maxon

First steps in motion control

A practice with maxon ESCON and EPOS4 controllers

Urs Kafader, Walter Schmid, Stefan Enz

First Edition 2020

Version 1.01, January 2021

maxon ESCON and EPOS4 practice

First steps in motion control

Intention and approach

The approach of this textbook is a practical and experimental one. Instead of first explaining the theory of motion control and then applying it to specific examples, here we will start with hands-on experimenting. Our investigation is carried out by means of the *ESCON* or *EPOS Studio* software on a maxon *ESCON* or *EPOS4* positioning control system. We explain all the relevant motion control principles and features as they appear on our journey. Therefore, the text contains mainly the exercises and practical work to do. Background information can be found in the colored boxes. I would like to stress that “playing” with and experiencing the system are important aspects of learning. Hence, widely use the suggestions for additional exercises!

Motion control is mechatronics. It is the combination of mechanics and electronics, of actuators (motors) and sensors all controlled by software. In this textbook, we identify and define the role, behavior and mutual interaction of the different elements of a motion control system. However, we need to know the basic internal construction and working principles of the elements only to a limited extent, but rather look at them as a black box. Given a certain input, what comes out of the box? How is this output generated and which parameter can be used to influence the output? For instance, we will not explain how feed forward control must be implemented, but we will explain the basic idea of it and how the parameters will look during axis tuning.

Boxes

The color-coded boxes contain additional information about motion control and programming in general and about the hardware and software at hand.

Motion Control Background
Gives general information and knowledge on motion control concepts.

ESCON or EPOS Info
Gives additional short input for better understanding of special features and behavior of the *EPOS4* system.

Best Practice
Gives tips and hints how to use the *EPOS4 and* the *EPOS Studio* software in the most efficient way.

Table of Content

Part 1: ESCON Preparation

The next two chapters serve to get to know the system, and to prepare the workbench.

In chapter 1, we will provide an overview of the hardware, go over the installation of the *ESCON Studio* software and briefly explain the main elements of the motion control system.

In chapter 2, the motion control system is prepared for the exercises that will follow. The properties of the motor and encoder must be defined, and the control loop tuned.

1 Setting-up the ESCON Starter Kit

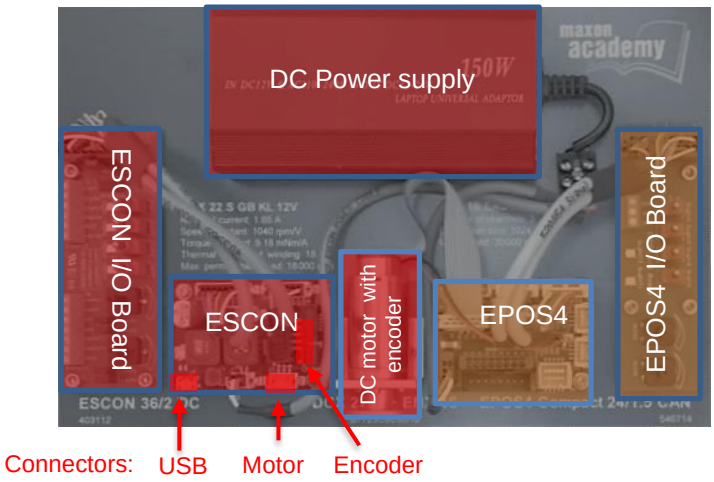


Figure 1: The content of the ESCON/EPOS4 Starter Kit.
The components used for the ESCON training are highlighted in red.

In the Starter Kit box, there is a DC motor with an encoder, and on the left an *ESCON 36/2 DC* controller connected to a printed circuit board (PCB) with switches, LEDs, and potentiometer on it. This PCB serves as an analog and digital input and output (I/O) device. You will also find a DC power supply and the necessary cables below the main board.

Besides, there is an EPOS4 position controller with its own I/O printed circuit board placed on the right side of the main board. The EPOS4 part will be treated from Part 3 on of this textbook.

1.1 Installation of ESCON Studio software

First, prepare the computer. Install the software from the maxon ESCON website: https://escon.maxongroup.com. On the product page, click on the *Download ESCON Setup* button.

The installation contains all necessary information and tools (ESCON Studio, documentation and firmware) required for setting up and exploiting the maxon *ESCON servo controllers*. The *ESCON Studio* is the user interface that allows access to the features and parameters of the servo controller.

Latest *ESCON Studio*
In the case that you have already the ESCON Studio installed, check the version. Do so by clicking menu *Help*, then select menu *About ESCON Studio*. If your version is older than the one on the maxon website (https://escon.maxongroup.com), uninstall the old version and download the latest version from the maxon website.
After installation, verify that you are using *ESCON Studio 2.2* (Revision 5) or higher.

Installation

Step 1 Follow the instructions given when running the installation program.
Please read every instruction carefully. Indicate location of working directory when prompted.

Working directory
We recommend the following location as a working directory:
C:\Program Files (x86)\maxon motor ag
Note that the designation of the program directory may vary depending on the system language installed.

Step 2 There will be new shortcuts and icons in the start menu of your computer.
The files have been copied to the menu *maxon motor ag*, where you can access the program, as well as the entire documentation set.
Clicking the *ESCON Studio* shortcut on your desktop will launch the program.

ESCON Studio Language selection
The ESCON Studio supports different languages selected during the installation of the EPOS Studio. However, you can change it later. In the File menu select Options and then Language or the corresponding word in the current studio language. Select the language from the drop-down list and restart the *ESCON Studio* for the language to be activated.

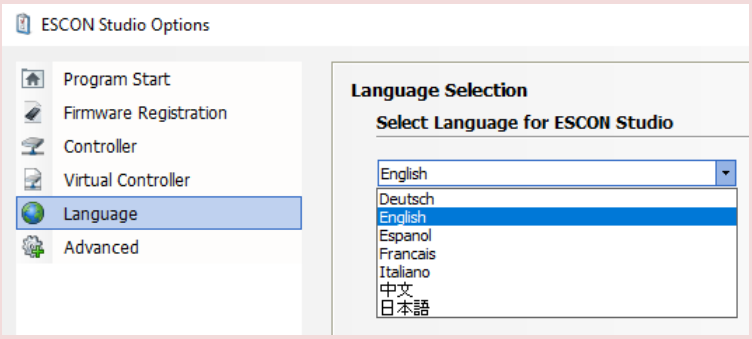


Figure 2: Language selection

1.2 Connecting the cables

The cabling is easy and basically mistake proof. First connect the DCX motor (2 wires: red, black with a tiny black connector) into the only white socket that fits. Then do the same with the encoder flat ribbon cable. The PCB with the I/Os is already connected. Then – and it is good engineering practice to do this last - connect the power supply. Finally, fit the *USB* cable to one of the *USB* outputs on your computer and to the *ESCON Micro USB* plug. Your setup should then look similar to Figure 3.

Figure 3: How to connect the ESCON Starter Kit.

1.3 Starting the ESCON Studio

Connect the ESCON to your computer by USB. Start the *ESCON Studio* by double-clicking the corresponding shortcut on your desktop. The *Startup Wizard* is launched automatically. Exit it by clicking on *Cancel*.

Check the Active Controller displayed in the drop-down box at the top. It should read *ESCON 36/2 DC - USB0* as in Figure 4, maybe with a different USB port number. If there is a different active controller displayed select *ESCON 36/2 DC - USB0* from the list.

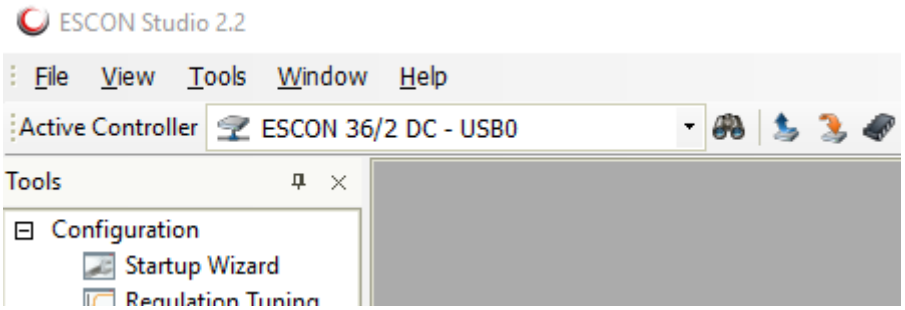


Figure 4: Active Controller drop-down menu.

1.4 Firmware Update Wizard (optional)

Objective: Download the actual firmware version to the ESCON.

Check the actual firmware of the ESCON (click on *Help – About ESCON Controller*) and compare with the latest available in the standard file directory
C:\Program Files (x86)\maxon motor ag\ESCON Servo Controller\ESCON 36-2 DC\Firmware

If the actual firmware should not be the latest available start the *Firmware Update* wizard in the *Configuration* section of the *Tools* window on the left

Step 1 Read the warning and confirm. Click Next.
Step 2: Select the Update Options for the firmware and parameter update mode. (Typically use the default settings: Update with latest installed firmware and Default parameters) Click Next.
(Step 3 Select the new firmware version. Click Next.)
Step 4 Start the download with the Start button.
Step 5 After the successful Firmware update, click on Finish.

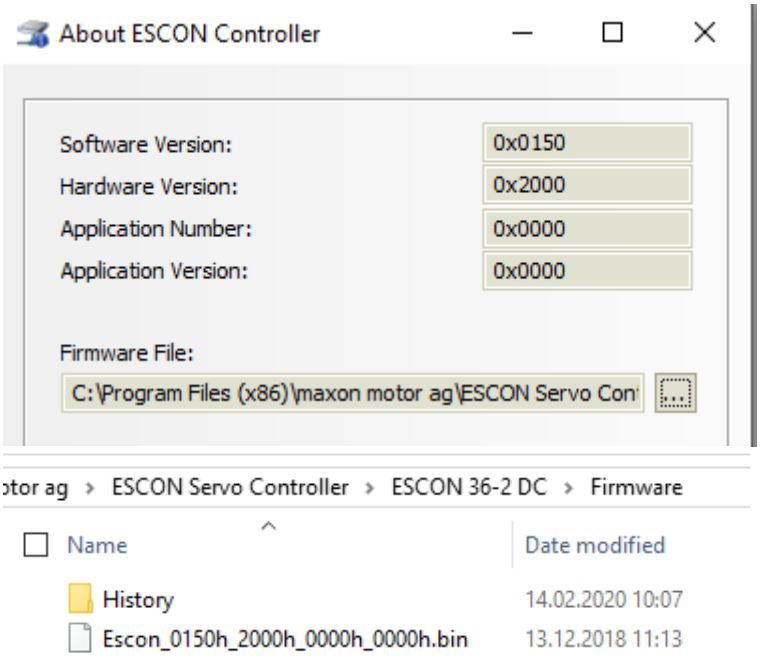


Figure 5: Firmware version comparison.
Top: the actual Firmware version of the ESCON as shown in the Help menu
Botton: the latest Firmware version available
Both show the same software and hardware version.

Remark: Check also the ESCON Video Tutorial about the Firmware Update Tool in the *Videos* tab of the product page of the ESCON 36/2 controller on www.maxongroup.com.

1.5 Startup Wizard)

Objective: Configure the *ESCON 36/2 DC* system in *Closed Loop Speed Control* for this particular DC motor with incremental encoder feedback. Set up an external *Enable* switch and an external potentiometer as an *Analog set value*.

Step 1 *Startup Wizard*: If the Startup Wizard should not start automatically, double click on *Startup Wizard* in the *Tools* on the left.

Step 2 *Safety Instructions*: Read the safety instructions and confirm. Press *Next*.
Remark: By clicking on *Help*, the corresponding section of the document *Hardware Reference* opens.

Step 3 *Introduction*: Verify that the controller description matches the one you have connected. Press *Next*.

Step 4 *Motor Data*: Enter the motor parameters of the DCX 22 S motor. You find the relevant motor data directly on the board or refer to Appendix 12.1 (Motor and encoder data sheets). Press *Next*.

Step 5 *System Data*: Enter the system data. Press *Next*.
Observe: Speed and current (or the corresponding torque) can be limited by any component of the drive system: Motor, gearhead, mechanics, power supply…. In our case, the maximum speed and the nominal current are motor limits; the max. current is limited by the ESCON controller.

Step 6 *Speed sensor*: Select *Digital Incremental Encoder*. Again, you find the *Encoder Resolution* (1024cpt) directly on the board. *Encoder Direction* is maxon standard. Press *Next*.

Step 7 Mode of Operation: Select Speed Controller (Closed Loop) with Inner Current Control Loop as the mode of operation. Press Next.

Step 8 *Enable*: From the drop-down menu select *Enable* and chose the *Digital Input 2* (high active) for the enable functionality. Press *Next*.

Step 9 *Set Value*: Select *Analog Set Value* and *Analog Input 1* for the set value functionality. Enter the desired scaling of the set value, e.g. minimum speed - 1000 rpm and maximum speed + 1000 rpm.
Observe: The voltage range of the external potentiometer (*Analog Input 1*) is approx. 0...5 V. Press *Next*.

Step 10 *Current Limit*: Select *Fixed current Limit* for our simple configuration and leave it at 4 A. Press *Next*.

Step 11 *Speed Ramp*: Select *Fixed Ramp* for our configuration and enter the desired acceleration and deceleration values, e.g. 500 rpm/s. (This is a rather low value, but it will allow us to better follow the acceleration) Press *Next*.

Step 12 *Offset*: For this basic configuration, leave the fixed offset value at zero rpm. Press *Next*.

Step 13 Digital Inputs and Outputs: Select the functionality Stop for the Digital Input 1 Press Next.

Step 14 *Analog Inputs*: For this basic configuration, there is no additional change of the analog input functionality required. Press *Next*.

Step 15 *Analog Outputs*: For this basic configuration, there is no change of the analog output functionality required. Press *Next*.

Step 16 *Digital Input 1 - Stop*: Set a relative high *Deceleration* rate, e.g. 20’000 rpm/s. Press *Next*.

Step 17 *Configuration Summary*: Check your settings.

Step 18 *Wiring*: Tick Show Wiring Overview (or open it via menu *File* ➔ *Wiring Overview*).
Remark: Additional information can also be found in chapter 4 of the document *Hardware Reference*.

Step 19 *Finish configuration*: Press *Finish* to save your configuration settings in the ESCON.

Motor and encoder parameters of the unit at hand
The motor used is a *DCX 22 S*, 12 V nominal voltage, with graphite brushes and ball bearing. Its exact specification can be found on the maxon website or in the Appendix. Important for us is the fact, that it is a brushed DC motor.
This motor has a maximum speed limit of 18 000 rpm which is lower than what we can achieve with the given voltage of the power supply (24 VDC). Typically, the motor is able to reach speeds of up to approximately 24'400 rpm at 24 V supply (speed constant of motor times motor voltage = 1040 rpm/V * 0.98 * 24V).
The nominal current is 1.65 A. The motor can draw that current continuously without overheating. For a few seconds it can support a much higher current, possibly up to 10 A. However, we are not able to get as much current out of our ESCON 36/2 DC, which exhibits a current limit of 4 A.

The encoder type is *ENX 16 EASY encoder* with 1024 counts per turn on each of the two channels. This results in 4096 signal edges (states) per turn of the motor shaft which are called increments (inc). The inc are the internal position units used by the EPOS controllers. Hence, we have a nominal resolution of 1/4096 of a turn or 0.088°.

1.6 Regulation Tuning Wizard

Remark: Check also the ESCON Video Tutorial in the *Videos* tab of the product page of the ESCON 36/2 controller on www.maxongroup.com about the *Auto Tuning Mode*. If you run into error during the tuning, try the *Diagnostics wizard* (refer to chapter 5.1)

Objective: Learn how auto tuning is done in ESCON systems. Prepare the optimum system reaction to set value commands.

Step 1 If the *Regulation Tuning* does not start automatically after configuration, double-click on *Regulation Tuning* in the *Tools* window on the left.
Step 2 *Tuning Type*: Choose the *Auto Tuning* method. Press *Next*.
Step 3 *Auto Tuning*: Click *Start*. Consider the warning message carefully. Make sure that the motor shaft is free running. Click *Yes* to initiate Auto Tuning.
Step 4 Upon successful completion of the Auto Tuning all status bars change to green. Click *Finish* to permanently save the tuning parameters.

Your system is now ready to operate. Activate on the *Enable* input switch to run the motor and change the speed with the set value potentiometer.

ESCON Auto Tuning
For each control loop (current, closed loop and open loop speed), a two-step procedure is executed in turn:

- First, optimum control parameters are identified. During this process, the motor shaft oscillates. The corresponding red status bars will be moving
- Second, the identified control parameters are verified by evaluating a step response. The motor shaft moves, and the moving status bars change to green. The result of the evaluation is shown graphically.

Option: Expert Tuning

Expert tuning is not part of this basic ESCON training. However, feel free to explore it as well. Check also the ESCON Video Tutorial in the *Videos* tab of the product page of the ESCON 36/2 controller on www.maxongroup.com about the *Expert Tuning* mode.

Step 1 Start the *Regulation Tuning Wizard* by double-clicking on *Regulation Tuning* in the *Tools* window on the left.
Step 2 *Tuning Type*: Select Expert Tuning. Press Next.
Step 3 Tuning Mode: Select Speed controller (Closed Loop). Press Next.
Step 4 *Expert Tuning*: Activate both *Identify* check boxes. The green *Speed* and *Current* loop indicators at the top will turn red. Click *Start*. Consider the warning message. Click *Yes* to initiate the tuning. *Expert Tuning* will now commence (see box below).
Step 5 Set different *Controller Stiffness* and observe how the parameter values and the result of the verification changes. Observe: there is no need to repeat the *Identification* step.
Step 6 Once your satisfied with the tuning result, click *Finish* to save the tuning parameters.

Your system is now ready to operate.

ESCON Expert and Manual Tuning

Expert tuning

For the current and speed control loop, a three-step procedure is executed in turn:

- First, the properties of the system (friction, inertias) are evaluated. During this process, the motor shaft oscillates. The corresponding red status bars will be moving. In *Expert tuning* you can influence this step by setting an identification *Amplitude*.
  Remark: Do not select the amplitudes too large, because this will cause strong system reaction. However, the selected speed amplitude should reflect a typical speed change of your application.
- Second, optimum control parameters are calculated from the result of the first step. This parameterization can be influenced by setting the *Controller Stiffness* sliders more to the soft or hard side. There is no need to repeat the identification step when evaluating different parameterizations.
  Remarks: You can check the calculated control parameters by clicking on *Show Parameters*. Here you could also manually change the parameters. The exact meaning of the ESCON control parameters is intellectual property of maxon and not disclosed to anyone.
- The final step verifies the tuning by executing a step response. The result is displayed in the *Verification* window. Upon successful completion of the tuning all status bars change to green. .
  Remark: You can zoom into the verification curves by selecting the interesting area with your mouse. Right click resets the original diagram.

Manual tuning

Should you not be satisfied with the tuning result or run into a problem or error, you can manually change the controller parameters. The *Start* command then executes a verification test with the manually set parameters. (Of course, no identification and automatic parameterization will be done)

Part 2: Exploring ESCON Functionalities

In Part 1 we have configured the system, tuned the motion controller and saved all the settings on the device. The *ESCON* servo controller is now ready for further exploration.

- Chapter 2 goes through the main functionality of the ESCON Studio including monitoring the control behavior and parameter setting.
- Chapter 3 shows special tools for diagnostics and explores the inputs and outputs associated with the motion controller.
- Chapter 4 explores alternative operation modes
- Chapter 5 gives information about special tools.

The *ESCON Studio* is a graphical user interface (GUI) for setup, parameterization and monitoring the performance of the ESCON. And the USB communication interface is just meant for these purposes; it is not a serial communication to a master system. The operation of the ESCON is controlled by its digital and analog inputs (see chapter 2.1).

All the following exercises are just starting points for your own experiments. My recommendation: Just play!

2 Operation

Objective: Learn what you can see on the *Controller Monitor* of the ESCON Studio.

2.1 Stand-alone operation

Having performed the *Startup Wizard* and the *Regulation Tuning*, your ESCON is ready to operate. As mentioned above, the ESCON is controlled by its I/Os only. Let us explore how the configured inputs can be used to operate the motor.

Step 1 Disconnect the USB connection to your PC.

Step 2 *Enable* the power stage of the ESCON by activating the *Digital Input switch 2* (second from top on the I/O board, next to the green LED). For permanent enable move the switch to the right. The motor starts spinning.

Step 3 *Disable* the motor by deactivating the *Digital Input switch 2.* The motor wheels free and stops.

Step 4 Enable the motor again. Change speed and direction with the *Analog Input 1* potentiometer on the I/O board. Can you bring the motor to standstill with the set value? Observe the slow reaction due to the low ramp settings.

Step 5 Set a high speed. Activate and deactivate the *Stop* function with the *Digital Input switch 1.* Deceleration to standstill is very fast because of the high deceleration ramp setting.

Step 6 Set a high speed and increase the load by slightly pressing your finger on the motor disc. What happens to the speed?

Step 7 Disable the motor.

2.2 Controller Monitor: Overview Tab

The *ESCON Studio* allows to monitor all kind of settings and parameters. For this, reconnect the USB communication.

Step 1 On the left in the ESCON Studio, select *Controller Monitor*. At the bottom activate the *Overview* tab.

Step 2 On the I/O PCB, activate the digital input switch 1 (*Stop*) and observe how its representation on the *Controller Monitor* changes.
Turn the analog input 1 potentiometer (*Set value*). Observe how the input voltage reading and the corresponding speed set value changes.

Step 3 *Enable* the ESCON (digital input switch 2 on the I/O-PCB) and set different speed values. Observe the reaction of the motor and the readings on the left panel (*Inputs/Outputs*) and on the right panel (*Motor/Sensors*) .

Step 4 How does the motor react to fast set value changes? Change the acceleration and/or deceleration ramps in the central panel (*Controller*) to very low (100 rpm/s) or much higher (10'000 rpm/s) values. How fast is the motor reaction to abrupt set value changes? Follow the reaction of the motor shaft and the speed reading in the *Controller Monitor*.

Step 5 Apply a little friction to the motor shaft with your finger. How does the motor current react? What happens to speed?

Step 6 Increase the friction or even block to motor. What do you feel on the motor shaft? How big is the current reading? What do the status indicators show in the central panel of the *Controller Monitor*?

Step 7 Activate the digital input 1 (*Stop*) and observe the motor reaction. Has the deceleration ramp setting any influence on the stopping duration?

2.3 Controller Monitor: Controller Tab

The *Controller* tab of the *Controller Monitor* shows the inputs and parameters regarding the control in detail. Repeat the steps from above and compare the reaction on the display.
As an option, explore the parameters on this monitor. Which parameters can be changed?

