EPOS2 P

Programmable Positioning Controllers

Command Library

epos.maxonmotor.com

Document ID: rel5893

TABLE OF CONTENTS

Function Group Overview
For a detailed overview on function groups see page 12-171.

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.*

EPOS2 P *programmables positioning controllers are considered as partly completed machinery according to EU Directive 2006/42/EC, Article 2, Clause (g) and* ***are intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment****.*
Therefore, you must not put the device into service,*…*

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

••page intentionally left blank••

1 About this Document

We strongly stress the following facts:

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

1.1 Intended Purpose

The present document provides instructions on the implemented functions of the Windows Dynamic-Link Libraries «EposPCmd.dll» and «EposPCmd64.dll», which can be used for EPOS2 P devices.

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

1.2 Target Audience

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

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

1.3 How to use

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

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

Table 1-1 Notations used in this Document

1.4 Symbols and Signs

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

Best Practice
Gives advice on the easiest and best way to proceed.

Material Damage
Points out information particular to potential damage of equipment.

1.5 Sources for additional Information

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

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

Table 1-2 Sources for additional Information

1.6 Trademarks and Brand Names

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

Brand Name	Trademark Owner
Borland C++ Builder™ Borland®	© Borland Software Corporation, USA-Rockville MD
CANopen® CiA®	© CiA CAN in Automation e.V, DE-Nuremberg
LabVIEW™ LabWindows™	© National Instruments Corporation, USA-Austin TX
NI-CAN™ NI-XNET™	© National Instruments Corporation, USA-Austin TX
Visual Basic® Visual C#® Visual C++®	© Microsoft Corporation, USA-Redmond WA
Windows®	© Microsoft Corporation, USA-Redmond WA

Table 1-3 Brand Names and Trademark Owners

1.7 Legal Notice

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

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

1.8 Copyright

© 2016 maxon motor. All rights reserved.

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

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

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

www.maxonmotor.com

••page intentionally left blank••

2 Introduction

2.1 Documentation Structure

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

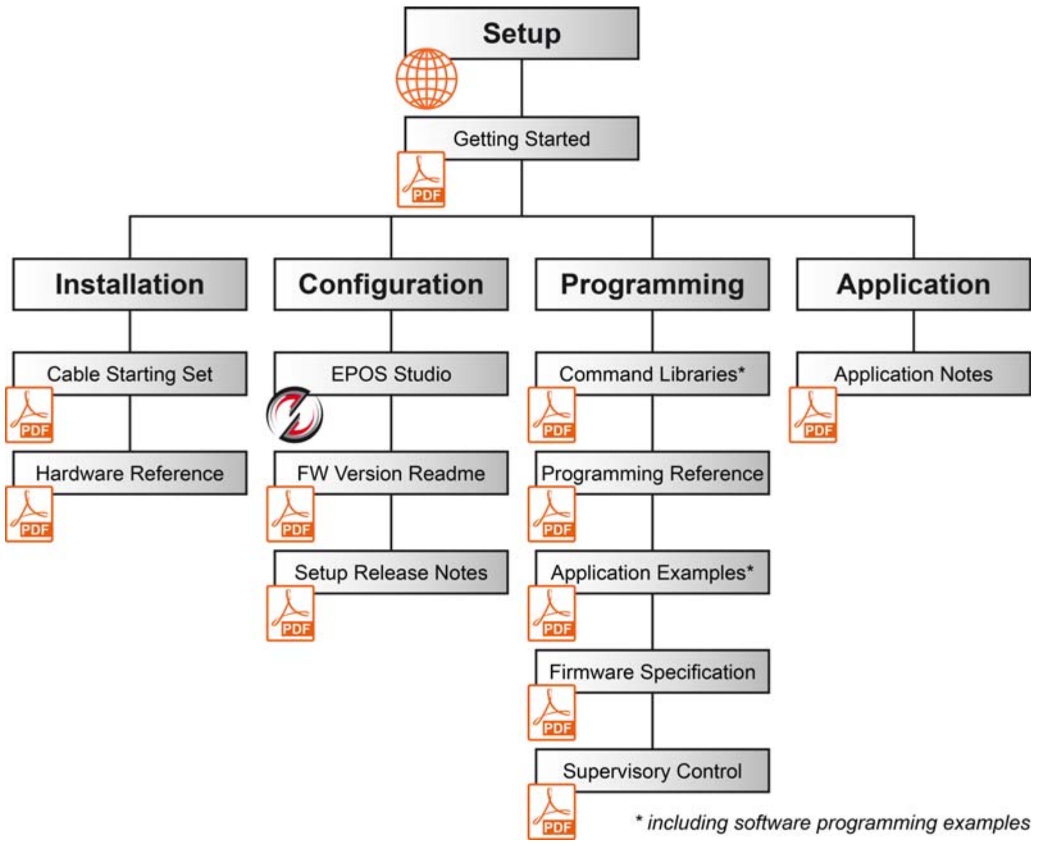


Figure 2-1 Documentation Structure

2.2 General Information

The «EPOS Command Libraries» (EposPCmd.dll and EposPCmd64.dll) 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 operating systems.
The document describes the interfaces between the control software and the libraries. They support maxon motor’s EPOS2 P devices and connected slave devices (e.g. EPOS2), which are connected to a serial RS232 interface or USB port or to a CAN board (➔Table 2-4).

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

Interface		Platform	
		Windows 32-Bit	Windows 64-Bit
RS232		X	X
USB		X	X
CAN Board	IXXAT	X	X
	Kvaser	X	X
	NI	X	X
	Vector	X	X

Table 2-4 Supported Platforms and Interfaces

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

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

Find the latest edition of the present document, as well as additional documentation and software to the EPOS2 P Programmable Positioning Controllers also on the Internet: ➔www.maxonmotor.com

2.3 Products by Third Party Suppliers

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

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

Table 2-5 Third Party Supplier Products

2.4 Communication Structure

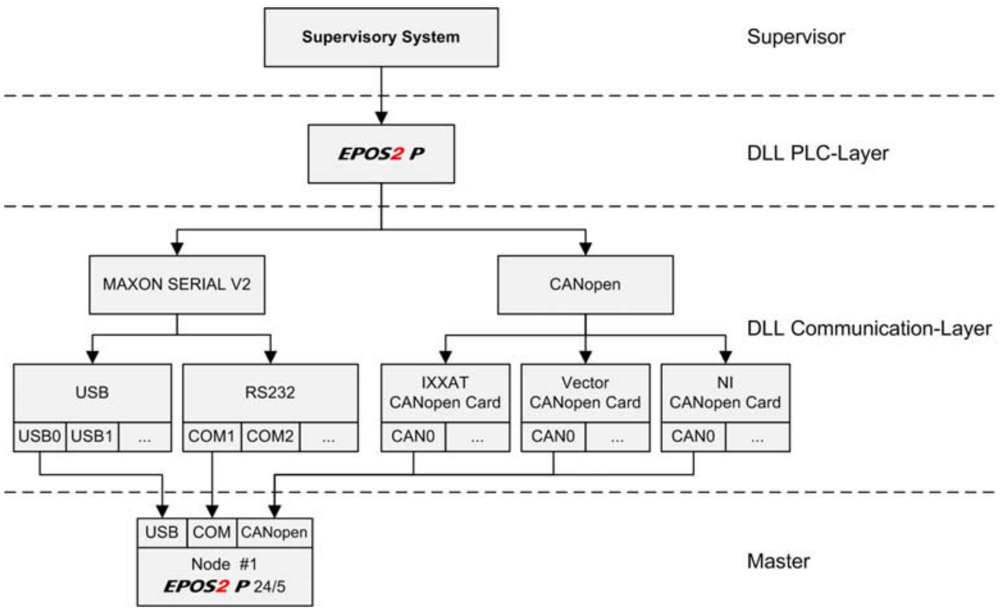


Figure 2-2 Communication Structure (Example)

2.5 Gateway Structure

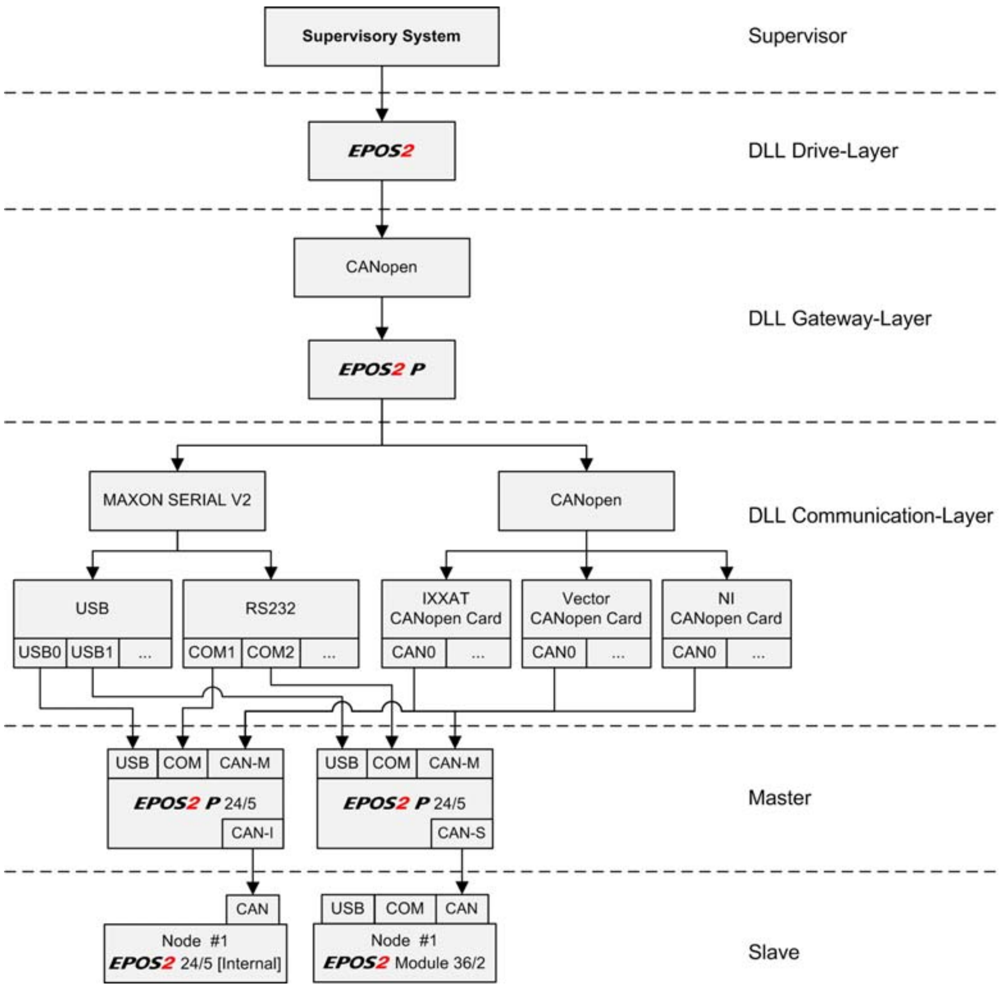


Figure 2-3 Gateway Structure (Example)

2.6 Data Type Definitions

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

Table 2-6 Data Type Definitions

3 Initialization Functions

3.1 Communication

3.1.1 Open Communication

FUNCTION

HANDLE VCM_OpenCommunication(char* ProtocolStackName, char* InterfaceName, char* PortName, DWORD* pErrorCode)

DESCRIPTION

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

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

PARAMETERS

ProtocolStackName	char*	Name of used communication protocol: • 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 functions ➔Open Communication Dialog or ➔Get Interface Name Selection 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”.

3.1.2 Open Communication Dialog

FUNCTION

HANDLE VCM_OpenCommunicationDlg(DWORD* pErrorCode)

DESCRIPTION

VCM_OpenCommunicationDlg recognizes available interfaces capable to operate with EPOS2 P and opens the selected interface for communication.

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

3.1.3 Set Protocol Stack Settings

FUNCTION

BOOL VCM_SetProtocolStackSettings(HANDLE CommunicationHandle, DWORD Baudrate, DWORD Timeout, DWORD* pErrorCode)

DESCRIPTION

VCM_SetProtocolStackSettings writes the communication parameters. For exact values on available baud rates, use function ➔*Get Baud Rate Selection*.

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

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.1.4 Get Protocol Stack Settings

FUNCTION

BOOL VCM_GetProtocolStackSettings(HANDLE CommunicationHandle, DWORD* pBaudrate, DWORD* pTimeout, DWORD* pErrorCode)

DESCRIPTION

VCM_GetProtocolStackSettings returns the baud rate and timeout communication parameters.

PARAMETERS

CommunicationHandle	HANDLE	Handle for communication port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.1.5 Find Communication Settings

FUNCTION

BOOL VCM_FindCommunicationSettings(HANDLE* pPlcOrDriveHandle, char* pDeviceName, char* pProtocolStackName, char* pInterfaceName, char* pPortName, WORD SizeName, HANDLE* pCommunicationHandle, int DialogMode, DWORD* pErrorCode)

DESCRIPTION

VCM_FindCommunicationSettings searches the communication setting parameters. Parameters can be defined to accelerate the process.

PARAMETERS

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

pPlcOrDriveHandle	HANDLE*	Handle from found device (PLC or drive)
pDeviceName	char*	Found device name
pProtocolStackName	char*	Found protocol stack name
pInterfaceName	char*	Found interface name
pPortName	char*	Found port name
pCommunicationHandle	HANDLE*	Handle from found communication port
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.1.6 Close All Communication

FUNCTION

BOOL VCM_CloseAllCommunication(DWORD* pErrorCode)

DESCRIPTION

VCM_CloseAllCommunication closes all opened ports and releases them for other applications.

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.1.7 Close Communication

FUNCTION

BOOL VCM_CloseCommunication(HANDLE CommunicationHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_CloseCommunication closes the port and releases it for other applications.

PARAMETERS

CommunicationHandle	HANDLE	Handle for communication port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.2 Gateway

3.2.1 Open Gateway

FUNCTION

HANDLE VCM_OpenGateway(char* GatewayProtocolStackName, char* GatewayDeviceName, HANDLE CommunicationHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_OpenGateway opens the gateway port for sending and receiving drive commands.

For correct designations on GatewayProtocolStackName and GatewayDeviceName, use the functions ➔*Get Gateway Protocol Stack Name Selection* and ➔*Get Gateway Name Selection*.

The function is used to connect the gateway port with the communication port.

PARAMETERS

GatewayProtocolStackName	char*	Name of used gateway communication protocol: • CANopen
GatewayDeviceName	char*	Name of used gateway device interface: • EPOS2 P
CommunicationHandle	HANDLE	Handle for communication port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

3.2.2 Open Gateway Dialog

FUNCTION

HANDLE VCM_OpenGatewayDlg(DWORD* pErrorCode)

DESCRIPTION

VCM_OpenGatewayDlg registers available gateway interfaces the device can be operated with, and opens the selected gateway interface for communication.

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

3.2.3 Set Gateway Settings

FUNCTION

BOOL VCM_SetGatewaySettings(HANDLE GatewayHandle, BYTE NodeId, BYTE RemoteNetworkId, DWORD* pErrorCode) . . . .

DESCRIPTION

VCM_SetGatewaySettings writes the gateway communication settings.

PARAMETERS

GatewayHandle	HANDLE	Handle for gateway port access
NodeId	BYTE	Node identification from the gateway
RemoteNetworkId	BYTE	Network identification from the network 1: internal 2: external

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.2.4 Get Gateway Settings

FUNCTION

BOOL VCM_GetGatewaySettings(HANDLE GatewayHandle, BYTE* pNodeId, BYTE* pRemoteNetworkId, DWORD* pErrorCode)

DESCRIPTION

VCM_GetGatewaySettings returns the gateway communication settings.

PARAMETERS

GatewayHandle	HANDLE	Handle for gateway port access

RETURN PARAMETERS

pNodeId	BYTE*	Node identification from the gateway
pRemoteNetworkId	BYTE*	Network identification from the network 1: internal 2: external
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.2.5 Find Gateway Communication Settings

FUNCTION

BOOL VCM_FindGatewayCommunicationSettings(HANDLE* pGatewayHandle, char* pGatewayProtocolStackName, char* pGatewayName, char* pProtocolStackName, char* pInterfaceName, char* pPortName, WORD SizeName, HANDLE* pCommunicationHandle, int DialogMode, DWORD* pErrorCode)

DESCRIPTION

VCM_FindGatewayCommunicationSettings searches the gateway communication setting parameters.

PARAMETERS

pGatewayProtocolStackName	char*	Gateway protocol stack name
pGatewayName	char*	Gateway name
pProtocolStackName	char*	Protocol stack name
pInterfaceName	char*	Interface name
pPortName	char*	Port name
SizeName	WORD	Reserved memory size for return parameters
pCommunicationHandle	HANDLE*	Handle from the opened communication interface
DialogMode	int	0: Show progress dialog 1: Show progress and confirmation dialog 2: Show confirmation dialog 3: Don't show any dialog

RETURN PARAMETERS

pGatewayHandle	HANDLE*	Handle from the found gateway
pGatewayProtocolStackName	char*	Gateway protocol stack name
pGatewayName	char*	Found gateway name
pProtocolStackName	char*	Found protocol stack name
pInterfaceName	char*	Found interface name
pPortName	char*	Found port name
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.2.6 Close All Gateways

FUNCTION

BOOL VCM_CloseAllGateways(HANDLE CommunicationHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_CloseAllGateways closes all opened gateway ports and releases them for other applications.

PARAMETERS

CommunicationHandle	HANDLE	Handle for gateway port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.2.7 Close Gateway

FUNCTION

BOOL VCM_CloseGateway(HANDLE GatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_CloseGateway closes the selected gateway port and releases it for other applications.

PARAMETERS

GatewayHandle	HANDLE	Handle for gateway port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3 Help

3.3.1 Library Information

3.3.1.1 Get Driver Error Info

FUNCTION

BOOL VCM_GetDriverErrorInfo(DWORD ErrorCodeValue, char* pErrorInfo, WORD MaxStrSize)

DESCRIPTION

VCM_GetDriverErrorInfo returns the error information on the executed function from a received error code. It returns communication and library errors (but not device error descriptions). For error codes ➔chapter “9 Error Overview” on page 9-153.

