

EPOS

Positioning Controller

Documentation

DLL Integration into LabVIEW

1 Table of contents

1	Table of contents.....	2
2	Table of figures	2
3	Introduction	3
4	Third party products	4
4.1	IXXAT	4
4.2	Vector	4
4.3	National Instruments	4
5	How to use this guide.....	5
6	Including Library Functions	6
6.1	General Information	6
6.2	Including.....	7
6.3	National Instruments LabVIEW.....	8
7	Programming.....	9
7.1	Fundamental Program Flow.....	9
7.2	National Instruments LabVIEW Example.....	10

2 Table of figures

Figure 1:	EPOS documentation hierarchy.....	5
Figure 2:	Library hierarchy	6
Figure 3:	EPOS library	8

3 Introduction

This documentation "DLL Integration into LabVIEW" explains how the 'Windows 32-Bit DLL' is integrated into a program. This document should simplify the programming of the control software based on Windows.

It is a short overview, which important functions are used and like a program must be developed. Everything is illustrated with the help of a small example. A complete overview of all existing functions is available in the document "Windows 32-Bit DLL".

This documentation is intended to cover most applications in automatisisation. It is based on the experience of maxon motor control. Maxon motor control certifies that the content of this library is correct according his best knowledge.

BECAUSE THIS LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THIS LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THIS LIBRARY IS WITH YOU. SHOULD THIS LIBRARY PROVE DEFECTIVE OR INSUFFICIENT, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION

The latest edition of these documentation "DLL Integration into LabVIEW", additional documentation and software to the EPOS positioning controller may also be found in the internet under <http://www.maxonmotor.com> category <Service>, subdirectory <Downloads>.

4 Third party products

Use one of the listed PC CANopen interface cards. For all of these manufacturers motion control library, example and documentation are available.

All other CANopen products of other manufacturers can also be used, however no motion control library is available.

4.1 IXXAT

CANopen boards from IXXAT operated with an universal driver VCI V2. The Windows DLL works with this universal driver VCI V2 (Version 2.14 and greater) from IXXAT.

See addresses below for ordering CANopen boards.

Distributors

- www.ixxat.de subdirectory <contact>

4.2 Vector

For use of Vector CANopen cards, the 'XL Driver Library' is needed. The latest edition of this 'XL Driver Library' may also be found in the internet under <http://www.vector-informatik.de/english/> category <Hardware Products> subdirectory <Interfaces>. The library has to be installed manually in the appropriate working or system directory. With this library, it is possible to write own CANopen applications based on Vector's XL hardware.

See address below for ordering CANopen boards.

Distributors

- [General distributors](#)

4.3 National Instruments

CAN Interfaces of National Instruments are supported. The NI-CAN Software and Hardware has to be installed.

See address below for ordering CAN boards.

