePosition	long	Holds the position that is compared with the position actual value

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 NodeId, BYTE PositionMarkerEdgeType, BYTE PositionMarkerMode, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionMarkerParameter writes all parameters for position marker method.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
PositionMarkerEdgeType	BYTE	Defines the type of edge of the position to be captured (➔Table 5-19)
PositionMarkerMode	BYTE	Defines the position marker capturing mode (➔Table 5-20)

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

POSITIONMARKEREDGETYPE

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

POSITIONMARKERMODE

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 NodeId, BYTE* pPositionMarkerEdgeType, BYTE* pPositionMarkerMode, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionMarkerParameter reads all parameters for position marker method.

PARAMETERS

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

RETURN PARAMETERS

pPositionMarkerEdge Type	BYTE*	Defines the type of edge of the position to be captured (➔Table 5-19)
pPositionMarkerMode	BYTE*	Defines the position marker capturing mode (➔Table 5-20)
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 NodeId, WORD DigitalInputNumber, BOOL Polarity, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivatePositionMarker enables the digital input to position marker method.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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 NodeId, WORD DigitalInputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivatePositionMarker disables the digital input to position marker method.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
DigitalInputNumber	WORD	Selected digital input for position marker

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.7.5 Read Position Marker Counter

FUNCTION

BOOL VCS_ReadPositionMarkerCounter(HANDLE KeyHandle, WORD NodeId, WORD* pCount, DWORD* pErrorCode)

DESCRIPTION

VCS_ReadPositionMarkerCounter returns the number of the detected edges.

PARAMETERS

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

RETURN PARAMETERS

pCount	WORD*	Counts the number of detected edges
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.7.6 Read Position Marker Captured Position

FUNCTION

BOOL VCS_ReadPositionMarkerCapturedPosition(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
CounterIndex	WORD	0: Read position marker captured position
		1: Read position marker history
		2: Read position marker history

RETURN PARAMETERS

pCapturedPosition	long*	Contains the captured position or the position marker history
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 NodeId, 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
NodeId	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

Availability of functions
The availability of certain functions depends on the used hardware. For an overview ➔“Appendix A — Hardware vs. Functions” on page 11-155.

6.1 Operation Mode

6.1.1 Set Recorder Parameter

FUNCTION

BOOL VCS_SetRecorderParameter(HANDLE KeyHandle, WORD NodeId, WORD SamplingPeriod, WORD NbOfPrecedingSamples, WORD PulseWidth, DWORD* pErrorCode)

DESCRIPTION

VCS_SetRecorderParameter writes parameters for data recorder.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node ID of the addressed device
SamplingPeriod	WORD	Sampling Period as a multiple of the current regulator cycle (n-times 0.1 ms)
NbOfPrecedingSamples	WORD	Number of preceding samples (data history)

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.1.2 Get Recorder Parameter

FUNCTION

BOOL VCS_GetRecorderParameter(HANDLE KeyHandle, WORD NodeId, WORD* pSamplingPeriod, WORD* pNbOfPrecedingSamples, WORD PulseWidth, DWORD* pErrorCode)

DESCRIPTION

VCS_GetRecorderParameter reads parameters for data recorder.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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)
pNbOfPrecedingSamples	WORD*	Number of preceding samples (data history)
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.1.3 Enable Trigger

FUNCTION

BOOL VCS_EnableTrigger(HANDLE KeyHandle, WORD NodeId, BYTE TriggerType, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableTrigger connects the trigger(s) for data recording.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	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.

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_DisableAllTriggers(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableAllTriggers sets data recorder configuration (0x2011-00) for triggers to zero.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.1.5 Activate Channel

FUNCTION

BOOL VCS_ActivateChannel(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
ChannelNumber	BYTE	Channel number [1…4]
ObjectIndex	WORD	Object index for data recording
ObjectSubIndex	BYTE	Object subindex for data recording
ObjectSize	BYTE	Object size in bytes for data recording

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.1.6 Deactivate all Channels

FUNCTION

BOOL VCS_DeactivateAllChannel(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivateAllChannel zeros all data recording objects (0x2014, 0x2015, and 0x2016).

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2 Data Recorder Status

6.2.1 Start Recorder

FUNCTION

BOOL VCS_StartRecorder(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_StartRecorder starts data recording. Not available with Linux.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.2 Stop Recorder

FUNCTION

BOOL VCS_StopRecorder(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_StopRecorder stops data recording. Not available with Linux.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.3 Force Trigger

FUNCTION

BOOL VCS_ForceTrigger(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCS_ForceTrigger forces the data recording triggers. Not available with Linux.

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.4 Is Recorder Running

FUNCTION

BOOL VCS_IsRecorderRunning(HANDLE KeyHandle, WORD NodeId, BOOL* pRunning, DWORD* pErrorCode)

DESCRIPTION

VCS_IsRecorderRunning returns the data recorder status “running”. Not available with Linux.

PARAMETERS

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

RETURN PARAMETERS

pRunning	BOOL	1: Data recorder running 0: Data recorder stopped
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.5 Is Recorder Triggered

FUNCTION

BOOL VCS_IsRecorderTriggered(HANDLE KeyHandle, WORD NodeId, BOOL* pTriggered, DWORD* pErrorCode)

DESCRIPTION

VCS_IsRecorderTriggered returns data recorder status “triggered”. Not available with Linux.

PARAMETERS

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

RETURN PARAMETERS

pTriggered	BOOL	1: Data recorder triggered 0: Data recorder not triggered
pErrorCode	DWORD*	Error information on the executed function

Return Value	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 NodeId, 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
NodeId	WORD	Node ID of the addressed device

RETURN PARAMETERS

pVectorSize	DWORD*	Maximal number of samples per variable
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 NodeId, 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
NodeId	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
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 NodeId, 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
NodeId	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”

6.3.4 Export Channel Data to File

FUNCTION

BOOL VCS_ExportChannelDataToFile(HANDLE KeyHandle, WORD NodeId, 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
NodeId	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4 Advanced Functions

6.4.1 Read Data Buffer

FUNCTION

BOOL VCS_ReadDataBuffer(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
BufferSizeToRead	DWORD	Buffer size

RETURN PARAMETERS

pDataBuffer	BYTE*	Data points
pBufferSizeRead	DWORD*	Size of read data buffer
pVectorStartOffset	WORD*	Offset to the start of the recorded data vector within the ring buffer
pMaxNbOfSamples	WORD*	Maximal number of samples per variable
pNbOfRecordedSamples	WORD*	Number of recorded samples
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.2 Extract Channel Data Vector

FUNCTION

BOOL VCS_ExtractChannelDataVector(HANDLE KeyHandle, WORD NodeId, 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
NodeId	WORD	Node ID of the addressed device
ChannelNumber	BYTE	Selected channel
pDataBuffer	BYTE	Data points
BufferSize	DWORD	Buffer size
VectorSize	DWORD	Vector size
VectorStartOffset	WORD	Offset to the start of the recorded data vector within the ring buffer
MaxNbOfSamples	WORD	Maximal number of samples per variable
NbOfRecordedSamples	WORD	Number of recorded samples

RETURN PARAMETERS

pDataVector	BYTE*	Data points of the channel
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7 Low Layer Functions

Availability of functions
The availability of certain functions depends on the used hardware. For an overview ➔"Appendix A — Hardware vs. Functions" on page 11-155.