2.4 Controller Monitor: Properties Tab

Objective: Learn how you can change configuration parameters without working through the *Startup Wizard*.

Step 1 Activate *Properties* at the bottom of the *Controller Monitor*. The *Properties* dialog window opens.
Observe: The ESCON needs to be disabled for changing configuration parameters.

Step 2 Change some of the parameter. E.g. the *Set Value* speed range, *Speed Ramp*, or the stop deceleration value in the *Digital Inputs and Outputs*. Press *OK* to confirm and to save the new configuration settings.

Step 3 Operate the motor and verify on the *Controller Monitor* the new settings.

Operation* tool: *Parameters
The *Operation* tool *Parameters* give some basic read-only information about the device (name, serial and version numbers) as well as setting and actual current and speed values.
The full list of parameters can be made available by changing the *User Level* to *Expert* within the *Advanced Options*.

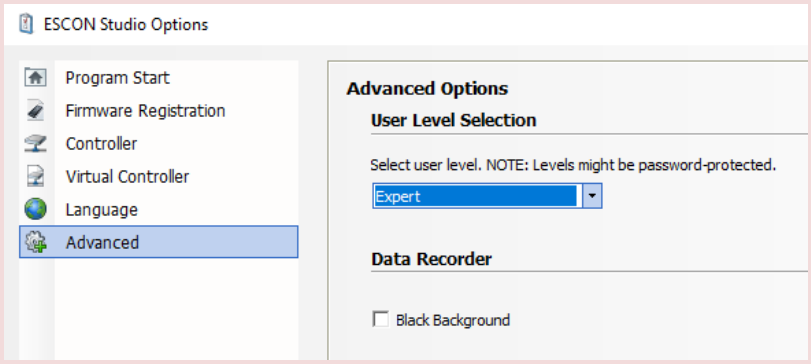


Figure 6: Advanced options

2.5 Data Recorder and Speed

Objective: Learn how set up the *Data Recorder* tool and how to interpret the displayed speed information.

Setting up the Data Recorder for speed measurement.

Step 1 Open the *Data Recorder* in the *Tools* section on the left of the *ESCON Studio* (double-click) or select from the *Tools* main menu.

Step 2 Click on the *Settings* button on the lower right.

Step 3 Activate channels 1 to 3. Click on the corresponding buttons. Select channel 1 to display *Demand Speed*, channel 2 to show the *Actual Speed Averaged*, and channel 3 *Actual Speed*.

Step 4 For all three channels, deactivate the *Auto Scale* checkboxes and set the minimum scale to 850 rpm and the maximum to 950 rpm.

Step 5 In the *Data Sampling* section on the right, select the minimum possible Sampling Time (0.187 ms). This corresponds to a sampling rate of the data recorder equal to the sampling rate of the speed control loop.

Step 6 Set the Trigger Configuration to Continuous Acquisition. Press OK.

Evaluation of the speed information in the Data Recorder.

Step 1 Set a speed of about 900 rpm and enable the ESCON.

Step 2 Examine the signals on the Data Recorder screen. You might see a picture like in Figure 7, upper diagram. Observe the scale information at the bottom.

Step 3 *Stop* the recording, click on *Actual Speed* in the *Available Curves* section on the right (What happens if you deactivate and reactivate the checkbox?). Change the Scale value of the Actual Speed to 25 rpm/div and confirm with Enter. (You can leave the Offset as it is.) The picture may now look like the lower diagram in Figure 7.

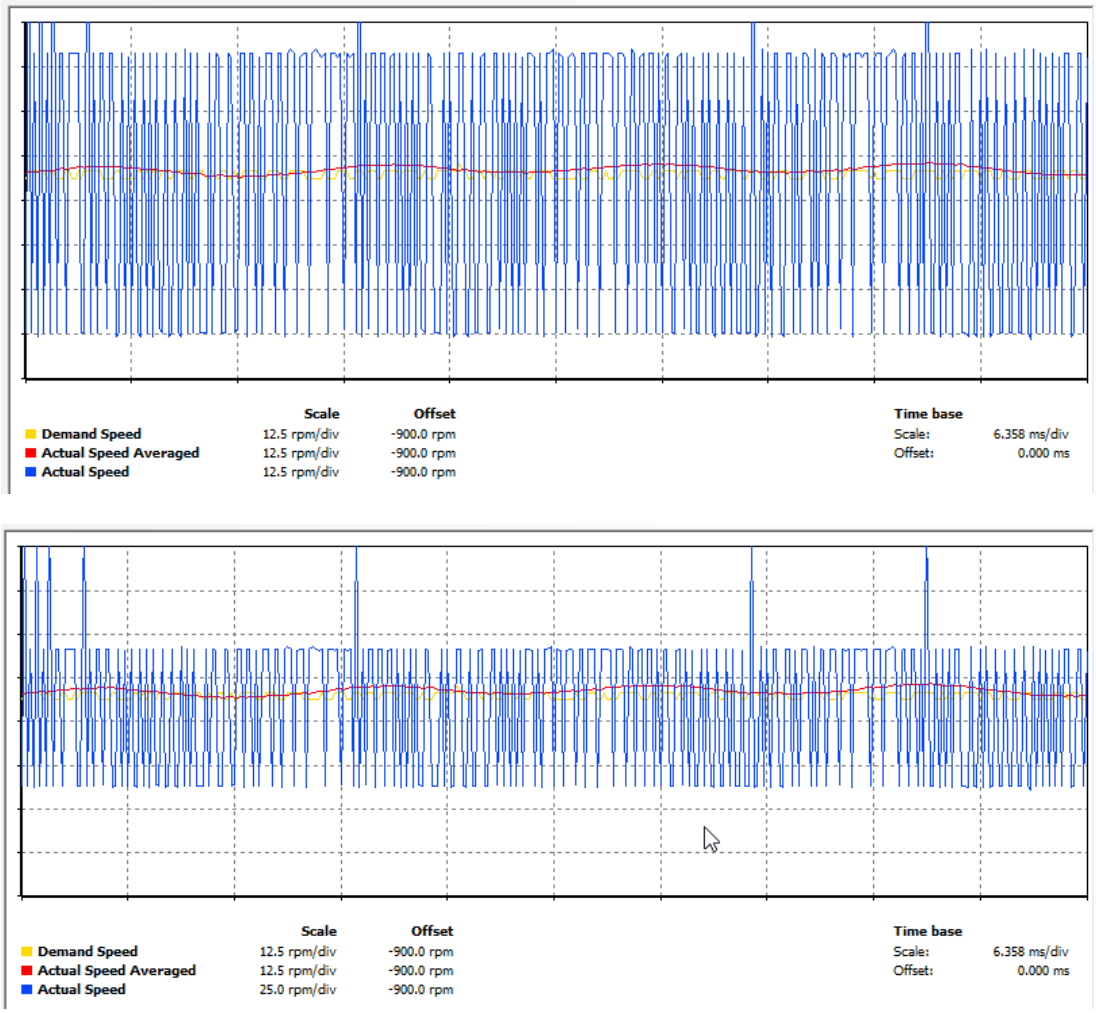


Figure 7: Speed signals on ESCON Data Recorder.
Top: All curves with the same scaling
Bottom: Actual Speed scaling changed.

Interpretation of the curves in the diagram

- The *Demand Speed* (yellow curve) is almost constant, having spikes of about 4 rpm. These spikes reflect the noise of the external potentiometer acting as the set value source. As soon as the noise exceeds the resolution of the 12-bit AD-converter we see a spike.
- The *Actual Speed* (blue curve) seems to jump between 3 main levels separated by a little bit more than 3 divisions of 25 rpm (>78 rpm). This cannot be real speed changes; the motor cannot react that fast. Instead, what we see is the effect of speed evaluation from a digital incremental encoder (refer also to the boxes on page 45 and on page 71). Speed can only be measured in steps of +/- 1 encoder increment per sampling period, i.e. 1 inc per 0.187 ms or about 320'000 inc/min. The encoder used here has 4096 inc/turn resulting in the observed speed resolution of about 78 rpm.
- Hence, the *Actual Speed Averaged* (red curve) is a better indicator for the real speed. It shows four fluctuations per motor turn of about +/-3 rpm. At 900 rpm one motor turn takes 67 ms; that's about the full displayed time range. Four fluctuations per turn are typical for the EASY encoders used here. Therefore, the red curve rather reflects the properties of the encoder than the real speed variations.

ESCON Data Recorder display options

Background color
For better visibility, you can change the background color of the display to black with right mouse button on the display area: *Swap Background Color*.

Cursor
On the right in the *Data Recorder* window you can turn a cursor on and off. The values at the curser position are also displayed below the diagram.

Data Recorder in single trigger mode

In an analogous way, the *Data Recorder* can be set up for all kind of signals. The procedure comprises the same main steps as above. Let us exercise this to explore system reaction upon activating the *Stop* input.

Step 1 Open the Settings in the Data Recorder
Step 2 Activate the channels 1 to 3 and select channel 1 to display Demand Speed, channel 2 to show the Actual Speed Averaged, channel 3 is Actual Current Averaged.
Step 3 For all channels, activate the *Auto Scale* checkbox.
Step 4 Select the *Sampling Time* (approx. 5 ms). The total time depends on the number of activated channels and the selected sampling time.

Step 5 Set the *Trigger Configuration* to *Single Trigger* and define the type as *Digital Signal* with *Stop* from *Running->Stopped* as the trigger mode. Set a *Time Delay* of a few percent or about 500 ms. Press *OK*.

Step 6 Enable the ESCON and set a relatively high speed. You can see that the *Data Recorder* is constantly recording the selected parameters.

Step 7 Activate the *Stop* switch. The recording stops and displays the selected parameters. The *Demand Speed* drops to zero very fast as requested from the high deceleration ramp for the Stop input. The *Actual Speed Averaged* curve shows that the motor takes much longer to really stop. It looks like the actual current being rather small.

3 Inputs and outputs

In this chapter we explore some of the input and output functionalities without being exhaustive.

3.1 Enable options

Up to present, we have used a simple input for enable and disable. The functionality is to connect and disconnect the power stage of the ESCON to the motor. There are other possibilities involving direction of rotation commands.

Enable & Direction

Enable & Direction allows to set additionally the rotation direction of the motor.

Step 1 Open the *Properties* tab of the *Controller Monitor*.

Step 2 Click on *Enable* in the left menu and select *Enable & Direction* from the drop-down. The *Enable* is still on *Digital Input 2*, the *Direction* input is on *Digital Input 3.* Press *OK* to save the new settings.

Step 3 Observe the motor behavior (speed and direction) for the following questions and watch the display on the *Controller Monitor*.
- How does the motor shaft rotate for positive or negative speed set values and the 2 positions of the *Direction* input?
- What happens if you change the direction on a running motor?

Enable CW

Step 4 Open the *Properties* tab again and set up your ESCON with *Enable CW*.

Step 5 How does the motor shaft rotate for positive or negative speed set values. Observe the signals on the *Controller Monitor* as well.

Step 6 What is the set value polarity (positive or negative) for CW motion?

Enable CW & CCW

Step 7 Set up your ESCON with *Enable CW & CCW*.

Step 8 How does the motor shaft rotate for positive or negative speed set values. Observe the signals on the *Controller Monitor* as well.

Step 9 What are the conditions for an enable command in one particular direction to be executed? What is different to *Enable & Direction*.

Enable and disable the power stage
The terms *Enabled* and *Disabled* refer to the state of the output power stage of the motion controller.
- *Enabled* The output stage of the controller powers the motor. The motor receives energy and can be controlled.
- *Disabled* The power stage is switched off. The motor is not powered; there is no torque produced. The control electronics however is still powered.

3.2 Set value options

We have used an analog input to set the speed command with a potentiometer. There are applications that run always at the same speed or maybe at two different fixed speeds, e.g. think of a fan with a low and high speed setting. Fixed speeds can also be set in the ESCON.

Step 1 Open the *Properties* dialog at the bottom of the *Controller Monitor*.
Observe: The ESCON needs to be disabled. Reconfigure the ESCON to simple *Enable*.

Step 2 Change *Set Value* type to *2 Fixed Set Values*. Configure *Digital Input 3* to toggle between the 2 values. Define 2 speed values that you can clearly distinguish, for instance a negative low value (e.g. -200 rpm for the *Inactive State*) and a positive high value (e.g. +2000 rpm for the *Active State*). Press *OK* to save the new configuration settings.

Step 3 Operate the motor and verify on the *Controller Monitor* the new settings.

Step 4 Explore the speed control behavior upon speed change. Compare with the settings of the speed ramps.

3.3 Comparator

Objective: Find out about the possibilities to supervise the speed (or current) with the help of the comparator functionality.

Step 1 Reconfigure the analog set value to be commanded by *Analog Input 1*.

Step 2 Set one of the digital outputs as a *Speed Comparator*.

Step 3 Explore the different modes in turn: *Limit*, *Range* or *Deviation*.
Set reasonable speed range parameters compatible with your motor (e.g. a few hundred rpm).
For being able to easily follow motor and output reaction, select relative low speed ramps (e.g. 25 rpm/s)
Select the direction you want to explore.

Step 4 Change the set value and observe the LED *Speed Comparator* output.

Observe: The LED on the I/0 board will only be functional if the corresponding input switch is activated. A HIGH or *Active* level of the output will extinguish the LED. A LOW or *Inactive* level will lighten the LED.)

DC tacho feedback
The ESCON can also be operated with a DC tacho as the feedback signal. Just select the corresponding *Speed Sensor* and follow the instructions:

- Define the analog input for the tacho to connect
- Set the DC Tacho Constant, i.e. DC voltage per speed)
- Select the DC Tacho Direction.

4 Special operation modes (optional)

4.1 Open loop speed control

Objective: Learn that Open Loop with the ESCON is equivalent to IxR compensation.

The ESCON can do open-loop speed control, i.e. without encoder or DC tacho as a speed feedback sensor. Instead, the ESCON gets simple speed information from the motor current and the applied voltage with the so-called IxR compensation method.

We can try this operation mode by setting up our DC motor without encoder (use the *Startup Wizard* or the *Properties* tab.)

Step 1 Configure the ESCON as *Speed Controller (Open Loop)*, i.e. without speed feedback sensor.

Step 2 Perform an *Expert Tuning* and study the speed tuning parameters. (They are completely different from standard PI parameters in closed loop).

Step 3 Compare the speed control behavior to the case with encoder feedback. Can you feel or see or observe in the *Data Recorder* any difference to *Speed Controller (Closed Loop)* mode.
Observe: For the open loop control you can keep the encoder connected. Just change the *Mode of Operation* in the *Properties* section.

4.2 Current control

Objective: Learn how current control works.

Setting the ESCON in Current Controller Mode.

Step 1 Open the Startup Wizard or the Properties tab. Change the Mode of Operation to Current Controller.

Step 2 Note the parameters: Nominal Current, Thermal Time Constant Winding and Max. Permissible Speed.

Step 3 Configure the *Set Value* to *Analog Set Value* from *Analog Input 1.* Define the current range from 0 A at 0V to maximum about 3 times the nominal current of the motor at 5V.

Step 4 Be aware of the maximum current capabilities of your power supply.

Explore the motor behavior in Current Controller Mode.

Step 1 Set a low current of approx. 50...100 mA. *Enable*. Follow the physical motor reaction and on the *Controller Monitor*.

- How large is the motor speed?
- How large is the real motor current compared to the set value?
- Are there any status indicators active (middle section of the *Controller Monitor* screen)?

Step 2 What changes if you increase the set value?

Step 3 Set zero current, fix the motor shaft by hand and increase the set value.

- What do you feel upon increasing the set value?
- What happens if the set value is higher than the nominal motor current?
  Attention: This can be difficult for larger motors because of the high torque.

You should have learned that the current control loop always needs a limitation by an external higher lying speed or position control loop or by some mechanical stop avoiding a “runaway” situation.

More background on control loops

Refer to the information given in the context of the EPOS4 controller

5 Special ESCON Tools

5.1 Diagnostics

Remark: Check also the ESCON Video Tutorial in the *Videos* tab of the product page of the ESCON 36/2 controller on www.maxongroup.com.

Objective: Find out about the *Diagnostics* tool of the ESCON Studio.

Step 1 Make a wiring mistake or a wrong sensor direction setup on purpose. The simplest is an *Inverted* encoder direction.
Step 2 Work through the *Diagnostics* tool of the ESCON Studio.
Step 3 Correct your wiring mistake or wrong sensor direction setup from step 1.
Step 4 Work again through Diagnostics.

5.2 Virtual controllers

Objective: Learn how to use the *Virtual Controller* for preparing without any hardware.

Configuration in Virtual Controller

Start without any hardware (motor, ESCON) connected, just work with your computer and the ESCON Studio

Step 1 In the ESCON Studio, open the *Options* dialog in the *File* menu. Select *Virtual Controller* and activate the checkbox *Scan for Virtual Controllers*. Press *OK*.

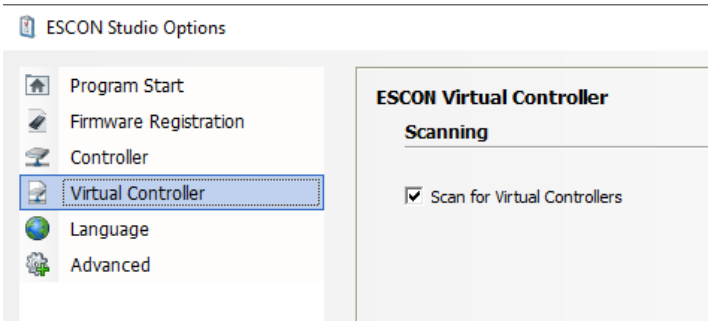


Figure 8: Scan for virtual controllers

Step 2 *Search Controllers …* either in the *File* menu or press the binocular icon. From the drop-down menu, select the ESCON type you are going to use. In our case, it is *ESCON 36/2 DC – IPC0*. The communication type is marked as an extension to the controller name. Virtual controller extensions are named IPCx.
Step 3 Configure the virtual ESCON with the *Startup Wizard* for use with our DCX motor. Operation mode is closed loop control with encoder feedback.
Step 4 Configure different parameters than before. E.g. a large speed set value range (-5000 .. +5000 rpm) on *Analog Input 2* (instead of 1), *Enable & Direction* on *Digital Input 1* and 2, *Stop* on *Digital Input 3*.

Transfer the configuration to the real ESCON controller

Step 5 Upload (i.e. save) the virtual configuration to your computer. Use the *Upload Parameters* command in the *File* menu or the corresponding icon with the blue upload arrow.

Step 6 Connect your real ESCON with your computer via USB. Set the *Active Controller* as the real ESCON controller type with the extension USBx.

Step 7 Download the saved parameters to the ESCON. Use the corresponding command in the *File* menu or the icon with the orange download arrow.

Step 8 With a correct wiring and after a tuning step you should be able to operate the motor according to the configuration performed in the virtual controller.

5.3 Restore Default Parameters

Please restore the factory settings in the ESCON at the end of the training session, so the next student can start from scratch.

Either use the restore parameter icon or the corresponding command in the *File* menu

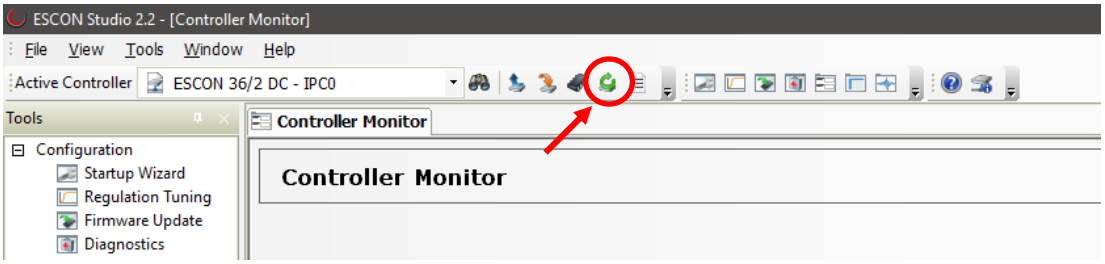


Figure 9: Restore default parameters

Part 3: EPOS4 Preparation

The next two chapters serve to get to know the EPOS4 system, and to prepare the workbench.

In chapter 6, we will provide an overview of the hardware, go over the installation of the *EPOS Studio* software and briefly explain the main elements of the motion control system.

In chapter 7, the motion control system is prepared for the exercises that will follow. The properties of the motor and encoder must be defined, and the control loop tuned.

6 Setting-up the EPOS4 Starter Kit

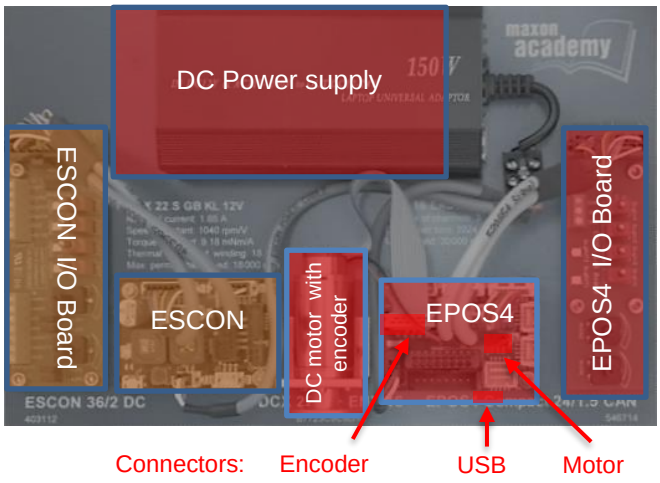


Figure 10: The content of the ESCON/EPOS4 Starter Kit.
The components used for the EPOS4 training are highlighted in red.

In the Starter Kit box on the right, you will find a motor with encoder and the *EPOS4 Compact 24/1.5 CAN* controller board connected to a printed circuit board (PCB) with switches, LEDs, and potentiometers. This PCB serves as an analog and digital input and output (I/O) device.

Besides, there is an ESCON servo controller with its own I/O printed circuit board placed on the left side of the main board. The ESCON part has been treated in the first 2 parts of this textbook.

6.1 Installation of EPOS Studio software

First, prepare the computer. Install the software from the maxon website: scroll down on the website https://epos.maxongroup.com/ until you find the button *Download EPOS setup*. The installation contains all necessary information and tools (such as manuals, firmware, Windows DLLs, drivers) required for setting up and exploiting the maxon *EPOS Positioning Controllers*. The most important tool is called *EPOS Studio*; it's the user interface that allows full access to all the features and parameters of the motion controller.

Step 1 Follow the instructions given when running the installation program.
Please read every instruction carefully. Indicate location of working directory when prompted.

Working directory
We recommend the following location as a working directory:
C:\Program Files (x86)\maxon motor ag
Note that the designation of the program directory may vary depending on the system language installed.

