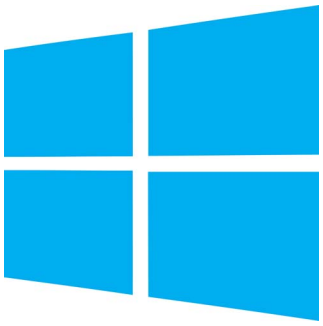
maxon

EPOS

Command Library Documentation



epos.maxongroup.com

EPOS Positioning Controllers
Command Library Documentation
CCMC | Edition 2021-03 | DocID rel9704

maxon

TABLE OF CONTENTS

Function Group Overview
For a detailed overview on function groups see page 12-179.

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!*

maxon

••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 see separate documents ➔«Firmware Specification» and ➔«Hardware Reference» of the respective positioning controller.
Index / Subindex	For detailed descriptions on used objects see separate document ➔«Firmware Specification».
IXXAT	www.ixxat.de
Kvaser	www.kvaser.com
maxon	www.maxongroup.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 see 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
Jetson™ NVIDIA®	© NVIDIA Corporation, USA-Santa Clara CA
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's experience. maxon explicitly states that its content is true and correct as to maxon'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 maxon's «EPOS Command Library» End User License Agreement (EULA) which is an integrated part of the library installation package.

1.8 Copyright

CCMC | EPOS Command Library Documentation | Edition 2021-03 | DocID rel9704

maxon motor ag
Brünigstrasse 220 +41 41 666 15 00
CH-6072 Sachseln www.maxongroup.com

maxon

••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:

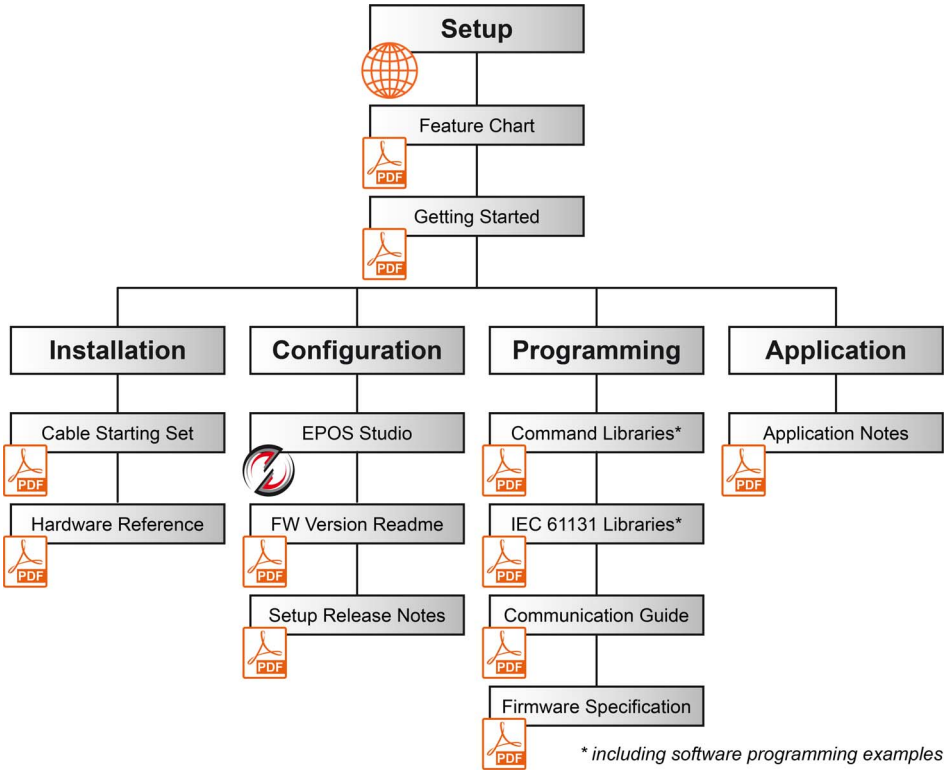


Figure 2-1 EPOS2 documentation structure

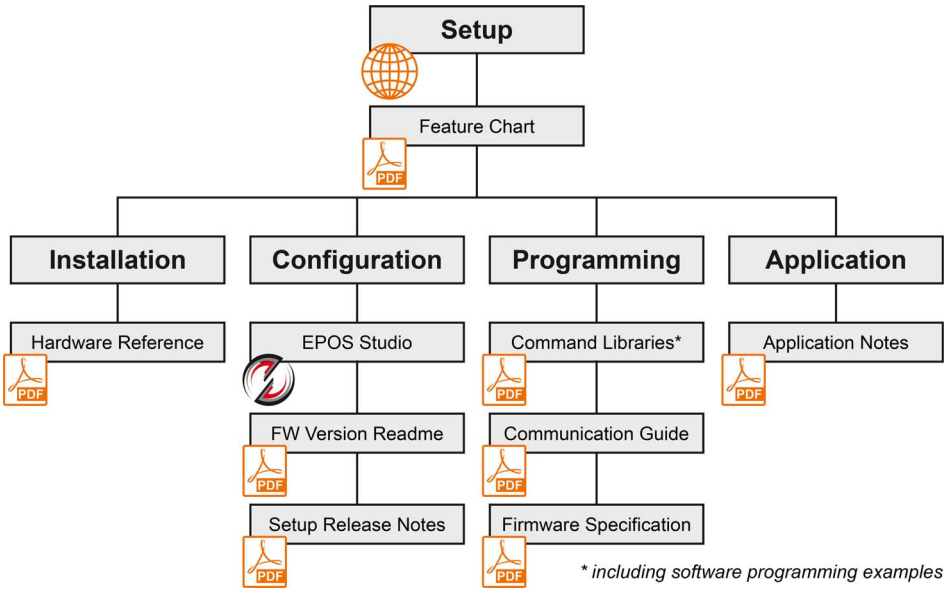


Figure 2-2 EPOS4 documentation structure

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's EPOS devices, which are connected to a serial RS232, USB, or CAN interface.

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.maxongroup.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 2-4 Third party supplier products

2.4 Communication Structure

2.4.1 Windows / Linux

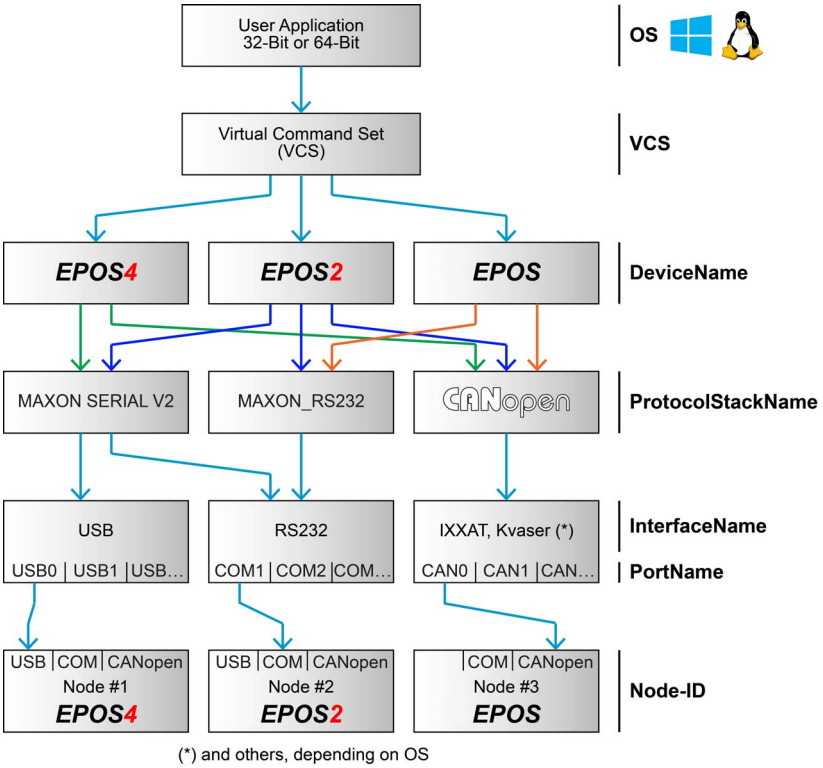


Figure 2-3 Windows / Linux – Communication structure (example)

2.4.2 Gateway

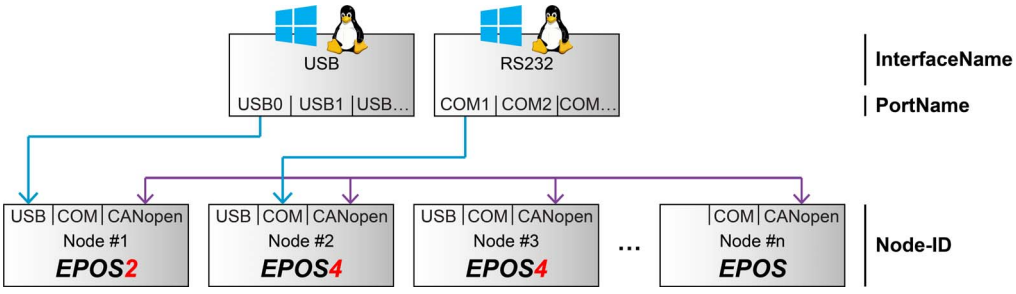


Figure 2-4 Gateway – Communication structure (example)

2.5 Data Type Definitions

Name	Data type	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	
		64	8	−2'147'483'648…2'147'483'647	Range independent of OS
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 2-5 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 11-173.*

3.1 Communication

3.1.1 VCS_OpenDevice

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 ➔*VCS_GetDeviceNameSelection*, ➔*VCS_GetProtocolStackNameSelection*, ➔*VCS_GetInterfaceNameSelection*, and ➔*VCS_GetPortNameSelection*.

For gateway topologies use function ➔*VCS_OpenSubDevice*.

PARAMETERS

DeviceName	char*	Name of connected device: • EPOS • EPOS2 • EPOS4 (Note: Also used for IDX drives)
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".

Continued on next page.

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 3-5 VCS_OpenDevice (programming example)

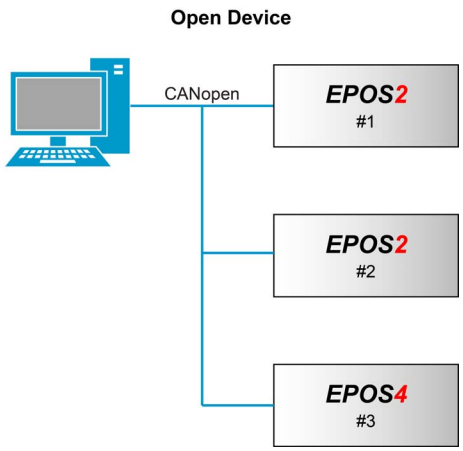


Figure 3-6 VCS_OpenDevice (example)

For gateway topologies use OpenSubDevice (➔chapter “3.1.8 VCS_OpenSubDevice” on page 3-18).

3.1.2 VCS_OpenDeviceDlg

FUNCTION

HANDLE VCS_OpenDeviceDlg(DWORD* pErrorCode)

DESCRIPTION

VCS_OpenDeviceDlg recognizes available interfaces capable to operate with EPOS and opens the selected interface for communication. Select “EPOS4” for IDX drives. 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 VCS_SetProtocolStackSettings

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 ➔*VCS_GetBaudRateSelection*.

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

In gateway topologies for subdevice use ➔*VCS_SetGatewaySettings* instead.

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 VCS_SetProtocolStackSettings (programming example)

3.1.4 VCS_GetProtocolStackSettings

FUNCTION

BOOL VCS_GetProtocolStackSettings(HANDLE KeyHandle, DWORD* pBaudrate, DWORD* pTimeout, DWORD* pErrorCode)

DESCRIPTION

VCS_GetProtocolStackSettings returns the baud rate and timeout communication parameters.

In gateway topologies for subdevice use ➔*VCS_GetGatewaySettings* instead.

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 VCS_FindDeviceCommunicationSettings

FUNCTION

BOOL VCS_FindDeviceCommunicationSettings(HANDLE* pKeyHandle, 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

pKeyHandle	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

pKeyHandle	HANDLE*	Handle for port access
pDeviceName	char*	Device name
pProtocolStackName	char*	Protocol stack name
pInterfaceName	char*	Interface name
pPortName	char*	Port name
pBaudrate	DWORD*	Baud rate [bit/s]
pTimeout	DWORD*	Timeout [ms]
pNodeId	WORD*	Node-ID
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

3.1.6 VCS_CloseAllDevices

FUNCTION

BOOL VCS_CloseAllDevices(DWORD* pErrorCode)

DESCRIPTION

VCS_CloseAllDevices closes all opened ports for devices and subdevices 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 VCS_CloseDevice

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”

3.1.8 VCS_OpenSubDevice

FUNCTION

HANDLE VCS_OpenSubDevice(HANDLE DeviceHandle, char* DeviceName, char* ProtocolStackName, DWORD* pErrorCode)

Description

VCS_OpenSubDevice opens the subdevice connected to the gateway device to send and receive commands.