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

Return Value	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

Return Value	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

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.4 Send NMT Frame

FUNCTION

BOOL VCS_SendNMTService(HANDLE KeyHandle, WORD NodeId, 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

Return Value	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 error	Communication was successful
0x0503 0000	Toggle error	Toggle bit not alternated
0x0504 0000	SDO timeout	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	CRC error	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	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 internal Incompatibility error	General internal incompatibility in device
0x0606 0000	Hardware error	Access failed due to a hardware error
0x0607 0010	Service parameter error	Data type does not match, length or service parameter does not match
0x0607 0012	Service parameter too high	Data type does not match, length or service parameter too high
0x0607 0013	Service Parameter too low	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	Value of parameter written too high
0x0609 0032	Value too low	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	CAN ID error	Wrong CAN ID
0x0F00 FFBC	Service mode error	Device is not in service mode
0x0F00 FFBE	Password error	Password is wrong
0x0F00 FFBF	Illegal command	RS232 command is illegal (does not exist)
0x0F00 FFC0	Wrong NMT state	Device is in wrong NMT state

Table 8-23 Communication errors

8.2 Library Errors

8.2.1 General Errors

Abort Code	Name	Error Cause
0x0000 0000	No error	Communication 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 protocol stack 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	Command failed	Error while executing command
0x1000 000B	Timeout	Timeout occurred during execution
0x1000 000C	Bad parameter	Bad parameter passed to function
0x1000 000D	Command aborted by user	Command was 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 is not supported
0x1000 0011	Parameter already used	Parameter is already in use
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	Error while opening interface
0x2000 0002	Closing Interface error	Error while closing interface
0x2000 0003	Interface is not open	Interface is not open
0x2000 0004	Opening port error	Error while opening port
0x2000 0005	Closing port error	Error while closing port
0x2000 0006	Port is not open	Port is not open
0x2000 0007	Resetting port error	Error while resetting port
0x2000 0008	Configuring port settings error	Error while configuring port settings
0x2000 0009	Configuring port mode error	Error while configuring port mode

Table 8-25 Interface layer errors

8.2.2.1 Interface Layer “RS232” Errors

Abort Code	Name	Error Cause
0x2100 0001	RS232 write data error	Error while writing RS232 data
0x2100 0002	RS232 read data error	Error while reading RS232 data

Table 8-26 Interface layer “RS232” errors

8.2.2.2 Interface Layer “CAN” Errors

Abort Code	Name	Error Cause
0x2200 0001	CAN receive frame error	Error while receiving CAN frame
0x2200 0002	CAN transmit frame error	Error while transmitting CAN frame

Table 8-27 Interface layer “CAN” errors

8.2.2.3 Interface Layer “USB” Errors

Abort Code	Name	Error Cause
0x2300 0001	USB write data error	Error while writing data
0x2300 0002	USB read data error	Error while reading data

Table 8-28 Interface layer “USB” errors

8.2.2.4 Interface Layer “HID” Errors

Abort Code	Name	Error Cause
0x2400 0001	HID write data error	Error while writing USB data to HID device
0x2400 0002	HID read data error	Error while reading USB data from HID device

Table 8-29 Interface layer “HID” errors

8.2.3 Protocol Layer Errors

8.2.3.1 Protocol Layer “MAXON_RS232” Errors

Abort Code	Name	Error Cause
0x3100 0001	Negative acknowledge received	Negative acknowledge received
0x3100 0002	Bad CRC received	Bad checksum received
0x3100 0003	Bad data received	Bad data size received

Table 8-30 Protocol layer “MAXON_RS232” errors

8.2.3.2 Protocol Layer “CANopen” Errors

Abort Code	Name	Error Cause
0x3200 0001	SDO response not received	CAN frame of SDO protocol not received
0x3200 0002	Requested CAN frame not received	Requested CAN frame not received
0x3200 0003	CAN frame not received	CAN frame not received

Table 8-31 Protocol layer “CANopen” errors

8.2.3.3 Protocol Layer “Maxon Serial V2” Errors

Abort Code	Name	Error Cause
0x3400 0001	Stuffing error	Failure while stuffing data
0x3400 0002	Destuffing error	Failure while destuffing data
0x3400 0003	Bad CRC received	Bad CRC received
0x3400 0004	Bad data size received	Bad data size received
0x3400 0005	Bad data size written	Bad data size written
0x3400 0006	Serial data frame not written	Failure occurred while writing data
0x3400 0007	Serial data frame not received	Failure occurred while reading data

Table 8-32 Protocol layer “Maxon Serial V2” errors

8.2.3.4 Device Layer Errors

Abort Code	Name	Error Cause
0x5100 0001	Bad data size received	Object data size does not correspond to requested data size
0x5100 0007	Sensor configuration not supported	Sensor configuration cannot be written to controller
0x5100 0008	Sensor configuration unknown	Sensor configuration read from controller is not supported by library

Table 8-33 Device layer 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 9-135) and Linux (➔as of page 9-147) 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

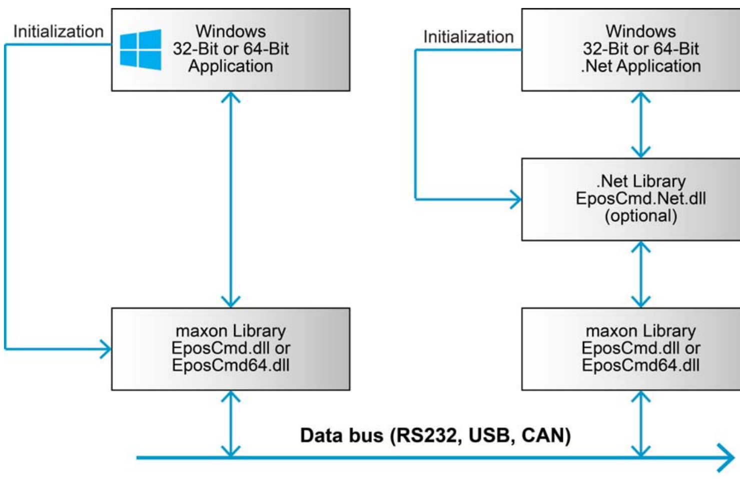

Initialization
Windows 32-Bit or 64-Bit Application
maxon Library EposCmd.dll or EposCmd64.dll
Initialization
Windows 32-Bit or 64-Bit .Net Application
.Net Library EposCmd.Net.dll (optional)
maxon Library EposCmd.dll or EposCmd64.dll
Data bus (RS232, USB, CAN)

Figure 9-21 Windows – Library hierarchy

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 9-144.