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”

3.3.1.2 Get Driver Info

FUNCTION

BOOL VCM_GetDriverInfo(char* pLibraryName, WORD MaxStrNameSize, char* pLibraryVersion, WORD MaxStrVersionSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDriverInfo returns the name and version from the “EPOS2 P 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”

3.3.2 Driver Information

3.3.2.1 Get Protocol Stack Name Selection

FUNCTION

BOOL VCM_GetProtocolStackNameSelection(BOOL StartOfSelection, char* pProtocolStackNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetProtocolStackNameSelection returns all available protocol stack names. For programming example ➔page 3-25.

PARAMETERS

StartOfSelection	BOOL	1: Get first selection string 0: Get next selection string
MaxStrSize	WORD	Reserved memory size for the protocol stack name

RETURN PARAMETERS

pProtocolStackNameSel	char*	Pointer to available protocol stack name
pEndOfSelection	BOOL*	1: No more string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.2 Get Interface Name Selection

FUNCTION

BOOL VCM_GetInterfaceNameSelection(char* ProtocolStackName, BOOL StartOfSelection, char* pInterfaceNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetInterfaceNameSelection returns all available interface names. For programming example ➔page 3-25.

PARAMETERS

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*	Interface name
pEndOfSelection	BOOL*	1: No more selection string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.3 Get Port Name Selection

FUNCTION

BOOL VCM_GetPortNameSelection(char* ProtocolStackName, char* InterfaceName, BOOL StartOf-Selection, char* pPortSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPortNameSelection returns all available port names. For programming example ➔page 3-25.

PARAMETERS

ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
StartOfSelection	BOOL	1: Get first selection string 0: Get next selection string
MaxStrSize	WORD	Reserved memory size for the port name

RETURN PARAMETERS

pPortSel	char*	Port name
pEndOfSelection	BOOL*	1: No more string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.4 Get Baud Rate Selection

FUNCTION

BOOL VCM_GetBaudrateSelection(char* ProtocolStackName, char* InterfaceName, char* PortName, BOOL StartOfSelection, DWORD* pBaudrateSel, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetBaudrateSelection returns all available baud rates for the connected port. For programming example ➔page 3-25.

PARAMETERS

ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
PortName	char*	Port name
StartOfSelection	BOOL	1: Get first selection value 0: Get next selection value

RETURN PARAMETERS

pBaudrateSel	DWORD*	Pointer to baud rate [Bit/s]
pEndOfSelection	BOOL*	1: No more value available 0: More value available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.5 Get Gateway Protocol Stack Name Selection

FUNCTION

BOOL VCM_GetGatewayProtocolStackNameSelection(BOOL StartOfSelection, char* pGatewayProtocolStackNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetGatewayProtocolStackNameSelection returns all available gateway protocol stack names.
For programming example ➔page 3-25.

PARAMETERS

StartOfSelection	BOOL	1: Get first selection string 0: Get next selection string
MaxStrSize	WORD	Reserved memory size for the gateway protocol stack name

RETURN PARAMETERS

pGatewayProtocolStackNameSel	char*	Gateway protocol stack name
pEndOfSelection	BOOL*	1: No more string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.6 Get Gateway Name Selection

FUNCTION

BOOL VCM_GetGatewayNameSelection(char* GatewayProtocolStackName, BOOL StartOfSelection, char* pGatewayNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetGatewayNameSelection returns all available gateway protocol stack names.

PARAMETERS

GatewayProtocolStackName	char*	Gateway protocol stack name
StartOfSelection	BOOL	1: Get first selection string 0: Get next selection string
MaxStrSize	WORD	Reserved memory size for the gateway name

RETURN PARAMETERS

pGatewayNameSel	char*	Gateway name
pEndOfSelection	BOOL*	1: No more string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.7 Programming Example

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

```
-------------------------------------------------------------------------------------
const WORD MAX_STRING_SIZE = 100;

char* strProtocolStackName[MAX_STRING_SIZE];
BOOL endOfSel;
DWORD errorCode;

//get first protocol stack name
if(VCM_GetProtocolStackNameSelection(TRUE, strProtocolStackName,
MAX_STRING_SIZE, &endOfSel, &errorCode))
{
         //get next protocol stack name (as long as endOfSel == FALSE)
         while(!endOfSel)
         {
                              VCM_GetProtocolStackNameSelection(FALSE, strProtocol-
StackName,
                              MAX_STRING_SIZE, &endOfSel, &errorCode);
         }
}
-------------------------------------------------------------------------------------
```

3.3.2.8 Get Communication Handle

FUNCTION

BOOL VCM_GetCommunicationHandle(char* ProtocolStackName, char* InterfaceName, char* PortName, HANDLE* pCommunicationHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_GetCommunicationHandle returns the communication handle from the opened interface.

PARAMETERS

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

RETURN PARAMETERS

pCommunicationHandle	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”

3.3.2.9 Get Gateway Handle

FUNCTION

BOOL VCM_GetGatewayHandle(char* ProtocolStackName, char* InterfaceName, WORD NodeId, HANDLE* pGatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_GetGatewayHandle returns the gateway handle from the opened gateway interface.

PARAMETERS

ProtocolStackName	char*	Protocol stack name
InterfaceName	char*	Interface name
NodeId	WORD	Node identification

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.10 Get Device Communication Handle

FUNCTION

BOOL VCM_GetDeviceCommunicationHandle(HANDLE PlcOrDriveOrGatewayHandle, HANDLE* pCommunicationHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDeviceCommunicationHandle returns the drive communication handle from the selected interface.

PARAMETERS

PlcOrDriveOrGatewayHandle	HANDLE	Opened PLC, drive, or gateway handle

RETURN PARAMETERS

pCommunicationHandle	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”

3.3.2.11 Get Device Gateway Handle

FUNCTION

BOOL VCM_GetGatewayHandle(HANDLE PlcOrDriveHandle, HANDLE* pGatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_GetGatewayHandle returns the gateway handle from the selected interface.

PARAMETERS

PlcOrDriveHandle	HANDLE	Opened PLC or drive handle

RETURN PARAMETERS

pGatewayHandle	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”

3.3.2.12 Get Protocol Stack Name

FUNCTION

BOOL VCM_GetProtocolStackName(HANDLE CommunicationHandle, char* pProtocolStackName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetProtocolStackName returns the protocol stack name to corresponding handle.

PARAMETERS

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

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.13 Get Interface Name

FUNCTION

BOOL VCM_GetInterfaceName(HANDLE CommunicationHandle, char* pInterfaceName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetInterfaceName returns the interface name to corresponding handle.

PARAMETERS

CommunicationHandle	HANDLE	Handle for communication port access
MaxStrSize	WORD	Reserved memory size for the interface name

RETURN PARAMETERS

pInterfaceName	char*	Pointer to interface name
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.3.2.14 Get Port Name

FUNCTION

BOOL VCM_GetPortName(HANDLE CommunicationHandle, char* pPortName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPortName returns the port name to corresponding handle.

PARAMETERS

CommunicationHandle	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”

3.3.2.15 Get Gateway Name

FUNCTION

BOOL VCM_GetGatewayName(HANDLE GatewayHandle, char* pGatewayName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetGatewayName returns the gateway name to corresponding handle.

PARAMETERS

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

RETURN PARAMETERS

pGatewayName	char*	Gateway name
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.4 CANopen

Note
The function “Send NMT Service” is no longer operative. ***Use ➔“Send NMT Service Ex” instead****.*

3.4.1 Send CAN Frame

FUNCTION

BOOL VCM_SendCANFrame(HANDLE DriveHandle, WORD CobID, WORD Length, void* pData, DWORD* pErrorCode)

DESCRIPTION

VCM_SendCANFrame sends a general CAN frame to the CAN bus.

PARAMETERS

DriveHandle	HANDLE	Handle for port access
CobID	WORD	CAN frame 11-bit identifier
Length	WORD	CAN frame length
pData	void*	CAN frame data

RETURN PARAMETERS

pData	void*	CAN frame data
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.4.2 Read CAN Frame

FUNCTION

BOOL VCM_ReadCANFrame(HANDLE DriveHandle, WORD CobID, WORD Length, void* pData, DWORD Timeout, DWORD* p ErrorCode)

DESCRIPTION

VCM_ReadCANFrame reads a general CAN frame from the CAN bus.

PARAMETERS

DriveHandle	HANDLE	Handle for port access
CobID	WORD	CAN frame 11-bit identifier
Length	WORD	CAN frame length
Timeout	WORD	Max. wait time

RETURN PARAMETERS

pData	void*	CAN frame data
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.4.3 Request CAN Frame

FUNCTION

BOOL VCM_RequestCANFrame(HANDLE DriveHandle, WORD CobID, WORD Length, void* pData, DWORD* pErrorCode)

DESCRIPTION

VCM_RequestCANFrame requests a general CAN frame from the CAN bus using Remote Transmit Request (RTR).

PARAMETERS

DriveHandle	HANDLE	Handle for port access
CobID	WORD	CAN frame 11-bit identifier
Length	WORD	CAN frame length

RETURN PARAMETERS

pData	void*	CAN frame data
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

3.4.4 Send NMT Service Ex

FUNCTION

BOOL VCM_SendNMTServiceEx(HANDLE DriveHandle, WORD NodeId, WORD CommandSpecifier, DWORD* pErrorCode)

DESCRIPTION

VCM_SendNMTServiceEx sends a NMT protocol from a master to a slave/all slaves in a network. Command is without acknowledge.

PARAMETERS

DriveHandle	HANDLE	Handle for port access
NodeId	WORD	1…127: NMT slave with given Node ID 0: All NMT slaves
CommandSpecifier	WORD	NMT service (➔Table 3-7)

RETURN PARAMETERS

pData	void*	CAN frame data
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 3-7 Command Specifier

••page intentionally left blank••

4 PLC Functions

4.1 Initialization

4.1.1 Open PLC

FUNCTION

HANDLE VCM_OpenPlc(char* DeviceName, HANDLE CommunicationOrGatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_OpenPlc opens the interface for sending and receiving commands to/from a PLC device.

For correct designations on DeviceName use the function ➔*Get PLC Device Name Selection*.

PARAMETERS

DeviceName	char*	Name of used device: EPOS2 P
CommunicationOrGatewayHandle	HANDLE	Handle from opened communication or gateway port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	HANDLE	Handle for PLC device access. Nonzero if successful; otherwise “0”.

4.1.2 Open PLC Dialog

FUNCTION

HANDLE VCM_OpenPlcDlg(CommunicationOrGatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_OpenPlcDlg registers available PLC devices the library can be operated with. And it opens the selected interface for communication.

PARAMETERS

CommunicationOrGatewayHandle	HANDLE	Handle from opened communication or gateway port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	HANDLE	Handle for PLC device access. Nonzero if successful; otherwise “0”.

4.1.3 Set PLC Settings

FUNCTION

BOOL VCM_SetPlcSettings(HANDLE PlcHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCM_SetPlcSettings writes the PLC device parameter.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC device access
NodeId	WORD	Node identification

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

4.1.4 Get PLC Settings

FUNCTION

BOOL VCM_GetPlcSettings(HANDLE PlcHandle, WORD* pNodeId, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPlcSettings returns the PLC device parameter.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC device access

RETURN PARAMETERS

NodeId	WORD*	Node identification
pErrorCode	DWORD*	Error information on the executed function

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

4.1.5 Find PLC Communication Settings

FUNCTION

BOOL VCM_FindPlcCommunicationSettings(HANDLE* pPlcHandle, HANDLE CommunicationOrGatewayHandle, char* pDeviceName, WORD SizeName, int DialogMode, DWORD* pErrorCode)

DESCRIPTION

VCM_FindPlcCommunicationSettings searches the PLC device communication settings.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle from the opened communication or gateway interface
pDeviceName	char*	Device 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

pPlcHandle	HANDLE	Handle from the found PLC device
pDeviceName	char*	Device name
pErrorCode	DWORD*	Error information on the executed function

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

4.1.6 Close All PLC

FUNCTION

BOOL VCM_CloseAllPlc(DWORD* pErrorCode)

DESCRIPTION

VCM_CloseAllPlc closes all opened PLC interfaces.

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

4.1.7 Close PLC

FUNCTION

BOOL VCM_ClosePlc(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_ClosePlc closes the selected PLC interface.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

4.2 Help

4.2.1 Get Version

FUNCTION

BOOL VCM_GetVersion(HANDLE PlcHandle, WORD* pHardwareVersion, WORD* pSoftwareVersion, WORD* pApplicationNumber, WORD* pApplicationVersion, DWORD* pErrorCode)

DESCRIPTION

VCM_GetVersion returns the version information (e. g. software version or hardware version).

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.2 Get Error Info

FUNCTION

BOOL VCM_GetErrorInfo(DWORD ErrorCodeValue, char* pErrorInfo, WORD MaxStrSize)

DESCRIPTION

VCM_GetErrorInfo returns the error information on the executed function from a received error code. It returns communication and library errors (but not device error descriptions). For error codes ➔chapter “9 Error Overview” on page 9-153.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	char*	Error string

Return Value	BOOL	Nonzero if error information found “0” if no error information string available

4.2.3 Driver Information

4.2.3.1 Get PLC Device Name Selection

FUNCTION

BOOL VCM_GetPlcDeviceNameSelection(BOOL StartOfSelection, char* pPlcDeviceNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPlcDeviceNameSelection returns all available device names. For programming example ➔page 3-25.

PARAMETERS

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

RETURN PARAMETERS

pPlcDeviceNameSel	char*	PLC device name
pEndOfSelection	BOOL*	1: No more selection string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.3.2 Get PLC Handle

FUNCTION

BOOL VCM_GetPlcHandle(HANDLE CommunicationOrGatewayHandle, char* DeviceName, BYE NodeId, HANDLE* pPlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPlcHandle returns the handle from the opened PLC interface.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle from the opened communication or gateway interface
DeviceName	char*	Device name
NodeId	BYTE	Node identification

RETURN PARAMETERS

pPlcHandle	HANDLE*	Handle for PLC interface access, if parameters are correct; otherwise “0”
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.2.3.3 Get Device Name

FUNCTION

BOOL VCM_GetDeviceName(HANDLE PlcHandle, char* pDeviceName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDeviceName returns the device name to corresponding handle.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
MaxStrSize	WORD	Reserved memory size for the device name

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.3 Configuration

4.3.1 General

4.3.1.1 Import Parameter

FUNCTION

BOOL VCM_ImportParameter(HANDLE PlcHandle, char* ParameterFileName, BOOL ShowDlg, BOOL ShowMsg, DWORD* pErrorCode)

DESCRIPTION

VCM_ImportParameter writes parameters from a file to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ParameterFileName	char*	Path to corresponding file
ShowDlg	BOOL	Dialog is shown
ShowMsg	BOOL	Message boxes are activated

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.3.1.2 Export Parameter

FUNCTION

BOOL VCM_ExportParameter(HANDLE PlcHandle, char* ParameterFileName, char* BinaryFile, char* UserID, char* Comment, BOOL ShowDlg, BOOL ShowMsg, DWORD* pErrorCode)

DESCRIPTION

VCM_ExportParameter reads all device parameters and writes them to the file.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ParameterFileName	char*	Path to corresponding file
BinaryFile	char*	Firmware file of the connected device
UserID	char*	User name
Comment	char*	Comment
ShowDlg	BOOL	Dialog is shown
ShowMsg	BOOL	Message box are activated

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.3.1.3 Set Object

FUNCTION

BOOL VCM_SetObject(HANDLE PlcHandle, WORD ObjectIndex, BYTE ObjectSubIndex, void* pData, DWORD NbOfBytesToWrite, DWORD* pNbOfBytesWritten, DWORD* pErrorCode)

DESCRIPTION

VCM_SetObject writes an object value at the given index and subindex.

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

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
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”

4.3.1.4 Get Object

FUNCTION

BOOL VCM_GetObject(HANDLE PlcHandle, WORD ObjectIndex, BYTE ObjectSubIndex, void* pData, DWORD NbOfBytesToRead, DWORD* pNbOfBytesRead, DWORD* pErrorCode)

DESCRIPTION

VCM_GetObject reads an object value at the given index and subindex.

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

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
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”

4.3.1.5 Restore

FUNCTION

BOOL VCM_Restore(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_Restore restores all default parameters.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.3.1.6 Store

FUNCTION

BOOL VCM_Store(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_Store stores all parameters.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.3.2 PLC Configuration

4.3.2.1 Set Bootup Behavior

FUNCTION

BOOL VCM_SetBootupBehavior(HANDLE PlcHandle, WORD ProgramControl, DWORD* pErrorCode)

DESCRIPTION

VCM_SetBootupBehavior writes the object to control the start of a stored application program.