PARAMETERS

DeviceHandle	HANDLE	Handle from opened device
DeviceName	char*	Name of connected subdevice: • EPOS • EPOS2 • EPOS4 (Note: Also used for IDX drives)
ProtocolStackName	char*	Name of used communication protocol: • CANopen

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
Return Value	HANDLE	Handle for gateway port access. Nonzero if successful; otherwise “0”.

Continued on next page.

PROGRAMMING EXAMPLE

```
// device (gateway)
HANDLE keyHandle = 0;
char* deviceName = "EPOS2";
char* protocolStackName = "MAXON SERIAL V2";
char* interfaceName = "RS232";
char* portName = "COM1";

DWORD errorCode = 0;
DWORD baudrate = 0;
DWORD timeout = 0;

// subdevice
HANDLE subkeyHandle = 0;
char* subdeviceName = "EPOS4";
char* subProtocolStackName = "CANopen";

keyHandle = VCS_OpenDevice(deviceName, protocolStackName, interfaceName, portName, &errorCode);

if (keyHandle > 0)
{
    if (VCS_GetProtocolStackSettings(keyHandle, &baudrate, &timeout, &errorCode))
    {
        timeout += 100;
        VCS_SetProtocolStackSettings(keyHandle, baudrate, timeout, &errorCode);
    }

    subkeyHandle = VCS_OpenSubDevice(keyHandle, subdeviceName, subProtocolStackName, &errorCode);

    if (subkeyHandle > 0)
    {
        if (VCS_GetGatewaySettings(keyHandle, &baudrate, &errorCode))
        {
            printf("Gateway baudrate = %u\r\n", baudrate);
        }

        //...application code...

        VCS_CloseSubDevice(subkeyHandle, &errorCode);
    }

    VCS_CloseDevice(keyHandle, &errorCode);
}
```

Figure 3-8 VCS_OpenSubDevice (programming example)

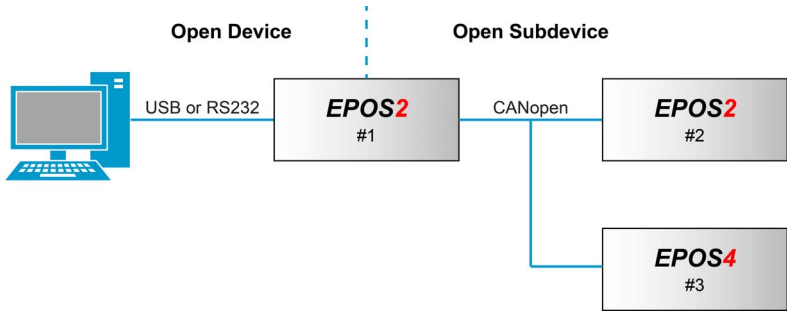


Figure 3-9 VCS_OpenSubDevice (example)

3.1.9 VCS_OpenSubDeviceDlg

FUNCTION

HANDLE VCS_OpenSubDeviceDlg(HANDLE DeviceHandle, DWORD* pErrorCode)

Description

VCS_OpenSubDeviceDlg recognizes available subdevices capable to operate with the gateway device and opens the selected device for communication. Select “EPOS4” for IDX drives. Not available with Linux.

PARAMETERS

DeviceHandle	HANDLE	Handle from opened device

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function
Return Value	HANDLE	Handle for gateway port access. Nonzero if successful; otherwise “0”.

3.1.10 VCS_SetGatewaySettings

FUNCTION

BOOL VCS_SetGatewaySettings(HANDLE KeyHandle, DWORD Baudrate, WORD* pErrorCode)

Description

VCS_SetGatewaySettings writes the gateway communication parameters to the device, stores them, and resets the gateway device.

The function does not set the communication parameters to all devices on the bus.

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

PARAMETERS

KeyHandle	HANDLE	Handle for port access
Baudrate	DWORD	Actual baud rate from opened port [bit/s]

RETURN PARAMETERS

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

3.1.11 VCS_GetGatewaySettings

FUNCTION

BOOL VCS_GetGatewaySettings(HANDLE KeyHandle, DWORD* pBaudrate, DWORD* pErrorCode)

Description

VCS_GetGatewaySettings returns the baud rate gateway communication parameter.

PARAMETERS

KeyHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pBaudrate	DWORD*	Actual baud rate from opened port [bit/s]
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”.

3.1.12 VCS_FindSubDeviceCommunicationSettings

FUNCTION

BOOL VCS_FindSubDeviceCommunicationSettings(HANDLE DeviceHandle, HANDLE* pKeyHandle, char* pDeviceName, char* pProtocolStackName, WORD SizeName, DWORD* pBaudrate, WORD* pNodeId, int DialogMode, DWORD* pErrorCode)

Description

VCS_FindSubDeviceCommunicationSettings searches the subdevice communication setting parameters. The parameters can be defined to accelerate the process. The search will be terminated as the first device is found. Not available with Linux.

PARAMETERS

DeviceHandle	HANDLE	Handle from opened device
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

pKeyHandle	HANDLE*	Handle for port access
pDeviceName	char*	Device name
pProtocolStackName	char*	ProtocolStack name
pBaudrate	DWORD*	Baud rate [bit/s]
pNodeId	WORD*	Node-ID
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”.

3.1.13 VCS_CloseAllSubDevices

FUNCTION

BOOL VCS_CloseAllSubDevices(HANDLE DeviceHandle, DWORD* pErrorCode)

Description

VCS_CloseAllSubDevices closes all opened subdevices and releases them for other applications.

PARAMETERS

DeviceHandle	HANDLE	Handle from opened device

RETURN PARAMETERS

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

3.1.14 VCS_CloseSubDevice

FUNCTION

BOOL VCS_CloseSubDevice(HANDLE KeyHandle, DWORD* pErrorCode)

Description

VCS_CloseSubDevice closes the subdevice and releases it for other applications.

PARAMETERS

KeyHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

3.2 Info

3.2.1 VCS_GetErrorInfo

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. For error codes ➔chapter “8 Error Overview” on page 8-145.

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 VCS_GetDriverInfo

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».

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 VCS_GetVersion

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 VCS_GetDeviceNameSelection

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 3-10 VCS_GetDeviceNameSelection (programming example)

3.3.2 VCS_GetProtocolStackNameSelection

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 3-11 VCS_GetProtocolStackNameSelection (programming example)

3.3.3 VCS_GetInterfaceNameSelection

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 3-12 VCS_GetInterfaceNameSelection (programming example)

3.3.4 VCS_GetPortNameSelection

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 3-13 VCS_GetPortNameSelection (programming example)