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¤.

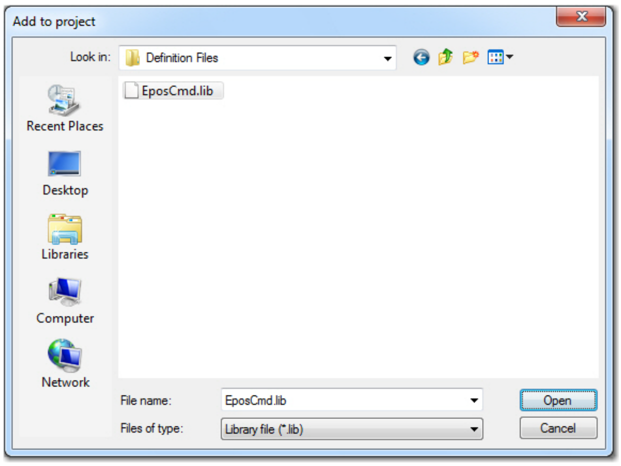


Figure 9-22 Borland C++Builder – Adding library

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

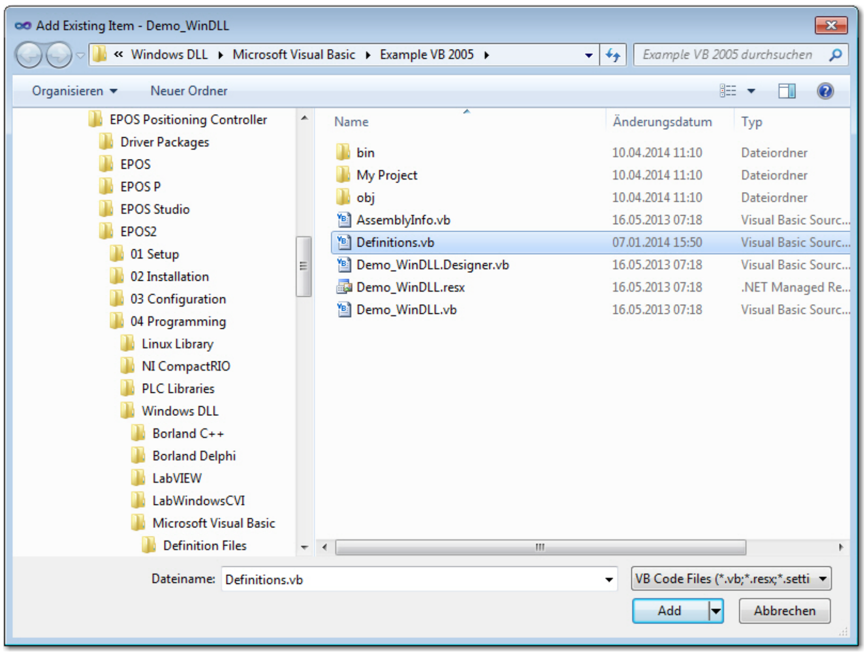
- **Definitions.vb** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library

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¤.


Add Existing Item - Demo_WinDLL

« Windows DLL ▸ Microsoft Visual Basic ▸ Example VB 2005 ▸

Example VB 2005 durchsuchen

Organisieren ▾ Neuer Ordner

- EPOS Positioning Controller
  - Driver Packages
  - EPOS
  - EPOS P
  - EPOS Studio
  - EPOS2
    - 01 Setup
    - 02 Installation
    - 03 Configuration
    - 04 Programming
      - Linux Library
      - NI CompactRIO
      - PLC Libraries
      - Windows DLL
        - Borland C++
        - Borland Delphi
        - LabVIEW
        - LabWindowsCVI
        - Microsoft Visual Basic
          - Definition Files

Name	Änderungsdatum	Typ
bin	10.04.2014 11:10	Dateiordner
My Project	10.04.2014 11:10	Dateiordner
obj	10.04.2014 11:10	Dateiordner
AssemblyInfo.vb	16.05.2013 07:18	Visual Basic Sourc...
Definitions.vb	07.01.2014 15:50	Visual Basic Sourc...
Demo_WinDLL.Designer.vb	16.05.2013 07:18	Visual Basic Sourc...
Demo_WinDLL.resx	16.05.2013 07:18	.NET Managed Re...
Demo_WinDLL.vb	16.05.2013 07:18	Visual Basic Sourc...

Dateiname: Definitions.vb

VB Code Files (*.vb;*.resx;*.setti

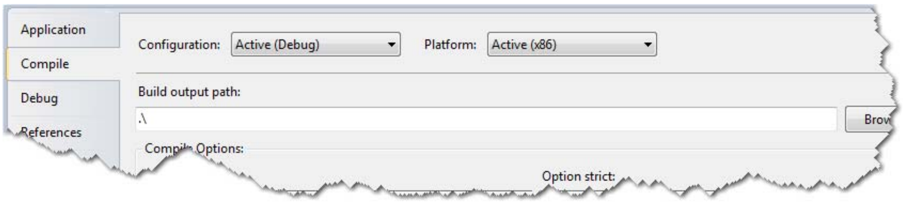
Add Abbrechen

Figure 9-23 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.


Application
Compile
Debug
References

Configuration: Active (Debug) Platform: Active (x86)

Build output path:
.\

Compile Options:

Option strict:

Figure 9-24 Visual Basic – Output path

4) Now, you can execute all library functions in your own code.