RETURN PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access	
ProgramControl	WORD	Startup Program Control (➔Table 4-8)	0x2F51-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Handle for port access Nonzero if successful; otherwise “0”

Description	Value	Name
Stopped	0	BOH_NOT_STARTED_AT_BOOTUP
Coldstart	1	BOH_COLD_START_AT_BOOTUP
Warmstart	2	BOH_WARM_START_AT_BOOTUP
Hotstart	3	BOH_HOT_START_AT_BOOTUP

Table 4-8 Startup Program Control

4.3.2.2 Get Bootup Behavior

FUNCTION

BOOL VCM_GetBootupBehavior(HANDLE PlcHandle, WORD* pProgramControl, DWORD* pEr-rorCode)

DESCRIPTION

VCM_GetBootupBehavior reads the object to control the start of a stored application program.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pProgramControl	WORD*	Startup Program Control (➔Table 4-8)	0x2F51-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4 Program Control

4.4.1 Download Program

FUNCTION

BOOL VCM_DownloadProgram(HANDLE PlcHandle, char* ProjectPathFileName, DWORD* pErrorCode)

DESCRIPTION

VCM_DownloadProgram downloads the program code to the PLC device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProjectPathFileName	char*	Path of corresponding file

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.2 Program Status

4.4.2.1 Coldstart PLC

FUNCTION

BOOL VCM_ColdstartPlc(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_ColdstartPlc writes program control state “cold start” to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.2.2 Warmstart PLC

FUNCTION

BOOL VCM_WarmstartPlc(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_WarmstartPlc writes program control state “warm start” to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.2.3 Hotstart PLC

FUNCTION

BOOL VCM_HotstartPlc(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_HotstartPlc writes program control state “hot start” to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.2.4 Stop PLC

FUNCTION

BOOL VCM_StopPlc(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_StopPlc stops the PLC program.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.2.5 Clear PLC Program

FUNCTION

BOOL VCM_ClearPlcProgram(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_ClearPlcProgram erases the PLC program.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.2.6 Get PLC Status

FUNCTION

BOOL VCM_GetPlcStatus(HANDLE PlcHandle, BOOL* pIsRunning, BOOL* pIsProgramAvailable, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPlcStatus reads the program control state of the PLC program.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pIsRunning	BOOL*	Program started	0x1F51-01
pIsProgramAvailable	BOOL*	Program available	
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.3 Error Handling

4.4.3.1 Get Number of Device Error

FUNCTION

BOOL VCM_GetNbOfDeviceError(HANDLE PlcHandle, BYTE* pNbDeviceError, DWORD* pErrorCode)

DESCRIPTION

VCM_GetNbOfDeviceError returns the number of actual errors that are recorded.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access	

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.3.2 Get Device Error Code

FUNCTION

BOOL VCM_GetDeviceErrorCode(HANDLE PlcHandle, BYTE ErrorNumber, DWORD* pDeviceErrorCode, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDeviceErrorCode returns the error code of the selected error number.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access	
ErrorNumber	BYTE	Number (object subindex) of device error (≥1)	0x1003-0x

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.3.3 Clear Device Errors

FUNCTION

BOOL VCM_ClearDeviceErrors(HANDLE PlcHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_ClearDeviceErrors deletes the error history.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.3.4 Programming Example

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

```
------------------------------------------------------------------------------------
//Global parameters
HANDLE PlcHandle = 1; //handle from opened device

//Functional parameters
BYTE nbOfDeviceError = 0;     //number of actual errors
DWORD functionErrorCode = 0;  //error code from function
DWORD deviceErrorCode = 0;    //error code from device

//get number of device errors
if(VCM_GetNbOfDeviceError(PlcHandle, &nbOfDeviceError, &functionErrorCode))
{
         //read device error code
         for(BYTE errorNumber = 1; errorNumber <= nbOfDeviceError; errorNumber++)
         {
                              if(!VCM_GetDeviceErrorCode(PlcHandle, errorNumber,
                              &deviceErrorCode, &functionErrorCode))
                              {
                              break;
                              }
         }
}
------------------------------------------------------------------------------------
```

4.4.4 Program Variable Access

4.4.4.1 Init Variable Table

FUNCTION

BOOL VCM_InitVariableTable(HANDLE PlcHandle, char* ProjectPathName, DWORD* pErrorCode)

DESCRIPTION

VCM_InitVariableTable writes the maxon variable table from a file to the device.

File Location
You may find the file “VariableInfo.xml”' in the OpenPCS project folder under “GEN\Resource”.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProjectPathName	char*	Project directory

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.4.2 Set Variable

FUNCTION

BOOL VCM_SetVariable(HANDLE PlcHandle, char* VariableName, void* pData, DWORD NbOfBytes-ToWrite, DWORD* pNbOfBytesWritten, DWORD* pErrorCode)

DESCRIPTION

VCM_SetVariable writes a PLC variable to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
VariableName	char*	Variable name declaration (e.g. COUNTER.COUNT)
pData	void*	Variable data
NbOfBytesToWrite	DWORD	Variable length to write (number of bytes)

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.4.3 Get Variable

FUNCTION

BOOL VCM_GetVariable(HANDLE PlcHandle, char* VariableName, void* pData, DWORD NbOfBytesToRead, DWORD* pNbOfBytesRead, DWORD* pErrorCode)

DESCRIPTION

VCM_GetVariable reads a PLC variable from the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
VariableName	char*	Variable name declaration (e.g. COUNTER.COUNT)
NbOfBytesToRead	DWORD	Variable length to read (number of bytes)

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.5 Process Input/Output Access

4.4.5.1 Set Process Input

FUNCTION

BOOL VCM_SetProcessInput(HANDLE PlcHandle, WORD ProcessInputType, BYTE ElementNumber, void* pData, DWORD NbOfBytesToWrite, DWORD* pNbOfBytesWritten, DWORD* pErrorCode)

DESCRIPTION

VCM_SetProcessInput writes process input variables to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProcessInputType	WORD	Type of the process input element (➔Table 4-9)
ElementNumber	BYTE	Number of the element
pData	void*	Variable data
NbOfBytesToWrite	DWORD	Variable length to write (number of bytes)

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Signed 8-Bit Object	0	PIT_PROCESS_INPUT_INT8
Unsigned 8-Bit Object	1	PIT_PROCESS_INPUT_UINT8
Signed 16 Bit Object	2	PIT_PROCESS_INPUT_INT16
Unsigned 16 Bit Object	3	PIT_PROCESS_INPUT_UINT16
Signed 32 Bit Object	4	PIT_PROCESS_INPUT_INT32
Unsigned 32 Bit Object	5	PIT_PROCESS_INPUT_UINT32
Signed 64 Bit Object	6	PIT_PROCESS_INPUT_INT64
Unsigned 64 Bit Object	7	PIT_PROCESS_INPUT_UINT64

Table 4-9 Process Input Type

4.4.5.2 Get Process Output

FUNCTION

BOOL VCM_GetProcessOutput(HANDLE PlcHandle, WORD ProcessOutputType, BYTE ElementNumber, void* pData, DWORD NbOfBytesToRead, DWORD* pNbOfBytesRead, DWORD* pErrorCode)

DESCRIPTION

VCM_GetProcessOutput reads process output variables from the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProcessOutputType	WORD	Type of the process output element (➔Table 4-10)
ElementNumber	BYTE	Number of the element
NbOfBytesToRead	DWORD	Variable length to read (number of bytes)

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Signed 8-Bit Object	0	PIT_PROCESS_OUTPUT_INT8
Unsigned 8-Bit Object	1	PIT_PROCESS_OUTPUT_UINT8
Signed 16 Bit Object	2	PIT_PROCESS_OUTPUT_INT16
Unsigned 16 Bit Object	3	PIT_PROCESS_OUTPUT_UINT16
Signed 32 Bit Object	4	PIT_PROCESS_OUTPUT_INT32
Unsigned 32 Bit Object	5	PIT_PROCESS_OUTPUT_UINT32
Signed 64 Bit Object	6	PIT_PROCESS_OUTPUT_INT64
Unsigned 64 Bit Object	7	PIT_PROCESS_OUTPUT_UINT64

Table 4-10 Process Output Type

4.4.5.3 Set Process Input Bit

FUNCTION

BOOL VCM_SetProcessInputBit(HANDLE PlcHandle, WORD ProcessInputType, BYTE ElementNumber, BYTE BitNumber, BYTE BitState, DWORD* pErrorCode)

DESCRIPTION

VCM_SetProcessInputBit writes the process input bit to the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProcessOutputType	WORD	Type of the process output element (➔Table 4-10)
ElementNumber	BYTE	Number of the element
BitNumber	BYTE	Bit number of the element
BitState	BYTE	Variable state

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.5.4 Get Process Output Bit

FUNCTION

BOOL VCM_GetProcessOutputBit(HANDLE PlcHandle, WORD ProcessOutputType, BYTE ElementNumber, BYTE BitNumber, BYTE* pBitState, DWORD* pErrorCode)

DESCRIPTION

VCM_GetProcessOutputBit reads the process output bit from the device.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProcessInputType	WORD	Type of the process input element (➔Table 4-9)
ElementNumber	BYTE	Number of the element
BitNumber	BYTE	Bit number of the element

RETURN PARAMETERS

pBitState	BYTE*	Variable state
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.6 Process Image Access

4.4.6.1 Set Process Input Image

FUNCTION

BOOL VCM_SetProcessInputImage(HANDLE PlcHandle, BYTE* ProcessInputImage, DWORD* pErrorCode)

DESCRIPTION

VCM_SetProcessInputImage writes the complete process image.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access
ProcessInputImage	BYTE*	Process image

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

4.4.6.2 Get Process Output Image

FUNCTION

BOOL VCM_GetProcessOutputImage(HANDLE PlcHandle, BYTE* pProcessOutputImage, DWORD* pErrorCode)

DESCRIPTION

VCM_GetProcessOutputImage reads the complete process image.

PARAMETERS

PlcHandle	HANDLE	Handle for PLC interface access

RETURN PARAMETERS

ProcessInputImage	BYTE*	Process image
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5 Drive Functions – Common

5.1 Initialization

5.1.1 Open Drive

FUNCTION

HANDLE VCM_OpenDrive(char* DeviceName, HANDLE GatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_OpenDrive opens the interface for sending and receiving commands to/from a drive device.

For correct designation on DeviceName use the function ➔*Get Drive Device Name Selection*.

PARAMETERS

DeviceName	char*	Name of used device: EPOS2
GatewayHandle	HANDLE	Handle from opened gateway port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

5.1.2 Open Drive Dialog

FUNCTION

HANDLE VCM_OpenDriveDlg(GatewayHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_OpenDriveDlg registers available drive devices the library can be operated with. And it opens the selected interface for communication.

PARAMETERS

GatewayHandle	HANDLE	Handle from opened gateway port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

5.1.3 Set Drive Settings

FUNCTION

BOOL VCM_SetDriveSettings(HANDLE DriveHandle, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCM_SetDriveSettings writes the drive device parameter.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access
NodeId	WORD	Node identification

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

5.1.4 Get Drive Settings

FUNCTION

BOOL VCM_GetDriveSettings(HANDLE DriveHandle, WORD* pNodeId, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDriveSettings returns the drive device parameter.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access

RETURN PARAMETERS

pNodeId	WORD*	Node identification
pErrorCode	DWORD*	Error information on the executed function

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

5.1.5 Find Drive Communication Settings

FUNCTION

BOOL VCM_FindDriveCommunicationSettings(HANDLE* pDriveHandle, HANDLE CommunicationOr-GatewayHandle, char* pDeviceName, WORD SizeName, int DialogMode, DWORD* pErrorCode)

DESCRIPTION

VCM_FindPlcCommunicationSettings searches the drive device communication settings.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle from the opened communication or gateway interface
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

pDriveHandle	HANDLE*	Handle from the found drive device
pDeviceName	char*	Device name
pErrorCode	DWORD*	Error information on the executed function

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

5.1.6 Close All Devices

FUNCTION

BOOL VCM_CloseAllDrives(DWORD* pErrorCode)

DESCRIPTION

VCM_CloseAllDrives closes all opened drive interfaces.

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

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

5.1.7 Close Drive

FUNCTION

BOOL VCM_CloseDrive(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_CloseDrive closes the selected drive interface.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Handle for PLC device access. Nonzero if successful; otherwise “0”.

5.2 Help

5.2.1 Get Version

FUNCTION

BOOL VCM_GetVersion(HANDLE DriveHandle, WORD* pHardwareVersion, WORD* pSoftwareVersion, WORD* pApplicationNumber, WORD* pApplicationVersion, DWORD* pErrorCode)

DESCRIPTION

VCM_GetVersion returns the version information (e. g. software version or hardware version).

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.2 Get Error Info

FUNCTION

BOOL VCM_GetErrorInfo(DWORD ErrorCodeValue, char* pErrorInfo, WORD MaxStrSize)

DESCRIPTION

VCM_GetErrorInfo returns the error information on the executed function from a received error code. It returns communication and library errors (but not device error descriptions). For error codes ➔chapter “9 Error Overview” on page 9-153.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	char*	Error string

Return Value	BOOL	Nonzero if error information found “0” if no error information string available

5.2.3 Driver Information

5.2.3.1 Get Drive Device Name Selection

FUNCTION

BOOL VCM_GetDriveDeviceNameSelection(BOOL StartOfSelection, char* pDriveDeviceNameSel, WORD MaxStrSize, BOOL* pEndOfSelection, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDriveDeviceNameSelection returns all available device names. For programming example ➔page 9-153.

PARAMETERS

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

RETURN PARAMETERS

pDriveDeviceNameSel	char*	Drive device name
pEndOfSelection	BOOL*	1: No more selection string available 0: More string available
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.3.2 Get Drive Handle

FUNCTION

BOOL VCM_GetDeviceName(HANDLE DriveHandle, char* pDeviceName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDeviceName returns the device name to corresponding handle.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle from the opened communication or gateway interface
DeviceName	char*	Device name
NodeId	BYTE	Node identification

RETURN PARAMETERS

pDriveHandle	HANDLE*	Handle for drive interface access, if parameters are correct; otherwise “0”
pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.2.3.3 Get Device Name

FUNCTION

BOOL VCM_GetDeviceName(HANDLE DriveHandle, char* pDeviceName, WORD MaxStrSize, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDeviceName returns the device name to corresponding handle.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access
MaxStrSize	WORD	Reserved memory size for the device name

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3 Configuration

5.3.1 General

5.3.1.1 Import Parameter

FUNCTION

BOOL VCM_ImportParameter(HANDLE DriveHandle, char* ParameterFileName, BOOL ShowDlg, BOOL ShowMsg, DWORD* pErrorCode)

DESCRIPTION

VCM_ImportParameter writes parameters from a file to the device.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access
ParameterFileName	char*	Path to corresponding file
ShowDlg	BOOL	Dialog is shown
ShowMsg	BOOL	Message boxes are activated

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.1.2 Export Parameter

FUNCTION

BOOL VCM_ExportParameter(HANDLE DriveHandle, char* ParameterFileName, char* BinaryFile, char* UserID, char* Comment, BOOL ShowDlg, BOOL ShowMsg, DWORD* pErrorCode)

DESCRIPTION

VCM_ExportParameter reads all device parameters and writes them to the file.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access
ParameterFileName	char*	Path to corresponding file
BinaryFile	char*	Firmware file of the connected device
UserID	char*	User name
Comment	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”

5.3.1.3 Set Object

FUNCTION

BOOL VCM_SetObject(HANDLE DriveHandle, WORD ObjectIndex, BYTE ObjectSubIndex, void* pData, DWORD NbOfBytesToWrite, DWORD* pNbOfBytesWritten, DWORD* pErrorCode)

DESCRIPTION

VCM_SetObject writes an object value at the given index and subindex.

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

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access
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”

5.3.1.4 Get Object

FUNCTION

BOOL VCM_GetObject(HANDLE DriveHandle, WORD ObjectIndex, BYTE ObjectSubIndex, void* pData, DWORD NbOfBytesToRead, DWORD* pNbOfBytesRead, DWORD* pErrorCode)

DESCRIPTION

VCM_GetObject reads an object value at the given index and subindex.

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

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access
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”

5.3.1.5 Restore

FUNCTION

BOOL VCM_Restore(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_Restore restores all default parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.1.6 Store

FUNCTION

BOOL VCM_Store(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCM_Store stores all parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for drive interface access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.2 Motor

5.3.2.1 Set Motor Type

FUNCTION

BOOL VCM_SetMotorType(HANDLE DriveHandle, WORD MotorType, DWORD* pErrorCode)

DESCRIPTION

VCM_SetMotorType writes the motor type.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
MotorType	WORD	Type of motor (➔Table 5-11)	0x6402-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

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

Table 5-11 Motor Types

5.3.2.2 Set DC Motor Parameter

FUNCTION

BOOL VCM_SetDcMotorParameter(HANDLE DriveHandle, WORD NominalCurrent, WORD MaxOutputCurrent, WORD ThermalTimeConstant, DWORD* pErrorCode)

DESCRIPTION

VCM_SetDcMotorParameter writes all DC motor parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
NominalCurrent	WORD	Maximal continuous current	0x6410-01
MaxOutputCurrent	WORD	Maximal peak current	0x6410-02
ThermalTimeConstant	WORD	Thermal time constant winding	0x6410-05

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.2.3 Set EC Motor Parameter

FUNCTION

BOOL VCM_SetEcMotorParameter(HANDLE DriveHandle, WORD NominalCurrent, WORD MaxOutputCurrent, WORD ThermalTimeConstant, BYTE NbOfPolePairs, DWORD* pErrorCode)

DESCRIPTION

VCM_SetEcMotorParameter writes all EC motor parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
NominalCurrent	WORD	Maximal continuous current	0x6410-01
MaxOutputCurrent	WORD	Maximal peak current	0x6410-02
ThermalTimeConstant	WORD	Thermal time constant winding	0x6410-05
NbOfPolePairs	BYTE	Number of pole pairs	0x6410-03

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.2.4 Get Motor Type

FUNCTION

BOOL VCM_GetMotorType(HANDLE DriveHandle, WORD* pMotorType, DWORD* pErrorCode)

DESCRIPTION

VCM_GetMotorType reads the motor type.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

MotorType	WORD	Type of motor (➔Table 5-11)	0x6402-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.2.5 Get DC Motor Parameter

FUNCTION

BOOL VCM_GetDcMotorParameter(HANDLE DriveHandle, WORD* pNominalCurrent, WORD* pMaxOutputCurrent, WORD* pThermalTimeConstant, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDcMotorParameter reads all DC motor parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.2.6 Get EC Motor Parameter

FUNCTION

BOOL VCM_GetEcMotorParameter(HANDLE DriveHandle, WORD* pNominalCurrent, WORD* pMaxOutputCurrent, WORD* pThermalTimeConstant, BYTE* pNbOfPolePairs, DWORD* pErrorCode)

DESCRIPTION

VCM_GetEcMotorParameter reads all EC motor parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pNominalCurrent	WORD*	Maximal continuous current	0x6410-01
pMaxOutputCurrent	WORD*	Maximal peak current	0x6410-02
pThermalTimeConstant	WORD*	Thermal time constant winding	0x6410-05
pNbOfPolePairs	WORD*	Number of pole pairs	0x6410-03
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3 Sensor

5.3.3.1 Set Sensor Type

FUNCTION

BOOL VCM_SetSensorType(HANDLE DriveHandle, WORD SensorType, DWORD* pErrorCode)

DESCRIPTION

VCM_SetSensorType writes the sensor type.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
SensorType	WORD	Position Sensor Type (➔Table 5-12)	0x2210-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

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

Table 5-12 Position Sensor Types

5.3.3.2 Set Incremental Encoder Parameter

FUNCTION

BOOL VCM_SetIncEncoderParameter(HANDLE DriveHandle, DWORD EncoderResolution, BOOL InvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCM_SetIncEncoderParameter writes the incremental encoder parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
EncoderResolution	DWORD	Encoder pulse number [pulse per turn	0x2210-01
InvertedPolarity	BOOL	Position sensor polarity	0x2210-04

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3.3 Set Hall Sensor Parameter

Function

BOOL VCM_SetHallSensorParameter(HANDLE DriveHandle, BOOL InvertedPolarity, DWORD* pErrorCode)

Description

VCM_SetHallSensorParameter writes the Hall sensor parameter.