3.3.5 VCS_ResetPortNameSelection

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 VCS_GetBaudRateSelection

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 = "EPOS4";
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 baudrate (as long as endOfSelection == FALSE)
    while(!endOfSelection)
    {
        VCS_GetBaudrateSelection(deviceName, protocolStackName, interfaceName, portName,
                                 FALSE, &baudrateSel, &endOfSelection, &errorCode);
    }
}
```

Figure 3-14 VCS_GetBaudrateSelection (programming example)

3.3.7 VCS_GetKeyHandle

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 VCS_GetDeviceName

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 VCS_GetProtocolStackName

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 VCS_GetInterfaceName

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 VCS_GetPortName

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”

maxon

••page intentionally left blank••

4 CONFIGURATION FUNCTIONS

For detailed information on the objects see 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 11-173.

4.1 General

4.1.1 VCS_ImportParameter

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 4-15 VCS_ImportParameter (programming example)

4.1.2 VCS_ExportParameter

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 4-16 VCS_ExportParameter (programming example)

4.1.3 VCS_SetObject

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 see 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 VCS_GetObject

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 see 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 VCS_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 VCS_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 VCS_UpdateFirmware

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 4-17 VCS_UpdateFirmware (programming example)

4.2 Advanced Functions

4.2.1 Motor

4.2.1.1 VCS_SetMotorType

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 4-6)

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 4-6 Motor types

4.2.1.2 VCS_SetDcMotorParameter

The function is no longer recommended for implementation. Use ➔VCS_SetDcMotorParameterEx instead.

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 VCS_SetDcMotorParameterEx

FUNCTION

BOOL VCS_SetDcMotorParameterEx(HANDLE KeyHandle, WORD NodeId, DWORD NominalCurrent, DWORD MaxOutputCurrent, WORD ThermalTimeConstant, DWORD* pErrorCode)

DESCRIPTION

VCS_SetDcMotorParameterEx writes all DC motor parameters.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
NominalCurrent	DWORD	Maximal continuous current
MaxOutputCurrent	DWORD	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”

4.2.1.4 VCS_SetEcMotorParameter

The function is no longer recommended for implementation. Use ➔VCS_SetEcMotorParameterEx instead.

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”

4.2.1.5 VCS_SetEcMotorParameterEx

FUNCTION

BOOL VCS_SetEcMotorParameterEx(HANDLE KeyHandle, WORD NodeId, DWORD NominalCurrent, DWORD MaxOutputCurrent, WORD ThermalTimeConstant, BYTE NbOfPolePairs, DWORD* pErrorCode)

DESCRIPTION

VCS_SetEcMotorParameterEx writes all EC motor parameters.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
NominalCurrent	DWORD	Maximal continuous current
MaxOutputCurrent	DWORD	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”

4.2.1.6 VCS_GetMotorType

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

pMotorType	WORD*	Type of motor (➔Table 4-6)
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.1.7 VCS_GetDcMotorParameter

The function is no longer recommended for implementation. Use ➔VCS_GetDcMotorParameterEx instead.

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”

4.2.1.8 VCS_GetDcMotorParameterEx

FUNCTION

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

DESCRIPTION

VCS_GetDcMotorParameterEx reads all DC motor parameters.

PARAMETERS

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

RETURN PARAMETERS

pNominalCurrent	DWORD*	Maximal continuous current
pMaxOutputCurrent	DWORD*	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”

4.2.1.9 VCS_GetEcMotorParameter

The function is no longer recommended for implementation. Use ➔VCS_GetEcMotorParameterEx instead.

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
pNbOfPolePairs	BYTE*	Number of pole pairs
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.1.10 VCS_GetEcMotorParameterEx

FUNCTION

BOOL VCS_GetEcMotorParameterEx(HANDLE KeyHandle, WORD NodeId, DWORD* pNominalCurrent, DWORD* pMaxOutputCurrent, WORD* pThermalTimeConstant, BYTE* pNbOfPolePairs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetEcMotorParameterEx reads all EC motor parameters.

PARAMETERS

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

RETURN PARAMETERS

pNominalCurrent	DWORD*	Maximal continuous current
pMaxOutputCurrent	DWORD*	Maximal peak current
pThermalTimeConstant	WORD*	Thermal time constant winding
pNbOfPolePairs	BYTE*	Number of pole pairs
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.2 Sensor

4.2.2.1 VCS_SetSensorType

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 4-7)

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
Incremental encoder 2 with index (3-channel)	6	ST_INC_ENCODER2_3CHANNEL
Incremental encoder 2 without index (2-channel)	7	ST_INC_ENCODER2_2CHANNEL
Analog incremental encoder with index (3-channel)	8	ST_ANALOG_INC_ENCODER_3CHANNEL
Analog incremental encoder without index (2-channel)	9	ST_ANALOG_INC_ENCODER_2CHANNEL

Table 4-7 Position sensor types

4.2.2.2 VCS_SetIncEncoderParameter

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 VCS_SetHallSensorParameter

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 VCS_SetSsiAbsEncoderParameter

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 VCS_SetSsiAbsEncoderParameterEx

FUNCTION

BOOL VCS_SetSsiAbsEncoderParameterEx(HANDLE KeyHandle, WORD NodeId, WORD DataRate, WORD NbOfMultiTurnDataBits, WORD NbOfSingleTurnDataBits, WORD NbOfSpecialDataBits, BOOL InvertedPolarity, WORD Timeout, WORD PowerupTime, DWORD* pErrorCode)

DESCRIPTION

VCS_SetSsiAbsEncoderParameterEx writes all parameters for EPOS4 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
NbOfSpecialDataBits	WORD	Number of bits special data
InvertedPolarity	BOOL	Position sensor polarity
Timeout	WORD	Timeout time
PowerupTime	WORD	Power up time

RETURN PARAMETERS

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

4.2.2.6 VCS_GetSensorType

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 4-7)
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.2.7 VCS_GetIncEncoderParameter

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”

4.2.2.8 VCS_GetHallSensorParameter

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”

4.2.2.9 VCS_GetSsiAbsEncoderParameter

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”

4.2.2.10 VCS_GetSsiAbsEncoderParameterEx

FUNCTION

BOOL VCS_GetSsiAbsEncoderParameterEx(HANDLE KeyHandle, WORD NodeId, WORD* pDataRate, WORD* pNbOfMultiTurnDataBits, WORD* pNbOfSingleTurnDataBits, WORD* pNbOfSpecialDataBits, BOOL* pInvertedPolarity, WORD* pTimeout, WORD* pPowerupTime, DWORD* pErrorCode)

DESCRIPTION

VCS_GetSsiAbsEncoderParameterEx reads all parameters from EPOS4 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
pNbOfSpecialDataBits	WORD*	Number of bits special data
pInvertedPolarity	BOOL*	Position sensor polarity
pTimeout	WORD*	Timeout time
pPowerupTime	WORD*	Power up time
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.3 Safety

4.2.3.1 VCS_SetMaxFollowingError

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 VCS_GetMaxFollowingError

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 VCS_SetMaxProfileVelocity

FUNCTION

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

DESCRIPTION

VCS_SetMaxProfileVelocity writes the maximal allowed velocity. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_GetMaxProfileVelocity

FUNCTION

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

DESCRIPTION

VCS_GetMaxProfileVelocity reads the maximal allowed velocity. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_SetMaxAcceleration

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 VCS_GetMaxAcceleration

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 Controller Gain

4.2.4.1 VCS_SetControllerGain

FUNCTION

VCS_SetControllerGain(HANDLE KeyHandle, WORD NodeId, WORD EController, WORD EGain, DWORD64 Value, DWORD* pErrorCode)

DESCRIPTION

VCS_SetControllerGain writes the controller gain.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
EController	WORD	Regulation controller (➔Table 4-8)
EGain	WORD	Regulation gain (➔Table 4-9 thru Table 4-13)
Value	DWORD64	Regulation value

RETURN PARAMETERS

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

4.2.4.2 VCS_GetControllerGain

FUNCTION

VCS_GetControllerGain(HANDLE KeyHandle, WORD NodeId, WORD EController, WORD EGain, DWORD64* pValue, DWORD* pErrorCode)

DESCRIPTION

VCS_SetControllerGain reads the controller gain.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
EController	WORD	Regulation controller (➔Table 4-8)
EGain	WORD	Regulation gain (➔Table 4-9 thru Table 4-13)

RETURN PARAMETERS

pValue	DWORD64	Regulation value
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

Continued on next page.

Description	Value	Name
PI current controller	1	EC_PI_CURRENT_CONTROLLER
PI velocity controller	10	EC_PI_VELOCITY_CONTROLLER
PI velocity controller with observer	11	EC_PI_VELOCITY_CONTROLLER_WITH_OBSERVER
PID position controller	20	EC_PID_POSITION_CONTROLLER
Dual loop position controller	21	EC_DUAL_LOOP_POSITION_CONTROLLER

Table 4-8 Controller Gain – Regulation controller

Description	Value	Name
Current controller P gain	1	EG_PICC_P_GAIN
Current controller I gain	2	EG_PICC_I_GAIN

Table 4-9 Controller Gain – PI current controller gains

Description	Value	Name
Velocity controller P gain	1	EG_PIVC_P_GAIN
Velocity controller I gain	2	EG_PIVC_I_GAIN
Velocity controller FF acceleration gain	10	EG_PIVC_FEED_FORWARD_VELOCITY_GAIN
Velocity controller FF acceleration gain	11	EG_PIVC_FEED_FORWARD_ACCELERATION_GAIN

Table 4-10 Controller Gain – PI velocity controller gains

Description	Value	Name
Velocity controller P gain	1	EG_PIVCWO_P_GAIN
Velocity controller I gain	2	EG_PIVCWO_I_GAIN
Velocity controller FF acceleration gain	10	EG_PIVCWO_FEED_FORWARD_VELOCITY_GAIN
Velocity controller FF acceleration gain	11	EG_PIVCWO_FEED_FORWARD_ACCELERATION_GAIN
Velocity observer position correction gain	20	EG_PIVCWO_OBSERVER_THETA_GAIN
Velocity observer velocity correction gain	21	EG_PIVCWO_OBSERVER_OMEGA_GAIN
Velocity observer load correction gain	22	EG_PIVCWO_OBSERVER_TAU_GAIN

Table 4-11 Controller Gain – PI velocity controller gains with observer

Continued on next page.

Description	Value	Name
Position controller P gain	1	EG_PIDPC_P_GAIN
Position controller I gain	2	EG_PIDPC_I_GAIN
Position controller D gain	3	EG_PIDPC_D_GAIN
Position controller FF velocity gain	10	EG_PIDPC_FEED_FORWARD_VELOCITY_GAIN
Position controller FF acceleration gain	11	EG_PIDPC_FEED_FORWARD_ACCELERATION_GAIN

Table 4-12 Controller Gain – PID position controller gains

Description	Value	Name
Auxiliary loop P gain	1	EG_DLPC_AUXILIARY_LOOP_P_GAIN
Auxiliary loop I gain	2	EG_DLPC_AUXILIARY_LOOP_I_GAIN
Auxiliary loop FF velocity gain	10	EG_DLPC_AUXILIARY_LOOP_FEED_FORWARD_VELOCITY_GAIN
Auxiliary loop FF acceleration gain	11	EG_DLPC_AUXILIARY_LOOP_FEED_FORWARD_ACCELERATION_GAIN
Auxiliary loop observer position correction gain	20	EG_DLPC_AUXILIARY_LOOP_OBSERVER_THETA_GAIN
Auxiliary loop observer velocity correction gain	21	EG_DLPC_AUXILIARY_LOOP_OBSERVER_OMEGA_GAIN
Auxiliary loop observer load correction gain	22	EG_DLPC_AUXILIARY_LOOP_OBSERVER_TAU_GAIN
Main loop P gain low	101	EG_DLPC_MAIN_LOOP_P_GAIN_LOW
Main loop P gain high	102	EG_DLPC_MAIN_LOOP_P_GAIN_HIGH
Main loop gain scheduling weight	110	EG_DLPC_MAIN_LOOP_GAIN_SCHEDULING_WEIGHT
Main loop filter coefficient A	120	EG_DLPC_MAIN_LOOP_FILTER_COEFFICIENT_A
Main loop filter coefficient B	121	EG_DLPC_MAIN_LOOP_FILTER_COEFFICIENT_B
Main loop filter coefficient C	122	EG_DLPC_MAIN_LOOP_FILTER_COEFFICIENT_C
Main loop filter coefficient D	123	EG_DLPC_MAIN_LOOP_FILTER_COEFFICIENT_D
Main loop filter coefficient E	124	EG_DLPC_MAIN_LOOP_FILTER_COEFFICIENT_E

Table 4-13 Controller Gain – Dual loop controller gains

4.2.5 Inputs/Outputs

4.2.5.1 VCS_DigitalInputConfiguration

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 4-14)
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 4-14 Digital input configuration

4.2.5.2 VCS_DigitalOutputConfiguration

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 4-15)
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 4-15 Digital output configuration

4.2.5.3 VCS_AnalogInputConfiguration

FUNCTION

BOOL VCS_AnalogInputConfiguration(HANDLE KeyHandle, WORD NodeId, WORD AnalogInputNb, WORD Configuration, BOOL 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 4-16)
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 setpoint	0	AIC_ANALOG_CURRENT_SETPOINT
Analog velocity setpoint	1	AIC_ANALOG_VELOCITY_SETPOINT
Analog position setpoint	2	AIC_ANALOG_POSITION_SETPOINT
General purpose H	8	AIC_GENERAL_PURPOSE_H
General purpose G	9	AIC_GENERAL_PURPOSE_G
General purpose F	10	AIC_GENERAL_PURPOSE_F
General purpose E	11	AIC_GENERAL_PURPOSE_E
General purpose D	12	AIC_GENERAL_PURPOSE_D
General purpose C	13	AIC_GENERAL_PURPOSE_C
General purpose B	14	AIC_GENERAL_PURPOSE_B
General purpose A	15	AIC_GENERAL_PURPOSE_A

Table 4-16 Analog input configuration

4.2.5.4 VCS_AnalogOutputConfiguration

FUNCTION

BOOL VCS_AnalogOutputConfiguration(HANDLE KeyHandle, WORD NodeId, WORD AnalogOutputNb, WORD Configuration, DWORD* pErrorCode)

DESCRIPTION

VCS_AnalogOutputConfiguration sets the configuration parameter for one analog output.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
AnalogOutputNb	WORD	Number of analog output
Configuration	WORD	Configures the functionality assigned to the analog input (➔Table 4-17)

RETURN PARAMETERS

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

Description	Value	Name
General purpose A	0	AOC_GENERAL_PURPOSE_A
General purpose B	1	AOC_GENERAL_PURPOSE_B

Table 4-17 Analog output configuration

4.2.6 Units

4.2.6.1 VCS_SetVelocityUnits

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 4-18)

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 4-18 Velocity notation index

4.2.6.2 VCS_GetVelocityUnits

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 4-18)
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

maxon

••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 11-173.*

5.1 Operation Mode

5.1.1 VCS_SetOperationMode

FUNCTION

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

DESCRIPTION

VCS_SetOperationMode sets the operation mode. Modes marked with a triple asterisk (***) are automatically mapped to EPOS4-compatible firmware operation modes as to ➔Table 5-20.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
Mode	__int8	Operation mode (➔Table 5-19)

RETURN PARAMETERS

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

Description	Value	Name
Profile Position Mode (PPM)	1	OMD_PROFILE_POSITION_MODE
Profile Velocity Mode (PVM)	3	OMD_PROFILE_VELOCITY_MODE
Homing Mode (HM)	6	OMD_HOMING_MODE
Interpolated Position Mode (IPM)	7	OMD_INTERPOLATED_POSITION_MODE
Position Mode (PM, CSP)***	−1	OMD_POSITION_MODE
Velocity Mode (VM, CSV)***	−2	OMD_VELOCITY_MODE
Current Mode (CM, CST)***	−3	OMD_CURRENT_MODE
Master Encoder Mode	−5	OMD_MASTER_ENCODER_MODE
Step Direction Mode	−6	OMD_STEP_DIRECTION_MODE

Table 5-19 Operation modes

Mapped from		Mapped to	
Name	Value	Name	Value
Position Mode (PM)	−1	Cyclic Synchronous Position Mode (CSP)	8
Velocity Mode (VM)	−2	Cyclic Synchronous Velocity Mode (CSV)	9
Current Mode (CM)	−3	Cyclic Synchronous Current Mode (CST)	10

Table 5-20 Mapped operation modes

5.1.2 VCS_GetOperationMode

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 5-19)
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 see separate document ➔«Firmware Specification».

5.2.1 VCS_ResetDevice

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 VCS_SetState

FUNCTION

BOOL VCS_SetState(HANDLE KeyHandle, WORD NodeId, WORD State, DWORD* pErrorCode)

DESCRIPTION

VCS_SetState writes 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 5-21)

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 5-21 State modes

5.2.3 VCS_SetEnableState

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 VCS_SetDisableState

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 VCS_SetQuickStopState

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 VCS_ClearFault

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 VCS_GetState

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 5-21)
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.8 VCS_GetEnableState

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 VCS_GetDisableState

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 VCS_GetQuickStopState

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 VCS_GetFaultState

FUNCTION

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

DESCRIPTION

VCS_GetFaultState returns the device state fault. Get error information if the device is in fault state (➔“Error Handling” on page 5-72).

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 VCS_GetNbOfDeviceError

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 5-18 VCS_GetNbOfDeviceError (programming example)

5.3.2 VCS_GetDeviceErrorCode

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_GetDeviceErrorCode(keyHandle, nodeId, errorNumber, &deviceErrorCode, &errorCode))
        {
            break;
        }
    }
}
//...
```

Figure 5-19 VCS_GetDeviceErrorCode (programming example)

5.4 Motion Info

5.4.1 VCS_GetMovementState

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 VCS_GetPositionIs

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 VCS_GetVelocityIs

FUNCTION

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

DESCRIPTION

VCS_GetVelocityIs reads the velocity actual value. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_GetVelocityIsAveraged

FUNCTION

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

DESCRIPTION

VCS_GetVelocityIsAveraged reads the velocity actual averaged value. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_GetCurrentIs

The function is no longer recommended for implementation. Use ➔VCS_GetCurrentIsEx instead.

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 VCS_GetCurrentIsEx

FUNCTION

BOOL VCS_GetCurrentIsEx(HANDLE KeyHandle, WORD NodeId, long* pCurrentIs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentIsEx returns the current actual value.

PARAMETERS

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

RETURN PARAMETERS

pCurrentIs	long*	Current actual value
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

5.4.7 VCS_GetCurrentIsAveraged

The function is no longer recommended for implementation. Use ➔VCS_GetCurrentIsAveragedEx instead.

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”

5.4.8 VCS_GetCurrentIsAveragedEx

FUNCTION

BOOL VCS_GetCurrentIsAveragedEx(HANDLE KeyHandle, WORD NodeId, long* pCurrentIsAveraged, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentIsAveragedEx returns the current actual averaged value.