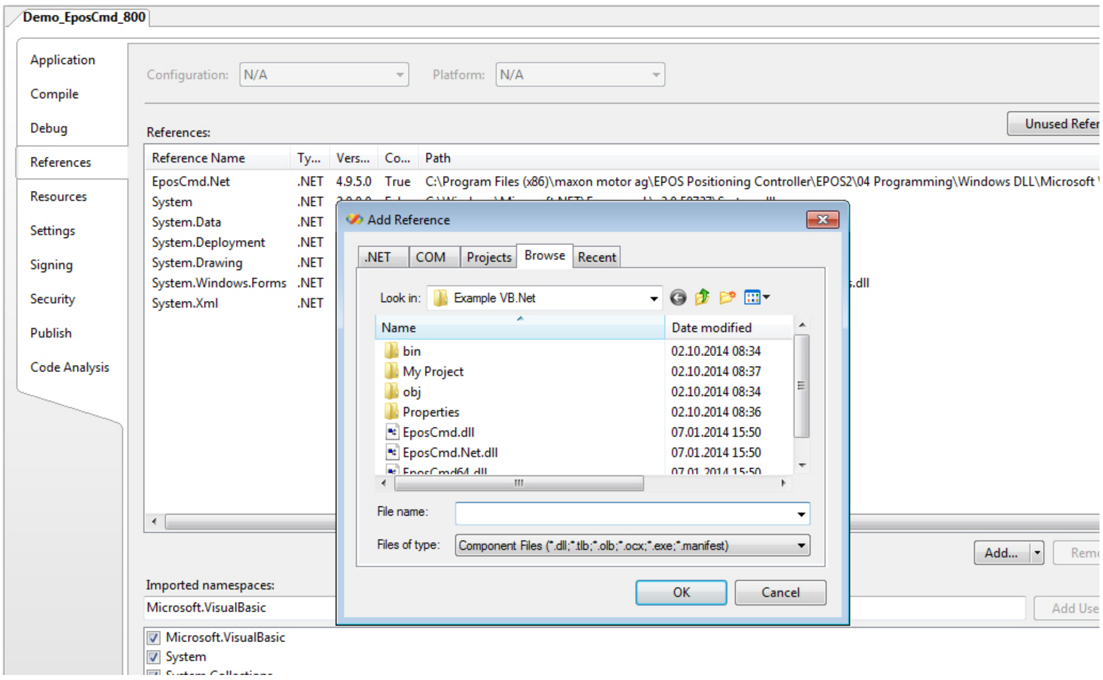
9.1.2.4 Microsoft Visual Basic .NET

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¤.



Configuration: N/A Platform: N/A

References:

Reference Name	Ty...	Vers...	Co...	Path
EposCmd.Net	.NET	4.9.5.0	True	C:\Program Files (x86)\maxon motor ag\EPOS Positioning Controller\EPOS2\04 Programming\Windows DLL\Microsoft
System	.NET			
System.Data	.NET			
System.Deployment	.NET			
System.Drawing	.NET			
System.Windows.Forms	.NET			
System.Xml	.NET			

Look in: Example VB.Net

Name	Date modified
bin	02.10.2014 08:34
My Project	02.10.2014 08:37
obj	02.10.2014 08:34
Properties	02.10.2014 08:36
EposCmd.dll	07.01.2014 15:50
EposCmd.Net.dll	07.01.2014 15:50
EposCmd64.dll	07.01.2014 15:50

File name:

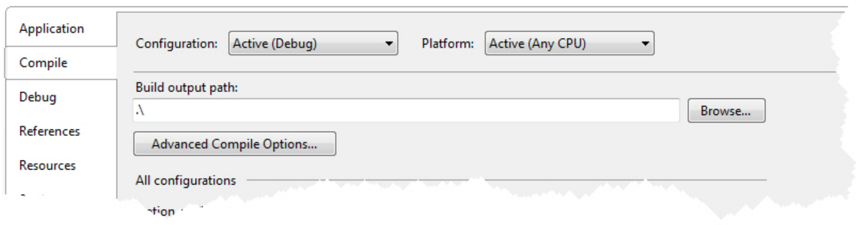
Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

Imported namespaces:
Microsoft.VisualBasic

☑ Microsoft.VisualBasic
☑ System

Figure 9-25 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.



Configuration: Active (Debug) Platform: Active (Any CPU)

Build output path:
.\

Figure 9-26 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».

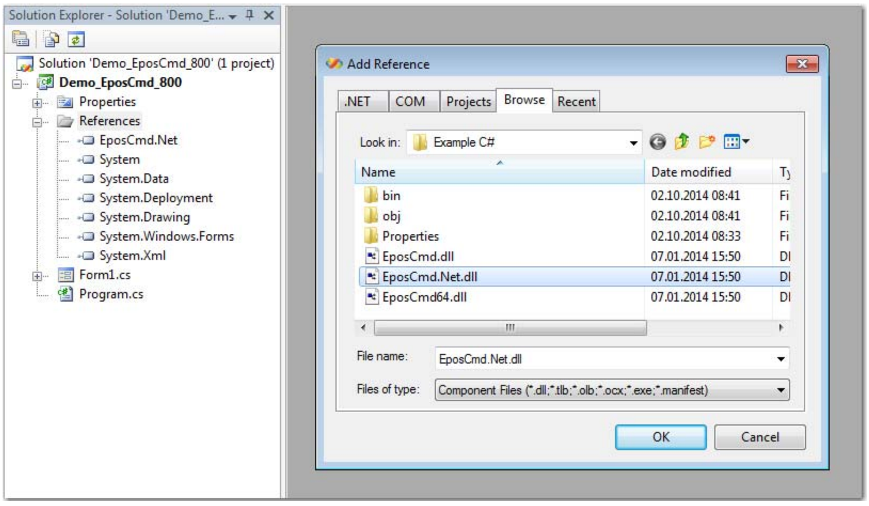
9.1.2.5 Microsoft Visual C#

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¤.



```
Solution 'Demo_EposCmd_800' (1 project)
  Demo_EposCmd_800
    Properties
    References
      EposCmd.Net
      System
      System.Data
      System.Deployment
      System.Drawing
      System.Windows.Forms
      System.Xml
    Form1.cs
    Program.cs
```

```
Look in:  Example C#
Name                Date modified       Ty
bin                 02.10.2014 08:41    Fi
obj                 02.10.2014 08:41    Fi
Properties          02.10.2014 08:33    Fi
EposCmd.dll         07.01.2014 15:50    Dl
EposCmd.Net.dll     07.01.2014 15:50    Dl
EposCmd64.dll       07.01.2014 15:50    Dl

File name:     EposCmd.Net.dll
Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)
```

Figure 9-27 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

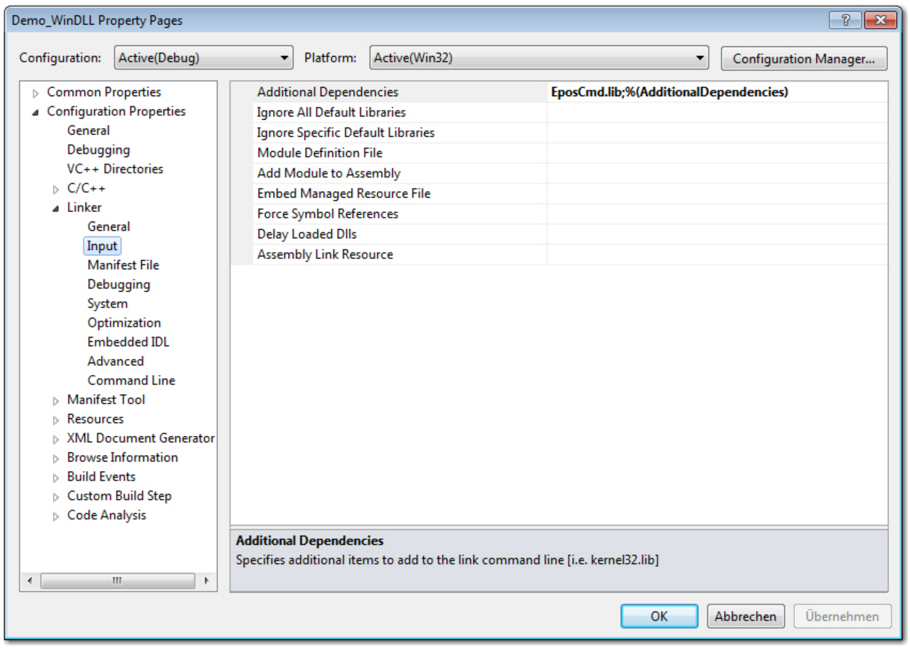
- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd.dll** – Dynamic link library
- **EposCmd.lib** – Import library (COFF format)

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.