Parameters

DriveHandle	HANDLE	Handle for port access	
InvertedPolarity	BOOL	Position sensor polarity	0x2210-04

Return Parameters

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3.4 Set SSI Absolute Encoder Parameter

Function

BOOL VCM_SetSsiAbsEncoderParameter(HANDLE DriveHandle, WORD DataRate, WORD NbOfMultiTurnDataBits, WORD NbOfSingleTurnDataBits, BOOL InvertedPolarity, DWORD* pErrorCode)

Description

VCM_SetSsiAbsEncoderParameter writes all parameters for SSI absolute encoder.

Parameters

DriveHandle	HANDLE	Handle for port access	
DataRate	WORD	SSI encoder data rate	0x2211-01
NbOfMultiTurnDataBits	WORD	Number of bits multi turn	0x2211-02
NbOfSingleTurnDataBits	WORD	Number of bits single turn	
InvertedPolarity	BOOL	Position sensor polarity	0x2210-04

Return Parameters

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3.5 Get Sensor Type

FUNCTION

BOOL VCM_GetSensorType(HANDLE DriveHandle, WORD* pSensorType, DWORD* pErrorCode)

DESCRIPTION

VCM_GetSensorType reads the sensor type.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	

RETURN PARAMETERS

pSensorType	WORD*	Position sensor type (➔Table 5-12)	0x2210-02
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3.6 Get Incremental Encoder Parameter

FUNCTION

BOOL VCM_GetIncEncoderParameter(HANDLE DriveHandle, DWORD* pEncoderResolution, BOOL* pInvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCM_GetIncEncoderParameter reads the incremental encoder parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3.7 Get Hall Sensor Parameter

FUNCTION

BOOL VCM_GetHallSensorParameter(HANDLE DriveHandle, BOOL* pInvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCM_GetHallSensorParameter reads the Hall sensor parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.3.8 Get SSI Absolute Encoder Parameter

FUNCTION

BOOL VCM_GetSsiAbsEncoderParameter(HANDLE DriveHandle, WORD* pDataRate, WORD* pNbOfMultiTurnDataBits, WORD* pNbOfSingleTurnDataBits, BOOL* pInvertedPolarity, DWORD* pErrorCode)

DESCRIPTION

VCM_GetSsiAbsEncoderParameter reads all parameters from SSI absolute encoder.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.4 Safety

5.3.4.1 Set Maximal Following Error

FUNCTION

BOOL VCM_SetMaxFollowingError(HANDLE DriveHandle, DWORD MaxFollowingError, DWORD* pErrorCode)

DESCRIPTION

VCM_SetMaxFollowingError writes the maximal allowed following error parameter.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
MaxFollowingError	DWORD	Maximal allowed difference of position actual value to position demand value	0x6065-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.4.2 Get Maximal Following Error

FUNCTION

BOOL VCM_GetMaxFollowingError(HANDLE DriveHandle, DWORD* pMaxFollowingError, DWORD* pErrorCode)

DESCRIPTION

VCM_GetMaxFollowingError reads the maximal allowed following error parameter.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.4.3 Set Maximal Profile Velocity

FUNCTION

BOOL VCM_SetMaxProfileVelocity(HANDLE DriveHandle, DWORD MaxProfileVelocity, DWORD* pErrorCode)

DESCRIPTION

VCM_SetMaxProfileVelocity writes the maximal allowed velocity.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
MaxProfileVelocity	DWORD	Used as velocity limit in a position (or velocity) move	0x607F-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.4.4 Get Maximal Profile Velocity

FUNCTION

BOOL VCM_GetMaxProfileVelocity(HANDLE DriveHandle, DWORD* pMaxProfileVelocity, DWORD* pErrorCode)

DESCRIPTION

VCM_GetMaxProfileVelocity reads the maximal allowed velocity.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.4.5 Set Maximal Acceleration

FUNCTION

BOOL VCM_SetMaxAcceleration(HANDLE DriveHandle, DWORD MaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCM_SetMaxAcceleration writes the maximal allowed acceleration/deceleration.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
MaxAcceleration	DWORD	Limiter of the other acceleration/ deceleration objects	0x60C5-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.4.6 Get Maximal Acceleration

FUNCTION

BOOL VCM_GetMaxAcceleration(HANDLE DriveHandle, DWORD* pMaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCM_GetMaxAcceleration reads the maximal allowed acceleration/deceleration.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.5 Position Regulator

5.3.5.1 Set Position Regulator Gain

FUNCTION

BOOL VCM_SetPositionRegulatorGain(HANDLE DriveHandle, WORD P, WORD I, WORD D, DWORD* pErrorCode)

DESCRIPTION

VCM_SetPositionRegulatorGain writes all position regulator gains. Determine the optimal parameters using “Regulation Tuning” in «EPOS Studio».

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
P	WORD	Position regulator P-Gain	0x60FB-01
I	WORD	Position regulator I-Gain	0x60FB-02
D	WORD	Position regulator D-Gain	0x60FB-03

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.5.2 Set Position Regulator Feed Forward

FUNCTION

BOOL VCM_SetPositionRegulatorFeedForward(HANDLE DriveHandle, WORD VelocityFeedForward, WORD AccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCM_SetPositionRegulatorFeedForward writes parameters for position regulation with feed forward.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
VelocityFeedForward	WORD	Velocity feed forward factor	0x60FB-04
AccelerationFeedForward	WORD	Acceleration feed forward factor	0x60FB-05

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.5.3 Get Position Regulator Gain

FUNCTION

BOOL VCM_GetPositionRegulatorGain(HANDLE DriveHandle, WORD* pP, WORD* pI, WORD* pD, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPositionRegulatorGain reads all position regulator gains.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.5.4 Get Position Regulator Feed Forward

FUNCTION

BOOL VCM_GetPositionRegulatorFeedForward(HANDLE DriveHandle, WORD* pVelocityFeedForward, WORD* pAccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCM_GetPositionRegulatorFeedForward reads parameters for position regulation feed forward.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.6 Velocity Regulator

5.3.6.1 Set Velocity Regulator Gain

FUNCTION

BOOL VCM_SetVelocityRegulatorGain(HANDLE DriveHandle, WORD P, WORD I, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
P	WORD	Velocity regulator P-Gain	0x60F9-01
I	WORD	Velocity regulator I-Gain	0x60F9-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

5.3.6.2 Set Velocity Regulator Feed Forward

FUNCTION

BOOL VCM_SetVelocityRegulatorFeedForward(HANDLE DriveHandle, WORD VelocityFeedForward, WORD AccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCM_SetVelocityRegulatorFeedForward writes parameters for velocity regulation with feed forward.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
VelocityFeedForward	WORD	Velocity feed forward factor	0x60F9-04
AccelerationFeedForward	WORD	Acceleration feed forward factor	0x60F9-05

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

5.3.6.3 Get Velocity Regulator Gain

FUNCTION

BOOL VCM_GetVelocityRegulatorGain(HANDLE DriveHandle, WORD* pP, WORD* pI, DWORD* pErrorCode)

DESCRIPTION

VCM_GetVelocityRegulatorGain reads all velocity regulator gains.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.6.4 Get Velocity Regulator Feed Forward

FUNCTION

BOOL VCM_GetVelocityRegulatorFeedForward(HANDLE DriveHandle, WORD* pVelocityFeedForward, WORD* pAccelerationFeedForward, DWORD* pErrorCode)

DESCRIPTION

VCM_GetVelocityRegulatorFeedForward reads parameters for velocity regulation feed forward.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pVelocityFeedForward	WORD*	Velocity feed forward factor	0x60F9-04
pAccelerationFeedForward	WORD*	Acceleration feed forward factor	0x60F9-05
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.7 Current Regulator

5.3.7.1 Set Current Regulator Gain

FUNCTION

BOOL VCM_SetCurrentRegulatorGain(HANDLE DriveHandle, WORD P, WORD I, DWORD* pErrorCode)

DESCRIPTION

VCS_SetCurrentRegulatorGain writes all current regulator gains. Determine the optimal parameters using “Regulation Tuning” in «EPOS Studio».

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
P	WORD	Current regulator P-Gain	0x60F6-01
I	WORD	Current regulator I-Gain	0x60F6-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.7.2 Get Current Regulator Gain

FUNCTION

BOOL VCM_GetCurrentRegulatorGain(HANDLE DriveHandle, WORD* pP, WORD* pI, DWORD* pErrorCode)

DESCRIPTION

VCM_GetCurrentRegulatorGain enables reading all current regulator gains.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

5.3.8 Inputs/Outputs

5.3.8.1 Digital Input Configuration

FUNCTION

BOOL VCM_DigitalInputConfiguration(HANDLE DriveHandle, WORD DigitalInputNb, WORD Configuration, BOOL Mask, BOOL Polarity, BOOL ExecutionMask, DWORD* pErrorCode)

DESCRIPTION

VCS_DigitalInputConfiguration sets the parameter for one digital input.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
DigitalInputNb	WORD	Number of digital input (object subindex)	0x2070-0x
Configuration	WORD	Configures the functionality assigned to the digital input (bit number) (➔Table 5-13)	
Mask	BOOL	1: Functionality state will be displayed 0: not displayed	0x2071-02
Polarity	BOOL	1: Low active 0: High active	0x2071-03
ExecutionMask	BOOL	1: Set the error routine Only for positive and negative switch	0x2071-04

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

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

Table 5-13 Digital Input Configuration

5.3.8.2 Digital Output Configuration

FUNCTION

BOOL VCM_DigitalOutputConfiguration(HANDLE DriveHandle, WORD DigitalOutputNb, WORD Configuration, BOOL State, BOOL Mask, BOOL Polarity, DWORD* pErrorCode)

DESCRIPTION

VCS_DigitalOutputConfiguration sets parameter for one digital output.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
DigitalOutputNb	WORD	Number of digital output (object subindex)	0x2079-0x
Configuration	WORD	Configures the functionality assigned to the digital output (bit number) (➔Table 5-14)	
State	BOOL	State of digital output	0x2078-01
Mask	BOOL	1: Register will be modified	0x2078-02
Polarity	BOOL	1: Output will be inverted	0x2078-03

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

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

Table 5-14 Digital Output Configuration

5.3.8.3 Analog Input Configuration

FUNCTION

BOOL VCM_AnalogInputConfiguration(HANDLE DriveHandle, WORD AnalogInputNb, WORD Configuration, BOOL ExecutionMask, DWORD* pErrorCode)

DESCRIPTION

VCS_DigitalOutputConfiguration sets parameter for one digital output.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
AnalogInputNb	WORD	Number of analog input (object subindex)	0x207B-0x
Configuration	WORD	Configures the functionality assigned to the analog input (bit number) (➔Table 5-15)	
ExecutionMask	BOOL	1: Register will be modified	0x207D-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

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

Table 5-15 Analog Input Configuration

5.3.9 Units

5.3.9.1 Set Velocity Units

FUNCTION

BOOL VCM_SetVelocityUnits(HANDLE DriveHandle, BYTE VelDimension, char VelNotation, DWORD* pErrorCode)

DESCRIPTION

VCS_SetVelocityUnits writes velocity unit parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
VelDimension	BYTE	Velocity dimension index VD_RPM = 0xA4	0x608C-00
VelNotation	char	Velocity notation index (➔Table 5-16)	0x608B-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Standard	0	VN_STANDARD
Deci (10^{-1})	−1	VN_DECI
Centi (10^{-2})	−2	VN_CENTI
Milli (10^{-3})	−3	VN_MILLI

Table 5-16 Velocity Notation Index

5.3.9.2 Get Velocity Units

FUNCTION

BOOL VCM_GetVelocityUnits(HANDLE DriveHandle, BYTE* pVelDimension, char* pVelNotation, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityUnits reads velocity unit parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pVelDimension	BYTE*	Velocity dimension index VD_RPM = 0xA4	0x608C-00
pVelNotation	char*	Velocity notation index (➔Table 5-16)	0x608B-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

••page intentionally left blank••

6 Drive Functions – Operation

6.1 Operation Mode

6.1.1 Set Operation Mode

FUNCTION

BOOL VCM_SetOperationMode(HANDLE DriveHandle, __int8 Mode, DWORD* pErrorCode)

DESCRIPTION

VCM_SetOperationMode sets the operation mode.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
Mode	__int8	Operation mode (➔Table 6-17	0x6060-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information about the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

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

Table 6-17 Operation Modes

6.1.2 Get Operation Mode

FUNCTION

BOOL VCM_GetOperationMode(HANDLE DriveHandle, __int8* pMode, DWORD* pErrorCode)

DESCRIPTION

VCS_GetOperationMode returns the activated operation mode.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pMode	__int8*	Operation mode (➔Table 6-17)	0x6061-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.2 State Machine

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

6.2.1 Reset Device

FUNCTION

BOOL VCM_ResetDevice(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ResetDevice is used to send the NMT service “Reset Node”. Command is without acknowledge.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.2 Set State

FUNCTION

BOOL VCM_SetState(HANDLE DriveHandle, WORD State, DWORD* pErrorCode)

DESCRIPTION

VCS_SetState writes the actual state machine state.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
State	WORD	Value of state machine (➔Table 6-18)	0x6040-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Get/Set Disable State	0	ST_DISABLED
Get/Set Enable State	1	ST_ENABLED
Get/Set Quickstop State	2	ST_QUICKSTOP
Get Fault State	3	ST_FAULT

Table 6-18 State Modes

6.2.3 Set Enable State

FUNCTION

BOOL VCM_SetEnableState(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_SetEnableState changes the device state to "enable".

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.2.4 Set Disable State

FUNCTION

BOOL VCM_SetDisableState(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_SetDisableState changes the device state to "disable".

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.2.5 Set Quick Stop State

FUNCTION

BOOL VCM_SetQuickStopState(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_SetQuickStopState changes the device state to "quick stop".

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.2.6 Clear Fault

FUNCTION

BOOL VCM_ClearFault(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.7 Get State

FUNCTION

BOOL VCM_GetState(HANDLE DriveHandle, WORD* pState, DWORD* pErrorCode)

DESCRIPTION

VCS_GetState reads the new state of the state machine.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pState	WORD*	Statusword value (➔Table 6-18)	0x6041-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.2.8 Get Enable State

FUNCTION

BOOL VCM_GetEnableState(HANDLE DriveHandle, BOOL* pIsEnabled, DWORD* pErrorCode)

DESCRIPTION

VCS_GetEnableState checks if the device is enabled.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

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”

6.2.9 Get Disable State

FUNCTION

BOOL VCM_GetDisableState(HANDLE DriveHandle, BOOL* pIsDisabled, DWORD* pErrorCode)

DESCRIPTION

VCM_GetDisableState checks if the device is disabled.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

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”

6.2.10 Get Quick Stop State

FUNCTION

BOOL VCM_GetQuickStopState(HANDLE DriveHandle, BOOL* pIsQuickStopped, DWORD* pEr-rorCode)

DESCRIPTION

VCM_GetQuickStopState returns the device state quick stop.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

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”

6.2.11 Get Fault State

FUNCTION

BOOL VCM_GetFaultState(HANDLE DriveHandle, BOOL* pIsInFault, DWORD* pErrorCode)

DESCRIPTION

VCS_GetFaultState returns the device state fault (pIsInFault = TRUE). Get error information if the device is in fault state (➔“Error Overview” on page 9-153).

PARAMETERS

DriveHandle	HANDLE	Handle for port access

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”

6.3 Error Handling

6.3.1 Get Number of Device Error

FUNCTION

BOOL VCM_GetNbOfDeviceError(HANDLE DriveHandle, BYTE* pNbDeviceError, DWORD* pErrorCode)

DESCRIPTION

VCS_GetNbOfDeviceError returns the number of actual errors that are recorded (➔“Programming Example” on page 6-93).

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.3.2 Get Device Error Code

FUNCTION

BOOL VCM_GetDeviceErrorCode(HANDLE DriveHandle, BYTE ErrorNumber, DWORD* pDeviceErrorCode, DWORD* pErrorCode)

DESCRIPTION

VCS_GetDeviceErrorCode returns the error code of the selected error number.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ErrorNumber	BYTE	Number (object subindex) of device error (≥1)	0x1003-0x

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.3.3 Programming Example

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

```
-------------------------------------------------------------------------------------
//Global parameters
HANDLE DriveHandle = 1;       //handle from opened interface

//Functional parameters
BYTE nbOfDeviceError = 0;     //number of actual errors
DWORD functionErrorCode = 0;  //error code from function
DWORD deviceErrorCode = 0;    //error code from device

//get number of device errors
if(VCM_GetNbOfDeviceError(DriveHandle, &nbOfDeviceError, &functionErrorCode))
{
         //read device error code
         for(BYTE errorNumber = 1; errorNumber <= nbOfDeviceError; errorNumber++)
         {
                              if(!VCM_GetDeviceErrorCode(DriveHandle, errorNumber,
                              &deviceErrorCode, &functionErrorCode))
                              {
                               break;
                              }
         }
}
-------------------------------------------------------------------------------------
```

6.4 Motion Info

6.4.1 Get Movement State

FUNCTION

BOOL VCM_GetMovementState(HANDLE DriveHandle, BOOL* pTargetReached, DWORD* pErrorCode)

DESCRIPTION

VCS_GetMovementState checks if the drive has reached target.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.2 Get Position Is

FUNCTION

BOOL VCM_GetPositionIs(HANDLE DriveHandle, long* pPositionIs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionIs returns the position actual value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.3 Get Velocity Is

FUNCTION

BOOL VCM_GetVelocityIs(HANDLE DriveHandle, long* pVelocityIs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityIs reads the velocity actual value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.4 Get Velocity Is Averaged

FUNCTION

BOOL VCM_GetVelocityIsAveraged(HANDLE DriveHandle, long* pVelocityIsAveraged, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityIsAveraged reads the velocity actual averaged value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.5 Get Current Is

FUNCTION

BOOL VCM_GetCurrentIs(HANDLE DriveHandle, short* pCurrentIs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentIs returns the current actual value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.6 Get Current Is Averaged

FUNCTION

BOOL VCM_GetCurrentIsAveraged(HANDLE DriveHandle, short* pCurrentIsAveraged, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentIsAveraged returns the current actual averaged value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.4.7 Wait For Target Reached

FUNCTION

BOOL VCM_WaitForTargetReached(HANDLE DriveHandle, DWORD Timeout, DWORD* pErrorCode)

DESCRIPTION

VCS_WaitForTargetReached waits until the state is changed to target reached or until the time is up.