Contact

- www.ni.com/can

5 How to use this guide

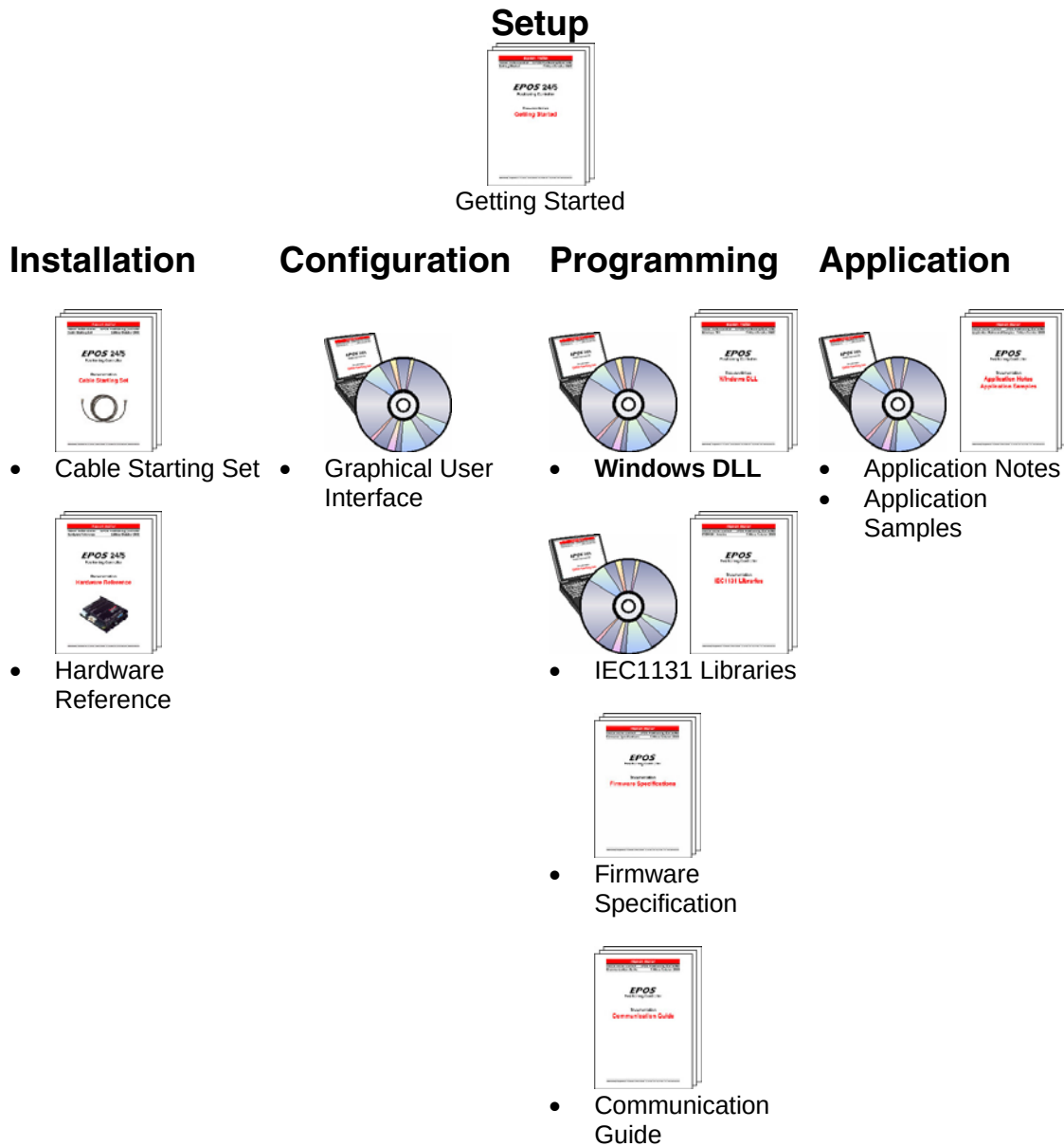


Figure 1: EPOS documentation hierarchy

6 Including Library Functions

6.1 General Information

The library 'EposCmd.dll' is an implementation of the EPOS RS232 and CANopen protocol for the communication between EPOS and a personal computer. Using this library is a simple way to develop your own application. The Windows DLL supports the interface RS232 and CANopen boards from the IXXAT and Vector Company.

This library is running on each windows 32-bit operating system. You can include it into different programming environments. All the EPOS commands are implemented and they can be called directly from your own program. You do not have to care about the protocol details. The only thing you have to ensure is a correct communication setting of the port.

The library 'EposCmd.dll' offers the whole set of EPOS commands. Generating data frames, sending and receiving these frames is the business of this communication library.

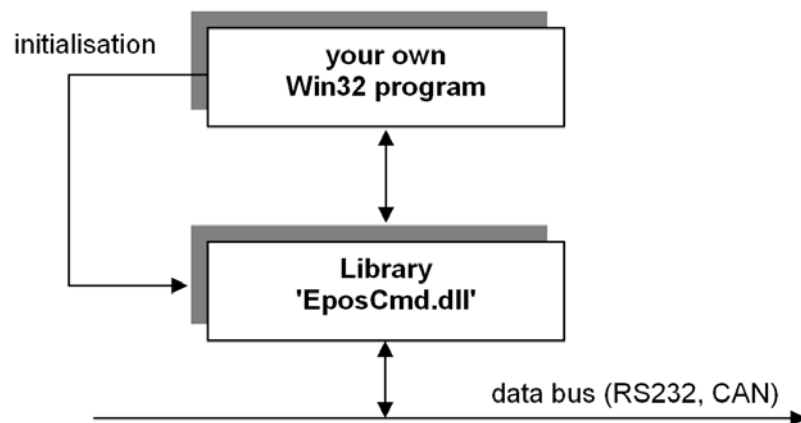


Figure 2: Library hierarchy

For your own program you have to include this library.

6.2 Including

The following chapter describes how to include all library functions in your own windows program. The way in which you do that, depends on the compiler and on the programming language you use. Thus we are going to explain the procedure to include the library based on some examples of the most popular programming languages. For several high-level programming languages are further documentations and its examples available.

In order to have a correctly working communication, you have to include the library

EposCmd.dll

to your programming environment. You have to copy this file to the working directory of your system.

To configure the library 'EposCmd.dll' you have to use the function "VCS_OpenDevice" if the settings are known. In addition, you can use the dialog "VCS_OpenDeviceDlg" to open a port. For a correct communication you have to select the baudrate and the timeout. Use the function "VCS_SetProtocolStackSettings". You have to do this before you can execute any EPOS command!

At the end of your program you have to close all opened ports.

For more detailed information about the initialisation procedure, have a look at the following chapter '[Programming](#)'.

6.3 National Instruments LabVIEW

For integration of the EPOS library LabVIEW offers a function block to include extern library functions. For each library function of the EPOS library a VI is created.

For an easy start with LabVIEW programming, most of the function blocks are already configured in a Instrument Driver (with LabVIEW 7.0 and higher). For each EPOS command we have a VI block.