Demo_WinDLL Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

Additional Dependencies	EposCmd.lib;%(AdditionalDependencies)
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line [i.e. kernel32.lib]

Figure 9-28 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¤.

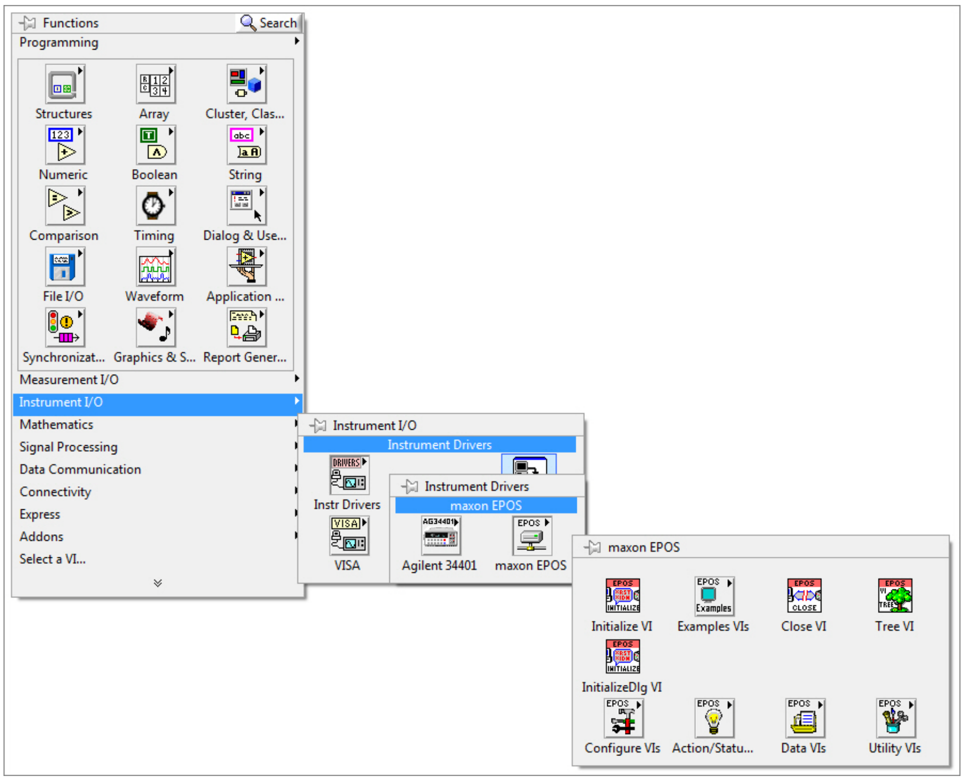


Figure 9-29 LabVIEW – maxon EPOS instrument driver

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¤.

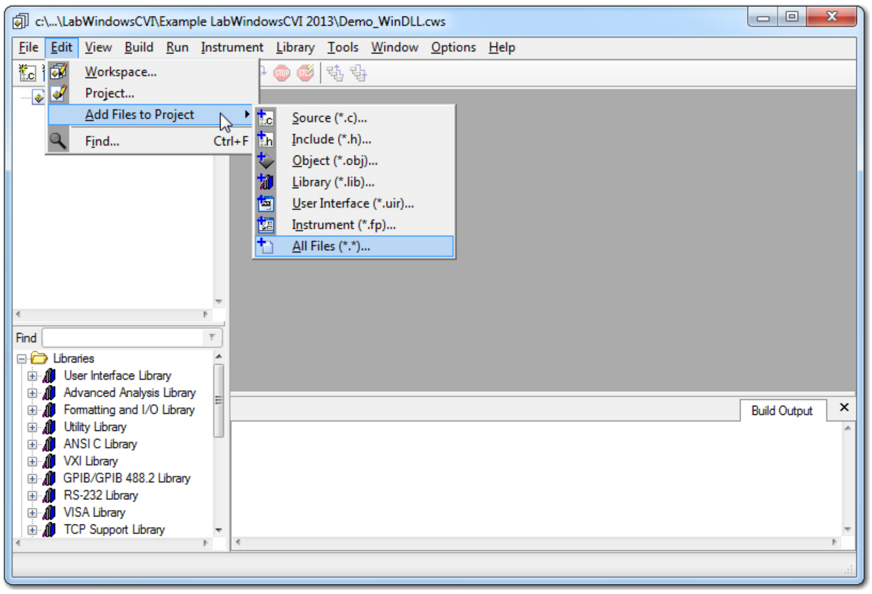


Figure 9-30 LabWindows – add files to project

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 ¤Device Settings¤ 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

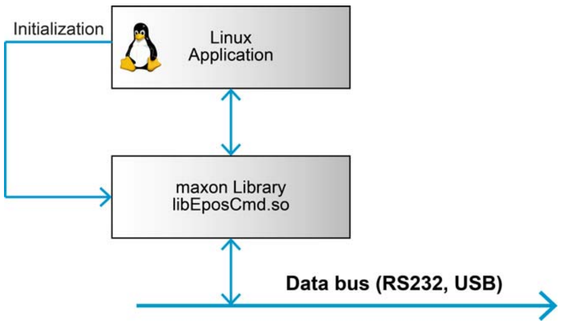


Figure 9-31 Linux – Library hierarchy

9.2.2 Framework Conditions

OPERATING SYSTEMS

The library has been tested with the following operating systems:

- x86/x64
  – Ubuntu 12.04, 14.04, 16.04
- ARMv7/8
  – Raspbian (Debian Jessie)*1)

**1) ARMv7 tested with Raspberry Pi2/3, Debian Jessie, armhf*

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 Eclipe with CDT plugin (➔http://eclipse.org/)
- gcc/g++ GNU compiler and libraries (➔http://gcc.gnu.org/)

DEVELOPMENT BOARDS (ARM)

- Raspberry Pi 2/3

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

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, armv7hf).
   a) x86: 32-bit operating system for Intel/AMD processors
   b) x86_64: 64-bit operating system for Intel/AMD processors
   c) armv7hf: 32-bit operating system (hard float) for ARMv7/8 processors
3) Take note that the following actions will require root privileges.
4) Copy the shared library file “libEposCmd.so.6.0.1.0” to the directory “/usr/local/lib”:
   a) sudo cp libEposCmd.so.6.0.1.0
   b) Create a soft link to this library. Execute the following commands:
      $ cd /usr/local/lib
      $ sudo ln -s libEposCmd.so.6.0.1.0 libEposCmd.so
   c) Create a soft link to this library into directory “/usr/lib” (or any directory listed in the LD_LIBRARY_PATH system variable).
      Execute the following commands from directory “/usr/lib”:
      $ cd /usr/lib
      $ sudo ln -s /usr/local/lib/libEposCmd.so libEposCmd.so
   d) Alternatively, if you do not plan to use Linux versioning/symlinks system, this single command does the job, too:
      $ sudo cp libEposCmd.so.6.0.1.0 /usr/lib/libEposCmd.so
5) Optional: Copy the file “Defintions.h” to the directory “/usr/include”.