PARAMETERS

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

RETURN PARAMETERS

pCurrentIsAveraged	long*	Current actual value averaged
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

5.4.9 VCS_WaitForTargetReached

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 VCS_ActivateProfilePositionMode

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 VCS_SetPositionProfile

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. Given in velocity units.
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 VCS_GetPositionProfile

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. Given in velocity units.
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 VCS_MoveToPosition

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 VCS_GetTargetPosition

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 VCS_HaltPositionMovement

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 VCS_EnablePositionWindow

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 VCS_DisablePositionWindow

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 VCS_ActivateProfileVelocityMode

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 VCS_SetVelocityProfile

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 VCS_GetVelocityProfile

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 VCS_MoveWithVelocity

FUNCTION

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

DESCRIPTION

VCS_MoveWithVelocity starts the movement with velocity profile to target velocity. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_GetTargetVelocity

FUNCTION

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

DESCRIPTION

VCS_GetTargetVelocity returns the profile velocity mode target value. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_HaltVelocityMovement

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 VCS_EnableVelocityWindow

FUNCTION

BOOL VCS_EnableVelocityWindow(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. Given in velocity units.
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 VCS_DisableVelocityWindow

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 VCS_ActivateHomingMode

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 VCS_SetHomingParameter

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 VCS_GetHomingParameter

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 VCS_FindHome

FUNCTION

BOOL VCS_FindHome(HANDLE KeyHandle, WORD NodeId, __int8 HomingMethod, DWORD* ErrorCode)

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 5-22)

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_POSITIVE_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 5-22 Homing methods

5.7.5 VCS_StopHoming

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 VCS_DefinePosition

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 VCS_GetHomingState

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 VCS_WaitForHomingAttained

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 VCS_ActivateInterpolatedPositionMode

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 VCS_SetIpmBufferParameter

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 VCS_GetIpmBufferParameter

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 VCS_ClearIpmBuffer

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 VCS_GetFreeIpmBufferSize

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 VCS_AddPvtValueToIpmBuffer

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 VCS_StartIpmTrajectory

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 VCS_StopIpmTrajectory

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 VCS_GetIpmStatus

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 VCS_ActivatePositionMode

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 VCS_SetPositionMust

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 VCS_GetPositionMust

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 VCS_ActivateAnalogPositionSetpoint

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 (for EPOS2, take note of below remarks)
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”

Remarks for the use with EPOS2:

The scaling value range is limited to -32767…+32767 and is depending on the decimal place accuracy:

- *0 decimal digits: ±32767*
- *1 decimal digit: ±3276.7*
- *2 decimal digits: ±327.67*

Values with more than two decimal digits are rounded to two decimal digits.

5.9.4.2 VCS_DeactivateAnalogPositionSetpoint

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 VCS_EnableAnalogPositionSetpoint

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 VCS_DisableAnalogPositionSetpoint

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 VCS_ActivateVelocityMode

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 VCS_SetVelocityMust

FUNCTION

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

DESCRIPTION

VCS_SetVelocityMust sets the velocity mode setting value. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_GetVelocityMust

FUNCTION

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

DESCRIPTION

VCS_GetVelocityMust returns the velocity mode setting value. The velocity is interpreted according to the currently configured velocity unit.

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 VCS_ActivateAnalogVelocitySetpoint

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 (for EPOS2, take note of below remarks)
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”

Remarks for the use with EPOS2:

The scaling value range is limited to -32767…+32767 and is depending on the decimal place accuracy:

- *0 decimal digits: ±32767*
- *1 decimal digit: ±3276.7*
- *2 decimal digits: ±327.67*

Values with more than two decimal digits are rounded to two decimal digits.

5.10.4.2 VCS_DeactivateAnalogVelocitySetpoint

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 VCS_EnableAnalogVelocitySetpoint

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 VCS_DisableAnalogVelocitySetpoint

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 VCS_ActivateCurrentMode

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 VCS_GetCurrentMust

The function is no longer recommended for implementation. Use ➔VCS_GetCurrentMustEx instead.

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 VCS_GetCurrentMustEx

FUNCTION

BOOL VCS_GetCurrentMustEx(HANDLE KeyHandle, WORD NodeId, long* pCurrentMust, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentMustEx reads the current mode setting value.

PARAMETERS

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

RETURN PARAMETERS

pCurrentMust	long*	Current mode setting value
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

5.11.4 VCS_SetCurrentMust

The function is no longer recommended for implementation. Use ➔VCS_SetCurrentMustEx instead.

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”

5.11.5 VCS_SetCurrentMustEx

FUNCTION

BOOL VCS_SetCurrentMustEx(HANDLE KeyHandle, WORD NodeId, long CurrentMust, DWORD* pErrorCode)

DESCRIPTION

VCS_SetCurrentMustEx writes current mode setting value.

PARAMETERS

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

RETURN PARAMETERS

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

5.11.6 Advanced Functions

5.11.6.1 VCS_ActivateAnalogCurrentSetpoint

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 (for EPOS2, take note of below remarks)
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”

Remarks for the use with EPOS2:

The scaling value range is limited to -32767…+32767 and is depending on the decimal place accuracy:

- *0 decimal digits: ±32767*
- *1 decimal digit: ±3276.7*
- *2 decimal digits: ±327.67*

Values with more than two decimal digits are rounded to two decimal digits.

5.11.6.2 VCS_DeactivateAnalogCurrentSetpoint

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”

5.11.6.3 VCS_EnableAnalogCurrentSetpoint

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”

5.11.6.4 VCS_DisableAnalogCurrentSetpoint

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 VCS_ActivateMasterEncoderMode

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 VCS_SetMasterEncoderParameter

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. Given in velocity units.
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 VCS_GetMasterEncoderParameter

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. Given in velocity units.
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 VCS_ActivateStepDirectionMode

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 VCS_SetStepDirectionParameter

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. Given in velocity units.
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 VCS_GetStepDirectionParameter

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. Given in velocity units.
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 see separate document ➔«Firmware Specification».

5.14.1 VCS_GetAllDigitalInputs

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 5-20 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 5-20 VCS_GetAllDigitalInputs (tInputs)

5.14.2 VCS_GetAllDigitalOutputs

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 5-21 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 5-21 VCS_GetAllDigitalOutputs (tOutputs)

5.14.3 VCS_SetAllDigitalOutputs

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 5-22 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 5-22 VCS_SetAllDigitalOutputs (tOutputs)

5.14.4 VCS_GetAnalogInput

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 VCS_GetAnalogInputVoltage

FUNCTION

BOOL VCS_GetAnalogInputVoltage(HANDLE KeyHandle, WORD NodeId, WORD InputNumber, long* pVoltageValue, DWORD* pErrorCode)

DESCRIPTION

VCS_GetAnalogInputVoltage returns the voltage 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

pVoltageValue	long*	Analog voltage value from input
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.6 VCS_GetAnalogInputState

FUNCTION

BOOL VCS_GetAnalogInputState(HANDLE KeyHandle, WORD NodeId, WORD Configuration, long* pStateValue, DWORD* pErrorCode)

DESCRIPTION

VCS_GetAnalogInputState returns the state value from an analog input functionality.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
Configuration	WORD	Analog input function configuration

RETURN PARAMETERS

pStateValue	long*	Analog state value from input
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Analog current setpoint	0	AIC_ANALOG_CURRENT_SETPOINT
Analog velocity setpoint	1	AIC_ANALOG_VELOCITY_SETPOINT
Analog position setpoint	2	AIC_ANALOG_POSITION_SETPOINT
General purpose H	8	AIC_GENERAL_PURPOSE_H
General purpose G	9	AIC_GENERAL_PURPOSE_G
General purpose F	10	AIC_GENERAL_PURPOSE_F
General purpose E	11	AIC_GENERAL_PURPOSE_E
General purpose D	12	AIC_GENERAL_PURPOSE_D
General purpose C	13	AIC_GENERAL_PURPOSE_C
General purpose B	14	AIC_GENERAL_PURPOSE_B
General purpose A	15	AIC_GENERAL_PURPOSE_A

Table 5-23 Analog input states

5.14.7 VCS_SetAnalogOutput

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
AnalogValue	WORD	Analog value for output

RETURN PARAMETERS

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

5.14.8 VCS_SetAnalogOutputVoltage

FUNCTION

BOOL VCS_SetAnalogOutputVoltage(HANDLE KeyHandle, WORD NodeId, WORD OutputNumber, long VoltageValue, DWORD* pErrorCode)

DESCRIPTION

VCS_SetAnalogOutputVoltage 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
VoltageValue	long	Analog voltage value for output

RETURN PARAMETERS

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

5.14.9 VCS_SetAnalogOutputState

FUNCTION

BOOL VCS_SetAnalogOutputState(HANDLE KeyHandle, WORD NodeId, WORD Configuration, long StateValue, DWORD* pErrorCode)

DESCRIPTION

VCS_SetAnalogOutputState sets the state value for an analog output functionality.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
Configuration	WORD	Analog output function configuration
StateValue	long	Analog state value for output

RETURN PARAMETERS

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

Description	Value	Name
General purpose A	0	AOC_GENERAL_PURPOSE_A
General purpose B	1	AOC_GENERAL_PURPOSE_B

Table 5-24 Analog output states

5.14.10 Position Compare

5.14.10.1 VCS_SetPositionCompareParameter

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 5-25)
IntervalMode	BYTE	Used interval mode in position sequence mode (➔Table 5-26)
DirectionDependency	BYTE	Used direction dependency in position sequence mode (➔Table 5-27)
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 5-25 Position compare – Operational modes

Continued on next page.

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 5-26 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 5-27 Position compare – Direction dependency

5.14.10.2 VCS_GetPositionCompareParameter

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 5-25)
pIntervalMode	BYTE*	Used interval mode in position sequence mode (➔Table 5-26)
pDirectionDependency	BYTE*	Used direction dependency in position sequence mode (➔Table 5-27)
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”

5.14.10.3 VCS_ActivatePositionCompare

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”

5.14.10.4 VCS_DeactivatePositionCompare

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”

5.14.10.5 VCS_EnablePositionCompare

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”

5.14.10.6 VCS_DisablePositionCompare

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”

5.14.10.7 VCS_SetPositionCompareReferencePosition

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”

5.14.11 Position Marker

5.14.11.1 VCS_SetPositionMarkerParameter

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 5-28)
PositionMarkerMode	BYTE	Defines the position marker capturing mode (➔Table 5-29)

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 5-28 Position marker edge types

POSITIONMARKERMODE

Description	Value	Name
Continuous	0	PM_CONTINUOUS
Single	1	PM_SINGLE
Multiple	2	PM_MULTIPLE

Table 5-29 Position marker modes

5.14.11.2 VCS_GetPositionMarkerParameter

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 5-28)
pPositionMarkerMode	BYTE*	Defines the position marker capturing mode (➔Table 5-29)
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

5.14.11.3 VCS_ActivatePositionMarker

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”

5.14.11.4 VCS_DeactivatePositionMarker

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”

5.14.11.5 VCS_ReadPositionMarkerCounter

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”

5.14.11.6 VCS_ReadPositionMarkerCapturedPosition

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”

5.14.11.7 VCS_ResetPositionMarkerCounter

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”

maxon

••page intentionally left blank••

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 11-173.*
Data recording functions are available for both Windows and Linux. Exemptions are marked accordingly.

6.1 Operation Mode

6.1.1 VCS_SetRecorderParameter

FUNCTION

BOOL VCS_SetRecorderParameter(HANDLE KeyHandle, WORD NodeId, WORD SamplingPeriod, WORD NbOfPrecedingSamples, 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 0.1 ms NOTE: For EPOS4, the sampling period is automatically rounded to a multiple of 0.4 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 VCS_GetRecorderParameter

FUNCTION

BOOL VCS_GetRecorderParameter(HANDLE KeyHandle, WORD NodeId, WORD* pSamplingPeriod, WORD* pNbOfPrecedingSamples, 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 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 VCS_EnableTrigger

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 6-30). 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 6-30 Data recorder trigger types

6.1.4 VCS_DisableAllTriggers

FUNCTION

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

DESCRIPTION

VCS_DisableAllTriggers sets data recorder configuration 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 VCS_ActivateChannel

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 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 VCS_DeactivateAllChannels

FUNCTION

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

DESCRIPTION

VCS_DeactivateAllChannels zeros all data recording objects.

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 VCS_StartRecorder

FUNCTION

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

DESCRIPTION

VCS_StartRecorder starts data recording.

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 VCS_StopRecorder

FUNCTION

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

DESCRIPTION

VCS_StopRecorder stops data recording.

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 VCS_ForceTrigger

FUNCTION

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

DESCRIPTION

VCS_ForceTrigger forces the data recording triggers.

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 VCS_IsRecorderRunning

FUNCTION

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

DESCRIPTION

VCS_IsRecorderRunning returns the data recorder status “running”.

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 VCS_IsRecorderTriggered

FUNCTION

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

DESCRIPTION

VCS_IsRecorderTriggered returns data recorder status “triggered”.

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 VCS_ReadChannelVectorSize

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.

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 VCS_ReadChannelDataVector

FUNCTION

BOOL VCS_ReadChannelDataVector(HANDLE KeyHandle, WORD NodeId, BYTE ChannelNumber, BYTE* pDataVectorBuffer, DWORD VectorBufferSize, DWORD* pErrorCode)

DESCRIPTION

VCS_ReadChannelDataVector returns the data points of a selected channel.