Step 2 There will be new shortcuts and icons in the start menu of your computer.
The files have been copied to the programs folder. You will find the EPOS Studio software in the start menu folder "maxon EPOS Studio" and the entire documentation about all the EPOS types in the corresponding menu folders.
Clicking the *EPOS Studio* shortcut on your desktop will launch the program.

Latest *EPOS Studio*
After installation, verify that you are using *EPOS Studio 3.6* (Revision 2) or higher. Do so by clicking menu *Help*, then select menu *About EPOS Studio*. If that is not the case, download the latest version from the maxon website https://epos.maxongroup.com.

Component	Minimum System Requirements
Operating System	Windows 10, 8, 7
Processor	Core2Duo 1.5 GHz
Drives	Hard disk drive, 1.5 GB available space
Port	USB 2.0 / USB 3.0
Memory	1 GB RAM
Monitor	Screen resolution 1024 x 768 pixels at high color (16-Bit)
Web Browser	Microsoft Internet Explorer 8.0
Software	Microsoft .NET Framework 2.0 & 4.0

Table 1: Minimum System Requirements (See Release Notes.txt in the EPOS Studio installation folder)

6.2 Connecting the cables

The cabling is easy and basically mistake proof. First connect the DCX motor (2 wires: red, black with a tiny black connector) into the only white socket that fits. Then do the same with the encoder flat ribbon cable. The PCB with the I/Os is already connected. Then – and it is good engineering practice to do this last – connect the power supply. Finally, fit the *USB* cable to one of the *USB* outputs on your computer and to the *EPOS4 Micro USB* plug.

Your setup should then look similar to Figure 11.

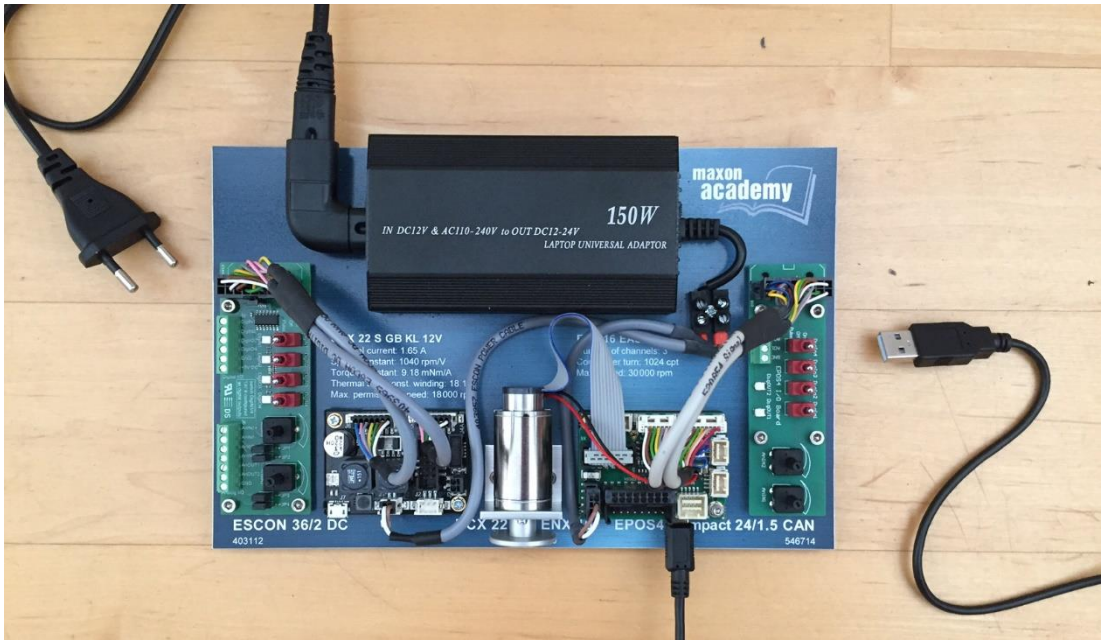


Figure 11: How to connect the EPOS4 Starter Kit.

Finding the *USB* driver

When you first connect the *EPOS4* to the *USB* port your computer will announce that it has found new hardware and will automatically install the corresponding drivers. This may take a while.

You can speed up the procedure by selecting manually the file location of the drivers. They are in the same folder where you installed the EPOS4 software: maxon motor ag\EPOS IDX\Driver Packages\EPOS USB Driver (*see Figure 9*). Follow the steps in the *EPOS USB Driver Installation.pdf* document. The procedure varies slightly depending on your computer's operating system.

6.3 The EPOS4 as a black box and its documentation

EPOS is an acronym for **E**asy to use **Po**sitioning **S**ystem and the **4** refers to the fourth - generation design. However, it's up to you to judge if it is really “EPOS” or just “POS”, i.e. if it is easy to use or not.

Considering the *EPOS4* as a Black Box, we get a main understanding of its functionality and its input and output. It is a motion controller that can regulate position, velocity or current of brushed or brushless DC motors with encoder feedback. There are also several digital and analog inputs and outputs that can be used for different purposes in the context of the motion axis.

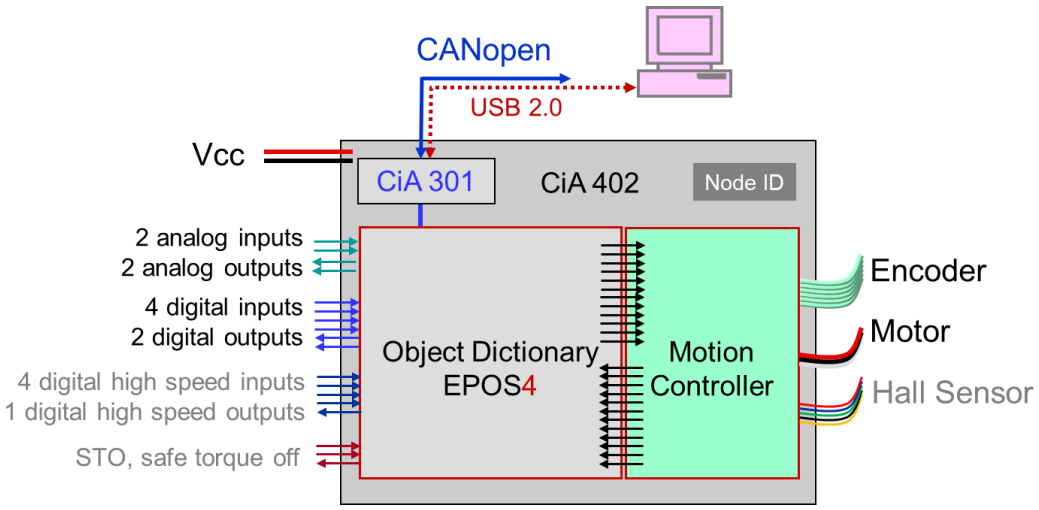


Figure 12: Schematic overview of the EPOS4 with external connections and communication. Observe that on our system no high speed I/Os are wired and that our brushed DCX motor contains no Hall sensors.

From the point of view of the system environment, the *EPOS4* is a CANopen device *Drives and Motion Control* (CiA 402). The *EPOS Studio* on your computer communicates with it by means of a *USB2.0* serial connection. Alternative ways for communication are serial *RS232* or *CANopen* bus connection. Most probably, your computer does not have a *CAN* interface, so we cannot use this bus.

EPOS4 motion controllers are CANopen slaves and require a master system with a program commanding them and other slaves. Masters can be PLCs, PCs, micro controllers and other programmable devices with a suitable communication. They control the process flow and all slave units in the network. In our situation, it will be the EPOS Studio taking the role of the master program. With an interface, EPOS4 controller allow even to be addressed by EtherCAT from an EtherCAT master system.

Manuals and software documentation
The installation not only sets up the *EPOS Studio* software on your computer but copies all the manuals and software you might need to program the device at hand or any other maxon EPOS device.

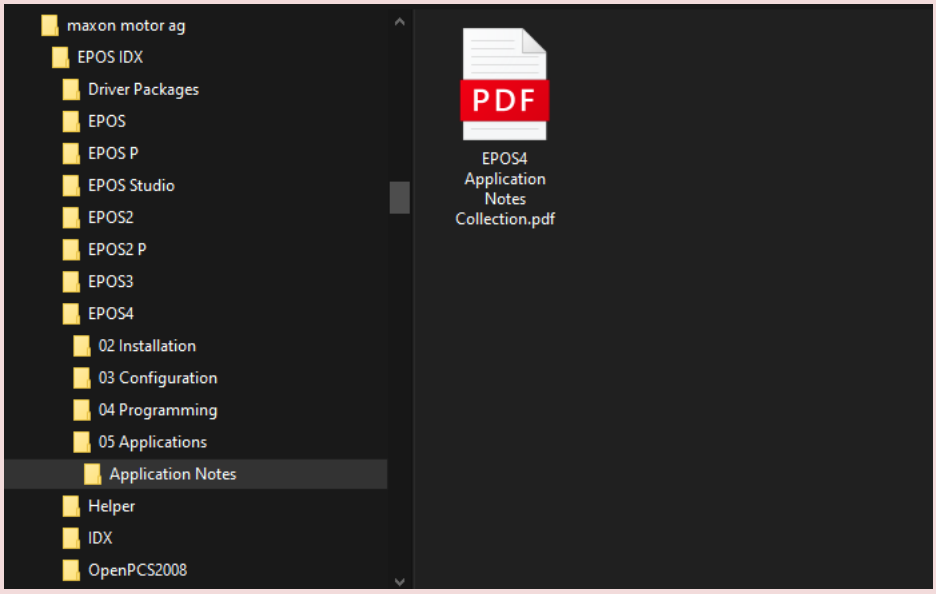


Figure 13: The file structure of the maxon EPOS software installation.

For the EPOS4 products there are 4 subfolders with the documents as shown in *Figure 14*:

- In *02 Installation,* the hardware and cabling information is listed.
- *03 Configuration* contains the actual and older *Firmware* files; that is the software running on the EPOS.
- *04 Programming* contains important documents for the programmer: the *Firmware Specification* (see chapter 10.3) where all the properties and parameters are described in detail. The programmer finds other useful information, programming examples and libraries in the *Programming Reference* document.
- *05 Application*: The *Application Notes* contain information on operation modes, special application conditions and situations.

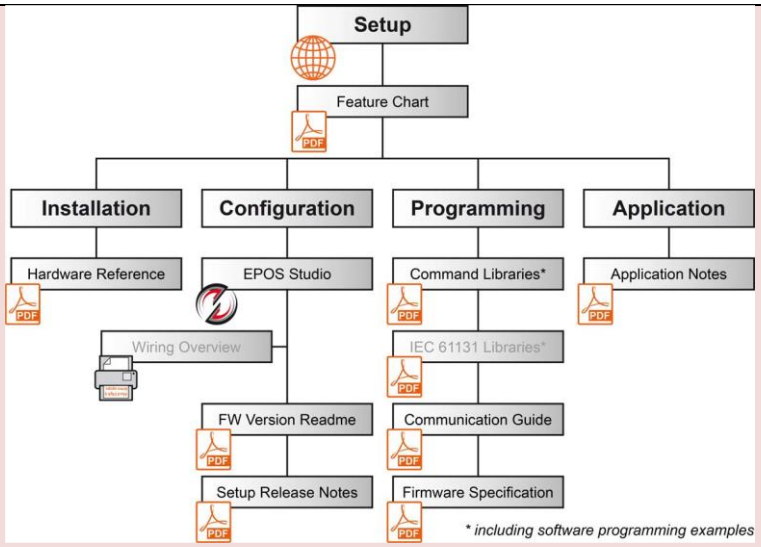


Figure 14: The document structure of the maxon EPOS4.

6.4 Starting the EPOS Studio

We are ready to begin. Start the *EPOS Studio* by clicking on the corresponding icon on your computer's desktop.

The *EPOS Studio* is the graphical user interface for all maxon EPOS products. The nice thing about the *EPOS Studio* is that it allows direct communication with any device in the network without any programming to be done. This is useful for setting up the system, training and any other exploration that you might want to do. Be aware however, that in a real application, you will have to program the master system to send the right commands to your EPOS.

To establish communication with the actual hardware, select the correct EPOS project in the *New Project Wizard, which should* open automatically. If not, click on the *New Project* icon on the menu bar.

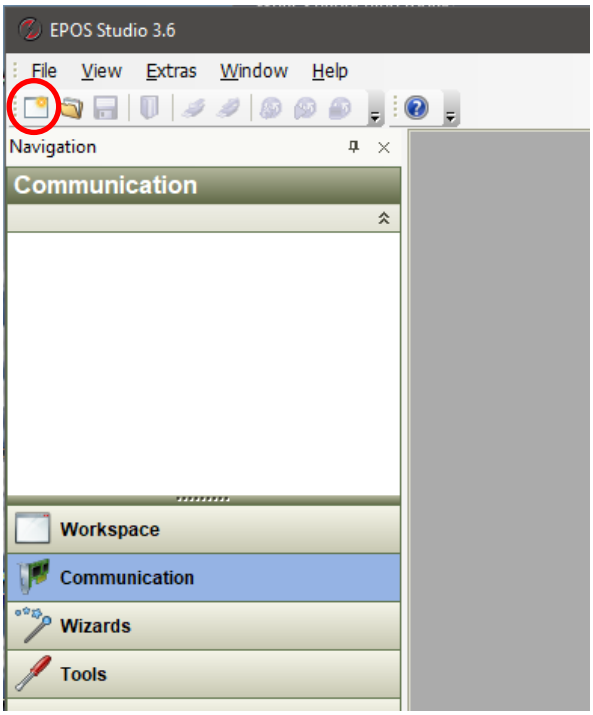


Figure 15: Where to find the New Project icon.

Projects in *EPOS Studio*
The *EPOS Studio* allows you to save the setup of your project and to open it again. The project in the *EPOS Studio* contains information about the hardware and communication channel used. A network of several EPOS linked to your computer will be saved as well. Saving projects under a particular name is useful whenever you are not using a simple standard project as we have here.

Step 1 Select *EPOS4 Project* from the list. Click *Next*.

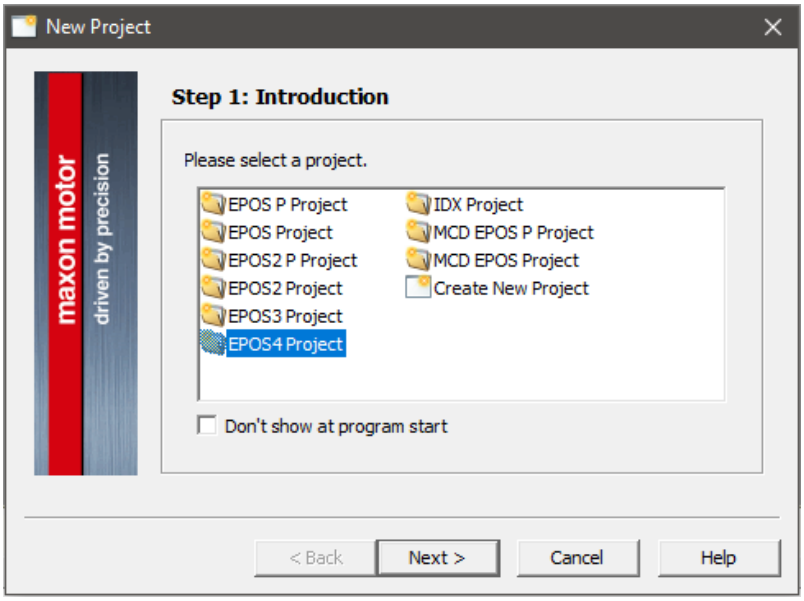


Figure 16: New Project Wizard, step 1.

Step 2 Now define a path and name for your project or use the suggested one. Click *Finish* to create a new project.

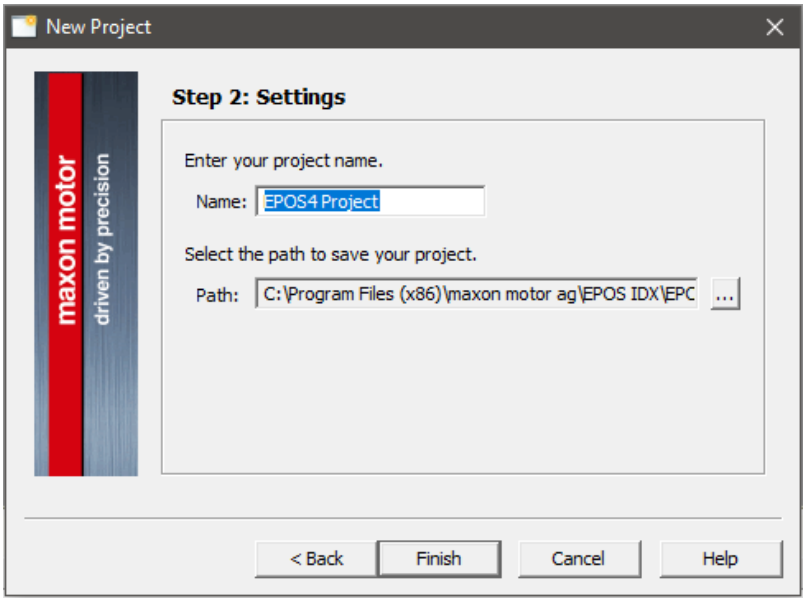


Figure 17: New Project Wizard, step 2.

If all goes well the *EPOS Studio* will announce that the project file has been saved – you will have to prompt this message – and it will establish communication with the *EPOS4.* The latter can be seen by the adjustment bars running from left to right in a pop-up window and indicating that the parameters of the *EPOS4* are being read.

The Workspace tab

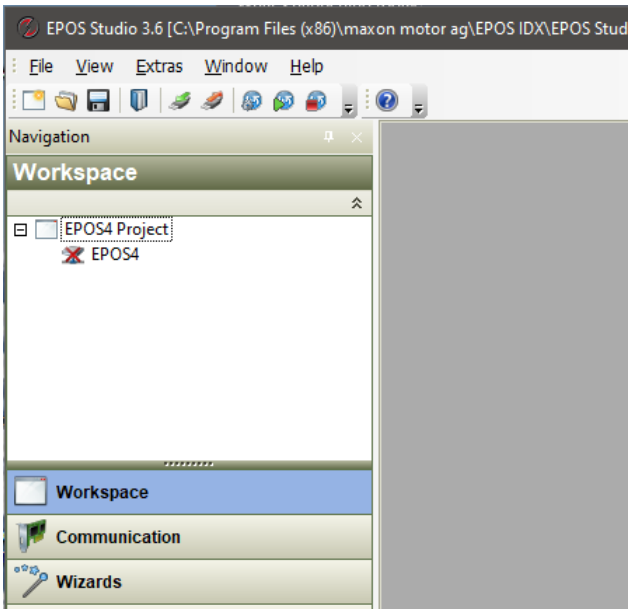


Figure 18: The EPOS Studio screen with the Workspace tab open.

The *Workspace* tab of the *Navigation Window* on the left displays the controller hardware of the project. In our case, the project contains the *EPOS4 [Node 1]* motion controller.

Errors and warnings
There might be errors and warnings appearing in the *Status* window at the bottom. For instance, if no *CAN* communication is established, the *Can Passive Error* might appear; or a *Hall sensor error* in our case where a brushed DC motor without Hall sensors is connected.
For the moment being, you can ignore the errors and clear them once you have made a proper setup. Errors are cleared with a right mouse click on the error list and selecting *Clear All Entries*.
In case other errors or warnings appear, check the wiring. For details on errors and warnings, see separate document *EPOS4 Firmware Specification*.

Status

Type	Node	Code	Name	Description
Error	EPOS4 [Node 1]	0x7388	Hall sensor error	Motor hall sensor report

Figure 19: Error and warning list in the Status *window at the bottom.*

The Communication tab

The *Communication* tab shows the setup of your project from a communication point of view. Schematically, you can see that your computer (*Local host*) connects via *USB* to the *EPOS4 [Node 1]*. There is an unused CAN connection available where other CANopen nodes (e.g. other EPOS4 controllers) could be added.

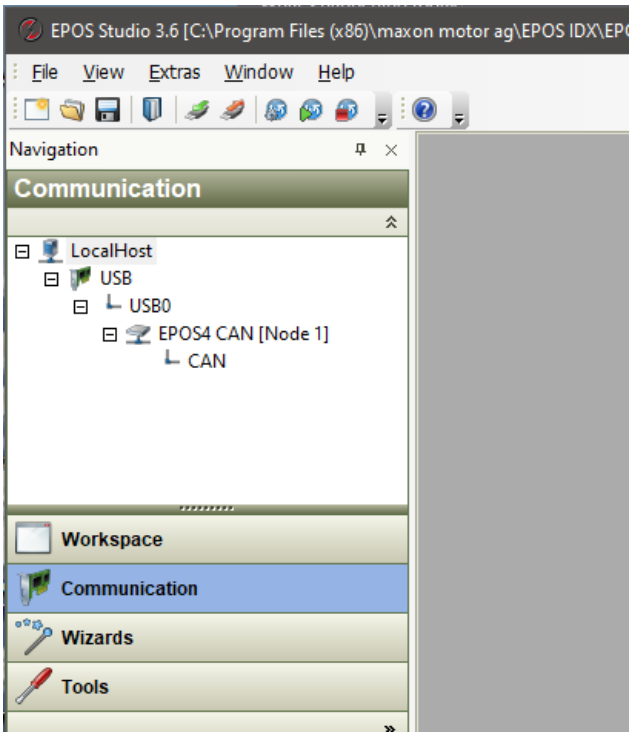


Figure 20: The Communication tab of the EPOS Studio screen.

The Wizards and Tools tabs

There is no need to look at the wizards and tools in detail at this time. Many of them will be used later on our journey and explained when they pop up.

Just one remark: Tools and wizards are assigned to the different hardware components in the project. Therefore, always select the correct device from the drop-down menu at the top. Here however, we have only one device at hand: the *EPOS4 [Node 1]*.

6.5 Downloading the latest Firmware

Before we start, we make sure our EPOS4 is up to date. For this purpose, we start the *Firmware Update* wizard by double clicking.

Step 1 Confirm, that you have read the warning and press Next.