9.2.4.2 EPOS2 FTDI Library

Best Practice
The «EPOS Command Library» for EPOS2 features built-in USB support. Additional configuration of FTDI device access rights may be required on your system.

1) To allow the non-root user to work with EPOS2 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 EPOS file “99-ftdi.rules” to the directory “/etc/udev/rules”.
2) Restart the udev service to adopt the changes:
   $ sudo service udev restart
   or, alternatively,
   $ sudo /etc/init.d/udev restart

9.2.4.3 EPOS4 HID driver

Best Practice
The «EPOS Command Library» features built-in HID support. Additional configuration of HID access rights may be required on your system.

1) To allow the non-root user to work with EPOS4 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 EPOS file "99-epos4.rules" to the directory "/etc/udev/rules".
2) Restart the udev service to adopt the changes:
   $ sudo service udev restart
   or, alternatively,
   $ sudo /etc/init.d/udev restart

9.2.5 HelloEposCmd

The Terminal C++ example project «HelloEposCmd» is delivered as source code. You may either…

A use the "**make**" program (see prerequisite below),

B compile the program with **g++** (see prerequisite below), or

C use the **Eclipse** project.

Notes
Prerequisite: libEposCmd.so located in LD_LIBRARY_PATH (i.e. /usr/lib)
The main advantage of the Eclipse project is the possibility to select different build configurations for the target platform (Debug/Release).

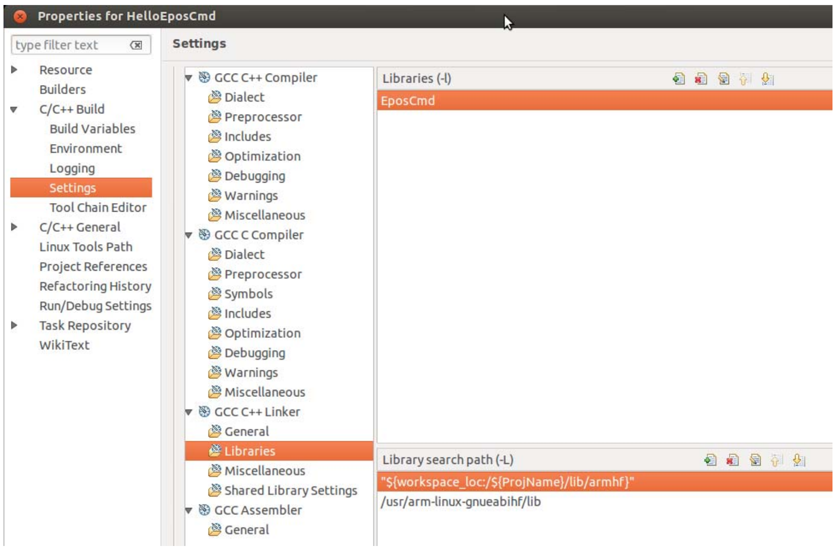
1) Compile the project using on of the following methods:

   A **make**
   Switch to the directory "HelloEposCmd" and execute the make* command without parameters:
   $ make

   B **g++**
   Switch to the directory "HelloEposCmd" and use the g++* command with parameters:
   $ g++ HelloEposCmd.cpp -l EposCmd -o HelloEposCmd

   C **Eclipse**
   1) Find the Eclipse project «HelloEposCmd» in the "examples" directory.
   2) Import the project into your workspace.
   3) Optional: Include the EposCmd library for C++ linker (-L option):
      EposCmd

Continued on next page.

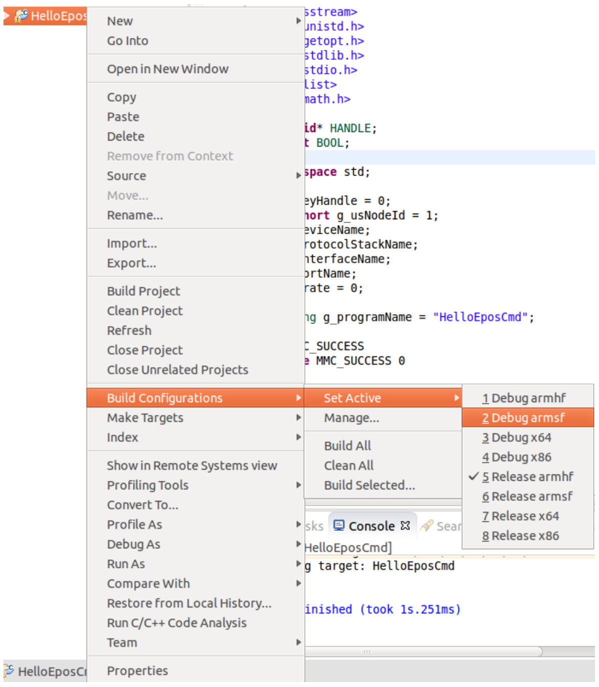


Libraries (-l)
EposCmd

Library search path (-L)
"${workspace_loc:/${ProjName}/lib/armhf}"
/usr/arm-linux-gnueabihf/lib

Figure 9-32 Eclipse – Libraries configuration

2) Set the target build configuration and start the build. Thereby, make use of the preconfigured debug and release build configurations for x86, x64, or armv7hf.