PARAMETERS

KeyHandle	HANDLE	Handle for port access
NodeId	WORD	Node-ID of the addressed device
ChannelNumber	BYTE	Selected channel
VectorBufferSize	DWORD	Size of data vector buffer

RETURN PARAMETERS

pDataVectorBuffer	BYTE*	Data points of selected channel
pErrorCode	DWORD*	Error information on the executed function
Return Value	BOOL	Nonzero if successful; otherwise “0”

Continued on next page.

```
BYTE channelNumber = 1;
BYTE objectSize = 2;

BOOL result = FALSE;
DWORD errorCode = 0;
HANDLE keyHandle = (HANDLE)0x00040001;
WORD nodeId = 1;

DWORD vectorSize = 0;
DWORD vectorBufferSize = 0;
BYTE* pDataVectorBuffer = 0;

// ...

// Read channel vector size
result = VCS_ReadChannelVectorSize(keyHandle, nodeId, &vectorSize, &errorCode);

if (result)
{
    // Calculate buffer size
    vectorBufferSize = vectorSize * objectSize;
    // Allocate memory
    pDataVectorBuffer = (BYTE*)malloc(vectorBufferSize);

    // Read channel data vector
    result = VCS_ReadChannelDataVector(keyHandle, nodeId, channelNumber, (BYTE*)pDataVectorBuffer, vectorBufferSize, &errorCode);

    // Frees memory
    free(pDataVectorBuffer);
}
```

Figure 6-23 VCS_ReadChannelVector (programming example)

6.3.3 VCS_ShowChannelDataDlg

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 VCS_ExportChannelDataToFile

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 VCS_ReadDataBuffer

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 VCS_ExtractChannelDataVector

FUNCTION

BOOL VCS_ExtractChannelDataVector(HANDLE KeyHandle, WORD NodeId, BYTE ChannelNumber, BYTE* pDataBuffer, DWORD BufferSize, BYTE* pDataVectorBuffer, DWORD VectorBufferSize, 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	Size of data buffer
VectorBufferSize	DWORD	Size of data vector buffer
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

pDataVectorBuffer	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 11-173.*

7.1 CAN Layer

7.1.1 VCS_SendCANFrame

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"

7.1.2 VCS_ReadCANFrame

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"

7.1.3 VCS_RequestCANFrame

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”

7.1.4 VCS_SendNMTService

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
NodeId	WORD	1…127: NMT slave with given Node-ID 0: All NMT slaves
CommandSpecifier	WORD	NMT service (➔Table 7-31)

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 7-31 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 8-32 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 0013	Bad device handle	Bad device handle
0x1000 0014	Bad protocol stack handle	Bad protocol stack handle
0x1000 0015	Bad interface handle	Bad interface handle
0x1000 0016	Bad port handle	Bad port handle
0x1000 0017	Address parameters are not correct	Address parameters are not correct
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
0x1000 0024	Cross thread error	(.NET only) Open device and close device called from different threads
0x1000 0026	Gateway support error	Gateway is not supported
0x1000 0027	Serial number update error	Serial number update failed
0x1000 0028	Communication interface error	Communication interface is not supported
0x1000 0029	Firmware support error	Firmware version does not support functionality
0x1000 002A	Firmware file hardware error	Firmware file does not match hardware version
0x1000 002B	Firmware file error	Firmware file does not match or is corrupt
0x1000 002C	Parameter access denied	Parameter access denied
0x1000 002D	Data recorder not configured	Data recorder not configured
0x1000 002E	File format not supported	File format not supported
0x1000 002F	Failed saving data	Failed saving data

Table 8-33 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
0x2000 000A	Getting port settings error	Error while getting port settings
0x2000 000B	Access denied	Access denied error

Table 8-34 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 8-35 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 8-36 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 8-37 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 8-38 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 8-39 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 8-40 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 8-41 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 0002	Homing error	Homing procedure failed
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
0x5100 0009	Configuration not supported	Configuration is not supported
0x5100 000A	Digital input mask not supported	Digital input mask is not supported
0x5100 000B	Controller gain not supported	Tuning mode does not support the gain

Table 8-42 Device layer errors

9 SUPPORTED OPERATING SYSTEMS

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 a Windows or Linux 32-bit or 64-bit operating system. 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 9-149) and Linux (➔as of page 9-163) operating systems and comprises the following sections:

a) Overview

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

9.1.1 Overview

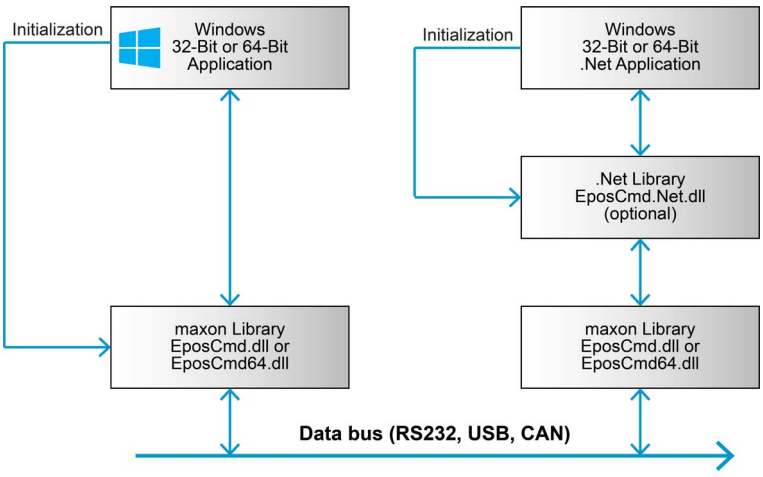


Figure 9-24 Windows – Library hierarchy

Continued on next page.

The Windows library supports communication interfaces and system architectures as shown in the following table:

Interface		Architecture	
		x86	x64
RS232		X	X
USB		X	X
CAN	IXXAT	X	X
	Kvaser	X	X
	NI	X	X
	Vector	X	X

Table 9-43 Supported platforms, architectures, and interfaces

9.1.1.1 Tested CAN Interfaces and Drivers

The following CAN adapters and driver versions were successfully tested:

IXXAT

- IXXAT USB-to-CAN V2 Professional
- IXXAT USB-to-CAN V2 Compact

Kvaser

- PCI canx II HS/HS
- Kvaser Leaf Light HS
- Kvaser Leaf Light v2

NI

- NI PCI-8512 CAN/HS
- NI PCI CAN, 2 Port

Vector

- Vector VN1610 CAN Interface
- Vector VN1611 CAN Interface

Other CAN adapters

Other CAN adapters might work with the library as well but have not been tested.

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 9-160.

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¤.

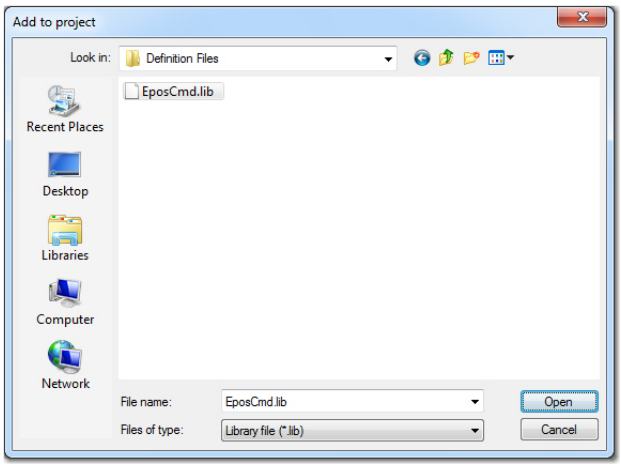


Figure 9-25 Borland C++Builder – Adding library

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

Best Practice

Use the calling convention __stdcall. It will manage how the parameters are put on the stack and how the stack will be cleaned once executed.

9.1.2.2 Borland Delphi

You will need to integrate the following files:

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

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Write the instruction “Definitions” into the uses clause of your program header.
3) Now, you can execute all library functions in your own code.

9.1.2.3 Microsoft Visual Basic

Remark

The «EPOS Command Library» was developed in programming language Microsoft Visual C++. Take note that data types in Microsoft Visual Basic and Microsoft Visual C++ differ. For more details consult the MSDN library, Visual Basic Concepts, ➔«Converting C Declarations to Visual Basic».

You will need to integrate the following files:

32-bit

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

64-bit

- **Definitions.vb** – Constant definitions and declarations of library functions
- **EposCmd64.dll** – Dynamic link library

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Add the file “Definitions.vb” to the project using the project tree in “Solution Explorer”. Click right on ¤Add¤, select ¤Existing Item¤, select the file, and click ¤Add¤.

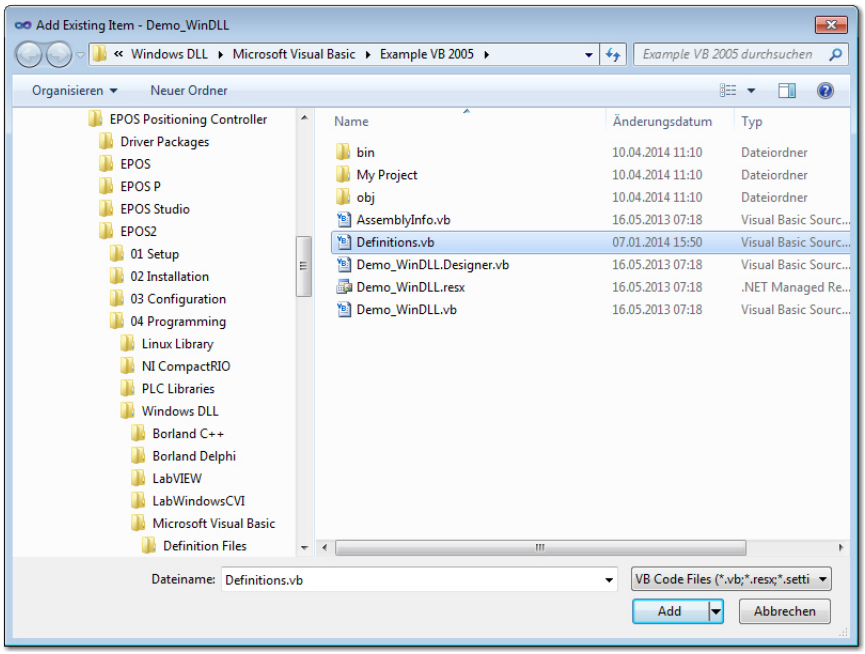


Figure 9-26 Visual Basic – Adding modules

3) Choose one of the two ways:
   a) Copy the file “EposCmd.dll” (for Windows 32-bit) or “EposCmd64.dll” for Windows 64-bit) into the release directory.
   b) Open menu ¤Properties¤, switch to the ¤Compile¤ tab and type “**.**” into the ¤Build output path¤ edit line.

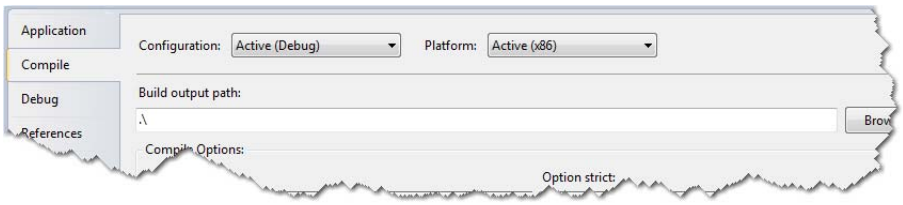


Figure 9-27 Visual Basic – Output path

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

9.1.2.4 Microsoft Visual Basic .NET

You will need to integrate the following files:

- **EposCmd.Net.dll** – .Net assembly
- **EposCmd.dll/ EposCmd64.dll** – Dynamic link library

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Add the .NET assembly "EposPCmd.Net.dll" to the project references using the project tree in “Solution Explorer”. Click right on ¤Add¤, select ¤Existing Item¤, select the file, and click ¤Add¤.