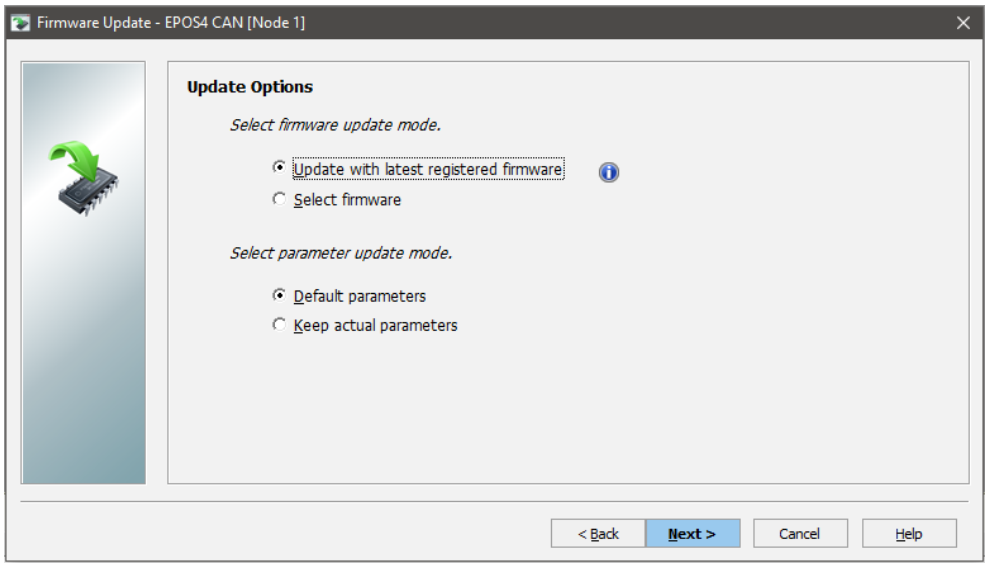


Figure 21: Firmware Update Wizard: Setting the options.

Step 2 There is no need to change the standard Update Options, just press *Next* again.

Step 3 Start the Firmware Update and wait until the Finish button turns active. Do not interrupt the procedure! Then press Finish.

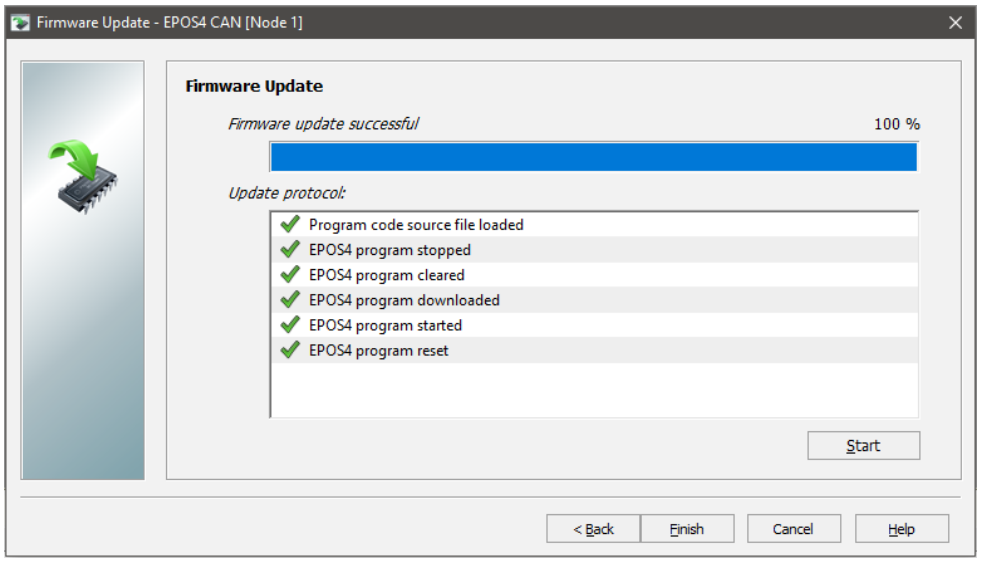


Figure 22: Firmware Update Wizard: After successful firmware update.

6.6 Restore Factory Settings Parameters

It' s always a good advice to restore all default parameter to the EPOS4 before performing any experiments. Right click on the EPOS controller in the Communication device tree

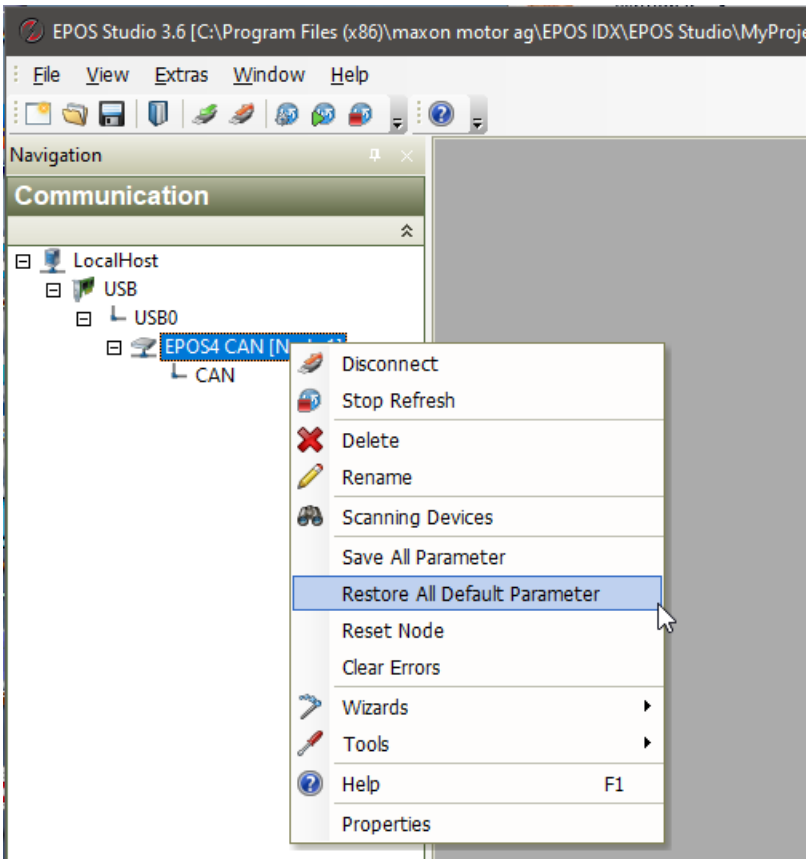


Figure 23: Restore all default parameters

7 Preparing the EPOS4 motion control system

7.1 Startup Wizard

An *EPOS4* can control many different drive configurations with a multitude of motors and possible sensors and – as we have learned earlier – with several possibilities of communication. Therefore, the first thing to do is to define the actual setup. For this purpose, we run the *Startup Wizard* that can be found in the *Wizards* tab of the *EPOS4.* Just double click on the corresponding icon.

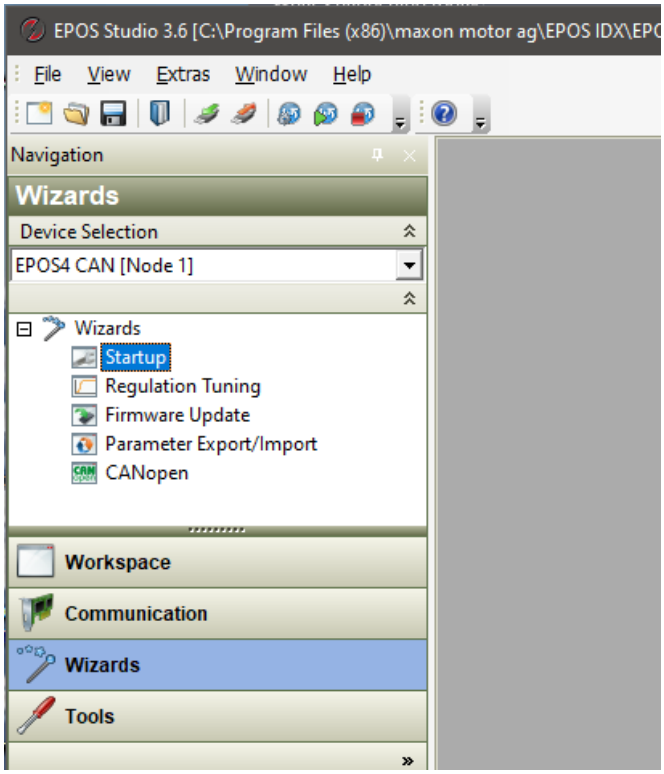


Figure 24: Double click for selecting the Startup Wizard in the Wizards tab of the EPOS Studio screen.

The left column in the wizard dialog indicates the different steps of defining the system, starting with the *Safety Instructions*. maxon wants to make sure that you are a qualified person. By following this textbook, you will become one. Hence, don't be afraid and check the box near the bottom "*Yes, I have read the above instructions*"; otherwise, you will not be able to proceed. For the configuration, we will follow the given order – from top to bottom in the left column. After completing each step, just click on the *Next* button at the bottom.

Fortunately, we have quite a simple setup and only a few parameters vary from the default settings. There are not too many options to choose from. However, if you have a more complex system (e.g. a Dual Loop configuration with feedback sensors on the motor and on the load) work carefully through the *Startup Wizard*.

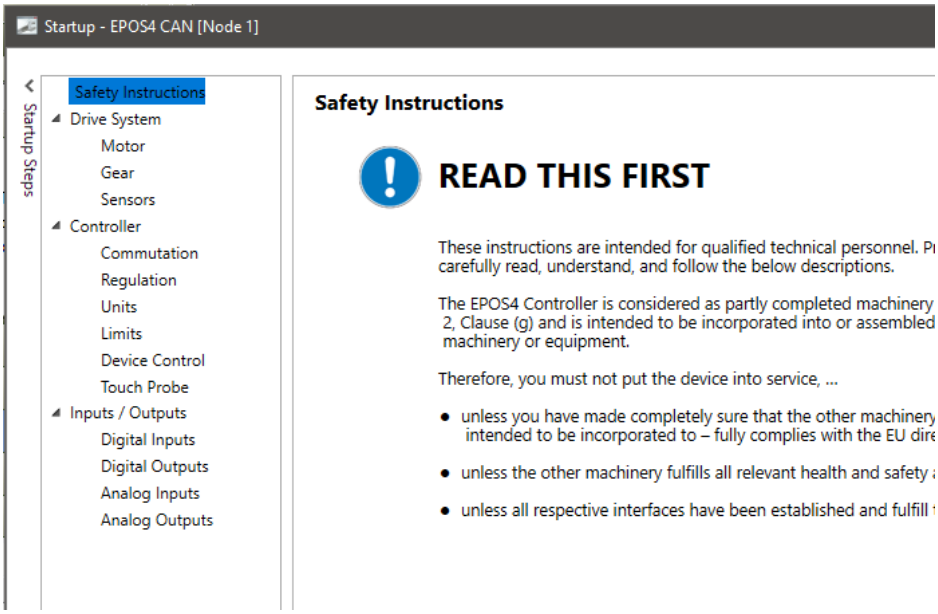


Figure 25: Startup Wizard, safety instructions.

Step 1 The first step is to define the type of motor used. The EPOS motion controllers can be used with brushed DC motors (maxon DC motor) or with brushless DC motors (maxon EC motor). Since the DCX motor at hand is a maxon DC motor select the first option. Accordingly, the following steps will adjust automatically.
Next, you will have to give e few key parameters of our DCX 22 S GB KL 12V motor; you can find these parameters printed on the main Starter Kit board. Because the continuous current of the motor (1650 mA) is larger than the continuous current of the EPOS4 (1500 mA), we are restricted to the latter, i.e. lower, value. By deselecting the corresponding checkbox, you can even enter the torque constant of the motor.

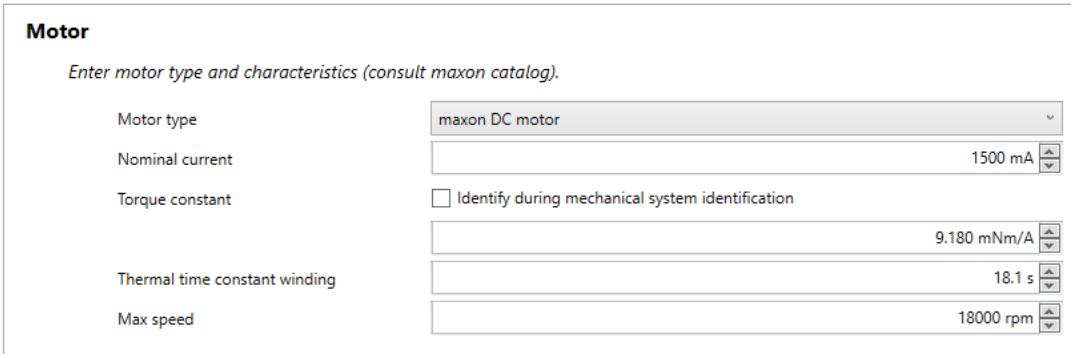


Figure 26: Startup Wizard, step 1, motor parameters.

The motor parameters should now look like as shown in Figure 26Figure 26. Click on *Next*.

Step 2 Gear. We can omit step 2 since there is no gear mounted on the motor.

Step 3 In step 3 we define the type of feedback sensor in use to close the position or speed control loop. The standard device for maxon EPOS controllers is an incremental digital encoder. This is what we have on our motor and it is called ENX 16 EASY. Again, its parameters can be found on the main board: It's a 3-channel encoder with 1024 cpt (counts per turn) or pulses/revolution.
Click on *Next* to continue.

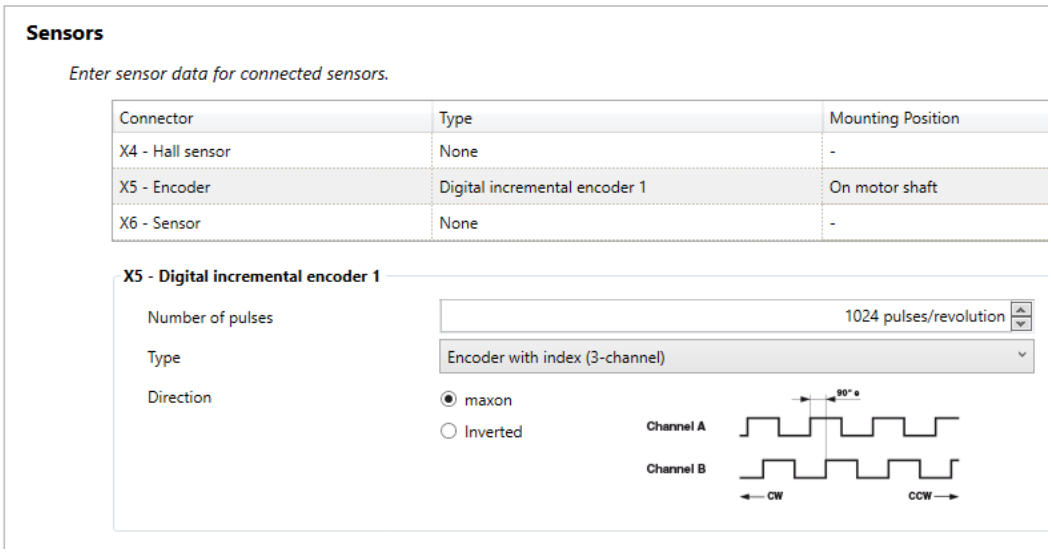


Figure 27: Startup Wizard, step 3, encoder parameters.

Step 4 *Regulation*. For our case, there are not many regulation options to choose from as you can see when exploring the drop-down buttons. Essentially, we could select an observer-based speed control, which might result in better speed control behavior at extremely low speed and at low feedback resolution. However, our encoder resolution is quite good. Hence, the default settings are perfect. Leave them as they are and just click *Next* to proceed.

Step 5 There is no need to change the controller units.

Step 6 In the *Limits* you can define general operating limits that you want to be respected all the time. In case a command exceeds such a limit, an error will occur.
The current limit has already been set to 1500 mA, the maximum possible value for this EPOS4 controller. We will keep the *Max output current* at 4500 mA to allow a fast acceleration.
The default value of the *Max acceleration* is unreasonably high. However, we leave it as it is for not limiting our operation unnecessarily.
Since the maximum permissible speed of the motor is 18'000 rpm, the *Maximum profile velocity* cannot be selected faster.
Remark: The maximum speed of the encoder (30'000 rpm) is much higher and will not be the limiting factor. In case of a gear, the maximum speed is often limited by the input speed of the gear.
We keep the following error window at its default value and do not limit the possible position range - as would be reasonable for a linear axis for instance.

Click on *Next* to continue.

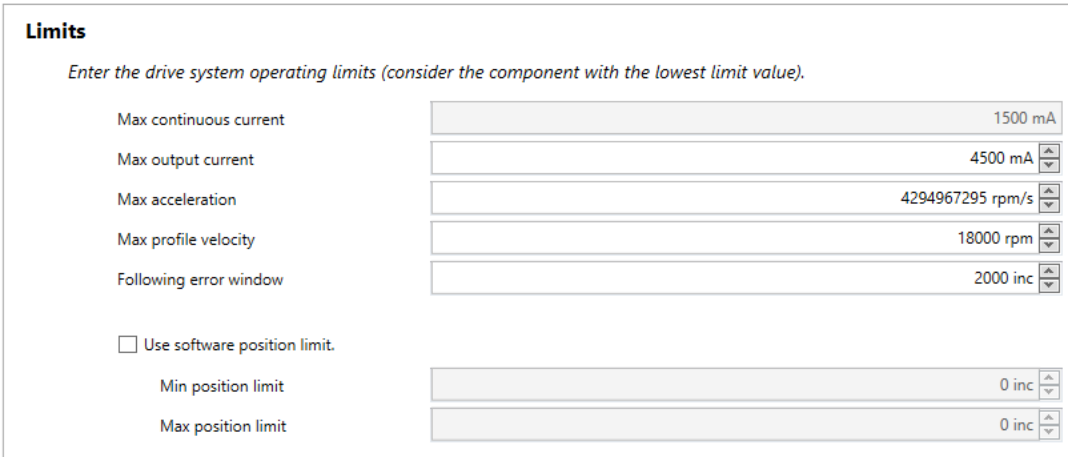


Figure 28: Startup Wizard, step 6, limitations.

Step 7 *Device Control*. In this configuration step, you define how the system should react on special incidents such as shutdown, disable, quick stop and others. You can also set some parameters governing these reactions. We don't want to bother with these settings now and keep the default values. For further information, refer to the document *Firmware Specification.*

Step 8 Configuration of the *Digital Inputs*. Again, we keep the configuration as it is.

Step 9 The same we do for the configuration of the Digital Outputs. Hence, all that needs to be done is press *Finish* to save the setup parameters in the EPOS4.

This is what we mean by *Easy to use*. With a few clicks and parameters, the system is set up.

Motor and encoder parameters of the unit at hand
The motor used is a *DCX 22 S*, 12 V nominal voltage, with graphite brushes and ball bearing. Its exact specification can be found on the maxon website or in the Appendix. Important for us is the fact, that it is a brushed DC motor.
This motor has a maximum speed limit of 18 000 rpm which is lower than what we can achieve with the given voltage of the power supply (24 VDC). Typically, the motor is able to reach speeds of up to approximately 22'500 rpm at 24 V supply (speed constant of motor times motor voltage = 1040 rpm/V * 0.9 * 24V).
The nominal current is 1.65 A. The motor can draw that current continuously without overheating. For a few seconds it can support a much higher current, possibly up to 10 A. However, we are not able to get as much current out of our EPOS4, which exhibits a current limit of 4.5 A.

The encoder type is *ENX 16 EASY encoder* with 1024 counts per turn on each of the two channels. This results in 4096 signal edges (states) per turn of the motor shaft which are called increments (inc). The inc are the internal position units used by the EPOS controllers. Hence, we have a nominal resolution of 1/4096 of a turn or 0.088°.
In addition, the encoder at hand has an *Index* channel that can be used for precise homing.

Commutation with brushed and brushless motors
The term *commutation* describes the way the supply voltage is applied to the different winding segments.

On brushed motors – e.g. maxon DCX motor – the commutation is done in the motor by means of graphite or precious metal brushes. To make the motor rotate all the controller needs to do is applying a voltage.
On brushed DC motors, there is no additional feedback device needed for proper commutation. However, for measuring speed and position in a control loop an encoder or other feedback device is needed.

On brushless DC motors (BLDC) – e.g. maxon EC motor - there are three winding segments (phases). The commutation, i.e. supplying the voltage correctly to these three phases in accordance with the actual rotor position, is an additional the task of the controller.
The simplest type of commutation for BLDC motors, called *block commutation* is achieved by the rotor position information provided by 3 digital Hall sensors. In short and without going into details, the Hall sensors are necessary to actuate the BLDC motor properly. However, Hall sensors only provide a low-level resolution feedback and block commutation has its limitation when it comes to smooth operation at low speeds.
A more sophisticated way of doing commutation is called *sinusoidal commutation* or *field-oriented control (FOC)*. It results in smooth motor operation even at very low speeds and gives a slightly higher motor efficiency. However, it requires the additional high-resolution feedback of an encoder. Fortunately, this is usually available anyway on servo systems with position or speed control.
Note: On brushless DC motors, feedback is needed for two separate tasks: for commutation – i.e. for motor operation – and for speed or position control.

The *EPOS4* motion controllers allow block commutation for motor speeds up to 100'000 rpm, while sinusoidal commutation has a maximum speed of 50'000 rpm. These values apply for BLDC motors with 1 magnetic pole pair. Higher numbers of pole pairs reduce the maximum speed accordingly. An EC-4pole motor with 2 pole pairs can only be operated at half these speed values. Flat motors can have even more pole pairs and – correspondingly – a lower maximum speed limit (refer to Table 2).

Position and speed evaluation with incremental encoder
Incremental digital encoders are the most common sensor type for measuring speed and position in micro-drives. They can be found in various designs and with various working principles, but all of them basically subdivide one revolution into a number of steps (increments) and send signal pulses to the controller each time an increment is detected. Incremental encoders generally supply square-wave signals in two channels, A and B, which are offset by a quarter signal length (or 90 electrical degrees).

The **characteristic parameter** of the encoder is the number of square pulses *N* per revolution, given in cpt (counts per turn). By counting the number of state transitions in both channels, the achievable resolution is four times higher. These states are called increments (inc) or quadrature counts (qc) and are used as the position unit in EPOS systems.
Note: When talking about the resolution of incremental encoders make sure whether the counts are meant before or after quadrature.

The higher the resolution, the more accurately the position can be measured, and the more accurately the speed can be derived from the change in position as a function of time.

The **direction of rotation** follows from the sequence of the signal pulses of channel A and B: Channel A leads in one direction; channel B leads in the other direction.

In addition, the encoder can have an **index channel**. That's one narrow pulse per turn. The index can be used for getting a very precise position reference (see Homing, chapter 3.2). Some controllers for brushless motors need the index as an absolute rotor reference during sinusoidal commutation as well.