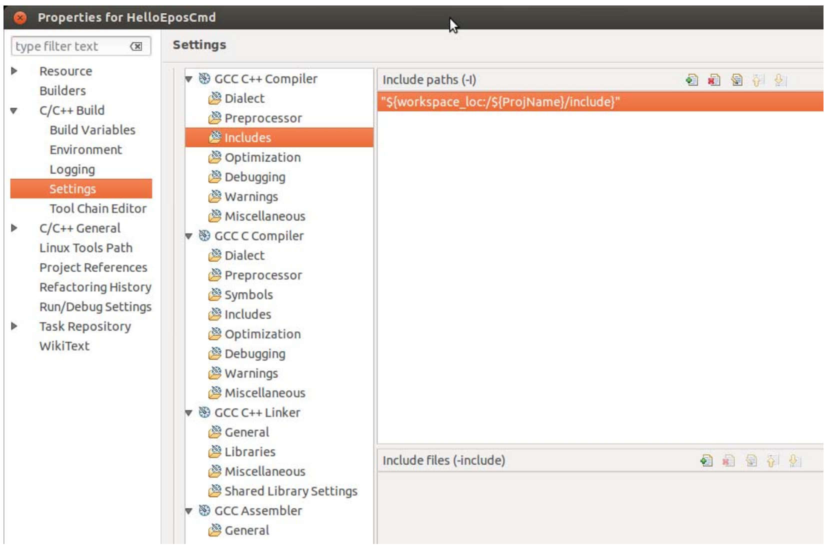


```
sstream>
unistd.h>
getopt.h>
stdlib.h>
stdio.h>
list>
math.h>

id* HANDLE;
t BOOL;

space std;

eyHandle = 0;
hort g_usNodeId = 1;
eviceName;
rotocolStackName;
nterfaceName;
ortName;
rate = 0;

ng g_programName = "HelloEposCmd";

_SUCCESS
e MMC_SUCCESS 0
```

```
HelloEposCmd]
g target: HelloEposCmd

inished (took 1s.251ms)
```

Figure 9-33 Eclipse – Build configuration

3) Optional: Include the file “Definition.h”.



Include paths (-I)
"${workspace_loc:/${ProjName}/include}"

Figure 9-34 Eclipse – Includes configuration

4) Set HelloEposCmd program options as to the following scheme:
HelloEposCmd [-h] [-d] device [-s] protocol stack [-i] interface [-p] port [-b] baudrate [-n] node id

h	usage information
b	baudrate; for example “115200” for RS232 or “1000000” for USB (default)
d	device name; for example “EPOS4” (default) or “EPOS2”
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”