PARAMETERS

DriveHandle	HANDLE	Handle for port access
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”

6.5 Profile Position Mode (PPM)

6.5.1 Activate Profile Position Mode

FUNCTION

BOOL VCM_ActivateProfilePositionMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.2 Set Position Profile

FUNCTION

BOOL VCM_SetPositionProfile(HANDLE DriveHandle, DWORD ProfileVelocity, DWORD ProfileAcceleration, DWORD ProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionProfile sets the position profile parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ProfileVelocity	DWORD	Position profile velocity	0x6081-00
ProfileAcceleration	DWORD	Position profile acceleration	0x6083-00
ProfileDeceleration	DWORD	Position profile deceleration	0x6084-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.3 Get Position Profile

FUNCTION

BOOL VCM_GetPositionProfile(HANDLE DriveHandle, DWORD* pProfileVelocity, DWORD* pProfile-Acceleration, DWORD* pProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionProfile returns the position profile parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.4 Move To Position

FUNCTION

BOOL VCM_MoveToPosition(HANDLE DriveHandle, long TargetPosition, BOOL Absolute, BOOL Immediately, DWORD* pErrorCode)

DESCRIPTION

VCS_MoveToPosition starts movement with position profile to target position.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
TargetPosition	long	Target position	0x607A-00
Absolute	BOOL	TRUE starts an absolute FALSE a relative movement	0x6040-00 Bit 6
Immediately	BOOL	TRUE starts immediately FALSE waits to end of last positioning	0x6040-00 Bit 5

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.5 Get Target Position

FUNCTION

BOOL VCM_GetTargetPosition(HANDLE DriveHandle, long* pTargetPosition, DWORD* pErrorCode)

DESCRIPTION

VCS_GetTargetPosition returns the profile position mode target value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.6 Halt Position Movement

FUNCTION

BOOL VCM_HaltPositionMovement(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_HaltPositionMovement stops the movement with profile deceleration.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.7 Advanced Functions

6.5.7.1 Enable Position Window

FUNCTION

BOOL VCM_EnablePositionWindow(HANDLE DriveHandle, DWORD PositionWindow, WORD PositionWindowTime, DWORD* pErrorCode)

DESCRIPTION

VCS_EnablePositionWindow activates the position window.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
PositionWindow	DWORD	Position window value	0x6067-00
PositionWindowTime	WORD	Position window time value	0x6068-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.5.7.2 Disable Position Window

FUNCTION

BOOL VCM_DisablePositionWindow(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_DisablePositionWindow deactivates the position window.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6 Profile Velocity Mode (PVM)

6.6.1 Activate Profile Velocity Mode

FUNCTION

BOOL VCM_ActivateProfileVelocityMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.2 Set Velocity Profile

FUNCTION

BOOL VCM_SetVelocityProfile(HANDLE DriveHandle, DWORD ProfileAcceleration, DWORD Profile-Deceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetVelocityProfile sets the velocity profile parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ProfileAcceleration	WORD	Velocity profile acceleration	0x6083-00
ProfileDeceleration	WORD	Velocity profile deceleration	0x6084-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.3 Get Velocity Profile

FUNCTION

BOOL VCM_GetVelocityProfile(HANDLE DriveHandle, DWORD* pProfileAcceleration, DWORD* pProfileDeceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityProfile returns the velocity profile parameters.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.4 Move With Velocity

FUNCTION

BOOL VCM_MoveWithVelocity(HANDLE DriveHandle, long TargetVelocity, DWORD* pErrorCode)

DESCRIPTION

VCS_MoveWithVelocity starts the movement with velocity profile to target velocity.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
TargetVelocity	long	Target velocity	0x60FF-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.5 Get Target Velocity

FUNCTION

BOOL VCM_GetTargetVelocity(HANDLE DriveHandle, long* pTargetVelocity, DWORD* pErrorCode)

DESCRIPTION

VCS_GetTargetVelocity returns the profile velocity mode target value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.6 Halt Velocity Movement

FUNCTION

BOOL VCM_HaltVelocityMovement(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_HaltVelocityMovement stops the movement with profile deceleration.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.7 Advanced Functions

6.6.7.1 Enable Velocity Window

FUNCTION

BOOL VCM_EnableVelocityWindow(HANDLE DriveHandle, DWORD VelocityWindow, WORD Velocity-WindowTime, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableVelocityWindow activates the velocity window.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
VelocityWindow	DWORD	Velocity window value	0x606D-00
VelocityWindowTime	WORD	Velocity window time value	0x606E-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.6.7.2 Disable Velocity Window

FUNCTION

BOOL VCM_DisableVelocityWindow(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableVelocityWindow deactivates the velocity window.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.7 Homing Mode (HM)

6.7.1 Activate Homing Mode

FUNCTION

BOOL VCM_ActivateHomingMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateHomingMode changes the operational mode to “homing mode”.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.7.2 Set Homing Parameter

FUNCTION

BOOL VCM_SetHomingParameter(HANDLE DriveHandle, 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

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.7.3 Get Homing Parameter

FUNCTION

BOOL VCM_GetHomingParameter(HANDLE DriveHandle, DWORD* pHomingAcceleration, DWORD* pSpeedSwitch, DWORD* pSpeedIndex, long* pHomeOffset, WORD* pCurrentThreshold, long* pHome-Position, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.7.4 Find Home

FUNCTION

BOOL VCM_FindHome(HANDLE DriveHandle, __int8 HomingMethod, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
HomingMethod	__int8	Homing method (➔Table 6-19)	0x6098-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

HOMING METHODS

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

Table 6-19 Homing Methods

6.7.5 Stop Homing

FUNCTION

BOOL VCM_StopHoming(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_StopHoming interrupts homing.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.7.6 Define Position

FUNCTION

BOOL VCM_DefinePosition(HANDLE DriveHandle, long HomePosition, DWORD* pErrorCode)

DESCRIPTION

VCS_DefinePosition uses homing method 35 (Actual Position) to set a new home position.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
HomePosition	long	Used to assign the present position as homing position	0x2081-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.7.7 Get Homing State

FUNCTION

BOOL VCM_GetHomingState(HANDLE DriveHandle, 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

DriveHandle	HANDLE	Handle for port access

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”

6.7.8 Wait For Homing Attained

FUNCTION

BOOL VCM_WaitForHomingAttained(HANDLE DriveHandle, DWORD Timeout, DWORD* pErrorCode)

DESCRIPTION

VCS_WaitForHomingAttained waits until the homing mode is successfully terminated or until the time has elapsed.

PARAMETERS

DriveHandle	HANDLE	Handle for port access
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”

6.8 Interpolated Position Mode (IPM)

6.8.1 Activate Interpolated Position Mode

FUNCTION

BOOL VCM_ActivateInterpolatedPositionMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.2 Set IPM Buffer Parameter

FUNCTION

BOOL VCM_SetIpmBufferParameter(HANDLE DriveHandle, WORD UnderflowWarningLimit, WORD OverflowWarningLimit, DWORD* pErrorCode)

DESCRIPTION

VCS_SetIpmBufferParameter sets warning borders of the data input.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
UnderflowWarningLimit	WORD	Gives lower signalization level of the data input FIFO	0x20C4-02
OverflowWarningLimit	WORD	Gives the higher signalization level of the data input FIFO	0x20C4-03

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.3 Get IPM Buffer Parameter

FUNCTION

BOOL VCM_GetIpmBufferParameter(HANDLE DriveHandle, WORD* pUnderflowWarningLimit, WORD* pOverflowWarningLimit, DWORD* pMaxBufferSize, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.4 Clear IPM Buffer

FUNCTION

BOOL VCM_ClearIpmBuffer(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ClearIpmBuffer clears the input buffer and enables access to the input buffer for drive functions.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.5 Get Free IPM Buffer Size

FUNCTION

BOOL VCM_GetFreeIpmBufferSize(HANDLE DriveHandle, DWORD* pBufferSize, DWORD* pErrorCode)

DESCRIPTION

VCS_GetFreeIpmBufferSize reads the available buffer size.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.6 Add PVT Value To IPM Buffer

FUNCTION

BOOL VCM_AddPvtValueToIpmBuffer(HANDLE DriveHandle, long Position, long Velocity, BYTE Time, DWORD* pErrorCode)

DESCRIPTION

VCS_AddPvtValueToIpmBuffer adds a new PVT reference point to the device.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
Position	long	Position of the reference point	0x20C1-00
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”

6.8.7 Start IPM Trajectory

FUNCTION

BOOL VCM_StartIpmTrajectory(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_StartIpmTrajectory starts the IPM trajectory.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.8 Stop IPM Trajectory

FUNCTION

BOOL VCM_StopIpmTrajectory(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_StopIpmTrajectory stops the IPM trajectory.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.8.9 Get IPM Status

Function

BOOL VCM_GetIpmStatus(HANDLE DriveHandle, BOOL* pTrajectoryRunning, BOOL* pIsUnderflow-Warning, BOOL* pIsOverflowWarning, BOOL* pIsVelocityWarning, BOOL* pIsAccelerationWarning, BOOL* pIsUnderflowError, BOOL* pIsOverflowError, BOOL* pIsVelocityError, BOOL* pIsAcceleration-Error, DWORD* pErrorCode)

Description

VCS_GetIpmStatus returns different warning and error states.

Parameters

DriveHandle	HANDLE	Handle for port access

Return Parameters

pTrajectoryRunning	BOOL*	State if IPM active	0x20C4-01
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”

6.9 Position Mode (PM)

6.9.1 Activate Position Mode

FUNCTION

BOOL VCM_ActivatePositionMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivatePositionMode changes the operational mode to “position mode”.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.9.2 Set Position Must

FUNCTION

BOOL VCM_SetPositionMust(HANDLE DriveHandle, long PositionMust, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionMust sets the position mode setting value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
PositionMust	long	Position mode setting value	0x2062-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.9.3 Get Position Must

FUNCTION

BOOL VCM_GetPositionMust(HANDLE DriveHandle, long* pPositionMust, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionMust reads the position mode setting value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.9.4 Advanced Functions

6.9.4.1 Activate Analog Position Setpoint

FUNCTION

BOOL VCM_ActivateAnalogPositionSetpoint(HANDLE DriveHandle, WORD AnalogInputNumber, float Scaling, long Offset, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateAnalogPositionSetpoint configures the selected analog input for analog position setpoint.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.9.4.2 Deactivate Analog Position Setpoint

FUNCTION

BOOL VCM_DeactivateAnalogPositionSetpoint(HANDLE DriveHandle, WORD AnalogInputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivateAnalogPositionSetpoint disables the selected analog input for analog position setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
AnalogInputNumber	WORD	Number of the used analog input	0x207B-01 or 0x207B-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.9.4.3 Enable Analog Position Setpoint

FUNCTION

BOOL VCM_EnableAnalogPositionSetpoint(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableAnalogPositionSetpoint enables the execution mask for analog position setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.9.4.4 Disable Analog Position Setpoint

FUNCTION

BOOL VCM_DisableAnalogPositionSetpoint(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableAnalogPositionSetpoint disables the execution mask for analog position setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.10 Velocity Mode (VM)

6.10.1 Activate Velocity Mode

FUNCTION

BOOL VCM_ActivateVelocityMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateVelocityMode changes the operational mode to "velocity mode".

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.10.2 Set Velocity Must

FUNCTION

BOOL VCM_SetVelocityMust(HANDLE DriveHandle, long VelocityMust, DWORD* pErrorCode)

DESCRIPTION

VCS_SetVelocityMust sets the velocity mode setting value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
VelocityMust	long	Velocity mode setting value	0x206B-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.10.3 Get Velocity Must

FUNCTION

BOOL VCM_GetVelocityMust(HANDLE DriveHandle, long* pVelocityMust, DWORD* pErrorCode)

DESCRIPTION

VCS_GetVelocityMust returns the velocity mode setting value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise "0"

6.10.4 Advanced Functions

6.10.4.1 Activate Analog Velocity Setpoint

FUNCTION

BOOL VCM_ActivateAnalogVelocitySetpoint(HANDLE DriveHandle, WORD AnalogInputNumber, float Scaling, long Offset, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateAnalogVelocitySetpoint configures the selected analog input for analog velocity setpoint.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.10.4.2 Deactivate Analog Velocity Setpoint

FUNCTION

BOOL VCM_DeactivateAnalogVelocitySetpoint(HANDLE DriveHandle, WORD AnalogInputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivateAnalogVelocitySetpoint disables the selected analog input for analog velocity setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
AnalogInputNumber	WORD	Number of the used analog input	0x207B-01 or 0x207B-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.10.4.3 Enable Analog Velocity Setpoint

FUNCTION

BOOL VCM_EnableAnalogVelocitySetpoint(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableAnalogVelocitySetpoint enables the execution mask for analog velocity setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.10.4.4 Disable Analog Velocity Setpoint

FUNCTION

BOOL VCM_DisableAnalogVelocitySetpoint(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableAnalogVelocitySetpoint disables the execution mask for analog velocity setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11 Current Mode (CM)

6.11.1 Activate Current Mode

FUNCTION

BOOL VCM_ActivateCurrentMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateCurrentMode changes the operational mode to “current mode”.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11.2 Get Current Must

FUNCTION

BOOL VCM_GetCurrentMust(HANDLE DriveHandle, short* pCurrentMust, DWORD* pErrorCode)

DESCRIPTION

VCS_GetCurrentMust reads the current mode setting value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11.3 Set Current Must

FUNCTION

BOOL VCM_SetCurrentMust(HANDLE DriveHandle, short CurrentMust, DWORD* pErrorCode)

DESCRIPTION

VCS_SetCurrentMust writes current mode setting value.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
CurrentMust	short	Current mode setting value	0x2030-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11.4 Advanced Functions

6.11.4.1 Activate Analog Current Setpoint

FUNCTION

BOOL VCM_ActivateAnalogCurrentSetpoint(HANDLE DriveHandle, WORD AnalogInputNumber, float Scaling, short Offset, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateAnalogCurrentSetpoint configures the selected analog input for analog current setpoint.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11.4.2 Deactivate Analog Current Setpoint

FUNCTION

BOOL VCM_DeactivateAnalogCurrentSetpoint(HANDLE DriveHandle, WORD AnalogInputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivateAnalogCurrentSetpoint disables the selected analog input for analog current setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
AnalogInputNumber	WORD	Number of the used analog input	0x207B-01 or 0x207B-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11.4.3 Enable Analog Current Setpoint

FUNCTION

BOOL VCM_EnableAnalogCurrentSetpoint(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableAnalogCurrentSetpoint enables the execution mask for analog current setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.11.4.4 Disable Analog Current Setpoint

FUNCTION

BOOL VCM_DisableAnalogCurrentSetpoint(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_DisableAnalogCurrentSetpoint disables the execution mask for analog current setpoint.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.12 Master Encoder Mode (MEM)

6.12.1 Activate Master Encoder Mode

FUNCTION

BOOL VCM_ActivateMasterEncoderMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.12.2 Set Master Encoder Parameter

FUNCTION

BOOL VCM_SetMasterEncoderParameter(HANDLE DriveHandle, WORD ScalingNumerator, WORD ScalingDenominator, BYTE Polarity, DWORD MaxVelocity, DWORD MaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetMasterEncoderParameter writes all parameters for master encoder mode.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.12.3 Get Master Encoder Parameter

FUNCTION

BOOL VCM_GetMasterEncoderParameter(HANDLE DriveHandle, WORD* pScalingNumerator, WORD* pScalingDenominator, BYTE* pPolarity, DWORD* pMaxVelocity, DWORD* pMaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_GetMasterEncoderParameter reads all parameters for master encoder mode.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.13 Step Direction Mode (SDM)

6.13.1 Activate Step Direction Mode

FUNCTION

BOOL VCM_ActivateStepDirectionMode(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.13.2 Set Step Direction Parameter

FUNCTION

BOOL VCM_SetStepDirectionParameter(HANDLE DriveHandle, WORD ScalingNumerator, WORD ScalingDenominator, BYTE Polarity, DWORD MaxVelocity, DWORD MaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_SetStepDirectionParameter writes all parameters for step direction mode.

PARAMETERS

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

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.13.3 Get Step Direction Parameter

FUNCTION

BOOL VCM_GetStepDirectionParameter(HANDLE DriveHandle, WORD* pScalingNumerator, WORD* pScalingDenominator, BYTE* pPolarity, DWORD* pMaxVelocity, DWORD* pMaxAcceleration, DWORD* pErrorCode)

DESCRIPTION

VCS_GetStepDirectionParameter reads all parameters for step direction mode.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14 Inputs & Outputs

For details ➔separate document «Firmware Specification».