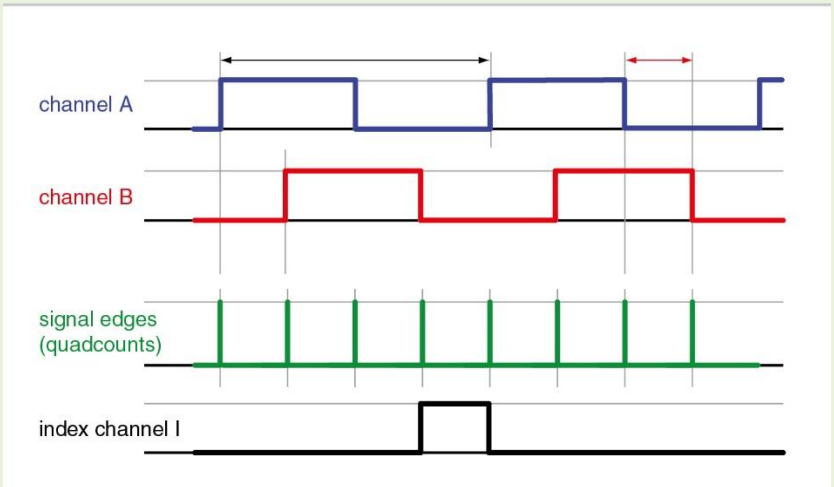


Figure 29: The signals of an incremental encoder.

Position evaluation is determined by counting the signal edges (increments) from a defined reference position (home position, see homing in chapter 8.2). The unit of positioning is called inc.

Speed evaluation is calculated from the position change per sampling interval. In our case, the natural unit of speed is increments per control cycle time. Therefore, speed can only be expressed in steps of 1 inc per 0.4ms, which is usually translated in the more common, and practical unit rpm. The effect of speed quantization from the incremental encoder can be nicely seen on speed measurement diagrams (see chapter 0).

Acceleration and deceleration values are given in the practical and easily interpreted unit rpm/s.

Encoder pulse count and control dynamics
A high encoder pulse count *N* not only results in a high position resolution, but also improves the control dynamics. Feedback information about a change in the actual position value is obtained more quickly and the control circuit can initiate corrective action that much faster. The controller can also be set up with more stiffness, i.e. with higher controller amplification, which reduces the risk of unstable oscillation.
With a low pulse count and high control parameter setting, the drive has more time to accelerate until a new position feedback signal, i.e. a new encoder pulse, is received. In such a case, the drive may overshoot the end position and produce a proportional correction value in the opposite direction. This again can result in an overshoot in the new direction. The motion becomes unstable and the drive oscillates.
Positioning with the signals from the Hall sensors alone typically leads to this unstable behavior. It is strongly recommended to use an encoder with sufficiently high resolution.

EPOS4 maximum velocity values on brushless motors
The micro-controller performance of the EPOS4 limits the maximum possible speed of brushless motors. Sinusoidal commutation (FOC) needs more calculation power than block commutation resulting in a lower possible commutation speed. The achievable motor speed also depends on the number of pole pairs of the motor.

maxon EC motor types (examples)	Number of pole pairs	Max. speed with block commutation	Max. speed with sinusoidal commutation
maxon EC motor, EC-max	1	100’000 rpm	50’000 rpm
EC-4pole	2	50’000 rpm	25’000 rpm
EC 20 flat, EC 32 flat	4	25'000 rpm	12'500 rpm
EC-i 40, EC 60 flat	7	14’280 rpm	7’140 rpm
EC-i 52, EC 45 flat	8	12’500 rpm	6’250 rpm
EC 90 flat*	11 (new) 12 (old)	9’090 rpm 8’330 rpm	4’540 rpm 4’160 rpm

Table 2: Maximum speed of EPOS4 systems for brushless DC motors as a function of the number of pole pairs and type of commutation. Note that some of the motors have lower maximum speed specification than what is possible with block commutation. (Check the actual version on the motor data sheet)*

7.2 Tuning

The next step for system preparation is tuning. The purpose is the same as tuning a sports car: We would like a quick, strong and optimized reaction of the drive system to any given motion commands. Again, we can use a wizard to make life easier. Start the *Regulation Tuning* wizard (double click) in the *Wizards* tab.

On the left, you can see the 3 possible tuning steps: current, velocity, position. The simplest way of performing a tuning with EPOS4 is to use the Auto tune functionality. This is what we will do here.

Tuning the electrical system

The tuning of the electrical system should be done first.

Click on the *Auto* tune button and prompt the next message stating that your system will move with *Yes*. An automatic tuning is performed with the parameters shown, optimizing the response of the motor current control loop. The goal is that the current is applied as fast and precisely as possible whenever it is demanded from the higher control level, such as position or velocity control. A well-tuned current control loop is the prerequisite for dynamic system reaction.

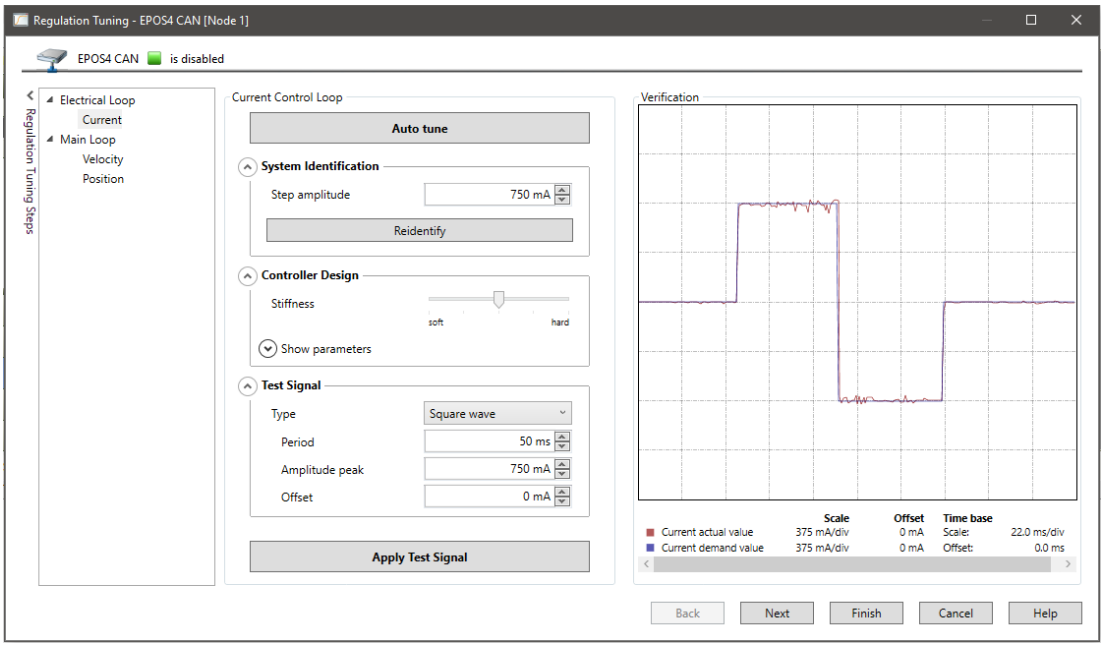


Figure 30: Regulation Tuning wizard: current
The result of auto tuning the current. The verification diagram shows the perfect response to a current set value step.

The *Auto tune* feature starts with a *System identification* step. An oscillating signal is applied to the motor. It serves to acquire information about the dynamic response of the system. System identification needs to be executed only once, unless there is a change in the drive layout.In a next tuning step, the optimum *Controller design* parameters are calculated according to the required stiffness. Controller design is based on the results of the

identification and uses a general model of the drive system layout. At the end, regulation tuning applies a *Test signal*, i.e. it executes a test motion, and records the actual system reaction. The diagram displays the result.

If you are interested in the meaning of all the parameters and how they can be used to improve the system reaction, please refer to the EPOS4 box *Tuning Parameters and Manual Tuning* below.

Press *Next* or select *Velocity* in the menu on the left.

Diagram zoom
You can zoom into the diagrams by selecting the interesting region with the mouse. Right mouse click zooms out.

Tuning the mechanical system

For simply tuning the velocity and position control, use the *Auto tune* functionality again. The procedure follows the same principles as described above. However, the *System identification* takes longer. It acquires information about the mechanic response of the system by oscillating the motor with increasing amplitude and at different frequencies. As before, the result is shown in the *Verification* diagram on the right.

The *Auto tune* processes provide quite good tuning parameters for most applications with just one mouse click. There is no need for cumbersome trial and error or time-consuming evaluation of recorded motion response.

Note: Tuning should always be done on the fully established system with the original power supply and all the mechanics, inertias and friction in place. In the end, it's the full machine that we want to respond in an optimum way and not just the motor alone.

Clicking on *Finish* when you exit the *Regulation Tuning* wizard saves the newly found tuning parameters permanently. The parameters apply to any future motion command. They are saved permanently and are functional, even after a restart of the system.

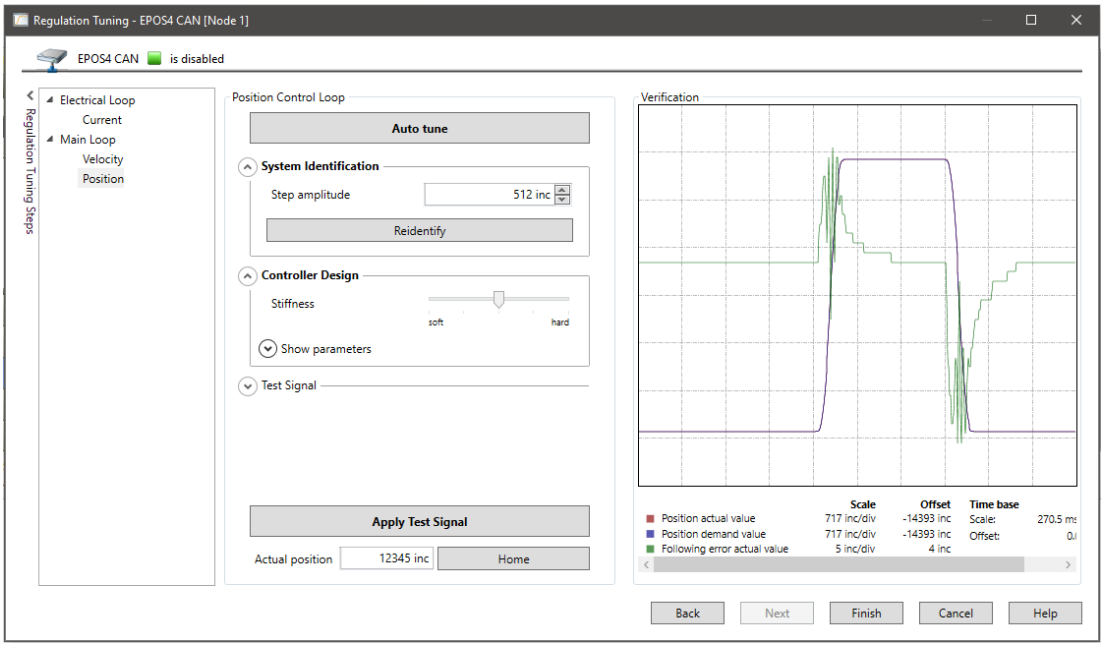


Figure 31: Regulation Tuning wizard: position.
The result of auto tuning the position. The verification diagram shows the response to a position move forth and back. The green signal is the amplified position error, i.e. the deviation between actual value and set value.

Tuning Parameters and Manual Tuning
The *Regulation Tuning* wizard allows to influence the tuning processes. We will show here some of the features regarding position control.

For *System identification,* we can define how large the motion amplitude of the motor should be. Default value is the eight part of a motor turn or 45°. In combination with gears and other mechanical layouts, higher amplitudes might be necessary in order to generate substantial motion of all the mechanical parts. For proper tuning, we need to identify all relevant mass inertias and friction, which is only possible if all the mechanics is set to motion.
Once the identification has been executed, the system behavior and the effects of friction and mass inertias are known. Therefore, there is no need to repeat the identification when you just change the parameter setting in the *Controller design* step.

The way the actual tuning parameters are calculated in the *Controller design* depends on the desired system behavior and your application. You might want a very stiff and fast reacting system and you don't care about overshooting the target position or speed. Or you set soft regulation stiffness with slower reaction and less overshoot.
Try different stiffness settings, compare the parameter values, press *Apply test signal* and observe the result in the *Verification* diagrams and. For better resolution in the diagrams, adjust the *Settling Time*. In our case 200ms or even less is perfect (see Figure below).

Parameters for a typical test motion can be set in the *Test signal* area. As a rule, test motion parameters should be similar to real motions in the applications, typically the most critical one.

Checking the box *Dimension manually* allows the writing access on the control parameters, i.e. manual tuning. Manual tuning is typically used to optimize the results of the auto tuning. It involves changing one of the parameters and checking the outcome graphically after applying the test signal.

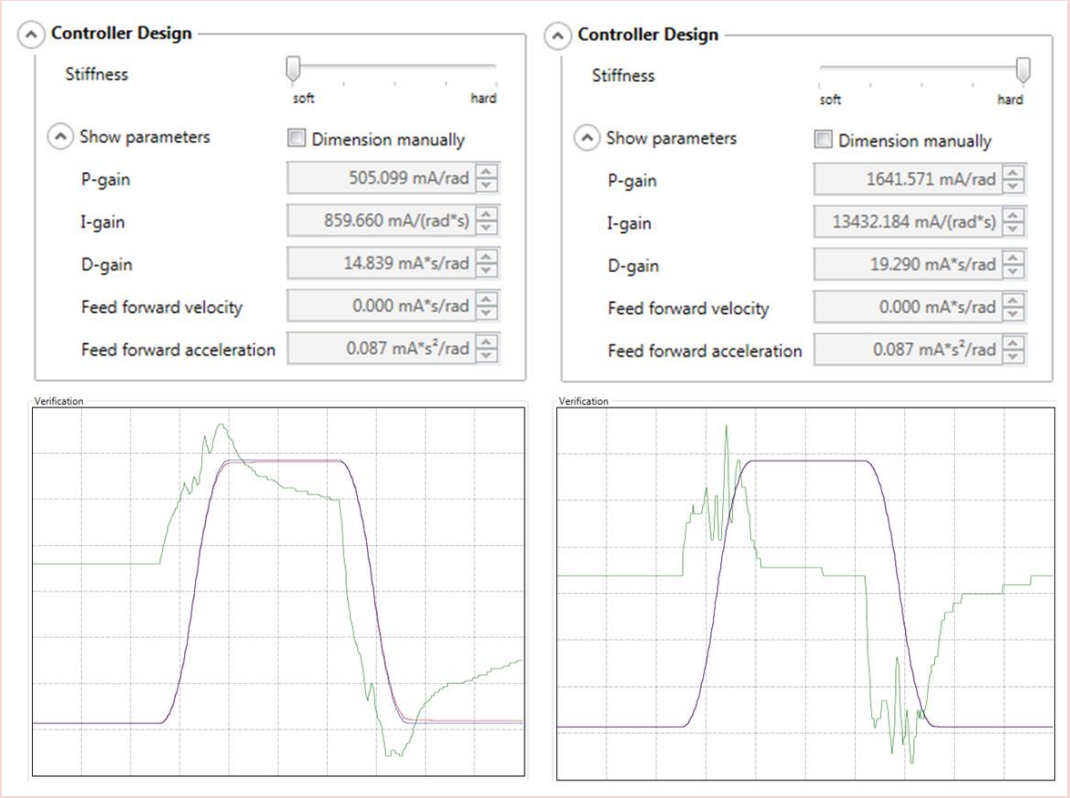


Figure 32: Regulation Tuning *wizard. Comparison of soft (left) and hard (right) regulation stiffness: PID parameters (top) and system response (diagram on bottom) resulting from different regulation stiffness settings.*
The green line gives the positioning error. The soft control on the left is not able to reach the end position for both movements within the short settling time of 200ms. The stiffer control parameters on the right allow the end position to be almost attained. Note the larger scale of the error signal in the left diagram; each step in the signals corresponding to one increment.

Feedback and Feed Forward

PID feedback amplification

PID stands for *Proportional*, I*ntegral* and *Derivative* control parameters. They describe how the error signal *e* (see *Figure 33Figure 33*) is amplified in order to produce an appropriate correction. The goal is to reduce this error, i.e. the deviation between the set (or *demand*) value and the measured (or *actual*) value. Low values of control parameters will usually result in a sluggish control behavior. High values will lead to a stiffer control with the risk of overshoot and, at too high an amplification, the system may start oscillating. This can easily be observed on our system by manually increasing the auto tuning parameters by a factor of about 5.

On speed controllers, a closed-loop control circuit with a simple PI algorithm is usually implemented. On positioning systems, a derivative term is also necessary. The three terms may influence each other and understanding this interaction is of particular importance for the fine-tuning of a positioning system. For optimal system performance, the coefficients K_P, K_I and K_D must be set depending on the specified motion and load inertia (Literature: Feinmess).

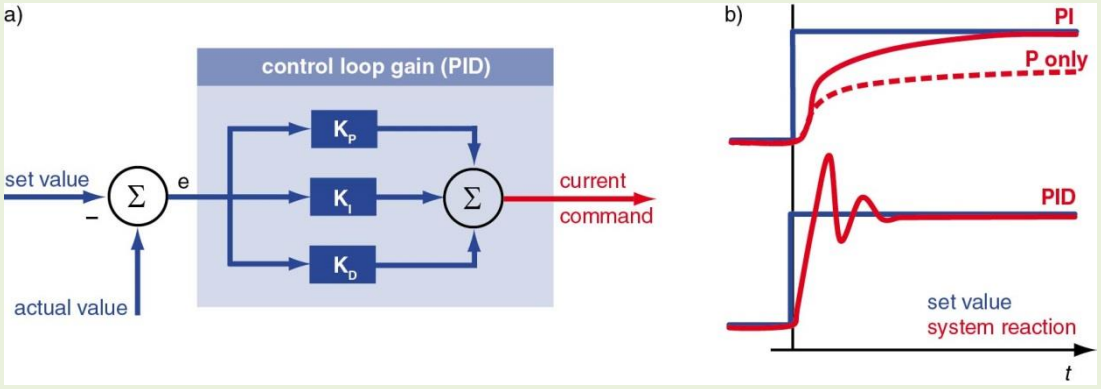


Figure 33: PID controller.
a) Schematic representation of a position or speed controller with the PID element as an amplifier for the error signal e.
b) System response to a set value change with a P, a PI and a PID element implemented.

Proportional controller (P): The error signal e (difference between actual and set value) is multiplied by a user-specified factor K_P and then transmitted as a new current command. An elevated K_P value accelerates error correction. If K_P is too large however, severe overshoot will occur. At even higher values of K_P the system may begin to oscillate, which in turn can result in instability if the damping in the system is insufficient.
K_P cannot completely eliminate the error e, since the proportional correction value ($K_P \cdot e$) becomes smaller as the error e decreases. The result is a residual error that is particularly important in systems that require a high current simply to maintain motion (e.g. because of high friction).

Integral controller (I): Here, the error is summed up over a certain time, multiplied by a user-specified factor K_I and then added to the new current command. This eliminates the residual error of the P only amplification because a longer lasting deviation will provoke increasing corrective action.
However, there is an inherent disadvantage to this method particularly in cases where the error oscillates between positive and negative values. As past errors have a delayed effect, this can lead to positive feedback coupling and destabilization of the entire control circuit. Without appropriate damping, high K_I values can cause severe system oscillations.

Derivative controller (D): The D controller considers the change in the error, which is multiplied by a user-specified factor K_D and added to the current command. Sudden errors, such as set value jumps, can be very quickly corrected with this method.
Properly set, this type of control can improve control stability. It can be considered a type of electronic damping. Increasing the K_D value results in greater system stability, but too high a K_D will lead to oscillations.

Feed-forward

With the PID algorithms, corrective action only occurs if there is a deviation between the set and actual values. For positioning systems, this means that there always is – in fact, there has to be – a position error while in motion. This is called *following error*. The objective of the feed-forward control is to minimize this following error by taking into account the set value changes in advance. Energy is provided in an open-loop controller set-up to compensate friction and for mass inertia acceleration. Generally, there are two parameters available in feed-forward. They must be determined for the specific application and motion task:

- *Speed feed-forward gain*: This component is multiplied by the demanded speed and compensates for speed-proportional friction.
- *Acceleration feed-forward correction*: This component is related to the mass inertia of the system and provides sufficient current to accelerate this inertia.

Incorporating the feed forward features reduces the average following error when accelerating and decelerating. By combining a feed-forward control and PID, the PID controller only has to correct the residual error remaining after feed-forward, thereby improving the system response and allowing very stiff control behavior.

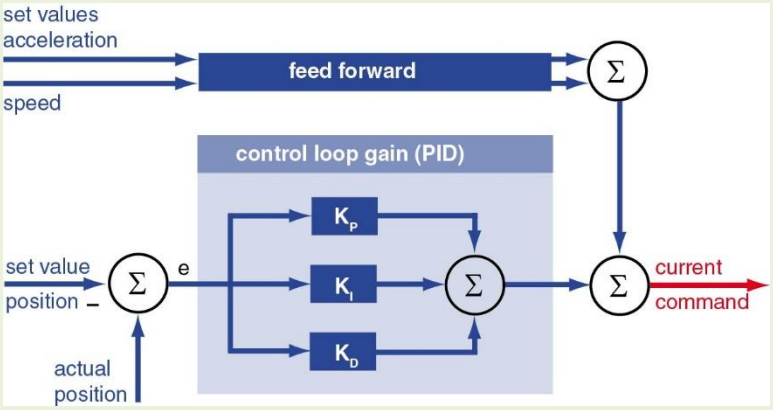


Figure 34: Schematic of feed-forward control.
The feed-forward values are calculated from the required set value changes, i.e. acceleration and speed. Both values are added to the PID current command value.

Part 4: The EPOS4 Motion Controller

In Part 3 we have configured the system, tuned the motion controller and saved all the settings on the device. The *EPOS4* motion controller is now ready for further exploration.

- Chapter 8 investigates the *EPOS4* motion controller with the tools to study the different operating modes.
- Chapter 9 explores the inputs and outputs associated with the motion controller.

In our setup, the *EPOS Studio* takes the role of the master. From the Studio, we send single motion commands (and others) and we can follow directly on the motor and in the Studio how they are executed. For the moment, there is no need to do any programming. Just play!

Master, **slaves and on-line single commands**
The *EPOS4* motion controllers are employed as slaves in a network. Slaves get their commands from a superior system, called *Master*.

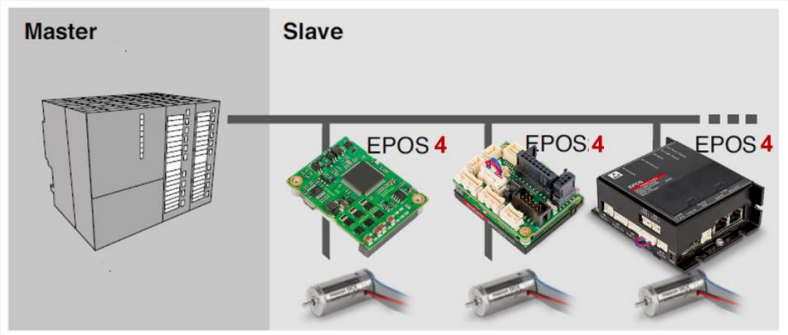


Figure 35: Master-Slave architecture with EPOS4 slaves.