In order to use the instrument driver the directory 'maxon EPOS' has to be copied into the directory 'instr.lib' of the 'LabVIEW' program directory.

Directory: ...\\National Instruments\\LabVIEW X.x\\instr.lib

The following files and directories have to be available to include the instrument driver.

Directory:

maxon EPOS

Home Directory

Files:

EposCmd.dll

Dynamic link library

EPOSLibrary.lib

LabView Library (incl. Example)

dir.mnu

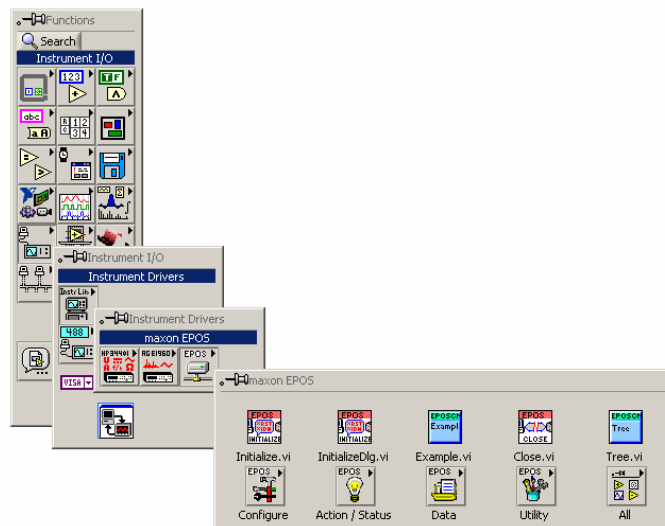
Menu Files

Configure.mnu

Action_Status.mnu

Data.mnu

Utility.mnu



The Vis of the maxon EPOS Instrument Driver can be inserted and used via:

=> Instrument I/O

=> Instrument Drivers

=> maxon EPOS

Figure 3: Instrument Driver maxon EPOS

7 Programming

The following chapter explains how the fundamental programming looks like. For several programming languages we deliver an example program.

7.1 Fundamental Program Flow

To configure the communication with the EPOS correctly, you have to execute an initialisation function before the first communication command. The exact program flow looks like this:

Initialisation procedures

These functions have to be executed at the beginning of the program.

Functions	Description
VCS_OpenDevice (..., ..., ..., ...)	Initialisation of the port with the user data. For some information about the interface settings use the help functions.
VCS_OpenDeviceDlg (...)	Initialisation of the port. The dialog shows all available communication ports (CANopen and RS232)
VCS_SetProtocolStackSettings (..., ..., ...)	Initialisation of the new baudrate and timeout.
VCS_ClearFault (...)	Deletes existing errors.

Help Functions

These functions are needed, if you do not know exactly how your interface is configured.

Functions	Description
VCS_GetDeviceNameSelection (..., ..., ..., ...)	This function returns all available 'DeviceNames' for the function "VCS_OpenDevice".
VCS_GetProtocolStackName- Selection (...)	This function returns all available 'ProtocolStackNames' for the function "VCS_OpenDevice".
VCS_GetInterfaceName- Selection (...)	This function returns all available 'InterfaceNames' for the function "VCS_OpenDevice".
VCS_GetPortNameSelection (..., ..., ..., ...)	This function returns all available 'PortNames' for the function "VCS_OpenDevice".

Communication with EPOS

Choose any of the EPOS commands.

Functions	Description
VCS_OperationMode (..., ..., ...)	Set the operation mode (Position, Profile Position, Current, ...).
VCS_GetEncoderParameter (..., ..., ...)	Read all encoder parameters.
etc.	

Closing procedures

Before closing the program you have to release the port.

Functions	Description
VCS_CloseDevice (...)	Release the opened port.
VCS_CloseAllDevices (...)	Release all opened ports.

How the interfaces are defined exactly, is described in the document "EPOS Windows 32-Bit DLL"!

7.2 National Instruments LabVIEW Example

Set the control parameters before you start this example program (e.g. motor and regulator parameters). Use the graphical user interface for this configuration.

The example 'Demo_WinDLL' in LabView is a dialog based application. The application shows you, how the communication with the EPOS has to be configured.

First a configuration dialog is opened where you adjust your communication settings.

At the beginning the EPOS is set into 'Profile Position Mode'. The whole initialisation is programmed in the member function "Create()" of the class 'Demo_WinDLL'. The opened port is released at the end in the function "Destroy()".

Clicking on the buttons you can execute the EPOS commands:

- "VCS_SetEnableState",
- "VCS_SetDisableState",
- "VCS_MoveToPosition"
- "VCS_HaltPositionMovement".

The function "VCS_MoveToPosition" can be used as absolute or relative positioning. Clicking the button "Device Settings" you can change your communication settings.

A timer is triggering a periodical update of the state and the actual position. Every 100ms the function "UpdateStatus()" is executed. If an error occurs during the update of the state the timer is stopped and an error report is displayed.