6.14.1 Get All Digital Inputs

FUNCTION

BOOL VCM_GetAllDigitalInputs(HANDLE DriveHandle, WORD* pInputs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetAllDigitalInputs returns state of all digital inputs.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.2 Get All Digital Outputs

FUNCTION

BOOL VCM_GetAllDigitalOutputs(HANDLE DriveHandle, WORD* pOutputs, DWORD* pErrorCode)

DESCRIPTION

VCS_GetAllDigitalOutputs returns state of all digital outputs.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.3 Set All Digital Outputs

FUNCTION

BOOL VCS_SetAllDigitalOutputs(HANDLE DriveHandle, WORD Outputs, DWORD* pErrorCode)

DESCRIPTION

VCS_SetAllDigitalOutputs sets the state of all digital outputs.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
Outputs	WORD	State of all digital outputs. Activated if a bit is written as “1”.	0x2078-01

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.4 Get Analog Input

FUNCTION

BOOL VCM_GetAnalogInput(HANDLE DriveHandle, WORD InputNumber, WORD* pAnalogValue, DWORD* pErrorCode)

DESCRIPTION

VCS_GetAnalogInput returns the value from an analog input.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pAnalogValue	WORD*	Analog value from input	0x207C-0x
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.5 Set Analog Output

FUNCTION

BOOL VCM_SetAnalogOutput(HANDLE DriveHandle, WORD OutputNumber, WORD AnalogValue, DWORD* pErrorCode)

DESCRIPTION

VCS_SetAnalogOutput sets the voltage level of an analog output.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
OutputNumber	WORD	Analog output number	
pAnalogValue	WORD*	Analog value for output	0x207E-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.6 Position Compare

6.14.6.1 Set Position Compare Parameter

FUNCTION

BOOL VCM_SetPositionCompareParameter(HANDLE DriveHandle, BYTE OperationalMode, BYTE IntervalMode, BYTE DirectionDependency, WORD IntervalWidth, WORD IntervalRepetitions, WORD PulseWidth, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionCompareParameter writes all parameters for position compare.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
OperationalMode	BYTE	Used operational mode in position sequence mode (➔Table 6-20)	0x207A-01
IntervalMode	BYTE	Used interval mode in position sequence mode (➔Table 6-21)	
DirectionDependency	BYTE	Used direction dependency in position sequence mode (➔Table 6-22)	
IntervalWidth	WORD	Holds the width of the position intervals	0x207A-03
IntervalRepetitions	WORD	Allows to configure the number of position intervals to be considered by position compare	0x207A-04
PulseWidth	WORD	Configures the pulse width of the trigger output	0x207A-05

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

OperationalMode

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

Table 6-20 Position Compare – Operational Modes

IntervalMode

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

Table 6-21 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 6-22 Position Compare – Direction Dependency

6.14.6.2 Get Position Compare Parameter

FUNCTION

BOOL VCM_GetPositionCompareParameter(HANDLE DriveHandle, BYTE* pOperationalMode, BYTE* pIntervalMode, BYTE* pDirectionDependency, WORD* pIntervalWidth, WORD* pIntervalRepetitions, WORD* pPulseWidth, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionCompareParameter reads all parameters for position compare.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pOperationalMode	BYTE*	Used operational mode in position sequence mode (➔Table 6-20)	0x207A-01
pIntervalMode	BYTE*	Used interval mode in position sequence mode (➔Table 6-21)	
pDirectionDependency	BYTE*	Used direction dependency in position sequence mode (➔Table 6-22)	
pIntervalWidth	WORD*	Holds the width of the position intervals	0x207A-03
pIntervalRepetitions	WORD*	Allows to configure the number of position intervals to be considered by position compare	0x207A-04
pPulseWidth	WORD*	Configures the pulse width of the trigger output	0x207A-05
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.6.3 Activate Position Compare

FUNCTION

BOOL VCM_ActivatePositionCompare(HANDLE DriveHandle, WORD DigitalOutputNumber, BOOL Polarity, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivatePositionCompare enables the output to position compare method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
DigitalOutputNumber	WORD	Selected digital output for position compare	0x2079
Polarity	BOOL	Polarity of the selected output	0x2078-03

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.6.4 Deactivate Position Compare

FUNCTION

BOOL VCM_DeactivatePositionCompare(HANDLE DriveHandle, WORD DigitalOutputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivatePositionCompare disables the output to position compare method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
DigitalOutputNumber	WORD	Selected digital output for position compare	0x2079-0x

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.6.5 Enable Position Compare

FUNCTION

BOOL VCM_EnablePositionCompare(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_EnablePositionCompare enables the output mask for position compare method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.6.6 Disable Position Compare

FUNCTION

BOOL VCM_DisablePositionCompare(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_DisablePositionCompare disables the output mask from position compare method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.6.7 Set Position Compare Reference Position

FUNCTION

BOOL VCM_SetPositionCompareReferencePosition(HANDLE DriveHandle, long ReferencePosition, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionCompareReferencePosition writes the reference position for position compare method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ReferencePosition	long	Holds the position that is compared with the position actual value	0x207A-02

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.7 Position Marker

6.14.7.1 Set Position Marker Parameter

FUNCTION

BOOL VCM_SetPositionMarkerParameter(HANDLE DriveHandle, BYTE PositionMarkerEdgeType, BYTE PositionMarkerMode, DWORD* pErrorCode)

DESCRIPTION

VCS_SetPositionMarkerParameter writes all parameters for position marker method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
PositionMarkerEdgeType	BYTE	Defines the type of edge of the position to be captured (➔Table 6-23)	0x2074-02
PositionMarkerMode	BYTE	Defines the position marker capturing mode (➔Table 6-24)	0x2074-03

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 6-23 Position Marker Edge Types

POSITIONMARKERMODE

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

Table 6-24 Position Marker Modes

6.14.7.2 Get Position Marker Parameter

FUNCTION

BOOL VCM_GetPositionMarkerParameter(HANDLE DriveHandle, BYTE* pPositionMarkerEdgeType, BYTE* pPositionMarkerMode, DWORD* pErrorCode)

DESCRIPTION

VCS_GetPositionMarkerParameter reads all parameters for position marker method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	

RETURN PARAMETERS

pPositionMarkerEdge Type	BYTE*	Defines the type of edge of the position to be captured (➔Table 6-23)	0x2074-02
pPositionMarkerMode	BYTE*	Defines the position marker capturing mode (➔Table 6-24)	0x2074-03
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.7.3 Activate Position Marker

FUNCTION

BOOL VCM_ActivatePositionMarker(HANDLE DriveHandle, WORD DigitalInputNumber, BOOL Polarity, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivatePositionMarker enables the digital input to position marker method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
DigitalInputNumber	WORD	Selected digital input for position marker	0x2070-0x
Polarity	BOOL	Polarity of the selected input	0x2071-03

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.7.4 Deactivate Position Marker

FUNCTION

BOOL VCM_DeactivatePositionMarker(HANDLE DriveHandle, WORD DigitalInputNumber, DWORD* pErrorCode)

DESCRIPTION

VCS_DeactivatePositionMarker disables the digital input to position marker method.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
DigitalInputNumber	WORD	Selected digital input for position marker	0x2070-0x

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.7.5 Read Position Marker Counter

FUNCTION

BOOL VCM_ReadPositionMarkerCounter(HANDLE DriveHandle, WORD* pCount, DWORD* pErrorCode)

DESCRIPTION

VCS_ReadPositionMarkerCounter returns the number of the detected edges.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pCount	WORD*	Counts the number of detected edges	0x2074-04
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.7.6 Read Position Marker Captured Position

FUNCTION

BOOL VCM_ReadPositionMarkerCapturedPosition(HANDLE DriveHandle, WORD CounterIndex, long* pCapturedPosition, DWORD* pErrorCode)

DESCRIPTION

VCS_ReadPositionMarkerCapturedPosition returns the last captured position or the position from the position marker history.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
CounterIndex	WORD	0: Read position marker captured position	0x2074-01
		1 or 2: Read position marker history	0x2074-05 or 0x2074-06

RETURN PARAMETERS

pCapturedPosition	long*	Contains the captured position or the position marker history	0x2074-01 or 0x2074-05 or 0x2074-06
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

6.14.7.7 Reset Position Marker Counter

FUNCTION

BOOL VCM_ResetPositionMarkerCounter(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ResetPositionMarkerCounter clears the counter and the captured positions by writing zero to object position marker counter (0x2074-04).

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7 Drive Functions – Data Recording

7.1 Operation Mode

7.1.1 Set Recorder Parameter

FUNCTION

BOOL VCM_SetRecorderParameter(HANDLE DriveHandle, WORD SamplingPeriod, WORD NbOfPrecedingSamples, WORD PulseWidth, DWORD* pErrorCode)

DESCRIPTION

VCS_SetRecorderParameter writes parameters for data recorder.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
SamplingPeriod	WORD	Sampling Period as a multiple of the current regulator cycle (n-times 0.1 ms)	0x2012-00
NbOfPrecedingSamples	WORD	Number of preceding samples (data history)	0x2013-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.1.2 Get Recorder Parameter

FUNCTION

BOOL VCM_GetRecorderParameter(HANDLE DriveHandle, WORD* pSamplingPeriod, WORD* pNbOfPrecedingSamples, WORD PulseWidth, DWORD* pErrorCode)

DESCRIPTION

VCS_GetRecorderParameter reads parameters for data recorder.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pSamplingPeriod	WORD*	Sampling Period as a multiple of the current regulator cycle (n-times 0.1 ms)	0x2012-00
pNbOfPrecedingSamples	WORD*	Number of preceding samples (data history)	0x2013-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.1.3 Enable Trigger

FUNCTION

BOOL VCM_EnableTrigger(HANDLE DriveHandle, BYTE TriggerType, DWORD* pErrorCode)

DESCRIPTION

VCS_EnableTrigger connects the trigger(s) for data recording.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
TriggerType	BYTE	Configuration of Auto Trigger functions. Activated if a bit is written as “1” (➔Table 7-25). Activation of more than one trigger at the same time is possible.	0x2011-00

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

Description	Value	Name
Trigger movement start	1	DR_MOVEMENT_START_TRIGGER
Error trigger	2	DR_ERROR_TRIGGER
Digital input trigger	4	DR_DIGITAL_INPUT_TRIGGER
Trigger movement end	8	DR_MOVEMENT_END_TRIGGER

Table 7-25 Data Recorder Trigger Types

7.1.4 Disable all Triggers

FUNCTION

BOOL VCM_DisableAllTrigger(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.1.5 Activate Channel

FUNCTION

BOOL VCM_ActivateChannel(HANDLE DriveHandle, BYTE ChannelNumber, WORD ObjectIndex, BYTE ObjectSubIndex, BYTE ObjectSize, DWORD* pErrorCode)

DESCRIPTION

VCS_ActivateChannel connects object for data recording.

Start with channel 1 (one)! Then, for every activated channel, the number of sampling variables (0x2014-00) will be incremented.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ChannelNumber	BYTE	Channel number [1…4]	
ObjectIndex	WORD	Object index for data recording	0x2015-Channel Number
ObjectSubIndex	BYTE	Object subindex for data recording	0x2016-Channel Number
ObjectSize	BYTE	Object size in bytes for data recording	

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.1.6 Deactivate all Channels

FUNCTION

BOOL VCM_DeactivateAllChannel(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

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

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.2 Data Recorder Status

7.2.1 Start Recorder

FUNCTION

BOOL VCM_StartRecorder(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_StartRecorder starts data recording.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.2.2 Stop Recorder

FUNCTION

BOOL VCM_StopRecorder(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_StopRecorder stops data recording.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.2.3 Force Trigger

FUNCTION

BOOL VCM_ForceTrigger(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ForceTrigger forces the data recording triggers.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.2.4 Is Recorder Running

FUNCTION

BOOL VCM_IsRecorderRunning(HANDLE DriveHandle, BOOL* pRunning, DWORD* pErrorCode)

DESCRIPTION

VCS_IsRecorderRunning returns the data recorder status “running”.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pRunning	BOOL	1: Data recorder running 0: Data recorder stopped	0x2017-00 (Bit 0)
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.2.5 Is Recorder Triggered

FUNCTION

BOOL VCM_IsRecorderTriggered(HANDLE DriveHandle, BOOL* pTriggered, DWORD* pErrorCode)

DESCRIPTION

VCS_IsRecorderTriggered returns data recorder status “triggered”.

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pTriggered	BOOL	1: Data recorder triggered 0: Data recorder not triggered	0x2017-00 (Bit 1)
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.3 Data Recorder Data

7.3.1 Read Channel Vector Size

FUNCTION

BOOL VCM_ReadChannelVectorSize(HANDLE DriveHandle, DWORD* pVectorSize, DWORD* pErrorCode)

DESCRIPTION

VCS_ReadChannelVectorSize returns the maximal number of samples per variable. It is dynamically calculated by the data recorder.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	

RETURN PARAMETERS

pVectorSize	DWORD*	Maximal number of samples per variable	0x2018-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.3.2 Read Channel Data Vector

FUNCTION

BOOL VCM_ReadChannelDataVector(HANDLE DriveHandle, BYTE ChannelNumber, BYTE* pDataVector, DWORD VectorSize, DWORD* pErrorCode)

DESCRIPTION

VCS_ReadChannelDataVector returns the data points of a selected channel.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ChannelNumber	BYTE	Selected channel	
VectorSize	DWORD	Size of data points	0x2018-00

RETURN PARAMETERS

pDataVector	BYTE*	Data points of selected channel	0x201B-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.3.3 Show Channel Data Dialog

FUNCTION

BOOL VCM_ShowChannelDataDlg(HANDLE DriveHandle, DWORD* pErrorCode)

DESCRIPTION

VCS_ShowChannelDataDlg opens the dialog to show the data channel(s).

PARAMETERS

DriveHandle	HANDLE	Handle for port access

RETURN PARAMETERS

pErrorCode	DWORD*	Error information on the executed function

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.3.4 Export Channel Data to File

FUNCTION

BOOL VCM_ExportChannelDataToFile(HANDLE DriveHandle, char* FileName, DWORD* pErrorCode)

DESCRIPTION

VCS_ExportChannelDataToFile saves the data point in a file.

PARAMETERS

DriveHandle	HANDLE	Handle for port access
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”

7.4 Advanced Functions

7.4.1 Read Data Buffer

Function

BOOL VCM_ReadDataBuffer(HANDLE DriveHandle, BYTE* pDataBuffer, DWORD BufferSizeToRead, DWORD* pBufferSizeRead, WORD* pVectorStartOffset, WORD* pMaxNbOfSamples, WORD* pNbOf-RecordedSamples, DWORD* pErrorCode)

Description

VCS_ReadDataBuffer returns the buffer data points.

Parameters

DriveHandle	HANDLE	Handle for port access
BufferSizeToRead	DWORD	Buffer size

Return Parameters

pDataBuffer	BYTE*	Data points	0x201B-00
pBufferSizeRead	DWORD*	Size of read data buffer	
pVectorStartOffset	WORD*	Offset to the start of the recorded data vector within the ring buffer	0x201A-00
pMaxNbOfSamples	WORD*	Maximal number of samples per variable	0x2018-00
pNbOfRecordedSamples	WORD*	Number of recorded samples	0x2019-00
pErrorCode	DWORD*	Error information on the executed function	

Return Value	BOOL	Nonzero if successful; otherwise “0”

7.4.2 Extract Channel Data Vector

FUNCTION

BOOL VCM_ExtractChannelDataVector(HANDLE DriveHandle, BYTE ChannelNumber, BYTE* pDataBuffer, DWORD BufferSize, BYTE* pDataVector, DWORD VectorSize, WORD VectorStartOffset, WORD MaxNbOfSamples, WORD NbOfRecordedSamples, DWORD* pErrorCode)

DESCRIPTION

VCS_ExtractChannelDataVector returns the vector of a data channel.

PARAMETERS

DriveHandle	HANDLE	Handle for port access	
ChannelNumber	BYTE	Selected channel	
pDataBuffer	BYTE	Data points	0x201B-00
BufferSize	DWORD	Buffer size	
VectorSize	DWORD	Vector size	
VectorStartOffset	WORD	Offset to the start of the recorded data vector within the ring buffer	0x201A-00
MaxNbOfSamples	WORD	Maximal number of samples per variable	0x2018-00
NbOfRecordedSamples	WORD	Number of recorded samples	0x2019-00

RETURN PARAMETERS

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

Return Value	BOOL	Nonzero if successful; otherwise “0”

••page intentionally left blank••

8 Drive Functions – Low Layer

Note
The function “Send NMT Service” is no longer operative. ***Use ➔Send NMT Service Ex instead****.*

8.1 Send CAN Frame

FUNCTION

BOOL VCM_SendCANFrame(HANDLE CommunicationOrGatewayHandle, WORD CobID, WORD Length, void* pData, DWORD* pErrorCode)

DESCRIPTION

VCM_SendCANFrame sends a general CAN frame to the CAN bus.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle for communication or gateway 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”

8.2 Read CAN Frame

FUNCTION

BOOL (HANDLE CommunicationOrGatewayHandle, WORD CobID, WORD Length, void* pData, DWORD Timeout, DWORD* p ErrorCode)

DESCRIPTION

VCM_ReadCANFrame reads a general CAN frame from the CAN bus. It **is not supported by the gateway**. Thus, the functionality is neither available for internal nor for external devices.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle for communication or gateway 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”

8.3 Request CAN Frame

FUNCTION

BOOL VCM_RequestCANFrame(HANDLE CommunicationOrGatewayHandle, WORD CobID, WORD Length, void* pData, DWORD* pErrorCode)

DESCRIPTION

VCM_RequestCANFrame requests a general CAN frame from the CAN bus using Remote Transmit Request (RTR). It **is not supported by the gateway**. Thus, the functionality is neither available for internal nor for external devices.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle for communication or gateway 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”

8.4 Send NMT Service Ex

FUNCTION

BOOL VCM_SendNMTServiceEx(HANDLE CommunicationOrGatewayHandle, WORD CommandSpecifier, WORD NodeId, DWORD* pErrorCode)

DESCRIPTION

VCM_SendNMTServiceEx is used to send a NMT protocol from a master to one slave/all slaves in a network. Command is without acknowledge.