In the master runs the software that controls all the processes and slaves in the network. A master can be a PLC, a computer or a microprocessor.
The master software used in this Part 2 is the *EPOS Studio*.

Important to note is that the slave *EPOS4* cannot store any command sequences; it is on-line commanded by single commands that are sent from the master and executed immediately one by one.

8 Exploring the Motion Controller

The tools for the exploration of the motion controller can be found in the *Tools* tab of the *EPOS4.*

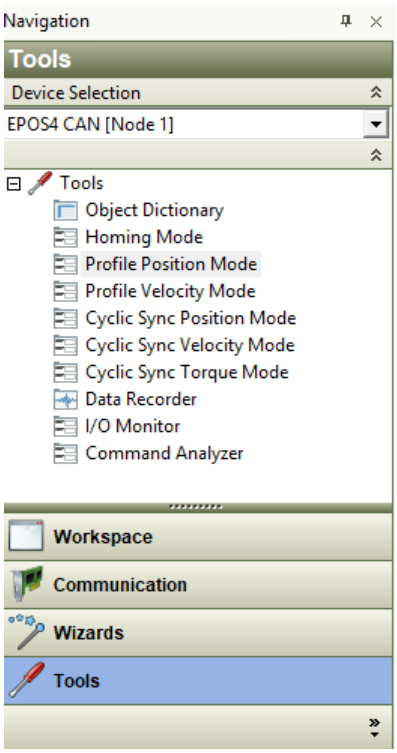


Figure 36: How to open Profile Position Mode*.*
Select the Tools *tab, then select the device* EPOS4 [Node 1]*, and then double click on* Profile Position Mode*.*

8.1 Profile Position Mode

Objectives: Executing a point-to-point movement and monitoring the motion by the built-in *Data Recorder*.

The standard positioning mode for most applications is the *Profile Position Mode*. Positioning is done by following a speed profile that is automatically created by the built-in path or trajectory generator. The path generator has a clock rate of 0.4 ms, adapted to the 2.5 kHz sample rate of the position control loop. Every 0.4 millisecond, a new *Position Demand Value* is calculated and fed into the position control loop as a new set value. The generated trajectory considers motion parameters such as acceleration, velocity, target position, profile type and others.

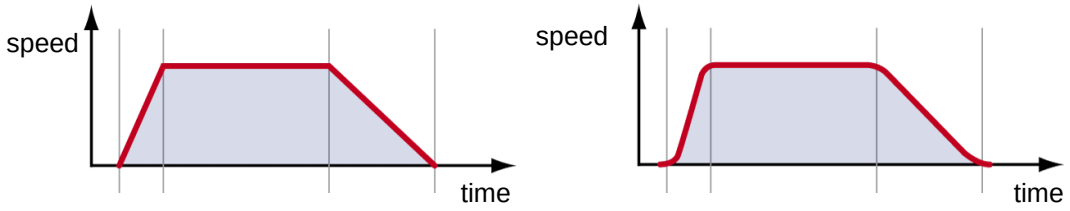


Figure 37: Speed profiles (speed vs. time) for a position move.
Left: A trapezoidal profile with fast acceleration and slower deceleration rate.
Right (not available yet): The same profile but with smooth acceleration changes (sinusoidal profile).

Point to point movements

Here is a first step-by-step recipe how to command a movement.

Step 1 Select the tool *Profile Position Mode* and activate the mode.
Step 2 Set *Target position* to 20'000 inc.
Step 3 Set *Profile velocity* to 500 rpm.
Step 4 Set *Profile acceleration* to 10000 rpm/s.
Step 5 Set *Profile deceleration* to 5000 rpm/s.
Step 6 Select Relative target.
Step 7 *Enable* the EPOS4.
Step 8 Start the movement by clicking on the *Move to target* button and observe the motor spinning. Follow the motion also in the parameters of the *Outputs* area in the upper right of the dialog window. Repeat at will.

Additional exercises

Try other movements with different parameters (velocity, target position …).
Certainly, you can figure out the difference between the settings *Absolute target* and *Relative target*. What is the end position if you click on *Move to zero*?
Try to move the disk on the motor shaft out of the target position by hand. What do you feel?

Observe also the *Status* indications on the lower right of the dialog window.

Green and red LED: Indication of the internal EPOS state
The green and red LED on the EPOS4 indicate the state of the internal EPOS4 motion controller.

- Green LED blinking — Power state disabled. The motor is not powered.
- Greed LED continuously ON — Power stage enabled. The motor is powered and controlled.
- Red LED ON — An error has occurred.

The state of these LEDs can also be seen on top of the dialog windows in the EPOS Studio.

The control loops in motion control
In motion control, the regulated parameter is generally a force or torque, velocity (speed) or position. Correspondingly, the controller is configured as a current, speed or position controller. It is often possible to switch between different types of control or operating modes.

Control of force/torque
This is the fundamental task of a drive in a closed-loop control system. It is the basis upon which the higher-level speed and positioning control systems build up.
Forces and torques are required to set masses in motion, to decelerate and stop their motion and to overcome friction forces. There can also be components of gravity against which work must be performed, either to lift and hold loads or to decelerate a downward motion. In some cases of torque control, it is not a question of moving anything, but simply a matter of pulling or pushing with a defined force against an obstacle.
Motor torque is a linear function of the motor current. Therefore, the basis of producing controlled forces and torques is to regulate and to control the motor current. It is the most basic control loop.

Control of velocity/speed of rotation
The velocity controller attempts to match the actual measured speed to a demanded speed. Rotating or moving a body at a specified velocity requires that the body be accelerated first. Later the forces and torques that affect the speed must be compensated. To accomplish this objective, the motor must generate the necessary torque - i.e. it needs current. Thus, the velocity controller sends a corresponding command to the lower-level current control loop (similar to the position controller in *Figure 38*).

Position control

To move an object to a specified final position, the object must first be set in motion (accelerated), moved and then decelerated. Finally, the specified position must be maintained against any interfering forces. All these operations require a motor torque that must be made available to the position control system. Ultimately, position control relies on the lower level current control loop for creating the necessary torque.

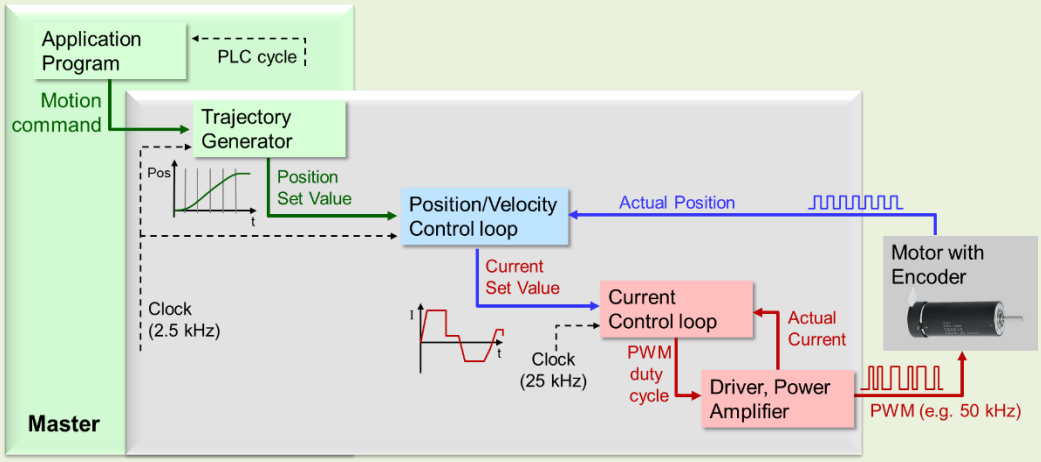


Figure 38: Diagram of a closed-loop position control system with a lower-level current controller. The Application Program in the master system sends motion commands to the motion controller. The commands are processed by the path or trajectory generator that calculates intermediate positions at given time intervals (0.4 ms on EPOS4 systems) on the path to reach the final position. These set values are fed into the position control loop, which, by comparison with the actual position, determines the set (or demand) values for the current controller.

The position and speed control loop in the EPOS4 *work at a sampling frequency of 2.5 kHz; the control loop receives new position information every 0.4 millisecond. This is fast enough for almost all applications, because mechanical reaction times are of the order of several milliseconds at least, as can be deduced from the mechanical time constants of the motors.*

The current controller, via the output stage or driver, controls the motor current which leads to the mechanical reaction of the drive. In the EPOS4 *the current control loop has a cycle time of 0.04 ms or 25 kHz. This is 10 times faster than the position control loop implying that the current is set almost instantly if needed from the position or speed controller.*

Recorder configuration

The *Data Recording* is a useful tool for the analysis of motion behavior. It works in a similar way as an oscilloscope.

We use the recorder to follow the motion parameters in detail. Here is a basic step-by-step configuration.

Step 1 Select and open the tool *Data Recorder*.
Step 2 In the *Data Channels* section select *Position demand value* from the pull-down menu for the topmost channel.
Step 3 Repeat with the second channel choosing *Position actual value* as the parameter to display.
Step 4 Set the *Data sampling time* to 2.0 ms.
Step 5 In the Triggers section, select *Single Trigger* and set the *Type* to *Motion states*, the *Trigger* to *Drive follows command* and the *Delay* to 100 ms. The data display will start 100 ms before the next motion command is performed.
Step 6 Press the *Start* button to activate the data recorder.

Recordings

Change to the tab *Profile Position Mode* and activate the mode if needed. Execute a relative movement, e.g. with the following parameters:

- *Target position* 8192 inc (that's 2 motor turns)
- Profile velocity 1000 rpm
- Profile acceleration 2000 rpm/s
- Profile deceleration 1000 rpm/s

Check the recorded data in the tab *Data Recording* and compare the position demand value with the actual position.

Additional exercises

Try different settings of the *Data Recorder* and try different motion profile parameters. Use the hints in the following *Best Practice* box.

Data Recorder
The data recorder is a powerful tool to examine all kind of signals: motion related parameters such as speed, position and current, but also encoder or Hall sensor feedback or I/O and communication signals.

Triggers: There are several possibilities to start the recording of data. Select the most appropriate depending on your signal.
For example, continuous acquisition may be used for speed measurements (refer to chapter 0), while single trigger acquisition with the trigger set to the start of the motion fits the recording of position moves better (see above). In case you are interested how the end position is attained, you may select *Target reached* as the trigger.

Cursor: Activate the cursor by right mouse button click on the diagram and selecting *Cursor*. At the bottom the curser values for each channel are shown. Use the checkboxes in the *Data Channels* section to select which curves to show.

Scaling: Deselecting the checkbox *Auto Scale* allows to set individual scales for each channel. Activate first the corresponding channel in the upper right box. Comparing signals of the same kind (e.g. position demand and actual value) is easier when displayed at the same scale. *Auto scale*, however, has the big advantage that the signal can always be seen and is not out of the defined display rage. Hence, record the data with *Auto Scale*, then analyze it with manual scaling, zooming and the cursor function.

Zoom into the diagram by marking the interesting area with the mouse, e.g. into the end of the position move to check the accuracy of positioning and the settling time. Zoom out with double mouse click.

Save and export of recorded data to a .csv file, e.g. suitable for Microsoft Excel import! (Hint: Use right mouse button on diagram).

Following error window
The *Following error window* specifies the maximum permissible difference between demanded and actual position at any time of evaluation. It serves as a safety and motion-supervising feature.
If the following error becomes too high, this is a sign of something going wrong: Either the drive cannot reach the required speed, or it is even blocked. However, allow for a certain amount of following error because feedback control will only work if there is a difference between the demanded and actual value. Hence, don't reduce the *Following error window* too much.

Position accuracy
On a well-tuned system, end positions are reached to within 1 encoder increment (inc). As soon as the position is 1 inc off the target the position control loop will take corrective actions. Remarks: The system reaction upon deviations from the required positions depends on the torque and speed capabilities of the system, on the friction and mass inertias and on the set feedback and feed-forward parameters. The controller parameter values have been established during the tuning process based on the system properties. One recognizes again that tuning should be done with the full system in place.

Another aspect of position accuracy concerns how the target position is assumed to be reached. There are different parameters that describe the *Standstill window* as can be seen in *Figure 39*.

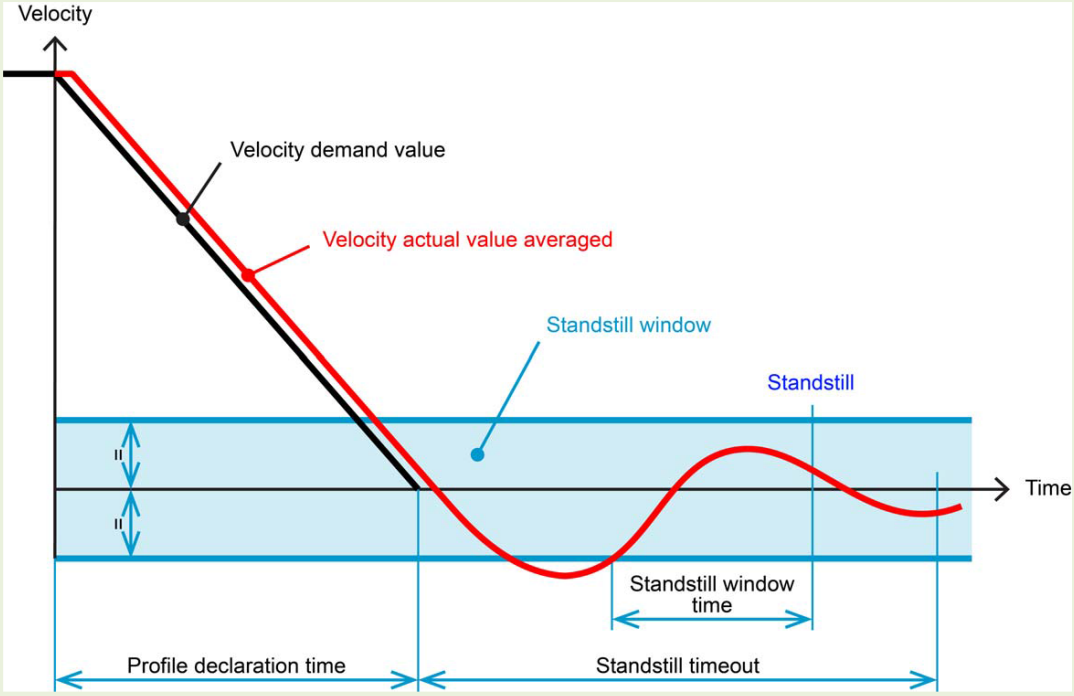


Figure 39: Standstill window*.*
The parameters that govern the reaching of the end position of a movement (from the document EPOS 4 Firmware Specification*). Only if the speed remains sufficiently small (i.e. within the* Standstill window*) for a specified time duration (*Standstill window time*), the target position is assumed to be reached.*

8.2 Homing

Objectives: Executing a homing on a mechanical stop and on a digital signal.
Knowing the different parameters of the homing modes.

Homing on mechanical stop

First, we execute a homing by the *Current Threshold* method. We simulate a blocking mechanical stop by an increasing current level. Just follow these steps:

Step 1 Select the tool *Homing Mode* and activate the mode.
Step 2 Select from the drop-down the homing method *Current Threshold Negative Speed.*
Step 3 Set the parameter *Speed for switch search* to 60 rpm.
Step 4 Set the parameter *Current threshold for homing mode* to 1000 mA.
Step 5 *Enable* the EPOS4.
Step 6 Start the homing with the *Start* button.
Step 7 Mimic the running into a mechanical obstacle by stopping the motor disk with your finger. As soon as the motor current rises above the 1000 mA threshold, the EPOS4 assumes the end of the motion range to be reached and defines this as the home or reference position.

Homing on limit switch

Before we start with the homing on a limit switch, we need to verify that we have a homing signal available on one of the digital inputs. For this, select the tool *I/O Monitor*. Set the digital inputs 1 to 4 according to *Figure 40*, which is their default setting. You can set input signals by activating the corresponding switches on the connected PCB.

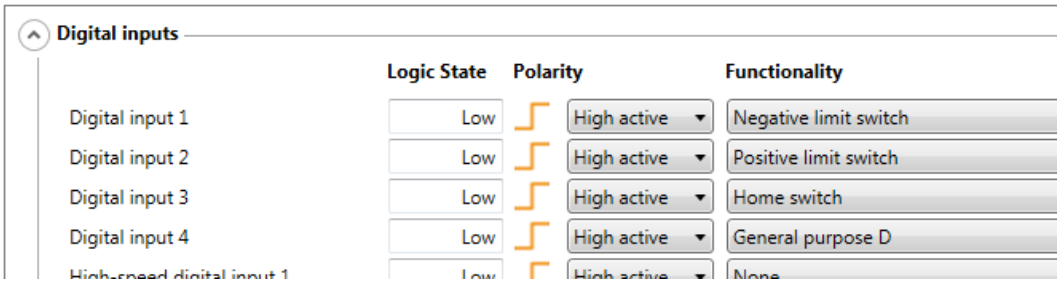


Figure 40: Configuration of the digital inputs for the homing exercise.

For this homing method, just follow these steps:

Step 1 Select from the drop-down the homing method *Negative Limit Switch and Index.*
Step 2 Set the parameter *Speed for switch search* to 100 rpm.
Step 3 Set the parameter *Speed for zero search* to 10 rpm.
Step 4 *Enable* the EPOS4.
Step 5 Start the homing with the *Start* button.
Step 6 Mimic the limit switch to be reached by activating the digital input 1, i.e. the switch 1 on the PCB.

As a result, the EPOS4 assumes the end of the motion range to be reached when the switch is activated. It then adds a motion back to the index pulse of the encoder and defines this as the home or reference position.

Other homing methods and homing parameters

Repeat the homing exercise using different parameters and try to find answers to the following questions (Refer to the green *Motion Control* boxes on the next page.):

- Which mechanical positions does the disk on motor shaft stop at?
- What changes if you select a homing method with *… Positive Speed*?
- Which mechanical positions does the disk on motor shaft stop at if you select a homing method with … *and Index*?
- If you execute a homing with index: What is the difference between the parameters *Speed for switch search* and *Speed for zero search*?
- How does homing methods with *Actual Position* work? And what is the functionality of the *Define position* button?
- What is the influence of setting a *Home offset move distance*?
- What is the meaning of the parameter *Home position*?

Observe the *Status* indication on the lower right as well.

What is homing and why is it needed?
The output signals of incremental encoders can only indicate relative position information or position differences. For absolute positioning, the system must be initialized to a zero or reference position first. This process is called homing, which is realized by moving the mechanism to a predefined position, relative to which all other positions are referenced. Such reference points are usually implemented by means of an inductive home or limit switch or by means of a mechanical stop. Reaching a mechanical stop leads to a current increase. When the current exceeds a predefined threshold, the home position is assumed to be attained.

Homing with additional move to index channel
To increase the repeatability and accuracy of the home position, an additional motion to the first state transition of the index channel (channel I or Z of the encoder) can be performed. The index channel I delivers one pulse per revolution, resulting in an absolute position within one motor turn.
By using this method of homing, it no longer matters if the reference switch actuates somewhat earlier or later; the homing position is now very precisely defined by the transition of the index channel state.

However, be careful in real systems. Using the index as a precise home position can only work, if the index pulse does not fall into the reference switch position error. That is why machine builders try to avoid using this function. If you need to change the motor or encoder, the index will not be at the same position and you will have to recalibrate the system and change the programmed positions.

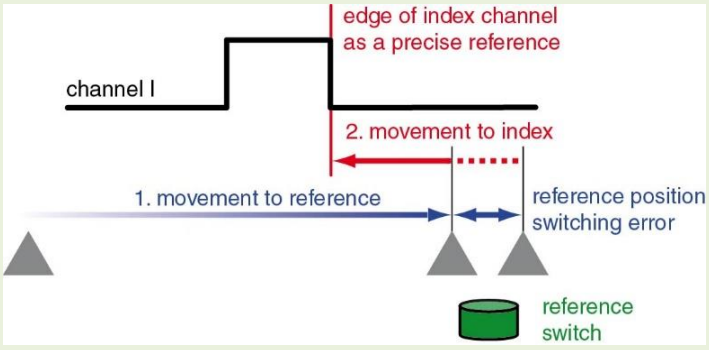


Figure 41: Principle of the homing mode with the index channel of the encoder.

The move to the reference switch (or mechanical stop) is followed by a movement back to the signal edge of the index channel. This always produces the same reference position, regardless of possible position errors during activation of the reference switch (due to fouling, for example).

8.3 Cyclic Synchronous Position (CSP) Mode

Objectives: Recognize the differences between *Profile Position Mode* and *Cyclic Sync Position Mode*.

The *Cyclic Sync Position (CSP)* mode allows positioning without profile, i.e. the built-in trajectory generator of the EPOS4 does not intervene. Instead, the position profile is generated in the master system. The basic idea of the CSP mode is to send a stream of position set values with short time intervals (ms or below) over the field bus to the EPOS4, typically using PDO communication. The short time difference between the set values results in only small position changes. The EPOS4 uses this flow of new target positions as immediate set values in the position control loop.
The *CSP Mode* allows the master system to synchronize several axes if required.
In the *CSP Mode* only absolute positioning is possible, there is no relative motion. Hence, check the actual position before executing a CSP command.

In the EPOS Studio the CSP Mode can only be exploited to a very limited extent, since there is no possibility to create a stream of set values as it is required to mimic a motion profile. All we can do is evaluating single step responses. In order to respect the nature of CSP only small position steps should be considered.
Again, first a recipe for the evaluation of the *CSP Mode*

Step 1 Set the actual position to 0, e.g. by homing. This avoids troubles with too high a following error.

Step 2 Configure the *Data Recorder*:
Select *Position Demand Value* and *Position Actual Value* as recorded parameters.
I recommend using fixed axis scales, e.g. -50 inc to 150 inc.
Optionally you can also record the *Actual Current*.
Set the minimum 0.4 ms as *Sampling Time*. That is the cycle time of the position control loop.
In the Triggers section, select *Single Trigger* and set the *Type* to *Motion states*, the *Trigger* to *Drive follows command* and the *Delay* to 10 ms.

Step 3 Change to Tab *Cyclic Synchronous Position Mode*, activate the mode and *Enable* the device.

Step 4 Start a move of 100 qc (*Apply Position*) and observe the motor reaction.

Step 5 Examine the move in the recorded diagram. Observe how the position demand value changes immediately and how the actual position tries to follow, overshoots and finally settles at the commanded target position.