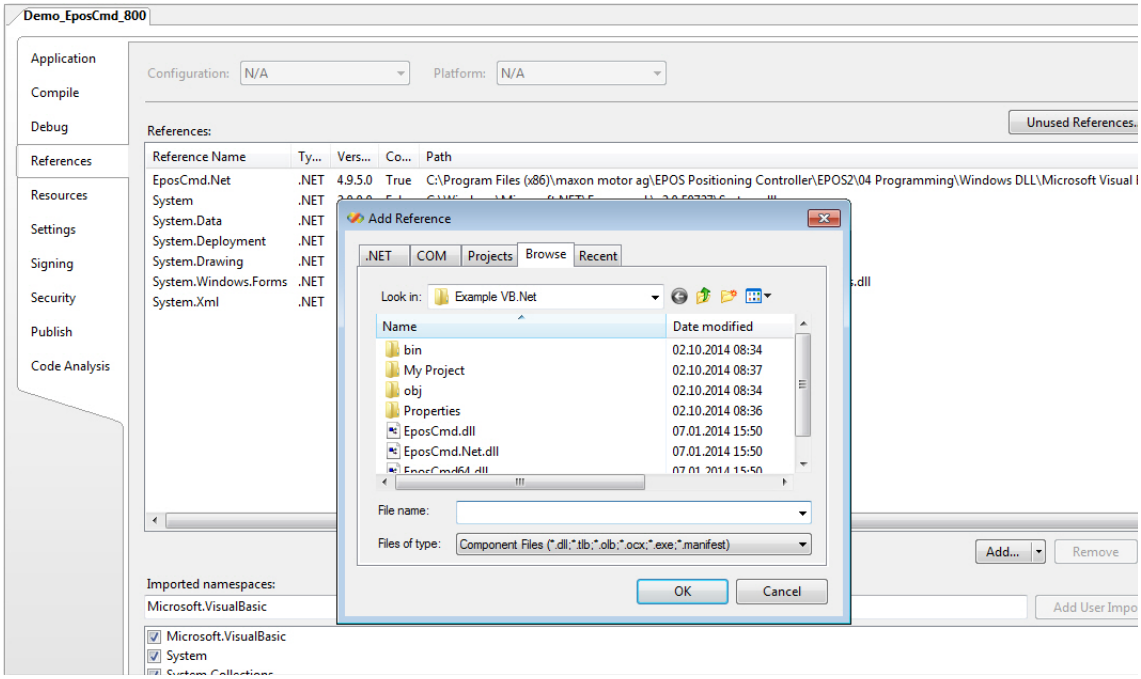


Figure 9-28 Visual Basic .NET – Adding modules

3) Choose one of the two ways:
   a) Copy the file “EposCmd.dll” (for Windows 32-bit) or “EposCmd64.dll” (for Windows 64-bit) into the release directory.
   b) Open menu ¤Properties¤, switch to the ¤Compile¤ tab and type “.\” into the ¤Build output path¤ edit line.

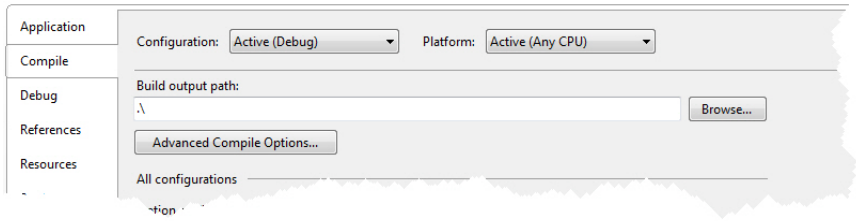


Figure 9-29 Visual Basic .NET – Output path

Continued on next page.

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

Remark
For further details and parameter description of the EposCmd.Net wrapper see separate document ➔«EposCmd.Net.chm».

9.1.2.5 Microsoft Visual C#

You will need to integrate the following files:

- **EposCmd.Net.dll** – .Net assembly
- **EposCmd.dll/ EposCmd64.dll** – Dynamic link library

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Setup the using directory in your program code using the instruction “using EposCmd.Net;”.
3) Add the file “EposCmd.Net” to the project using the project tree in “Solution Explorer”. Click right on ¤References¤, select ¤Add Reference¤, select the file, and click ¤OK¤.

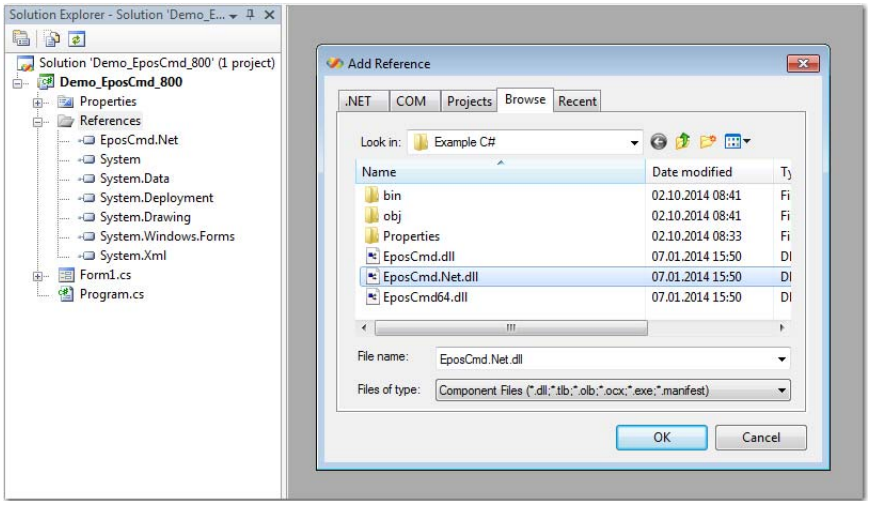


Figure 9-30 Visual C# – Project settings

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

Remark
For further details and parameter description of the EposCmd.Net wrapper see separate document ➔«EposCmd.Net.chm».

9.1.2.6 Microsoft Visual C++

You will need to integrate the following files:

32-bit

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

64-bit

- **Definitions.h** – Constant definitions and declarations of library functions
- **EposCmd64.dll** – Dynamic link library
- **EposCmd64.lib** – Import library (COFF format)

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Include the file “Definition.h” to your program code using the instruction “#include Definitions.h”.
3) Add the library to your project using menu ¤Project\Properties¤. Select ¤Linker\Input¤ from the tree and type the file name “EposCmd.lib” (for Windows 32-bit) or “EposCmd64.lib” (for Windows 64-bit) into the ¤Additional Dependencies¤ edit line.

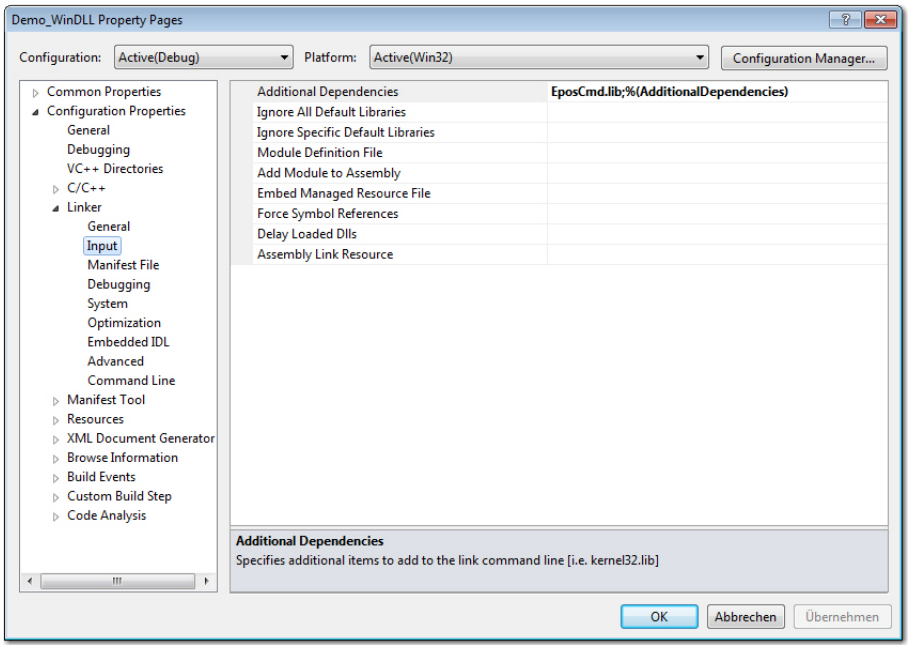


Figure 9-31 Visual C++ – Project settings

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

For an easy start with LabVIEW programming, most of the function blocks are already configured in a LabVIEW project structure.
VIs are supported with LabVIEW 2010 and higher.

Proceed as follows:

Either start the LabVIEW project “maxon EPOS.lvproj” or add the complete folder “maxon EPOS” to your project.

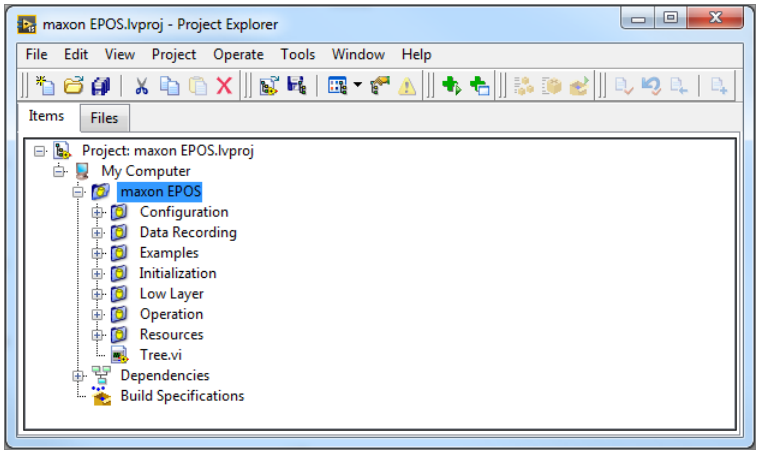


Figure 9-32 LabVIEW – Project Structure

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 the directory "…\borland".*
- *For Microsoft Visual C++ compiler use the file from the 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¤.

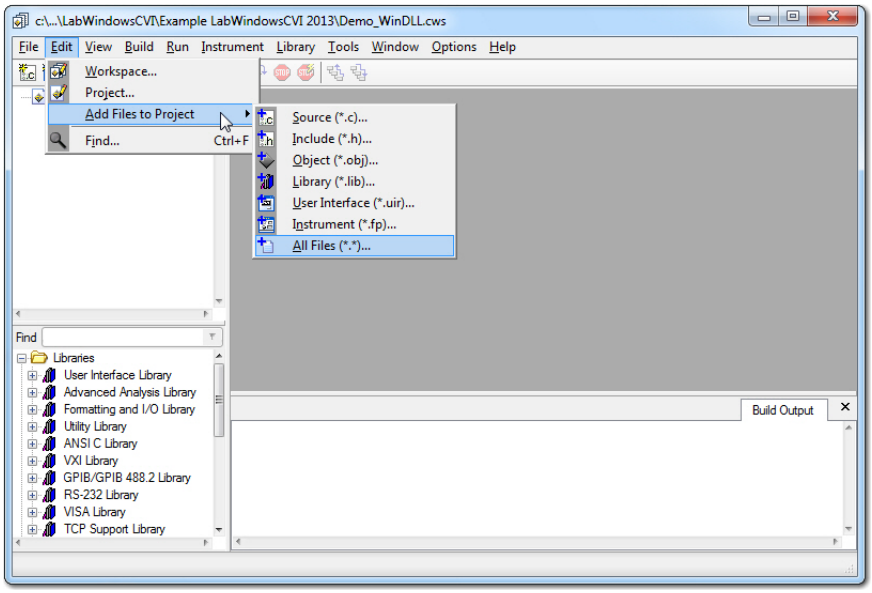


Figure 9-33 LabWindows – add files to project

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

9.2.1 Overview

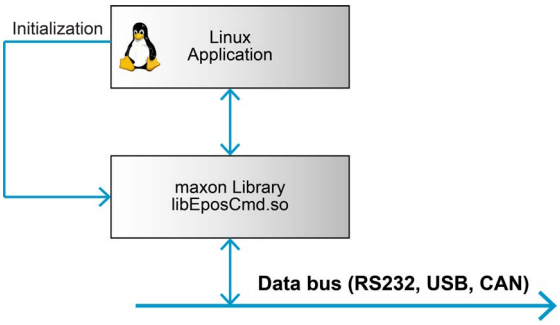


Figure 9-34 Linux – Library hierarchy

The Linux library supports communication interfaces and system architectures as shown in the following table:

Interface		Architecture				
		Intel		ARM		
		32-bit	64-bit	32-bit		64-bit
		X86	X86_64	V6	V7/V8	V8
RS232		X	X	—	X	—
USB		X	X	X	X	X
CAN	IXXAT	X	X	—	X	—
	Kvaser	X	X	—	X	—
	PiCAN2	—	—	—	X	—
	MTTCAN	—	—	—	—	X

Table 9-44 Supported platforms, architectures, and interfaces

Tested setups

- ***x86 / x86_64****: Tested on Ubuntu 12.04, 14.04 and 16.04 32/64-bit*
- ***ARMv6 32-bit****: Tested on Raspberry Pi Zero, Raspbian Stretch, Raspbian Buster*
- ***ARMv7/v8****: Tested on Raspberry Pi 2/3, Raspbian Stretch 32-bit, Raspbian Buster 32-bit*
- ***ARMv7/v8 - IXXAT****: Requires modification of the official IXXAT installation script (install USB only)*
- ***ARMv8 64-bit****: Tested on NVIDIA Jetson TX2, Ubuntu 16.04, R28, revision 2.1*

9.2.1.1 Tested CAN Interfaces and Drivers

Communication via CAN interfaces works through the SocketCAN driver and networking stack. Depending on the CAN interface model and brand you might need to install special drivers or upgrade the Linux kernel of your system.