PARAMETERS

CommunicationOr GatewayHandle	HANDLE	Handle for communication or gateway port access
NodeId	WORD	1…127: NMT slave with given Node ID 0: All NMT slaves
CommandSpecifier	WORD	NMT service (➔Table 8-26)

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 8-26 Command Specifier

••page intentionally left blank••

9 Error Overview

9.1 Communication Errors

Abort Code	Name	Error Cause
0x0000 0000	No Communication Error	Communication was successful
0x0503 0000	Toggle Error	Toggle bit not alternated
0x0504 0000	SDO Time Out	SDO protocol timed out
0x0504 0001	Client/server specifier Error	Client/server command specifier not valid or unknown
0x0504 0002	Invalid block size	Invalid block size (block mode only)
0x0504 0003	Invalid sequence	Invalid sequence number (block mode only)
0x0504 0004	CrcError	CRC error (block mode only)
0x0504 0005	Out of Memory Error	Out of Memory
0x0601 0000	Access Error	Unsupported access to an object (e.g. write command to a read-only object)
0x0601 0001	Write Only	Read command to a write only object
0x0601 0002	Read Only	Write command to a read only object
0x0602 0000	Object does not exist Error	Last read or write command had a wrong object index or subindex
0x0604 0041	PDO mapping Error	Object cannot be mapped to PDO
0x0604 0042	PDO length Error	Number and length of objects to be mapped would exceed PDO length
0x0604 0043	General parameter Error	General parameter incompatibility
0x0604 0047	General Intern Incompatibility Error	General internal incompatibility in device
0x0606 0000	Hardware Error	Access failed due to an hardware error
0x0607 0010	Service Parameter Error	Data type does not match, length or service parameter does not match
0x0607 0012	Service Parameter Error too High Error	Data type does not match, length or service parameter too high
0x0607 0013	Service Parameter Error too Low Error	Data type does not match, length or service parameter too low
0x0609 0011	Object Subindex Error	Last read or write command had a wrong subindex
0x0609 0030	Value Range Error	Value range of parameter exceeded
0x0609 0031	Value too High Error	Value of parameter written too high
0x0609 0032	Value too Low Error	Value of parameter written too low
0x0609 0036	Maximum less Minimum Error	Maximum value is less than minimum value
0x0800 0000	General Error	General error
0x0800 0020	Transfer or store Error	Data cannot be transferred or stored
0x0800 0021	Local control Error	Data cannot be transferred or stored to application because of local control
0x0800 0022	Wrong Device State	Data cannot be transferred or stored to application because of present device state
0x0A00 0001	Network Id unknown	Network identification is unknown (does not exist in routing list)
0x0A00 0002	Node Id unknown	Node identification is unknown
0x0F00 FFB9	Error CAN id	Wrong CAN id
0x0F00 FFBC	Error Service Mode	Device is not in service mode
0x0F00 FFBE	Password Error	Password is wrong
0x0F00 FFBF	Illegal Command Error	Command is illegal (does not exist)
0x0F00 FFC0	Wrong NMT State Error	Device is in wrong NMT state

Abort Code	Name	Error Cause
0x0F00 FFC2	Segmented Transfer Required	Segmented transfer required (initialization is already done)
0x0FFF FFF0	Communication Sequence Error	Error during function block execution
0x0FFF FFF1	Communication Aborted	Communication aborted
0x0FFF FFF2	Communication Buffer Overflow	Communication buffer overflow
0x0FFF FFF9	Segmented Transfer Communication Error	Segmented transfer communication error
0x0FFF FFFA	Wrong Axis Number	Axis number was not within 0…32
0x0FFF FFFB	Wrong CAN Device	CAN device number was not within 0…127
0x0FFF FFFC	Wrong CAN Port	CAN port is not valid (not 1 or 2)
0x0FFF FFFD	Wrong Parameter	Internal function calling parameters wrong
0x0FFF FFFE	General Communication Error	General communication error occurred
0x0FFF FFFF	Communication Timeout	Communication timeout occurred

Table 9-27 Communication Errors

9.2 EPOS Command Library-specified Errors

9.2.1 General Errors

Abort Code	Name	Error Cause
0x0000 0000	No Error	Function was successful
0x1000 0001	Internal Error	Internal error
0x1000 0002	Null Pointer	Null pointer passed to function
0x1000 0003	Handle not Valid	Handle passed to function is not valid
0x1000 0004	Bad Virtual Device Name	Virtual device name is not valid
0x1000 0005	Bad Device Name	Device name is not valid
0x1000 0006	Bad ProtocolStack Name	Protocol stack name is not valid
0x1000 0007	Bad Interface Name	Interface name is not valid
0x1000 0008	Bad Port Name	Port is not valid
0x1000 0009	Library not Loaded	Could not load external library
0x1000 000A	Executing Command	Command failed
0x1000 000B	Timeout	Timeout occurred during execution
0x1000 000C	Bad Parameter	Bad parameter passed to function
0x1000 000D	Command Aborted By User	Command aborted by user
0x1000 000E	Buffer Too Small	Buffer is too small
0x1000 000F	No Communication Found	No communication settings found
0x1000 0010	Function Not Supported	Function not supported
0x1000 0011	Parameter Already Used	Parameter already used
0x1000 0012	Bad Virtual Device Handle	Virtual device handle is not valid
0x1000 0013	Bad Device Handle	Device handle is not valid
0x1000 0014	Bad Protocol Stack Handle	Protocol stack handle is not valid
0x1000 0015	Bad Interface Handle	Interface handle is not valid
0x1000 0016	Bad Port Handle	Port handle is not valid
0x1000 0017	Bad Address Parameter	Address parameters are not correct
0x1000 0018	Bad Variable Info File	Variable info file is not initialized
0x1000 0019	Variable Name Not Found	Variable name not found

Abort Code	Name	Error Cause
0x1000 0020	Bad Device State	Bad device state
0x1000 0021	Bad File Content	Bad file content
0x1000 0022	Path Does Not Exist	System cannot find specified path

Table 9-28 General Errors

9.2.2 Interface Layer Errors

Abort Code	Name	Error Cause
0x2000 0001	Opening Interface	Error opening interface
0x2000 0002	Closing Interface	Error closing interface
0x2000 0003	Interface not Open	Interface is not open
0x2000 0004	Opening Port	Error opening port
0x2000 0005	Closing Port	Error closing port
0x2000 0006	Port not Open	Port is not open
0x2000 0007	Reset Port	Error resetting port
0x2000 0008	Set Port Settings	Error configuring port settings
0x2000 0009	Set Port Mode	Error configuring port mode

Table 9-29 Interface Layer Errors

9.2.2.1 Interface Layer “RS232” Errors

Abort Code	Name	Error Cause
0x2100 0001	Write Data	Error writing data
0x2100 0002	Read Data	Error reading data

Table 9-30 Interface Layer “RS232” Errors

9.2.2.2 Interface Layer “CAN” Errors

Abort Code	Name	Error Cause
0x2200 0001	Receive CAN Frame	Error receiving CAN frame
0x2200 0002	Transmit CAN Frame	Error transmitting CAN frame
0x2300 0003	Rescan	Error rescanning USB device
0x2300 0004	Reload	Error reloading USB device

Table 9-31 Interface Layer “CAN” Errors

9.2.2.3 Interface Layer “USB” Errors

Abort Code	Name	Error Cause
0x2300 0001	Write Data	Error writing data
0x2300 0002	Read Data	Error reading data

Table 9-32 Interface Layer “USB” Errors

9.2.3 Protocol Layer Errors

9.2.3.1 Protocol Layer “MaxonRS232” Errors

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

Table 9-33 Protocol Layer “MaxonRS232” Errors

9.2.3.2 Protocol Layer “CANopen” Errors

Abort Code	Name	Error Cause
0x3200 0001	SdoReceiveFrameNotReceived	CAN frame of SDO protocol not received
0x3200 0002	RequestedCanFrameNotReceived	Requested CAN frame not received
0x3200 0003	CanFrameNotReceived	Can frame not received

Table 9-34 Protocol Layer “CANopen” Errors

9.2.3.3 Protocol Layer “USB” Errors

Abort Code	Name	Error Cause
0x3400 0001	Stuffing	Failed stuffing data
0x3400 0002	Destuffing	Failed destuffing data
0x3400 0003	BadCrcReceived	Bad CRC received
0x3400 0004	BadDataSizeReceived	Bad data received
0x3400 0005	BadDataSizeWritten	Bad data size written
0x3400 0006	SendFrame	Failed writing data
0x3400 0007	ReceiveFrame	Failed reading data

Table 9-35 Protocol Layer “USB” Errors

10 Integration

Consider this chapter as a “How To” on the integration of the library into your programming environment.

The EPOS2 P Command Library is an implementation of protocols to communicate between an EPOS Positioning Controller and a PC running Windows 32-Bit and 64-Bit operating systems. All EPOS commands (including generating/sending/receiving data frames) are implemented and they can be called directly from your own program.

Use the library as an easy and simple way to develop your own application. Do not bother about protocol details; the only thing you need to ensure are the correct communication port settings.

The chapter comprises the following sections:

a) Library hierarchy

b) Integration and programming environment-specific information on how to incorporate the library

c) Programming and a programming environment-specific example on how to configure and establish communication

10.1 Library Hierarchy

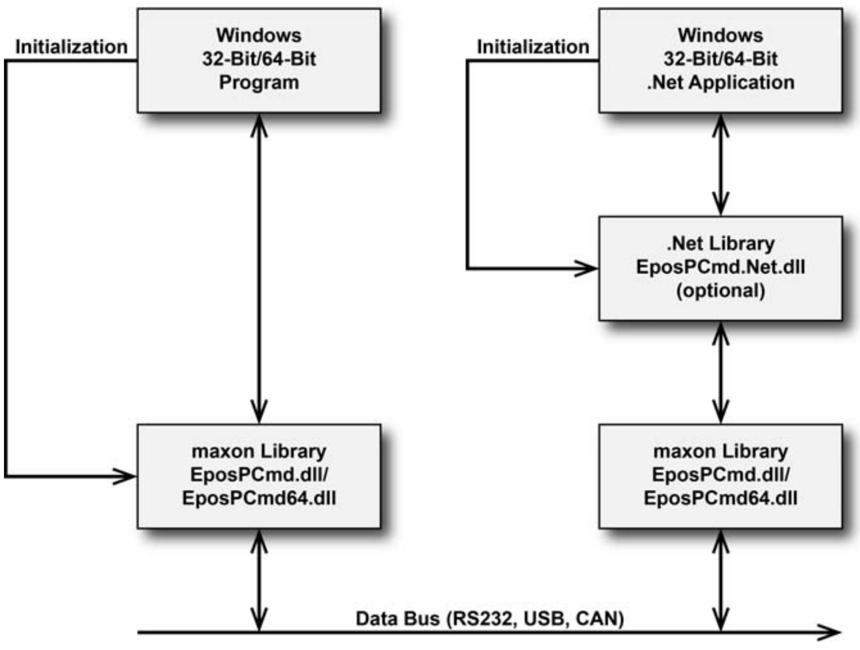

Initialization
Windows 32-Bit/64-Bit Program
maxon Library EposPCmd.dll/ EposPCmd64.dll
Initialization
Windows 32-Bit/64-Bit .Net Application
.Net Library EposPCmd.Net.dll (optional)
maxon Library EposPCmd.dll/ EposPCmd64.dll
Data Bus (RS232, USB, CAN)

Figure 10-4 Windows – Library Hierarchy

10.2 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 “EposPCmd.dll” (for Windows 32-Bit) or “EposPCmd64.dll” for Windows 64-Bit) to your working directory.
2) Use the open functionalities to choose the communication port if the settings are known. Or you may use the open dialogs to select a port.
3) Use the settings functionalities to set baud rate, timeout, and IDs.
4) Close all opened ports at the end of your program.
5) For detailed information on the initialization procedure ➔chapter “10.3 Programming” on page 10-165.

10.2.1 Borland Delphi

You will need to integrate the following files:

32-BIT

- **Win32_EposPCmd_Comm.pas** – Communication functions
- **Win32_EposPCmd_Plc.pas** – PLC/EPOS2 P functions
- **Win32_EposPCmd_Drive.pas** – Drive/EPOS2 functions
- **EposPCmd.dll** – Dynamic link library

64-BIT

- **Win64_EposPCmd_Comm.pas** – Communication functions
- **Win64_EposPCmd_Plc.pas** – PLC/EPOS2 P functions
- **Win64_EposPCmd_Drive.pas** –Drive/EPOS2 functions
- **EposPCmd64.dll** – Dynamic link library

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Write the instructions “Win32_EposPCmd_Comm”, “Win32_EposPCmd_Plc”, and “Win32_EposPCmd_Drive” into the uses clause of your program header.
3) Now, you can execute all library functions in your own code.

10.2.2 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

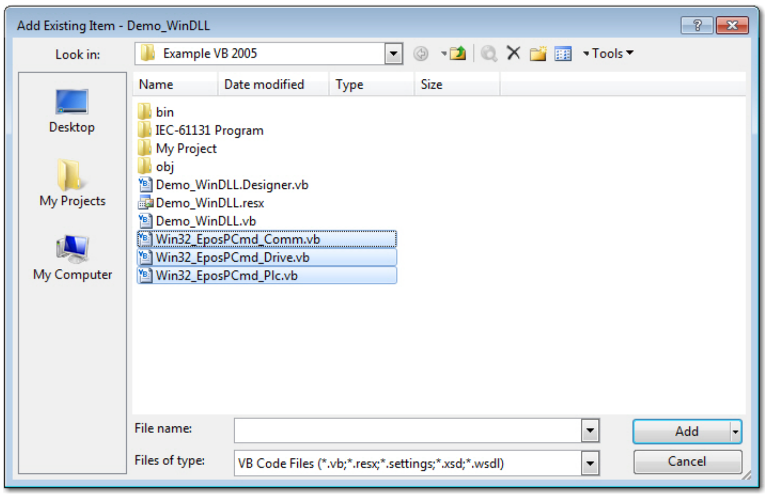
- **Win32_EposPCmd_Comm.vb** – Communication functions
- **Win32_EposPCmd_Plc.vb** – PLC/EPOS2 P functions
- **Win32_EposPCmd_Drive.vb** – Drive/EPOS2 functions
- **EposPCmd.dll** – Dynamic link library

64-BIT

- **Win64_EposPCmd_Comm.vb** – Communication functions
- **Win64_EposPCmd_Plc.vb** – PLC/EPOS2 P functions
- **Win64_EposPCmd_Drive.vb** – Drive/EPOS2 functions
- **EposPCmd64.dll** – Dynamic link library

Proceed as follows:

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


Add Existing Item - Demo_WinDLL

Look in: Example VB 2005

Name	Date modified	Type	Size
bin			
IEC-61131 Program			
My Project			
obj			
Demo_WinDLL.Designer.vb			
Demo_WinDLL.resx			
Demo_WinDLL.vb			
Win32_EposPCmd_Comm.vb			
Win32_EposPCmd_Drive.vb			
Win32_EposPCmd_Plc.vb			

File name:
Files of type: VB Code Files (*.vb;*.resx;*.settings;*.xsd;*.wsdl)

Figure 10-5 Visual Basic – Adding Modules

3) Choose one of the two ways:
   a) Copy the file “EposPCmd.dll” (for Windows 32-Bit) or “EposPCmd64.dll” for (Windows 64-Bit) into the release directory.
   b) Open menu ¤Properties¤, switch to the ¤Compile¤ tab and type “**.**” into the ¤Build output path¤ edit line.

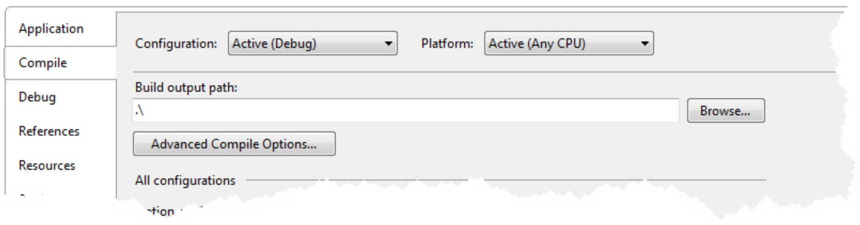

Configuration: Active (Debug)
Platform: Active (Any CPU)
Build output path: .\

Figure 10-6 Visual Basic – Output Path

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

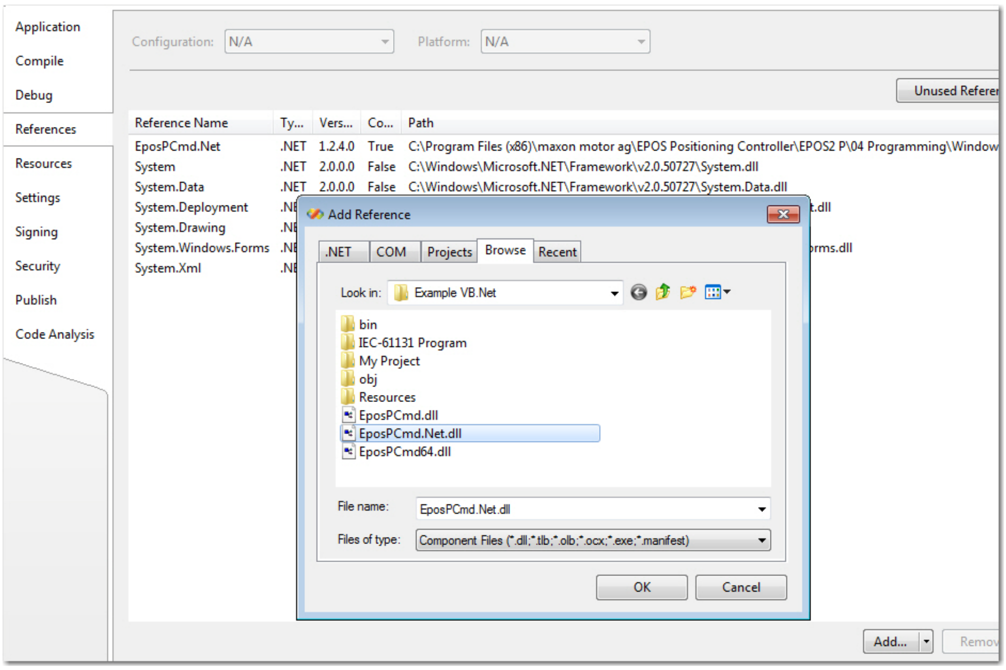
10.2.3 Microsoft Visual Basic .NET

You will need to integrate the following files:

- **EposPCmd.Net.dll** – .Net assembly
- **EposPCmd.dll/EposPCmd64.dll** – Dynamic link library

Proceed as follows:

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



Configuration: N/A   Platform: N/A

Reference Name	Ty...	Vers...	Co...	Path
EposPCmd.Net	.NET	1.2.4.0	True	C:\Program Files (x86)\maxon motor ag\EPOS Positioning Controller\EPOS2 P\04 Programming\Window
System	.NET	2.0.0.0	False	C:\Windows\Microsoft.NET\Framework\v2.0.50727\System.dll
System.Data	.NET	2.0.0.0	False	C:\Windows\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
System.Deployment	.NE			t.dll
System.Drawing	.NE			
System.Windows.Forms	.NE			rms.dll
System.Xml	.NE			

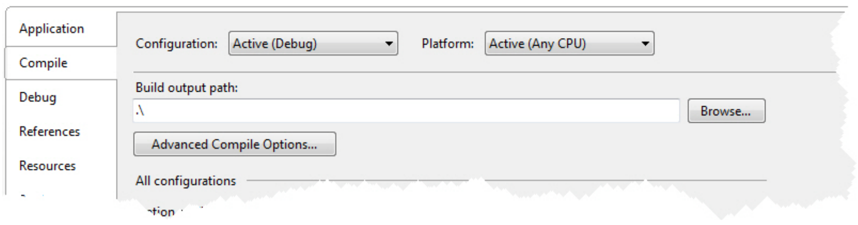
Look in: Example VB.Net

```
bin
IEC-61131 Program
My Project
obj
Resources
EposPCmd.dll
EposPCmd.Net.dll
EposPCmd64.dll
```

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

Figure 10-7 Visual Basic .NET – Adding Reference

3) Choose one of the two ways:
   a) Copy the file “EposPCmd.dll” (for Windows 32-Bit) or “EposPCmd64.dll” for Windows 64-Bit) into the release directory.
   b) Open menu ¤Properties¤, switch to the ¤Compile¤ tab and type “**.**” into the ¤Build output path¤ edit line.