Additional exercises

Repeat the procedure with larger and smaller position commands. Adjust the *Data Recorder* scale accordingly.
Change the position control tuning parameters (hard vs. soft) in the *Regulation Tuning* wizard. What is their influence on the system reaction?

Cyclic Synchronous Modes

In the cyclic synchronous position and velocity modes (*CSP* and *CSV*), the trajectory generation is done in the commanding master. New set values are sent at a sufficiently high rate from the master to the motion controller (e.g. an EPOS4) typically using PDO communication (refer to chapter 10.6).

The *CSP* and *CSV* modes are useful operating modes for a situation where the axis acts as a slave axis commanded by progressive position or speed set values without large jumps. Accordingly, the built-in trajectory generator is not needed. Typically, the cyclic synchronous modes are used to synchronize the motion of multiple axes, where the combined path planning for all the axes is done in the master. The master generates new position set values for all the axes based on each axis actual position and speed. This requires a high-performance communication (typically EtherCAT) for reading the actual values of all the axes and sending new commands with a sample rate in the milliseconds range. CANopen communication very soon comes to a limit concerning when it is about synchronizing several axes at reasonably fast synchronization times. Therefore, the cyclic synchronous modes with EPOS4 very often require the much more performant EtherCAT communication.

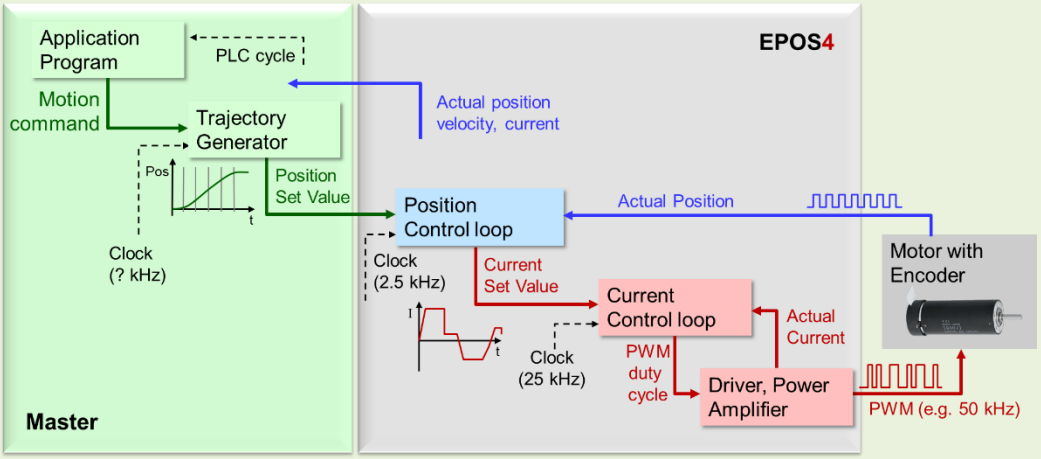


Figure 42: Schematic of the CSP mode.
Comparison to Figure 38 shows that trajectory generation is now done in the master system. The EPOS has to provide the master with actual position information.

Following Error* Window and *CSP Mode

Set the *Following Error Window* to a value higher than the largest position step to be expected. If not, there might be an immediate error message as soon as the EPOS tries to execute the command since the new set value (i.e. the target position) differs too much from the actual position (i.e. the starting position).

***Cyclic Synchronous Position* (CSP) Mode**

As mentioned above, the EPOS Studio does not allow to exploit the *Cyclic Synchronous Position* (*CSP*) Mode properly. There is no possibility to send a stream of position set values in short (ms) intervals. Nevertheless, we can study the reaction upon a single motion step to get a feeling for it.

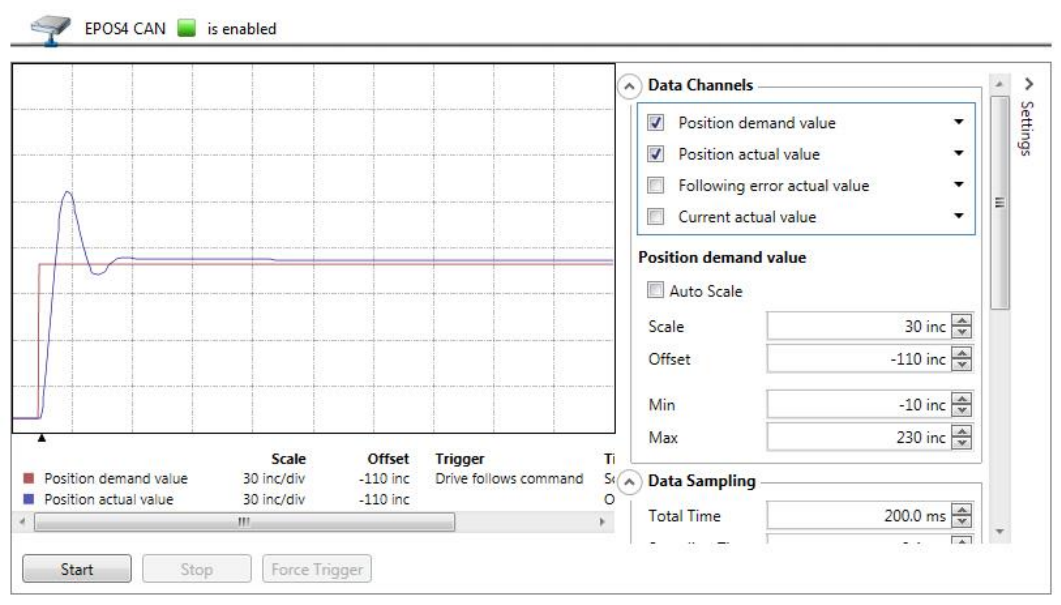


Figure 43: Position Recording in CSP mode.

The red and blue curve show the demanded position step of 100 inc and the actual system reaction, respectively. There is a considerable overshoot and it takes about 30 ms for the motor to settle. The overshoot is due to the quite stiff regulation tuning and the motor speed of up to about 200 rpm in this case.
Be aware that in real CSP mode applications the motion profile in the master creates new position steps that would be much smaller than 100 inc. This is particularly true towards the end of a motion leading to a much smoother motion and less overshoot (see Figure 44).

Figure 44: Position Recording in CSP mode for a small step of 10 inc. All the parameters are the same as in Figure 43.

8.4 Profile Velocity Mode

Objectives Setting a speed in the *EPOS4* (velocity control).
Monitoring the speed in the built-in *Data Recorder*.

The next operating mode to explore is the *Profile Velocity Mode*. This mode controls motor speed and not position. The new target speed is achieved by a speed ramp - or if you like a velocity profile, hence the name of the mode - with acceleration or deceleration values that can be set. The *Profile Velocity Mode* allows smooth speed changes.

As a starting point for your exploration in the *EPOS Studio* execute the following velocity profile and record the movement.

Step 1 Select the tab *Data Recorder* and configure the recorder.
Set the data channels as.
Channel 1: *Velocity demand value*
Channel 2: *Velocity actual value*
Channel 3: *Velocity actual value averaged*
Set the minimum *Sampling Time* of 0.4 ms (corresponding to the velocity control sampling frequency of 2.5 kHz)
Select the trigger as *Continuous Acquisition* and press *Start*.

Step 2 Select the tab *Profile Velocity Mode* and activate the mode.

Step 3 Set *Target Velocity* to 2000 rpm. If you set a relatively low acceleration value of e.g. 200 rpm/s you can observe the disk on the motor shaft speeding up.

Step 4 *Enable* the device and start the motion with the button *Set velocity*.

Step 5 Observe the velocity signals on the data recorder. For a better view and analysis of the signals, *Stop* the data recorder and adjust the scales of the different velocity signals. For instance, set the Min and Max of the velocity scales to 1900 rpm and 2100 rpm, respectively. Observe, that at 2000 rpm one motor turn takes 30 ms. Refer to the next green *Motion Control* box on *Understanding Velocity Signals on Data Recorder.*

Recording the speed ramp

The speeding up of the motor can be followed if the *Single Trigger* method is selected in the *Data Recorder*.

Step 1 Set the *Type* to *Parameter values*, the *Trigger* to *Channel 1*, the *Mode* to *Rising above value* 10 rpm and the *Delay* to 100 ms. For displaying the full velocity range, set the scales of all channels to -100 rpm to 2500 rpm. Keep all the other settings of the Data Recorder. Press *Start*.

Step 2 Select the tab *Profile Velocity Mode* and activate the mode.

Step 3 Keep the *Target Velocity* at 2000 rpm but increase the acceleration to 20000 rpm/s.

Step 4 *Enable* the device and start the motion with the button *Set velocity*.

Step 5 Observe the velocity signals on the data recorder. Keep a mental record of the diagram for later comparison with the CSV mode (chapter 8.5). Remark: The average speed signal lags due to the averaging time needed.

Additional exercises

Repeat the procedure with larger and smaller velocity commands. Adjust the *Data Recorder* scales accordingly.
Change the velocity control tuning parameters (hard vs. soft) in the *Regulation Tuning* wizard. What is their influence on the system reaction?

Understanding Velocity Signals on Data Recorder
Looking at the speed signals on the data recorder you might see something resembling the following picture. (Make sure that you have set the same scale for all the signals; e.g. by giving fixed *Min Max* values.)

Obviously, the horizontal red line is the commanded speed value of 2000 rpm. The blue signal is the speed reading of the sensor. It looks quite noisy and a closer look shows that it jumps in steps of about 37 rpm. Additionally, there is a certain periodicity in the signal with a period of 30 ms. This periodic behavior (with 4 variations per 30 ms) can more clearly be seen in the green signal that shows the averaged velocity.

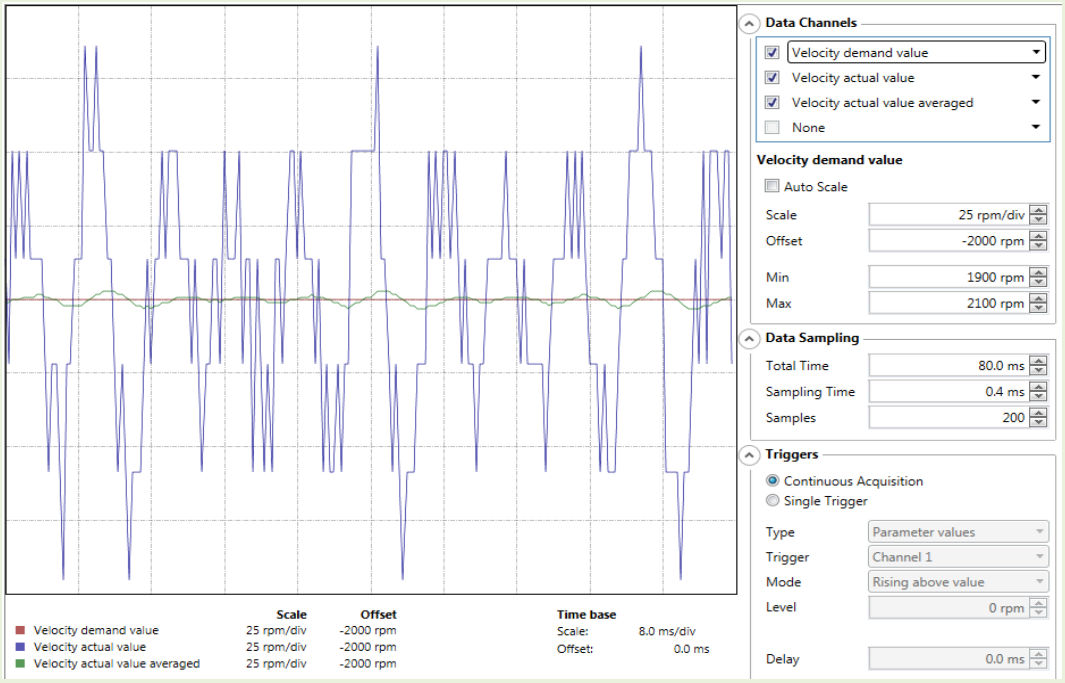


Figure 45: Velocity signals recorded on a DCX motor with EASY encoder.
Screenshot from the Data Recorder.

Where do the steps of 37 rpm in the blue signal come from? Certainly, the motor is not able to change its speed in the way suggested by this signal; its mechanical time constant being too high (> 4 ms). Instead, what we observe is a quantization phenomenon in

speed measurement stemming from the fact that velocity is calculated from the position change per sampling interval of 0.4 millisecond. The position can change in steps of 1 encoder increment (inc). 1 inc per 0.4 millisecond is equivalent to 150'000 inc per minute and this corresponds to 36.6 rpm for an encoder with 4096 inc per turn. Therefore, if you want to know the real speed of your motor it's better to look at the *Velocity actual value averaged*. i.e the green signal in the diagram above. It makes more sense.
By the way, the speed accuracy of the averaged signal is about 3 rpm deviation at a speed of 2000 rpm. And this is rather good!

Where does the periodicity stem from?
The periodicity in the *Velocity actual value* (blues signal) and in the *Velocity actual value averaged* (green signal) correlate with the rotation of the motor. At 2000 rpm one turn of the shaft takes exactly 30 ms, as observed in the signal periodicity.
One can think of two possible reasons for these oscillations: they stem either from some irregularity in the motor, e.g. an enhanced friction on some motor positions, or from imperfections of the measuring device, i.e. the encoder.
Here, I think it is rather the second reason. EASY encoders are known to have this sort of waviness with 4 periods per motor turn stemming from the interpolation process in the encoder. A comparison with a very accurate optical encoder clearly shows the difference (*Figure 46*).

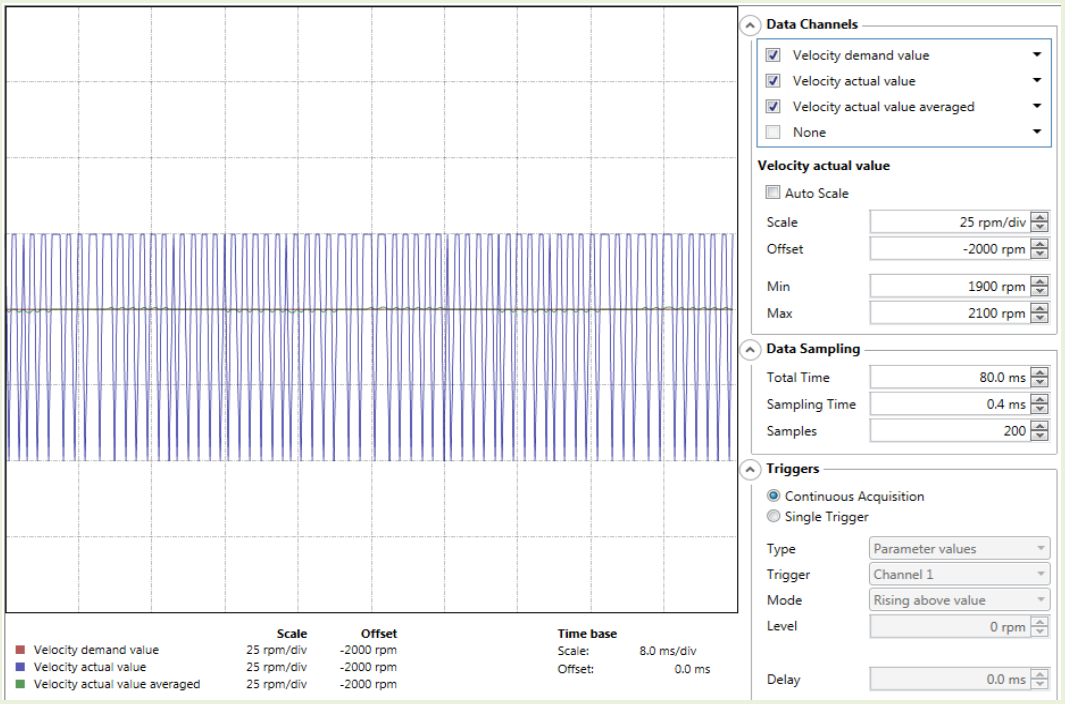


Figure 46: Velocity signals recorded on a motor with optical encoder (500 cpt).
Screenshot from the Data Recorder.

8.5 Cyclic Synchronous Velocity (CSV) Mode

Objectives: Recognize the differences between *Profile Velocity Mode* and *Cyclic Sync Velocity* (CSV) *Mode*.

The *Cyclic Synchronous Velocity* (CSV) *Mode* works in a similar way as the *Cyclic Synchronous Position* (CSP) of paragraph 8.3. The *CSV* mode, however, uses velocity control (instead of position control in *CSP* mode). Again, the master generates the motion trajectory and speed ramps and feeds the EPOS4 slave with a stream of gradually changing velocity demand values.

In the EPOS Studio the CSV Mode can only be exploited to a very limited extent, since there is no possibility to create a stream of set values as it is required to mimic a speed ramp. All we can do is evaluating single velocity step response.
As a starting point for comparing the speed changes in *Profile Velocity Mode* and *Cyclic Sync Velocity* (CSV) *Mode* we use the same velocity change and Data Recorder settings as in chapter 8.4 for monitoring the speed ramp.

Step 1 Select the tab *Data Recorder* and configure the recorder.
Set the data channels as.
Channel 1: *Velocity demand value*
Channel 2: *Velocity actual value*
Channel 3: *Velocity actual value averaged*
For displaying the full velocity range, set the scales of all channels to -100 rpm to 2500 rpm. Set the minimum *Sampling Time* of 0.4 ms (corresponding to the velocity control sampling frequency of 2.5 kHz)
Select *Single Trigger* and set the *Type* to *Parameter values*, the *Trigger* to *Channel 1*, the *Mode* to *Rising above value* 10 rpm and the *Delay* to 100 ms.
Press *Start*.

Step 2 Select the tool *Cyclic Sync Position Mode* and activate the mode.

Step 3 Keep the *Target Velocity* at 2000 rpm

Step 4 *Enable* the device and start the motion with the button *Apply velocity*.

Step 5 Observe the velocity signals on the data recorder. Compare to the acceleration in the *Profile Velocity Mode* (chapter 8.4).

Additional exercises

Repeat the procedure with larger and smaller speed steps. Adjust the *Data Recorder* settings accordingly.

***Cyclic Synchronous Velocity* (CSV) Mode**
***Cyclic Synchronous Torque* (CST) Mode**
Possible drive architectures of the CSV or CST modes, respectively, are shown in the following figure.

The CSV mode architecture is very similar to the CSP, only replacing the position control loop and the position set values by a velocity control loop and velocity set values. In the master there is very often a higher level position controller active generating the velocity set values.

In the CST mode, the motion controller acts as a current controller using the power stage for motor operation. The position control loop is closed in the master itself.

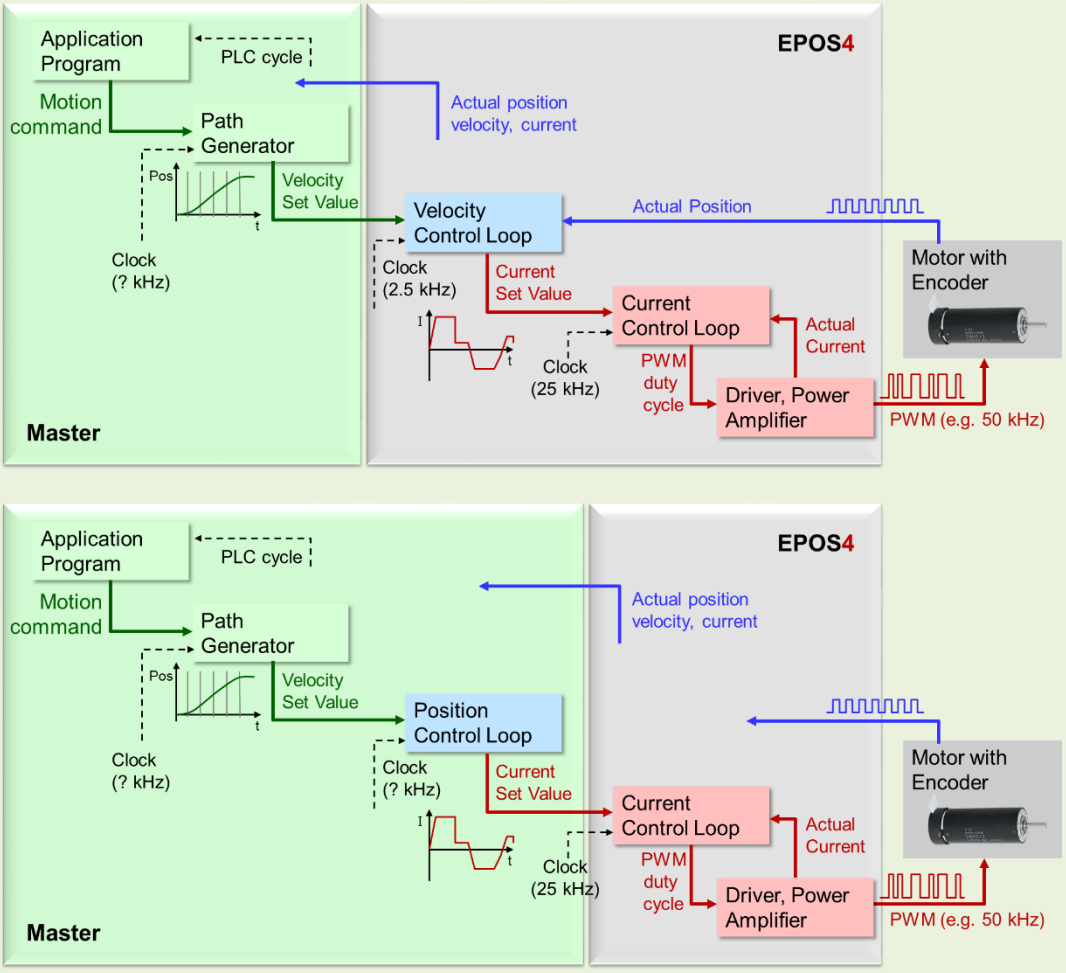


Figure 47: Schematic architecture of the CSV (top) and CST (bottom) modes.
Compare also to Figure 42.

8.6 Cyclic Synchronous Torque (CST) Mode

Objective Learn how the motor behaves upon current control only.

In the *Cyclic Synchronous Torque (CST) Mode*, the motor current is controlled. Since motor current is proportional to torque, regulating motor current is the same as regulating the torque. We directly address the innermost control loop – the current control loop (refer to *Figure 38*) – and bypass the velocity or position control loop.

Explore the *CST Mode*, e.g. with the following steps.

Step 1 Change to tool *Cyclic Sync Torque Mode* and activate it.

Step 2 *Enable* the device.

Step 3 Set the *Target torque* value to 10% of the *Motor rated torque* and press the *Apply Torque* button. What Happens? Observe the speed and current *Output* values on the right part of the screen. Press *Apply zero* to stop the motor.