The following CAN adapters and driver versions were successfully tested:

IXXAT (IXXAT SocketCAN Driver 1.1.138)

- IXXAT USB-to-CAN V2 Professional
- IXXAT USB-to-CAN V2 Compact

Kvaser (SocketCAN kernel driver)

- Kvaser Leaf Light v1
- Kvaser Leaf Light v2 (supported by Ubuntu 14.04 and newer)

SK Pang (SocketCAN kernel driver)

- PiCAN2 (based on MCP2511 CAN transceiver)
- Nvidia Jetson TX2 (MTTCAN SocketCAN driver), built-in CAN Interface with additional CAN transceiver

Other CAN adapters

Other CAN adapters might work with the library as well (SocketCAN interface driver required) but have not been tested.

9.2.2 Installation / Uninstallation

9.2.2.1 Unzipping the EPOS_Linux_Library

Unzip the EPOS_Linux_Library package:

```
$unzip EPOS_Linux_Library.zip -d .
```

NOTE: If the unzip program is not available, you can install it using the following command:

```
$sudo apt-get install unzip
```

9.2.2.2 Installing the EPOS Command Library

Go to the directory “EPOS_Linux_Library”:

```
$cd EPOS_Linux_Library
```

The install.sh script installs the EPOS Command Library and associated files to the directory “/opt/EposCmdLib_<version>” and configures device access rights on the system:

```
$sudo bash ./install.sh
```

NOTE: install.sh script requires sudo (root privileges)

Continued on next page.

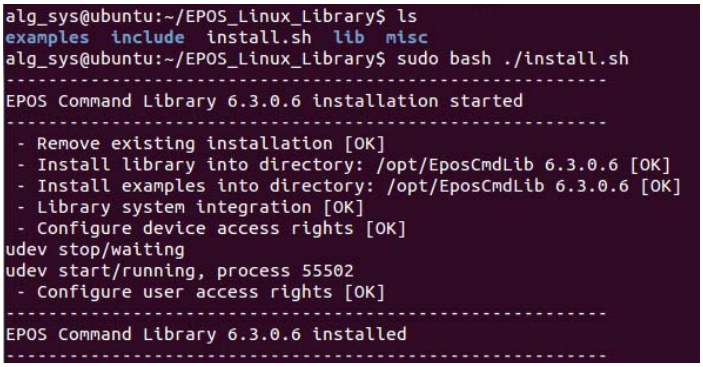


Figure 9-35 EPOS Command Library installation

After successful installation, the EPOS Command Library is ready for use.

9.2.2.3 Uninstalling the EPOS Command Library

Go to the package directory:

```
$cd EPOS_Linux_Library
```

Execute the uninstall script:

```
$sudo bash ./install.sh –u
or
$sudo bash ./install.sh --uninstall
```

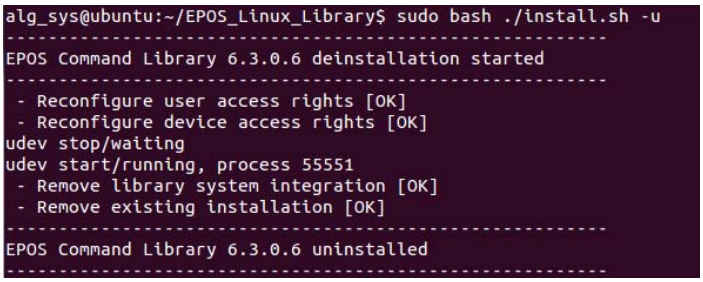


Figure 9-36 EPOS Command Library uninstallation

NOTE: The script will only uninstall the library version equal to the scripts package version.

9.2.3 Integration into Programming Environment

You will need to integrate the following files in your projects:

- **Definitions.h** – Constant definitions and declarations of library functions
- **libEposCmd.so.<major>.<minor>.<rev>.0** – EPOS Linux shared library

9.2.4 Programming

For details ➔Windows OS; chapter “9.1.3 Programming” on page 9-160.

The EPOS Linux library supports most of the EPOS commands. However, **not supported** are the following commands:

- Export/Import parameters commands
- GUI-related commands (such as VCS_OpenDeviceDlg)

9.2.4.1 Examples

HelloEposCmd

The demo program (source code) is available either in the package “EPOS_Linux_Library.zip” or after library installation in the directory “/opt/EposCmdLib_<version>/examples/HelloEposCmd”.

HelloEposCmd build and execution

```
$cd /opt/EposCmdLib_<version>/examples/HelloEposCmd
$make
$./HelloEposCmd
```

NOTE: If the make program is not available, you can install it using the following command:

```
$sudo apt-get install build-essential
```

The main purpose of HelloEposCmd is to show the basic concept of how to use the EPOS Command Libarary in a custom C++ application:

- Open and close communication with the device
- Get and set communication parameters
- Selected mode demo: Profile Velocity Mode (PVM), Profile Position Mode (PPM)

The HelloEposCmd application contains usefull command line parameters. Some of them can be used to identify the controller’s communication interface parameters for use with the EPOS Command Library later on.

-h: print out the command line parameters overview

```
alg_sys@ubuntu:~/HelloEposCmd$ ./HelloEposCmd -h
------------------------------------------------------------
Epos Command Library Example Program, (c) maxonmotor ag 2014-2017
------------------------------------------------------------
Usage: HelloEposCmd
        -h : this help
        -n : node id (default 1)
        -d   : device name (EPOS2, EPOS4, default - EPOS4)
        -s   : protocol stack name (MAXON_RS232, CANopen, MAXON SERIAL V2, default - MAXON SERIAL V2)
        -i   : interface name (RS232, USB, CAN_ixx_usb 0, CAN_kvaser_usb 0,... default - USB)
        -p   : port name (COM1, USB0, CAN0,... default - USB0)
        -b   : baudrate (115200, 1000000,... default - 1000000)
        -l   : list available interfaces (valid device name and protocol stack required)
        -r   : list supported protocols (valid device name required)
        -v   : display device version
alg_sys@ubuntu:~/HelloEposCmd$
```

Figure 9-37 HelloEposCmd – Parameters list

Continued on next page.

-r: list available protocol stacks for a selected device

```
alg_sys@ubuntu:~/HelloEposCmd$ ./HelloEposCmd -r
--------------------------------------------------------------
Epos Command Library Example Program, (c) maxonmotor ag 2014-2017
--------------------------------------------------------------
default settings:
node id              = 1
device name          = 'EPOS4'
protocal stack name = 'MAXON SERIAL V2'
interface name       = 'USB'
port name            = 'USB0'
baudrate             = 1000000
--------------------------------------------------------------
protocol stack name = MAXON SERIAL V2
protocol stack name = CANopen
--------------------------------------------------------------
```

Figure 9-38 HelloEposCmd – list available protocols

-l: list available interfaces and ports

```
alg_sys@ubuntu:~/HelloEposCmd$ ./HelloEposCmd -l -s 'CANopen'
--------------------------------------------------------------
Epos Command Library Example Program, (c) maxonmotor ag 2014-2017
--------------------------------------------------------------
default settings:
node id              = 1
device name          = 'EPOS4'
protocal stack name = 'CANopen'
interface name       = 'USB'
port name            = 'USB0'
baudrate             = 1000000
--------------------------------------------------------------
interface = CAN_ixx_usb 0
            port = CAN0
            port = CAN1
--------------------------------------------------------------
```

Figure 9-39 HelloEposCmd – list available interfaces

-v: read device version information

```
alg_sys@ubuntu:~/HelloEposCmd$ ./HelloEposCmd -s 'CANopen' -i 'CAN_ixx_usb 0' -p 'CAN0' -v
--------------------------------------------------------------
Epos Command Library Example Program, (c) maxonmotor ag 2014-2017
--------------------------------------------------------------
default settings:
node id              = 1
device name          = 'EPOS4'
protocal stack name = 'CANopen'
interface name       = 'CAN_ixx_usb 0'
port name            = 'CAN0'
baudrate             = 1000000
--------------------------------------------------------------
Open device...
EPOS4 Hardware Version    = 0x6552
      Software Version    = 0x0130
      Application Number  = 0xff00
      Application Version = 0x0050
Close device
```

Figure 9-40 HelloEposCmd – read device version

maxon

••page intentionally left blank••

10 VERSION HISTORY

10.1 Windows Operating Systems

Date [D / M / Y]	Library version	Documentation edition	Description
12.04.2021	6.7.1.0	2021-03	New: API VCS_SetEcMotorParameterEx, VCS_GetEcMotorParameterEx New: API VCS_SetDcMotorParameterEx, VCS_GetDcMotorParameterEx New: API VCS_SetCurrentMustEx, VCS_GetCurrentMustEx New: API VCS_GetCurrentIsEx, VCS_GetCurrentIsAveragedEx New: Error codes added Obsolete functions (do no longer use): VCS_SetEcMotorParameter, VCS_GetEcMotorParameter, VCS_SetDcMotorParameter, VCS_GetDcMotorParameter, VCS_SetCurrentMust, VCS_GetCurrentMust, VCS_GetCurrentIs, VCS_GetCurrentIsAveraged
21.04.2020	6.6.2.0	2020-04	Bugfix: EPOS4 VCS_GetAllDigitalInputs - Enable, Quickstop Bugfix: EPOS4 Data Recording Functions - Channel size, time unit Bugfix: EPOS4 VCS_UpdateFirmware - Not blocked via CANopen Bugfix: CANopen IXXAT Interfaces VCI-V4 bugfixes
02.12.2019	6.6.1.0	2019-11	New: Support of data recording functions for EPOS4 devices Bugfix: EPOS4 VCS_SetEnableState - Returns an error code when command fails Bugfix: EPOS2 VCS_ActivateAnalogCurrentSetpoint, VCS_ActivateAnalogVelocitySetpoint, VCS_ActivateAnalogPositionSetpoint - Support negative scaling values
18.12.2018	6.5.1.0	December 2018	New: API VCS_AnalogOutputConfiguration New: API VCS_GetAnalogInputVoltage New: API VCS_GetAnalogInputState New: API VCS_SetAnalogOutputVoltage New: API VCS_SetAnalogOutputState New: API VCS_GetControllerGain, VCS_SetControllerGain New: API VCS_SendNMTService support for EPOS4 Obsolete functions (do not use): VCS_SetCurrentRegulatorGain, VCS_SetPositionRegulatorGain, VCS_SetPositionRegulatorGainFeedForward, VCS_SetVelocityRegulatorGainFeedForward, VCS_SetVelocityRegulatorGain, VCS_GetCurrentRegulatorGain, VCS_GetPositionRegulatorGain, VCS_GetPositionRegulatorGainFeedForward, VCS_GetVelocityRegulatorGainFeedForward, VCS_GetVelocityRegulatorGain
06.08.2018	6.4.2.0	August 2018	Bugfix: VCS_UpdateFirmware - EPOS4 firmware update stability fixed
08.06.2018	6.4.1.0	May 2018	New: Support firmware update EPOS4 Bugfix: VCS_SetGatewaySettings - Resetting for EPOS2 fixed Bugfix: VCS_MoveToPosition - Reset 'TargetReached' bit after start
14.12.2017	6.3.1.0	November 2017	Bugfix: Duplicate issue “Error Cluster From Code.vi” in LabView Instrument Driver resolved Bugfix: USB port enumeration conflicts between EPOS2 and EPOS4 resolved
07.06.2017	6.2.1.0	May 2017	New: API for mixed gateway topologies EPOS, EPOS2, EPOS4 New: LabView Instrument Driver Update Bugfix: .Net Library: IPM mode starting fixed
20.01.2017	6.1.2.0	January 2017	Bugfix: EPOS2 USB communication with Windows 10 and USB 3.0 Bugfix: EPOS2 Interpolated Position Mode is not starting profile
25.10.2016	6.1.1.0	October 2016	New: EPOS4 RS232 communication New: EPOS4 SSI absolute encoder New: Support for IXXAT VCI4

Continued on next page.

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

Continued on next page.

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 10-45 Version history – Windows OS

10.2 Linux Operating Systems

Date [D / M / Y]	Library version	Documentation edition	Description
12.04.2021	6.7.1.0	2021-03	New: API VCS_SetEcMotorParameterEx, VCS_GetEcMotorParameterEx New: API VCS_SetDcMotorParameterEx, VCS_GetDcMotorParameterEx New: API VCS_SetCurrentMustEx, VCS_GetCurrentMustEx New: API VCS_GetCurrentIsEx, VCS_GetCurrentIsAveragedEx New: Error codes added Obsolete functions (do no longer use): VCS_SetEcMotorParameter, VCS_GetEcMotorParameter, VCS_SetDcMotorParameter, VCS_GetDcMotorParameter, VCS_SetCurrentMust, VCS_GetCurrentMust, VCS_GetCurrentIs, VCS_GetCurrentIsAveraged
21.04.2020	6.6.2.0	2020-04	Bugfix: EPOS4 VCS_GetAllDigitalInputs - Enable, Quickstop
02.12.2019	6.6.1.0	2019-11	New: Support of data recording functions for EPOS4 devices Bugfix: EPOS4 VCS_SetEnableState - Returns an error code when command fails Bugfix: EPOS2 VCS_ActivateAnalogCurrentSetpoint, VCS_ActivateAnalogVelocitySetpoint, VCS_ActivateAnalogPositionSetpoint - Support negative scaling values Bugfix: Support of IXXAT PCIE CAN adapters Bugfix: Socket CAN interface/port enumeration

Continued on next page.