Configuration: Active (Debug)   Platform: Active (Any CPU)
Build output path: .\

Figure 10-8 Visual Basic .NET – Output Path

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

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

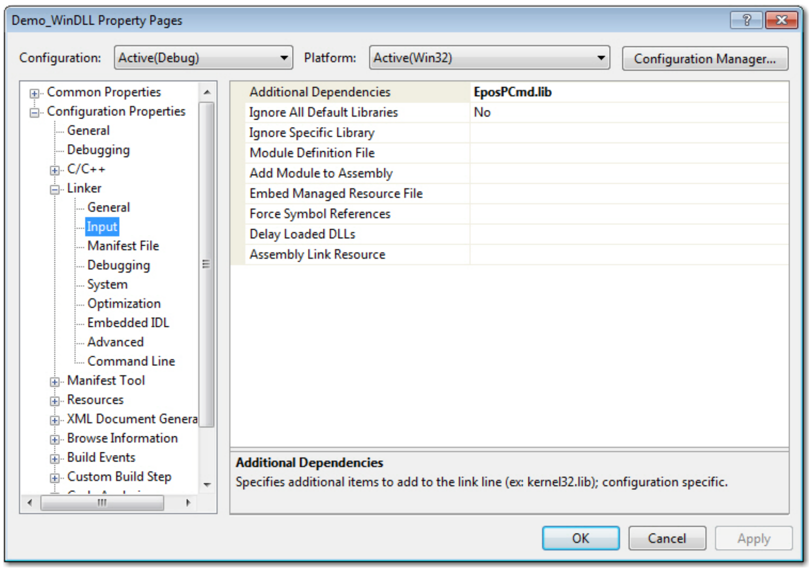
10.2.4 Microsoft Visual C++

You will need to integrate the following files:

- **EposPCmd_Comm.h** – Communication functions
- **EposPCmd_Plc.h** – PLC/EPOS2 P functions
- **EposPCmd_Drive.h** – Drive/EPOS2 functions
- **EposPCmd.dll/EposPCmd64.dll** – Dynamic link library
- **EposPCmd.lib/EposPCmd64.lib** – Import library (COFF format)

Proceed as follows:

1) Copy the files to the working directory of your project.
2) Include the files to your program code using the instruction “#include EposPCmd_xxxx.h”.
3) Add the library to your project using menu ¤Project\Properties¤. Select ¤Linker\Input¤ from the tree and type the file name “EposPCmd.lib”/”EposPCmd64.libl” into the ¤Additional Dependencies¤ edit line.



Additional Dependencies	EposPCmd.lib
Ignore All Default Libraries	No
Ignore Specific Library	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded DLLs	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

Figure 10-9 Visual C++ – Project Settings

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

10.2.5 Microsoft Visual C#

You will need to integrate the following files:

- **EposPCmd.Net.dll** – .Net assembly
- **EposPCmd.dll/EposPCmd64.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 EposPCmd.Net;”.
3) Add the file “EposPCmd.Net” to the project using the project tree in “Solution Explorer”. Click right on ¤References¤, select ¤Add Reference¤, select the file, and click ¤OK¤.

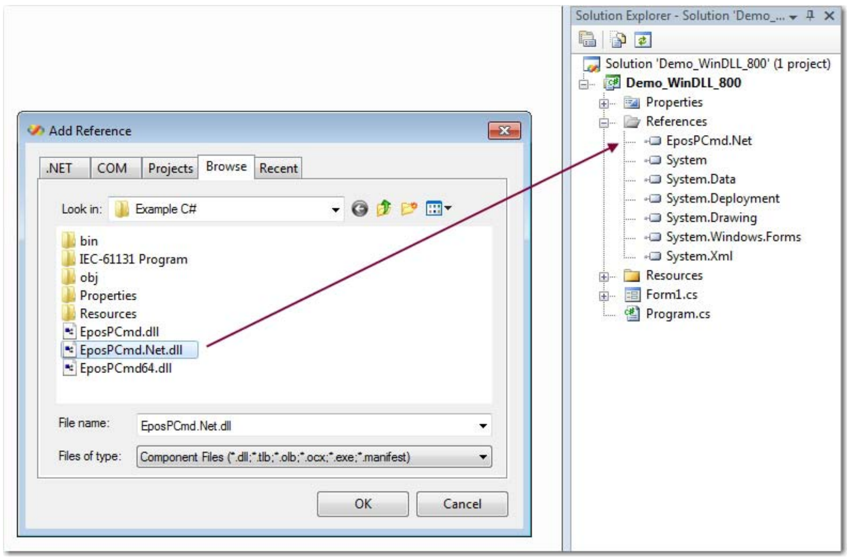


Figure 10-10 Visual C# – Project Settings

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

Remark
For further details and parameter description of the EposPCmd.Net wrapper → separate document «EposPCmd.Net.chm».

10.2.6 National Instruments LabVIEW

LabVIEW offers a function block to include external library functions. For each library function of the EPOS2 P library, a VI will be created.

For an easy start with LabVIEW programming, most of the function blocks are already configured in an instrument driver. Each EPOS command has a VI block:

- 32-Bit VIs are supported with LabVIEW 7.1 and higher
- 64-Bit VIs are supported with LabVIEW 2010 64-Bit and higher

You will need to integrate the following files:

32-BIT

- **EposPLibrary.llb** – LabVIEW library (incl. example)
- **EposPCmd.dll** – Dynamic link library
- *** .mnu** – Menu files

64-BIT

- **EposPLibrary.llb** – LabVIEW library (incl. example)
- **EposPCmd64.dll** – Dynamic link library
- *** .mnu** – Menu files

Proceed as follows:

1) Copy the directory “maxon EPOS P” to the LabVIEW program directory in path “…\National Instruments\LabVIEW X.x\instr.lib”.
2) Make the file “EposPCmd.dll” (for Windows 32-Bit) or “EposPCmd64.dll” (for Windows 64-Bit) available.
3) Include (and use) the maxon EPOS P Instrument Driver VIs via the functions ¤maxon EPOS P¤, ¤Instrument Drivers¤, or ¤Instrument I/O¤.

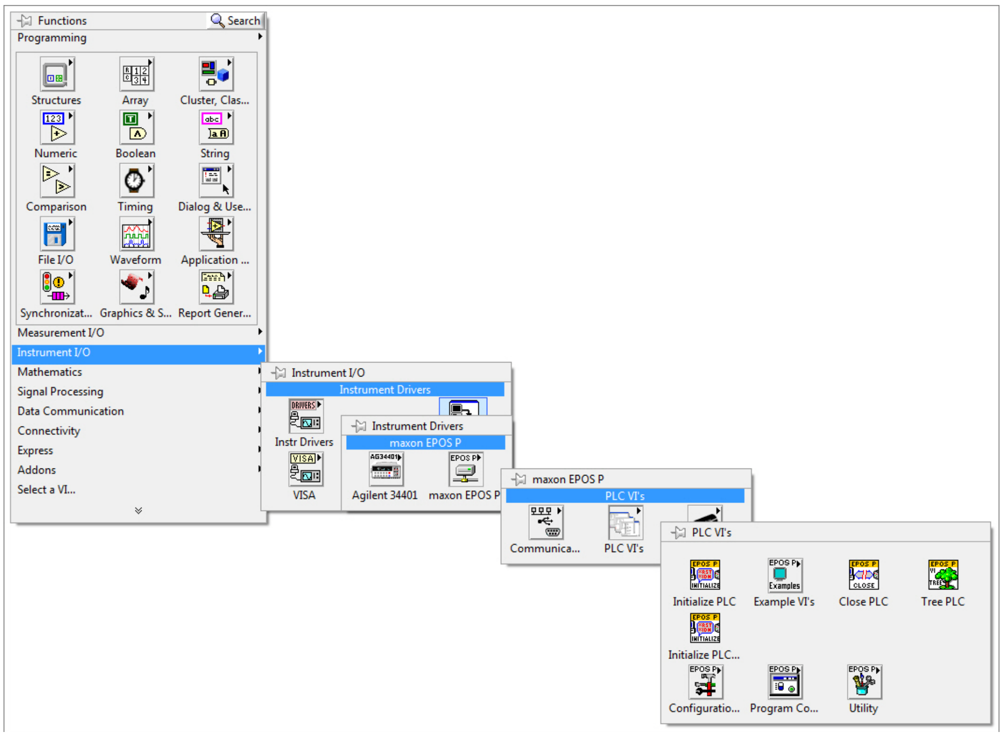


Figure 10-11 LabVIEW – maxon EPOS P Instrument Driver

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

10.3 Programming

For correct communication with the EPOS2 P, you must execute an initialization function before the first communication command. The fundamental program flow is as follows:

Communication Initialization

Use the functions to configure the communication settings.

Function	Description
VCM_OpenCommunication	Configuration of protocol, interface, and port
VCM_SetProtocolStackSettings	Configuration of baud rate and timeout

Gateway Initialization (optional)

You may optionally use the functions to configure the gateway settings.

Function	Description
VCM_OpenGateway	Configuration of remote protocol and gateway device
VCM_SetGatwaySettings	Configuration of gateway node ID and remote network ID

PLC Initialization

Use the functions to configure communication to the PLC/EPOS2 P.

Function	Description
VCM_OpenPlc	Configuration of PLC device
VCM_SetPlcSettings	Configuration of PLC node ID

Drive Initialization (optional)

You may optionally use the functions to configure the communication to a drive/EPOS.

Function	Description
VCM_OpenDrive	Configuration of drive device
VCM_SetDriveSettings	Configuration of drive node ID

Process Control

Choose any of the PLC or Drive methods.

Function	Description
VCM_ColdstartProgram	Start IEC-61131 Program
VCM_SetVariable	Write variable using a symbolic name (i.e TASK1.Var1)
VCM_GetVariable	Read variable using a symbolic name (i.e TASK1.Var1)
VCM_SetProcessInput	Write process input variable (direct variable)
VCM_GetProcessOutput	Read process output variable (direct variable)
VCM_SetProcessImage	Write complete process input image
VCM_GetProcessImage	Read complete process output image

Closing Procedure

Before closing the program, release all open handles in reverse order as they were opened.

Function	Description
VCM_CloseDrive	Release drive device
VCM_ClosePlc	Release PLC device
VCM_CloseGateway	Release gateway
VCM_CloseCommunication	Release communication

10.3.1 Examples

Applicability
The following universally valid example (➔Generic) applies for most programming environments. 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.

Generic

The example demonstrates how to implement a supervisory application in charge of controlling a cyclic movement.

1) Upon starting the application, the following configuration dialog is displayed: Select the communication settings and click ¤Open¤.

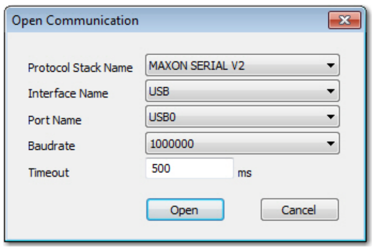

Open Communication
Protocol Stack Name: MAXON SERIAL V2
Interface Name: USB
Port Name: USB0
Baudrate: 1000000
Timeout: 500 ms

Figure 10-12 Generic Example: Open Communication

2) The demo program start screen is displayed, which splits in three parts – Communication, Program Control, and Cycle Application.

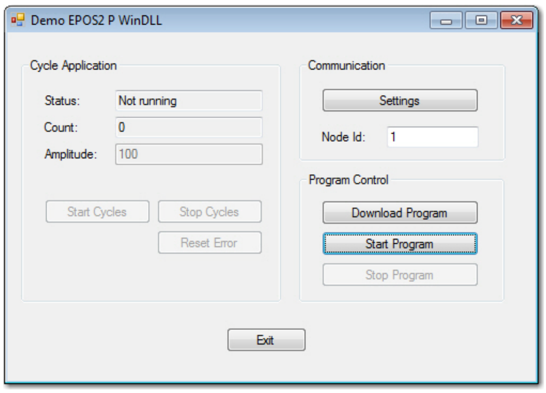

Demo EPOS2 P WinDLL
Cycle Application
Status: Not running
Count: 0
Amplitude: 100
Communication
Node Id: 1

Figure 10-13 Generic Example: “Demo EPOS2 P WinDLL” Screen

3) The configuration comprises the following functions and controls:

Communication

Settings – to define the communication settings (protocol, interface, port, baud rate)

Node Id – to configure the Node ID of EPOS2 P

Program Control

Download Program – to download the IEC-61131 program (after download, select the file “ProgramData.mem” from directory “IEC-61131 Program”)

Start Program – to start the IEC-61131 program

Stop Program – to stop the IEC-61131 program

Cycle Application

Status – to view the status of the cycle application

Count – the number of executed cycles

Amplitude – to enter the amplitude [qc] of the cycle movement

Start / Stop Cycles – to start or stop the cycle movement

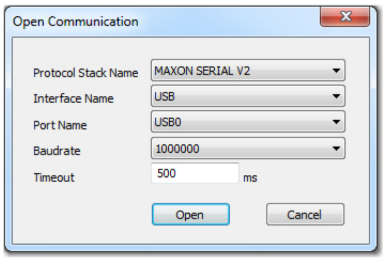
Reset Error – to acknowledge the error detected by the IEC-61131 program

4) Click ¤Exit¤ when done.

LabVIEW

The PLC example demonstrates how to implement a supervisory application in charge of controlling a cyclic movement.

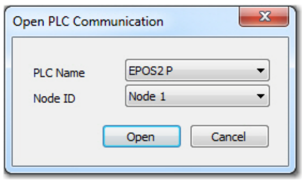
1) Upon starting the application, two configuration dialogs are displayed: Select the communication settings and click ¤Open¤.


Open Communication

Protocol Stack Name	MAXON SERIAL V2
Interface Name	USB
Port Name	USB0
Baudrate	1000000
Timeout	500 ms

Figure 10-14 LabVIEW Example: Open Communication

2) Select the PLC communication settings and click ¤Open¤.


Open PLC Communication

PLC Name	EPOS2 P
Node ID	Node 1

Figure 10-15 LabVIEW Example: Open PLC Communication

3) The demo program start screen is displayed, which splits in three parts – Communication, Program Control, and Cycle Application.

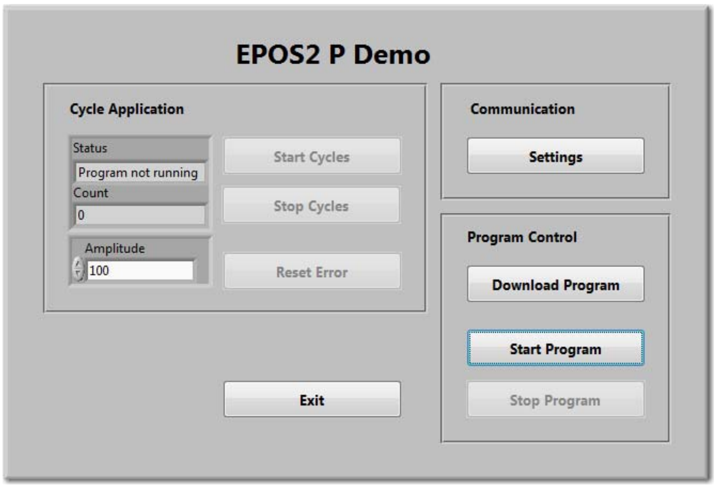


Figure 10-16 LabVIEW Example: “Demo EPOS2 P WinDLL” Screen

4) The configuration comprises the following functions and controls:

Communication

Settings – to define the communication settings (protocol, interface, port, baud rate)

Program Control

Download Program – to download the IEC-61131 program (after download, select the file “ProgramData.mem” from directory “IEC-61131 Program”)

Start Program – to start the IEC-61131 program

Stop Program – to stop the IEC-61131 program

Cycle Application

Status – to view the status of the cycle application

Count – the number of executed cycles

Amplitude – to enter the amplitude [qc] of the cycle movement

Start / Stop Cycles – to start or stop the cycle movement

Reset Error – to acknowledge the error detected by the IEC-61131 program

5) Click ¤Exit¤ when done.

11 Version History

Date [d.m.y]	DLL Version	Documentation Edition	Description
20.10.2014	1.3.1.0	October 2014	Documentation update New: Support for Kvaser CAN interfaces New: Support for NI-XNET driver
17.12.2013	1.2.4.0	December 2013	New functions: VCM_GetHomingState, VCM_WaitForHomingAttained, VCM_GetVelocityIsAveraged, VCM_GetCurrentIsAveraged Changed functions: VCM_SendNmtService replaced with VCM_SendNmtServiceEx
30.10.2013	1.2.3.0	October 2011	Bugfix: VCM_GetVariable using last variable address sent by VCM_SetVariable
01.12.2011	1.2.2.0	December 2011	Documentation update
02.02.2011	1.2.2.0	February 2011	Bugfix: NI-LIN USB device
28.01.2011	1.2.1.0	December 2010	New: Expand to 64-Bit Windows OS Bugfix: Segmented Write
30.08.2010	1.1.1.0	August 2010	New parameters: DialogMode for Findxxx functions
30.04.2010	1.0.1.0	April 2010	Initial release

Table 11-36 Version History

••page intentionally left blank••

Appendix A — Function Groups Overview

••page intentionally left blank••

List of Figures

LIST OF TABLES

Index

••page intentionally left blank••

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

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

www.maxonmotor.com