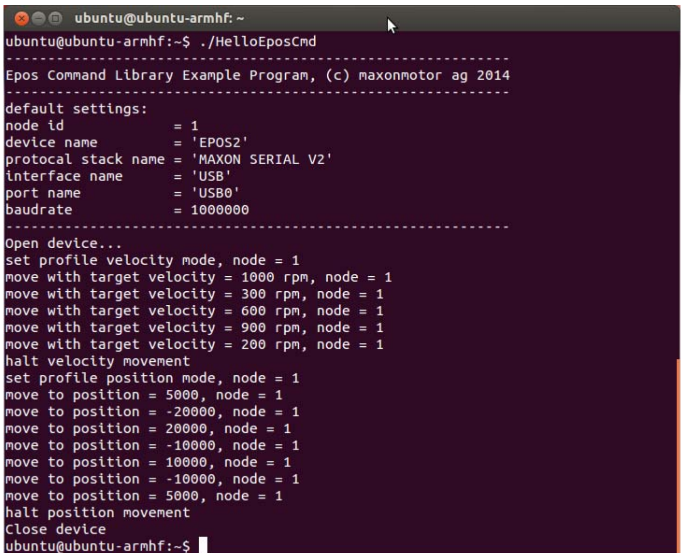

```
ubuntu@ubuntu-armhf: ~
ubuntu@ubuntu-armhf:~$ ./HelloEposCmd
------------------------------------------------------------
Epos Command Library Example Program, (c) maxonmotor ag 2014
------------------------------------------------------------
default settings:
node id             = 1
device name         = 'EPOS2'
protocal 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 9-35 HelloEposCmd executed with default settings

5) Sample call:
   EPOS2 controller connected over USB with node id = 2
   $ HelloEposCmd -n 2
   EPOS2 controller connected over RS-232 with node id = 1
   $HelloEposCmd -d EPOS2 -s MAXON_RS232 -i RS232 -p /dev/ttyS0 -b 115200
6) A properly configured and tuned controller presumed, you now will get several movements in «Position Mode» and «Velocity Mode».

10 Version History

10.1 Windows Operating Systems

Date [d.m.y]	Library Version	Documentation Edition	Description
04.07.2016	6.0.1.0	May 2016	Documentation update New: Implementation of EPOS4 New: Error codes added New: Appendix A featuring matrix on hardware and supported functions
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_GetVelocityRegulatorFeedForward, 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

Date [d.m.y]	Library 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_GetDeviceNameSelection(…), VCS_GetDriverInfo(…), VCS_GetInterfaceName(…), VCS_GetInterfaceNameSelection(…), VCS_GetPortName(…), VCS_GetPortNameSelection(…), VCS_GetProtocolStackModeSelection(…), VCS_GetProtocolStackName(…), VCS_GetProtocolStackNameSelection(…)
11.11.2003	1.00	November 2003	Initial release

Table 10-34 Version history – Windows OS

10.2 Linux Operating Systems

Date [d.m.y]	Library Version	Documentation Edition	Description
25.10.2016	6.0.1.0	October 2016	New: Implementation of EPOS4
10.10.2014	5.0.1.0	October 2014	New: x86_64, arm sf/hf support New functions: VCS_GetDriverInfo Bugfix: VCS_GetErrorInfo
26.04.2013	4.9.2.0	March 2013	New functions: VCS_GetHomingState, VCS_WaitForHomingAttained, VCS_GetVelocityIsAveraged, VCS_GetCurrentIsAveraged Bugfix: rs232 baudrate
27.07.2012	4.9.1.0	December 2013	New: kernel 2.6 support Bugfix: IPM mode Update: ftdi driver
14.03.2011	4.8.2.0	February 2011	Bugfix: USB interface
15.12.2010	4.8.1.0	January 2011	Initial release

Table 10-35 Version history – Linux OS

Appendix A — Hardware vs. Functions

In below tables you may find an overview on the available software functions versus their availability in the respective hardware versions. The tables are compiled in groups for initialization, configuration, operation, data recording, and low layer functions and are sorted in alphabetical order.

A click on the function's designation leads you directly to the detailed functional description.

INITIALIZATION FUNCTIONS

Designation	EPOS	EPOS2	EPOS4
Close All Devices	X	X	X
Close Device	X	X	X
Find Device Communication Settings	X	X	X
Get Baud Rate Selection	X	X	X
Get Device Name	X	X	X
Get Device Name Selection	X	X	X
Get Driver Info	X	X	X
Get Error Info	X	X	X
Get Interface Name	X	X	X
Get Interface Name Selection	X	X	X
Get Key Handle	X	X	X
Get Port Name	X	X	X
Get Port Name Selection	X	X	X
Get Protocol Stack Name	X	X	X
Get Protocol Stack Name Selection	X	X	X
Get Protocol Stack Settings	X	X	X
Get Version	X	X	X
Open Device	X	X	X
Open Device Dialog	X	X	X
Reset Port Name Selection	X	X	X
Set Protocol Stack Settings	X	X	X

Table 11-36   Hardware and their supported functions – Initialization functions

Configuration Functions

Designation	EPOS	EPOS2	EPOS4
Analog Input Configuration		X	
Digital Input Configuration	X	X	X
Digital Output Configuration	X	X	X
Export Parameter	X	X	X
Update Firmware		X	
Get Current Regulator Gain	X	X	
Get DC Motor Parameter	X	X	X
Get EC Motor Parameter	X	X	X
Get Hall Sensor Parameter	X	X	X
Get Incremental Encoder Parameter	X	X	X
Get Maximal Acceleration		X	X
Get Maximal Following Error	X	X	X
Get Maximal Profile Velocity	X	X	X
Get Motor Type	X	X	X
Get Object	X	X	X
Get Position Regulator Feed Forward	X	X	
Get Position Regulator Gain	X	X	
Get Sensor Type	X	X	X
Get SSI Absolute Encoder Parameter		X	
Get Velocity Regulator Feed Forward		X	
Get Velocity Regulator Gain	X	X	
Get Velocity Units		X	X
Import Parameter	X	X	X
Restore	X	X	X
Set Current Regulator Gain	X	X	
Set DC Motor Parameter	X	X	X
Set EC Motor Parameter	X	X	X
Set Hall Sensor Parameter	X	X	X
Set Incremental Encoder Parameter	X	X	X
Set Maximal Acceleration		X	X
Set Maximal Following Error	X	X	X
Set Maximal Profile Velocity	X	X	X
Set Motor Type	X	X	X
Set Object	X	X	X
Set Position Regulator Feed Forward	X	X	
Set Position Regulator Gain	X	X	
Set Sensor Type	X	X	X
Set SSI Absolute Encoder Parameter		X	

Designation	EPOS	EPOS2	EPOS4
Set Velocity Regulator Feed Forward		X	
Set Velocity Regulator Gain	X	X	
Set Velocity Units		X	X
Store	X	X	X

Table 11-37 Hardware and their supported functions – Configuration functions

Operation Functions

Designation	EPOS	EPOS2	EPOS4
Activate Analog Current Setpoint		X	
Activate Analog Position Setpoint		X	
Activate Analog Velocity Setpoint		X	
Activate Current Mode	X	X	X
Activate Homing Mode	X	X	X
Activate Interpolated Position Mode		X	
Activate Master Encoder Mode	X	X	
Activate Position Compare		X	
Activate Position Marker		X	
Activate Position Mode	X	X	
Activate Profile Position Mode	X	X	X
Activate Profile Velocity Mode	X	X	X
Activate Step Direction Mode	X	X	
Activate Velocity Mode	X	X	
Add PVT Value To IPM Buffer		X	
Clear Fault	X	X	X
Clear IPM Buffer		X	
Deactivate Analog Current Setpoint		X	
Deactivate Analog Position Setpoint		X	
Deactivate Analog Velocity Setpoint		X	
Deactivate Position Compare		X	
Deactivate Position Marker		X	
Define Position	X	X	X
Disable Analog Current Setpoint		X	
Disable Analog Position Setpoint		X	
Disable Analog Velocity Setpoint		X	
Disable Position Compare		X	
Disable Position Window	X	X	
Disable Velocity Window		X	
Enable Analog Current Setpoint		X	
Enable Analog Position Setpoint		X	

Designation	EPOS	EPOS2	EPOS4
Enable Analog Velocity Setpoint		X	
Enable Position Compare		X	
Enable Position Window	X	X	
Enable Velocity Window		X	
Find Home	X	X	X
Get All Digital Inputs	X	X	X
Get All Digital Outputs	X	X	X
Get Analog Input	X	X	X
Get Current Is	X	X	X
Get Current Is Averaged	X	X	X
Get Current Must	X	X	X
Get Device Error Code	X	X	X
Get Disable State	X	X	X
Get Enable State	X	X	X
Get Fault State	X	X	X
Get Free IPM Buffer Size		X	
Get Homing Parameter	X	X	X
Get Homing State	X	X	X
Get IPM Buffer Parameter		X	
Get IPM Status		X	
Get Master Encoder Parameter		X	
Get Movement State	X	X	X
Get Number of Device Error	X	X	X
Get Operation Mode	X	X	X
Get Position Compare Parameter		X	
Get Position Is	X	X	X
Get Position Marker Parameter		X	
Get Position Must	X	X	X
Get Position Profile	X	X	X
Get Quick Stop State	X	X	X
Get State	X	X	X
Get Step Direction Parameter		X	
Get Target Position	X	X	X
Get Target Velocity	X	X	X
Get Velocity Is	X	X	X
Get Velocity Is Averaged	X	X	X
Get Velocity Must	X	X	
Get Velocity Profile	X	X	X
Halt Position Movement	X	X	X
Halt Velocity Movement	X	X	X

Designation	EPOS	EPOS2	EPOS4
Move To Position	X	X	X
Move With Velocity	X	X	X
Read Position Marker Captured Position		X	
Read Position Marker Counter		X	
Reset Device	X	X	X
Reset Position Marker Counter		X	
Set All Digital Outputs	X	X	X
Set Analog Output	X	X	
Set Current Must	X	X	X
Set Disable State	X	X	X
Set Enable State	X	X	X
Set Homing Parameter	X	X	X
Set IPM Buffer Parameter		X	
Set Master Encoder Parameter		X	
Set Operation Mode	X	X	X
Set Position Compare Parameter		X	
Set Position Compare Reference Position		X	
Set Position Marker Parameter		X	
Set Position Must	X	X	
Set Position Profile	X	X	X
Set Quick Stop State	X	X	X
Set State	X	X	X
Set Step Direction Parameter		X	
Set Velocity Must	X	X	
Set Velocity Profile	X	X	X
Start IPM Trajectory		X	
Stop Homing	X	X	X
Stop IPM Trajectory		X	
Wait For Homing Attained	X	X	X
Wait For Target Reached	X	X	X

Table 11-38 Hardware and their supported functions – Operation functions

Data Recording Functions

Designation	EPOS	EPOS2	EPOS4
Activate Channel	X	X	
Deactivate all Channels	X	X	
Disable all Triggers	X	X	
Enable Trigger	X	X	
Export Channel Data to File	X	X	
Extract Channel Data Vector	X	X	
Force Trigger	X	X	
Get Recorder Parameter	X	X	
Is Recorder Running	X	X	
Is Recorder Triggered	X	X	
Read Channel Data Vector	X	X	
Read Channel Vector Size	X	X	
Read Data Buffer	X	X	
Set Recorder Parameter	X	X	
Show Channel Data Dialog	X	X	
Start Recorder	X	X	
Stop Recorder	X	X	

Table 11-39 Hardware and their supported functions – Data recording functions

Low Layer Functions

Designation	EPOS	EPOS2	EPOS4
Read CAN Frame	X	X	X
Request CAN Frame	X	X	X
Send CAN Frame	X	X	X
Send NMT Frame	X	X	X

Table 11-40 Hardware and their supported functions – Low layer functions

Appendix B — Function Groups Overview

••page intentionally left blank••

List of Figures

LIST OF TABLES

INDEX

••page intentionally left blank••

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