Date [D / M / Y]	Library version	Documentation edition	Description
18.12.2018	6.5.1.0	December 2018	New: API VCS_AnalogOutputConfiguration New: API VCS_GetAnalogInputVoltage New: API VCS_GetAnalogInputState New: API VCS_SetAnalogOutputVoltage New: API VCS_SetAnalogOutputState New: API VCS_GetControllerGain, VCS_SetControllerGain New: API VCS_SendNMTService support for EPOS4 New: ARMv8 64-bit (Nvidia Jetson TX2) New: ARMv6 32-bit (Raspberry Pi Zero) Update: FTDI driver 1.4.8 Obsolete functions (do not use): VCS_SetCurrentRegulatorGain, VCS_SetPositionRegulatorGain, VCS_SetPositionRegulatorGainFeedForward, VCS_SetVelocityRegulatorGainFeedForward, VCS_SetVelocityRegulatorGain, VCS_GetCurrentRegulatorGain, VCS_GetPositionRegulatorGain, VCS_GetPositionRegulatorGainFeedForward, VCS_GetVelocityRegulatorGainFeedForward, VCS_GetVelocityRegulatorGain
08.06.2018	6.4.1.0	May 2018	New: API for mixed gateway topologies EPOS, EPOS2, EPOS4 Improved function not supported handling
14.12.2017	6.3.1.0	November 2017	New: Support CANopen communication interfaces New: EPOS Linux Library install script Bugfix: Intel Skylake architecture lock elision issue resolved Bugfix: USB port enumeration conflicts resolved
07.06.2017	6.2.1.0	May 2017	Bugfix: Missing makefile for example added Bugfix: Wrong datatype in Definitions.h fixed
20.01.2017	6.1.1.0	January 2017	Bugfix: Make file added for example “HelloEposCmd”
25.10.2016	6.1.1.0	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 10-46 Version history – Linux OS

maxon

Appendix A — Hardware vs. Functions

In the following tables you can 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
VCS_CloseAllDevices	X	X	X
VCS_CloseAllSubDevices	X	X	X
VCS_CloseDevice	X	X	X
VCS_CloseSubDevice	X	X	X
VCS_FindDeviceCommunicationSettings	X	X	X
VCS_FindSubDeviceCommunicationSettings	X	X	X
VCS_GetBaudRateSelection	X	X	X
VCS_GetDeviceName	X	X	X
VCS_GetDeviceNameSelection	X	X	X
VCS_GetDriverInfo	X	X	X
VCS_GetErrorInfo	X	X	X
VCS_GetInterfaceName	X	X	X
VCS_GetInterfaceNameSelection	X	X	X
VCS_GetKeyHandle	X	X	X
VCS_GetPortName	X	X	X
VCS_GetPortNameSelection	X	X	X
VCS_GetProtocolStackName	X	X	X
VCS_GetProtocolStackNameSelection	X	X	X
VCS_GetProtocolStackSettings	X	X	X
VCS_GetVersion	X	X	X
VCS_OpenDevice	X	X	X
VCS_OpenDeviceDlg	X	X	X
VCS_OpenSubDevice	X	X	X
VCS_OpenSubDeviceDlg	X	X	X
VCS_ResetPortNameSelection		X	X
VCS_SetProtocolStackSettings	X	X	X

Table 11-47 Hardware and their supported functions – Initialization functions

CONFIGURATION FUNCTIONS

Designation	EPOS	EPOS2	EPOS4
VCS_AnalogInputConfiguration		X	
VCS_AnalogOutputConfiguration			X
VCS_DigitalInputConfiguration	X	X	X
VCS_DigitalOutputConfiguration	X	X	X
VCS_ExportParameter	X	X	X
VCS_GetControllerGain	X	X	X
VCS_GetDcMotorParameter	X	X	X
VCS_GetDcMotorParameterEx	X	X	X
VCS_GetEcMotorParameter	X	X	X
VCS_GetEcMotorParameterEx	X	X	X
VCS_GetHallSensorParameter	X	X	X
VCS_GetIncEncoderParameter	X	X	X
VCS_GetMaxAcceleration		X	X
VCS_GetMaxFollowingError	X	X	X
VCS_GetMaxProfileVelocity	X	X	X
VCS_GetMotorType	X	X	X
VCS_GetObject	X	X	X
VCS_GetSensorType	X	X	X
VCS_GetSsiAbsEncoderParameter		X	X
VCS_GetSsiAbsEncoderParameterEx			X
VCS_GetVelocityUnits		X	X
VCS_ImportParameter	X	X	X
VCS_Restore	X	X	X
VCS_SetControllerGain	X	X	X
VCS_SetDcMotorParameter	X	X	X
VCS_SetDcMotorParameterEx	X	X	X
VCS_SetEcMotorParameter	X	X	X
VCS_SetEcMotorParameterEx	X	X	X
VCS_SetHallSensorParameter	X	X	X
VCS_SetIncEncoderParameter	X	X	X
VCS_SetMaxAcceleration		X	X
VCS_SetMaxFollowingError	X	X	X
VCS_SetMaxProfileVelocity	X	X	X
VCS_SetMotorType	X	X	X
VCS_SetObject	X	X	X
VCS_SetSensorType	X	X	X
VCS_SetSsiAbsEncoderParameter		X	X
VCS_SetSsiAbsEncoderParameterEx			X
VCS_SetVelocityUnits		X	X
VCS_Store	X	X	X
VCS_UpdateFirmware		X	X

Table 11-48 Hardware and their supported functions – Configuration functions

Operation Functions

Designation	EPOS	EPOS2	EPOS4
VCS_ActivateAnalogCurrentSetpoint		X	
VCS_ActivateAnalogPositionSetpoint		X	
VCS_ActivateAnalogVelocitySetpoint		X	
VCS_ActivateCurrentMode	X	X	X
VCS_ActivateHomingMode	X	X	X
VCS_ActivateInterpolatedPositionMode		X	
VCS_ActivateMasterEncoderMode	X	X	
VCS_ActivatePositionCompare		X	
VCS_ActivatePositionMarker	X	X	
VCS_ActivatePositionMode	X	X	X
VCS_ActivateProfilePositionMode	X	X	X
VCS_ActivateProfileVelocityMode	X	X	X
VCS_ActivateStepDirectionMode	X	X	
VCS_ActivateVelocityMode	X	X	X
VCS_AddPvtValueToIpmBuffer		X	
VCS_ClearFault	X	X	X
VCS_ClearIpmBuffer		X	
VCS_DeactivateAnalogCurrentSetpoint		X	
VCS_DeactivateAnalogPositionSetpoint		X	
VCS_DeactivateAnalogVelocitySetpoint		X	
VCS_DeactivatePositionCompare		X	
VCS_DeactivatePositionMarker	X	X	
VCS_DefinePosition	X	X	X
VCS_DisableAnalogCurrentSetpoint		X	
VCS_DisableAnalogPositionSetpoint		X	
VCS_DisableAnalogVelocitySetpoint		X	
VCS_DisablePositionCompare		X	
VCS_DisablePositionWindow	X	X	
VCS_DisableVelocityWindow		X	
VCS_EnableAnalogCurrentSetpoint		X	
VCS_EnableAnalogPositionSetpoint		X	
VCS_EnableAnalogVelocitySetpoint		X	
VCS_EnablePositionCompare		X	
VCS_EnablePositionWindow	X	X	
VCS_EnableVelocityWindow		X	
VCS_FindHome	X	X	X
VCS_GetAllDigitalInputs	X	X	X
VCS_GetAllDigitalOutputs	X	X	X
VCS_GetAnalogInput	X	X	X
VCS_GetAnalogInputState		X	
VCS_GetAnalogInputVoltage	X	X	X

Continued on next page.

Designation	EPOS	EPOS2	EPOS4
VCS_GetCurrentIs	X	X	X
VCS_GetCurrentIsAveraged	X	X	X
VCS_GetCurrentIsAveragedEx	X	X	X
VCS_GetCurrentIsEx	X	X	X
VCS_GetCurrentMust	X	X	X
VCS_GetCurrentMustEx	X	X	X
VCS_GetDeviceErrorCode	X	X	X
VCS_GetDisableState	X	X	X
VCS_GetEnableState	X	X	X
VCS_GetFaultState	X	X	X
VCS_GetFreeIpmBufferSize		X	
VCS_GetHomingParameter	X	X	X
VCS_GetHomingState	X	X	X
VCS_GetIpmBufferParameter		X	
VCS_GetIpmStatus		X	
VCS_GetMasterEncoderParameter		X	
VCS_GetMovementState	X	X	X
VCS_GetNbOfDeviceError	X	X	X
VCS_GetOperationMode	X	X	X
VCS_GetPositionCompareParameter		X	
VCS_GetPositionIs	X	X	X
VCS_GetPositionMarkerParameter	X	X	
VCS_GetPositionMust	X	X	X
VCS_GetPositionProfile	X	X	X
VCS_GetQuickStopState	X	X	X
VCS_GetState	X	X	X
VCS_GetStepDirectionParameter		X	
VCS_GetTargetPosition	X	X	X
VCS_GetTargetVelocity	X	X	X
VCS_GetVelocityIs	X	X	X
VCS_GetVelocityIsAveraged	X	X	X
VCS_GetVelocityMust	X	X	X
VCS_GetVelocityProfile	X	X	X
VCS_HaltPositionMovement	X	X	X
VCS_HaltVelocityMovement	X	X	X
VCS_MoveToPosition	X	X	X
VCS_MoveWithVelocity	X	X	X
VCS_ReadPositionMarkerCapturedPosition	X	X	
VCS_ReadPositionMarkerCounter	X	X	
VCS_ResetDevice	X	X	X
VCS_ResetPositionMarkerCounter	X	X	
VCS_SetAllDigitalOutputs	X	X	X
VCS_SetAnalogOutput	X	X	X

Continued on next page.

Designation	EPOS	EPOS2	EPOS4
VCS_SetAnalogOutputState			X
VCS_SetAnalogOutputVoltage		X	X
VCS_SetCurrentMust	X	X	X
VCS_SetCurrentMust	X	X	X
VCS_SetDisableState	X	X	X
VCS_SetEnableState	X	X	X
VCS_SetHomingParameter	X	X	X
VCS_SetIpmBufferParameter		X	
VCS_SetMasterEncoderParameter		X	
VCS_SetOperationMode	X	X	X
VCS_SetPositionCompareParameter		X	
VCS_SetPositionCompareReferencePosition		X	
VCS_SetPositionMarkerParameter	X	X	
VCS_SetPositionMust	X	X	X
VCS_SetPositionProfile	X	X	X
VCS_SetQuickStopState	X	X	X
VCS_SetState	X	X	X
VCS_SetStepDirectionParameter		X	
VCS_SetVelocityMust	X	X	X
VCS_SetVelocityProfile	X	X	X
VCS_StartIpmTrajectory		X	
VCS_StopHoming	X	X	X
VCS_StopIpmTrajectory		X	
VCS_WaitForHomingAttained	X	X	X
VCS_WaitForTargetReached	X	X	X

Table 11-49 Hardware and their supported functions – Operation functions

Data Recording Functions

Designation	EPOS	EPOS2	EPOS4
VCS_ActivateChannel	X	X	X
VCS_DeactivateAllChannels	X	X	X
VCS_DisableAllTriggers	X	X	X
VCS_EnableTrigger	X	X	X
VCS_ExportChannelDataToFile	X	X	X
VCS_ExtractChannelDataVector	X	X	X
VCS_ForceTrigger	X	X	X
VCS_GetRecorderParameter	X	X	X
VCS_IsRecorderRunning	X	X	X
VCS_IsRecorderTriggered	X	X	X
VCS_ReadChannelDataVector	X	X	X
VCS_ReadChannelVectorSize	X	X	X
VCS_ReadDataBuffer	X	X	X
VCS_SetRecorderParameter	X	X	X
VCS_ShowChannelDataDlg	X	X	X
VCS_StartRecorder	X	X	X
VCS_StopRecorder	X	X	X

Table 11-50 Hardware and their supported functions – Data recording functions

Low Layer Functions

Designation	EPOS	EPOS2	EPOS4
VCS_ReadCANFrame	X	X	
VCS_RequestCANFrame	X	X	
VCS_SendCANFrame	X	X	
VCS_SendNMTService	X	X	X

Table 11-51 Hardware and their supported functions – Low layer functions

maxon

Appendix B — Function Groups Overview

maxon

LIST OF FIGURES

maxon

LIST OF TABLES

maxon

••page intentionally left blank••

INDEX

maxon

P

S

V

W

X

••page intentionally left blank••

maxon

CCMC | EPOS Command Library Documentation | Edition 2021-03 | DocID rel9704

maxon motor ag
Brünigstrasse 220
CH-6072 Sachseln

+41 41 666 15 00
www.maxongroup.com