Step 4 Block the motor shaft with your hand and press the *Apply Torque* button! How do the output values change now? Why?

Step 5 What do you feel on the blocked shaft if you increase the torque 50% or higher? Don't forget to press the *Apply Torque* button.

Step 6 Apply a torque of about 200% with blocked shaft. Can you feel the thermal protection being activated after a few seconds, i.e. the current being reduced? Follow the change of current value in the *Output* section on the right. How long can you apply a torque of 300%?

Runaway in current control mode

Not blocking the motor shaft in current control may result in the motor speed increasing up to very high levels. Why does this happen? Current control means torque control. The controller tries to maintain the current (torque) at the set value. Torque on the motor usually means acceleration => A higher speed results in a higher induced voltage (back EMF) that counteracts the applied voltage. => More applied voltage is needed to maintain the current at the set value. => Higher voltage means the motor accelerates further => higher back EMF => more applied voltage needed => resulting higher speed => …

As a result, the motor speed runs away up to the maximum speed that is possible with the given supply voltage. Such a runaway condition can occur if not restrained by a higher-level control loop (velocity or position control), a mechanical stop or an additional safety speed limitation (as the *Max permissible speed* set in the *Limits* section of the *Startup wizard*).

With the *Cyclic Synchronous Torque (CST) Mode* we have finished the exploration of the basic motion control functionality.

9 Using the I/O Monitor tool

Objectives Using the *I/O Monitor* and practical understanding of the I/O-functionality. (as far as the functionalities are implemented yet, June 2020)

There are several digital inputs and outputs available on the *EPOS4* as well as two analog outputs. These I/Os are primarily intended to be used for special tasks in the periphery of the corresponding motion axis. One example for an input is a limit switch that indicates that the mechanical drive has left the save operation area on a linear drive. Moving the device into the limit switch will cause the motion controller to signal this specific error to the master system. Limit switches can also be used as a reference for homing; in this case without provoking an error as you have seen when executing a homing in chapter 8.2.

The I/Os of the EPOS are freely configurable. You can use them for the predefined functionalities (e.g. home switch functionality) or for any other purpose you like (*General purpose*). You might use one of the digital inputs to start a particular part of your program in the master PLC or read the signal of a temperature sensor with an analog input or signal the end of a subroutine on one of the digital outputs.

There are two kinds of digital I/O: the standard ones (called *Digital inputs 1* to *4* and *Digital outputs 1* to *2*) are used for status signals such as *Ready* or *Limit switches*. The *High-speed digital inputs* serve for reading fast varying signals, e.g. the signals of analog or absolute encoders.

In the EPOS Starter Kit on your table, switches on the PCB can activate the standard digital inputs. The analog inputs read the voltage of potentiometers and the standard digital outputs illuminate colored LEDs. There are no high-speed I/Os connected. Open the *I/O Monitor* tool and try the following exercises to learn how the inputs and outputs work.

For more details, refer to the corresponding chapters in the *Firmware Reference.*

9.1 Inputs

Analog input reading

Turn the potentiometers on the PCB and observe the value reading on the *I/O Monitor*.

Remark: There is a delay between the activation of the input switches or potentiometers on the PCB and the reaction of the *I/O Monitor*. There is a lot of information to be transmitted over the USB communication. Hence, it takes a while for the refreshment of the *I/O Monitor* display in the *EPOS Studio*. Don’t worry, the EPOS4 internal state of the inputs are changed immediately.

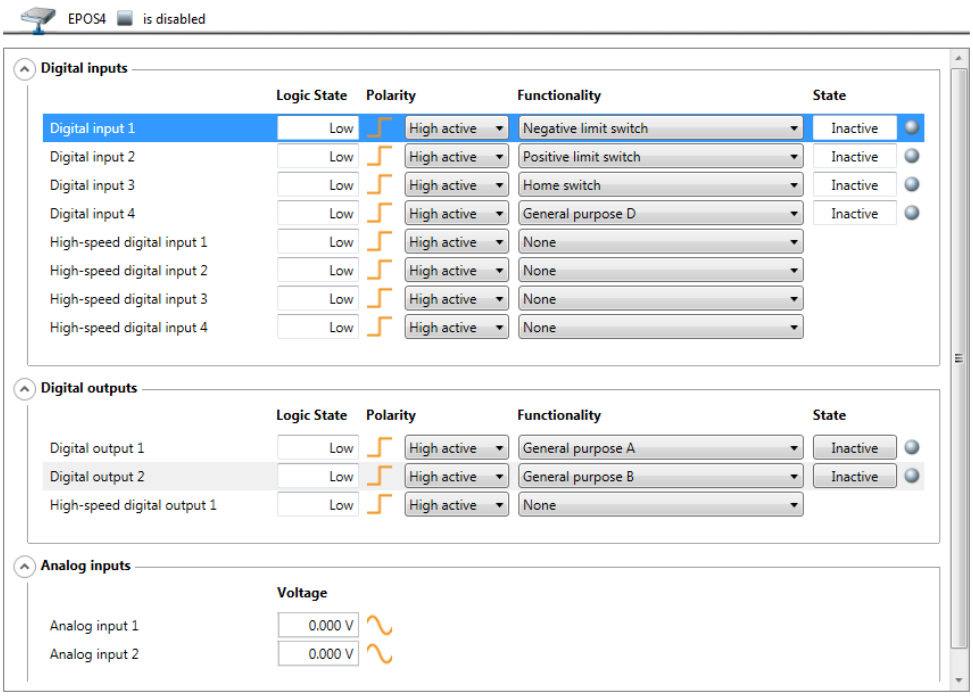


Figure 48: The I/O Monitor tool of the EPOS4.

Digital inputs

By default, *Digital input 4* is set as *General purpose*, i.e. it can be used to read any digital signal typically to be processed in the master program. Try the following

Step 1 Activate the *DigIn4* switch on the PCB and observe the reaction on the *I/O Monitor*. How do the parameters *Logic State* and *State* change?

Step 2 Set the polarity of Digital Input to Low active. What happens to the parameters *Logic State* and *State*? Which of these two parameters stands for the physical signal level (voltage) at the input, which for the Boolean signal level that will result?

Next, we examine the *Digital inputs 1* to *4* that have a functionality assigned to them.

Step 1 Activate the *DigIn1* or *DigIn2* switch on the PCB (limit switches) and observe the reaction on the *I/O Monitor*. What happens to the LED at the back of the EPOS4 controller board or the EPOS status indication at the very top of the *I/O Monitor* window? Also, check the *Status* window at the bottom of the EPOS Studio window for errors and warnings.

Step 2 Reset the activated switch on the PCB and clear the error in the *Status* window (right mouse click). Observe the reaction of the LED.

Step 3 Activate the *DigIn3* switch on the PCB (Home switch) and observe the reaction on the *I/O Monitor*. Why does no error occur in this case?

Step 4 Change the functionality of the *Digital input 4* to *Drive enable* with *High active* polarity. Activate the corresponding switch on the PCB. Change to the *Profile Position Mode* tool and try to disable the EPOS4. What do you observe? Disable the EPOS4 with the switch and try to enable in the *Profile Position Mode* tool.

Step 5 Change the functionality of the *Digital input 4* to *Quick stop.* It's best to observe the behavior of motor speed in the *Profile Velocity Mode* (alternatively in the *Profile Position Mode* with long positioning moves). How does the hardware *Quick stop* (*Digital input 4*) differ from the software *Quick stop* (button in the EPOS Studio communication windows)?

Axis related digital input functionalities

Limit Switches* and *Home Switch

We have already seen what *Limit Switches* can be used for. They limit the mechanical working range on both sides. Very often inductive proximity switches are used as limit switches; if a metallic part comes in close proximity they change their output state. When a limit switch is activated the drive assumes to have moved out of the permitted area. An error signal is generated; the drive comes to a stop and is disabled.

You can use one of the limit switches for homing after power up. No extra *Home Switch* is needed, and it is clear in which direction to look for a limit switch; at the corresponding positive or negative end of the mechanical travel. When using the limit switch for homing no error is generated of course.

For homing, you can also use a specially designed *Home Switch* and its input functionality.

Device enable

In most cases the enable and disable of the power stage in the controller is governed by the commands from the program in the master that controls the application. The *Drive enable* input functionality offers an alternative way to control the power stage by an external digital signal. An *active* state on this input enables the power stage; an *inactive* state disables it.

Observe that defining an input as *Drive enable* will overrule any enable or disable command sent over the fieldbus connection from the master.

This functionality might be useful in situations where you want to remove the power from the motor without referring to the program in the master. However, be careful, this feature does not comply with safety regulations!

Quick stop

The *Quick stop* brings the axis to a stop with the quick stop deceleration ramp and holds the final position. The motor is still powered, i.e. the power stage remains enabled. Software Quick stop and hardware Quick stop work in parallel; if either of the two is activated the motor will stop and hold position. However, as long as the hardware input is activated, no more positioning or speed control is possible. While the software Quick stop is switched off as soon as a new position or speed command is set.

Axis related digital input functionalities

Touch probe
The *Touch probe* input records the actual position when this digital input is activated. A typical situation for this functionality is the following: When an object on a conveyor belt passes a certain point (e.g. a light curtain) this is signaled to the motion controller on the *Touch probe* (position marker) input and the corresponding position of the conveyor is stored. The process control program may now add the offset for the object on the belt to stop at a defined position.

Axis related digital output functionalities

Ready/Fault
The *Ready/Fault* output can be used to signal the correct operation of the device. Think of the common green and red light accessory on production machinery.

Holding Brake* and *Set brake (GPIO)
Holding brakes are used to maintain a position without the motor being powered. This is useful to save energy in applications where the drive has to rest at the same position for a long time or where holding a position requires a lot of torque (e.g. on a vertical drive). Holding brakes hold position when they are not powered. Therefore, holding brakes may also be used as an emergency stopping device upon power shut down.
The powerful ***Holding Brake*** output of the EPOS4 allows the direct activation of brakes without additional power supply, taking into consideration the reaction time of the brake activation.
The ***Set brake (GPIO)*** output is used to drive a brake operated by the host (master) in conformance to CiA 402 specification

Position Compare
Position Compare output is not implemented yet (June 2020).
Position Compare allows sending signals at a predefined position or position intervals. This might be useful to activate another device or process by a digital signal if your axis moves through a specific position.

Axis related analog input and output functionalities
Analog set value reading is implemented with firmware version 0160h or higher.

- Set value current
- Set value velocity

But the following analog output functionalities are not implemented yet (June. 2020)

- Current monitor, Position monitor, Velocity monitor
- Temperature monitor

9.2 Outputs

As a first exercise let us set a digital output and learn how we can mask the physical output.

Step 1 Click on *Functionality* column in the table and set the digital outputs 1 and 2 as *General purpose A* and *General purpose B*, respectively. Set the *Polarity* to *High active*.

Step 2 Set the *State* of one of the digital outputs to *Active* and observe the reaction of the LED outputs on the PCB. How is it related to the indication of the *Logic State.*

Step 3 Change the *Polarity* to *Low Active*. How does this affect the LED output? What is the influence of changing the *State* to *Inactive* or *Active*.

Now we assign a specific functionality to a digital output.

Step 1 Set the *Functionality* of the *Digital output 1* (or any other) to *Ready*. What do the *Logic State* and the LED on the PCB show?
Remark: The *State* cannot be changed manually anymore.

Step 2 Watch the behavior of *Logic State* and LED if you create an error, e.g. by activating a limit switch.

Step 3 Set the *Functionality* of the *Digital output 2* to *Holding Brake*. The output will be set (LED shining, brake released) if the power stage is disabled.
Observe that you can define parameters of brake operation, i.e. the holding brake rise and fall times. (For more information refer to the *Firmware Reference* document.)

Step 4 *Enable* the power stage, e.g. in the *Profile Position Mode*. What happens to the *Holding Brake* output?

Save parameters
In case of a power shut down your parameter settings in the EPOS Studio are lost unless saved permanently in the EPOS4. Save the changed I/O parameters (and others) in the *Object Dictionary* tool. Just right-click on the *Object Dictionary* table and select *Save All Parameter*.
Alternatively, you can open the *Startup Wizard* and press the *Finish* button.

Part 5: CANopen and EtherCAT communication

The *EPOS4* is a *CANopen Device*, or more accurately a *Motion Controller Device* according to the CiA 402 definition. The purpose of this chapter is to give you the most important features and ideas connected with *CANopen Devices*. This provides you with a better understanding of how communication is organized. This will later be useful for programming.

However, this is just an introduction and not a full description of what CANopen is. Please refer to the CiA (*CAN in Automation*) website for further information (www.can-cia.org).

In the optional EtherCAT communication – to be precise the EPOS4 uses CANopen over EtherCAT – one uses the much faster information flow of EtherCAT but maintains the basic device and object structure as described in the following chapter 10. Therefore, we keep the additional information on EtherCAT rather basic.

10 An introduction to CANopen and CAN

CANopen is a communication protocol applied to a field bus that is usually CAN. CAN covers the lower communication levels focusing on technical aspects, while CANopen defines the meaning of the data and devices in the network. In other, rather simplistic words: the CAN bus describes the vehicle how data are transmitted, and the CANopen protocol describes what kind of data are transmitted.
CANopen is a standard maintained and supervised by the independent user organization *CAN in Automation*.

Here, we don't have to bother about the details of how messages are transmitted and received. All we need to know is that CANopen and CAN provide a reliable, low cost bus framework for sending messages between different devices.
Some limitations include:

- Maximum numbers of nodes per bus	127 nodes
- Maximum bit rate	1 Mbit/s up to 40 m total bus length All nodes need to have the same bit rate!
- Typical length of a CAN telegram	100 – 130 microseconds at maximum bit rate

The most important concept we must deal with is the *Device Model* and the *Object Dictionary* therein. But first a few words on CAN and CANopen.

10.1 CAN

CAN stands for *Controller Area Network*. It is a field bus designed originally for automotive applications but is now also used in other areas such as industrial automation and medical equipment.

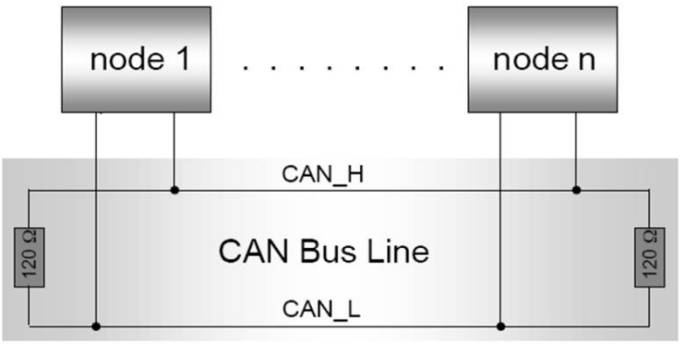


Figure 49: Physical layout of the CAN bus.

On a *physical level,* each device - called a node and given its distinctive address (node number) - is mounted in parallel on the bus. The bus transmits the signals on a differential line which needs proper termination on both sides to avoid signal reflections (120 Ohm resistances). Signal bits require a certain time to spread over the whole bus. Therefore, the transmission rate depends on the bus length. The maximum bit rate of 1 Mbit/s can only be achieved up to approx. 40 m bus length.

"The *Transfer Layer* represents the kernel of the CAN protocol." It is where the structure, transmission and reception of messages are defined. "The transfer layer is responsible for bit timing and synchronization, message framing, arbitration, acknowledgement, error detection and signaling, and fault confinement." (Wikipedia)

In short, CAN gives a framework for microcontrollers and devices to communicate in a safe and reliable way with each other.

10.2 CANopen

"In terms of the OSI model, CANopen implements the higher layers above and including the network layer. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/de-segmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication (such as Ethernet Powerlink, EtherCAT) can also implement the CANopen device profile." (Wikipedia)

In short, CANopen uses the CAN framework for sending meaningful messages in between specified nodes (devices) in a network.

10.3 CANopen device profile

The CANopen device profile is one of the central elements of CANopen. It describes how the nodes in a network must be structured. You can think of a CANopen Device as consisting of three sections: a communication unit, an object dictionary and the application part (Figure 50). In addition, there has to be a state machine implemented for starting and resetting the device. It must contain the states Initialization, Pre-operational, Operational and Stopped. Typically, the pre-operational state is used for configuration while real time communication is restricted to the operational state.

The *communication unit* is responsible for communication with the other nodes in the network. It contains all CAN and CANopen communication features: Receiving and sending network management information as well as writing and reading data in or from the object dictionary.

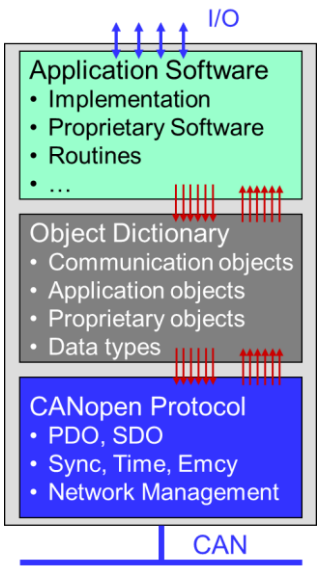
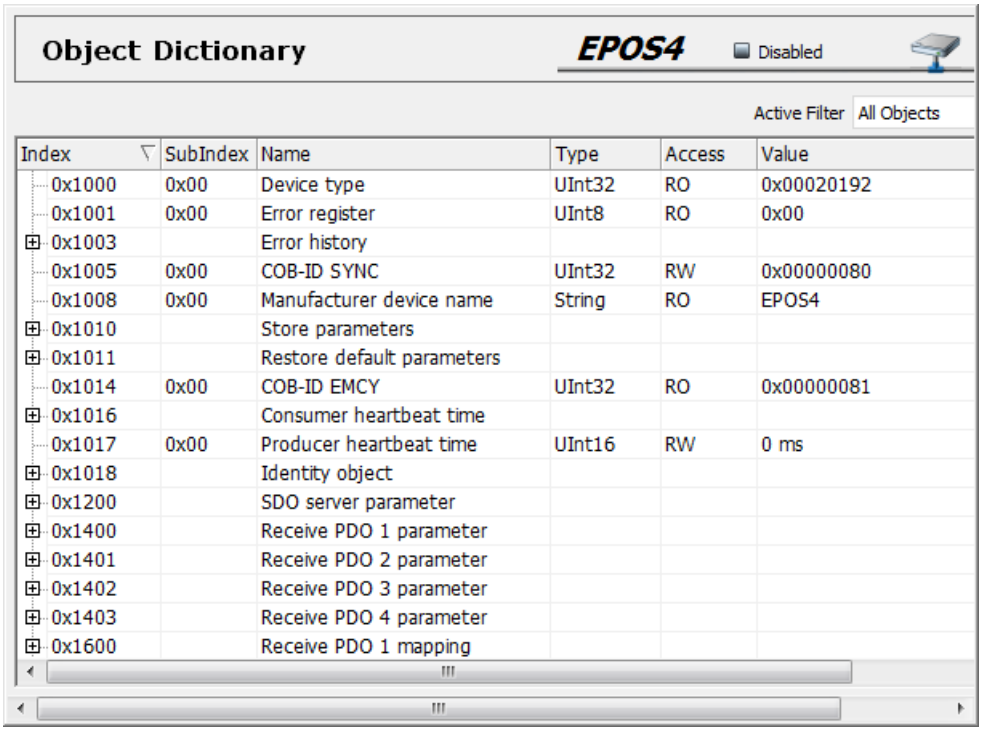


Figure 50: CANopen device pofile

The *Application* part performs the desired function of the device. This can be a motion controller as the *EPOS4*, an I/O device, a programmable logic controller (PLC) or any other functionality. The application is configured and controlled by parameters and variables in the *Object Dictionary*.

Object dictionary

The *Object Dictionary* is the heart of the device. It represents the link between the CANopen communication world and the application. The application uses the data from the object dictionary as inputs to perform its task and writes its results and outputs to the same object dictionary. On the other side, any communication between devices in the CANopen network is based on data exchange between corresponding object dictionaries. Hence, communication with a CANopen device comes down to writing or reading parameters from the corresponding object dictionary.



Index	SubIndex	Name	Type	Access	Value
0x1000	0x00	Device type	UInt32	RO	0x00020192
0x1001	0x00	Error register	UInt8	RO	0x00
0x1003		Error history			
0x1005	0x00	COB-ID SYNC	UInt32	RW	0x00000080
0x1008	0x00	Manufacturer device name	String	RO	EPOS4
0x1010		Store parameters			
0x1011		Restore default parameters			
0x1014	0x00	COB-ID EMCY	UInt32	RO	0x00000081
0x1016		Consumer heartbeat time			
0x1017	0x00	Producer heartbeat time	UInt16	RW	0 ms
0x1018		Identity object			
0x1200		SDO server parameter			
0x1400		Receive PDO 1 parameter			
0x1401		Receive PDO 2 parameter			
0x1402		Receive PDO 3 parameter			
0x1403		Receive PDO 4 parameter			
0x1600		Receive PDO 1 mapping			

Figure 51: Access to the Object Dictionary of the EPOS4 in the EPOS Studio Tools tab.

The *Object Dictionary* consists of a table of parameters describing all the properties of the device: its communication channels, its configuration and settings, but also the application configuration, input and output data.
An entry in the object dictionary is defined by

- Address (16-bit index and 8-bit subindex)
- Name a string describing the parameter
- Type the data type of the parameter
- Access rights read/write, read-only, write-only or read-only constant
- Value of the variable

The basic data types for object dictionary values such as Booleans, Integers and Floating Numbers are given by the CANopen standard, as well as composite data types such as Arrays, Records and Strings.

Restrictions on variable types of EPOS systems
The EPOS4 systems does not contain any float type variables. Numeric variables can only be integers of different size.

Firmware Specification
All the objects of the EPOS4 are described in the *Firmware Specification*. In there, you find information as well about the value range and the meaning of the parameter values (units).

Standard CANopen Device Profiles

Object dictionaries cannot be defined at will. First, there are certain rules to be observed regarding which data can be found where in the object dictionary. Second, there are standard device profiles for a variety of applications. The EPOS4 follows the CiA 402 profile for drives and motion control.

Profile number	Device class
CiA 401	Generic I/O Modules
CiA 402	Drives and Motion Control
CiA 404	Measuring devices and Closed Loop Controllers
CiA 405	IEC 61131-3 Programmable Devices
CiA 406	Rotating and Linear Encoders
CiA 408	Hydraulic Drives and Proportional Valves
CiA 410	Inclinometers
CiA 412	Medical Devices
CiA 413	Truck Gateways
CiA 414	Yarn Feeding Units (Weaving Machines)
CiA 415	Road Construction Machinery
CiA 416	Building Door Control
CiA 417	Lift Control Systems
CiA 418	Battery Modules

Figure 52: Some sandard CANopen Device Profiles (from CiA website).

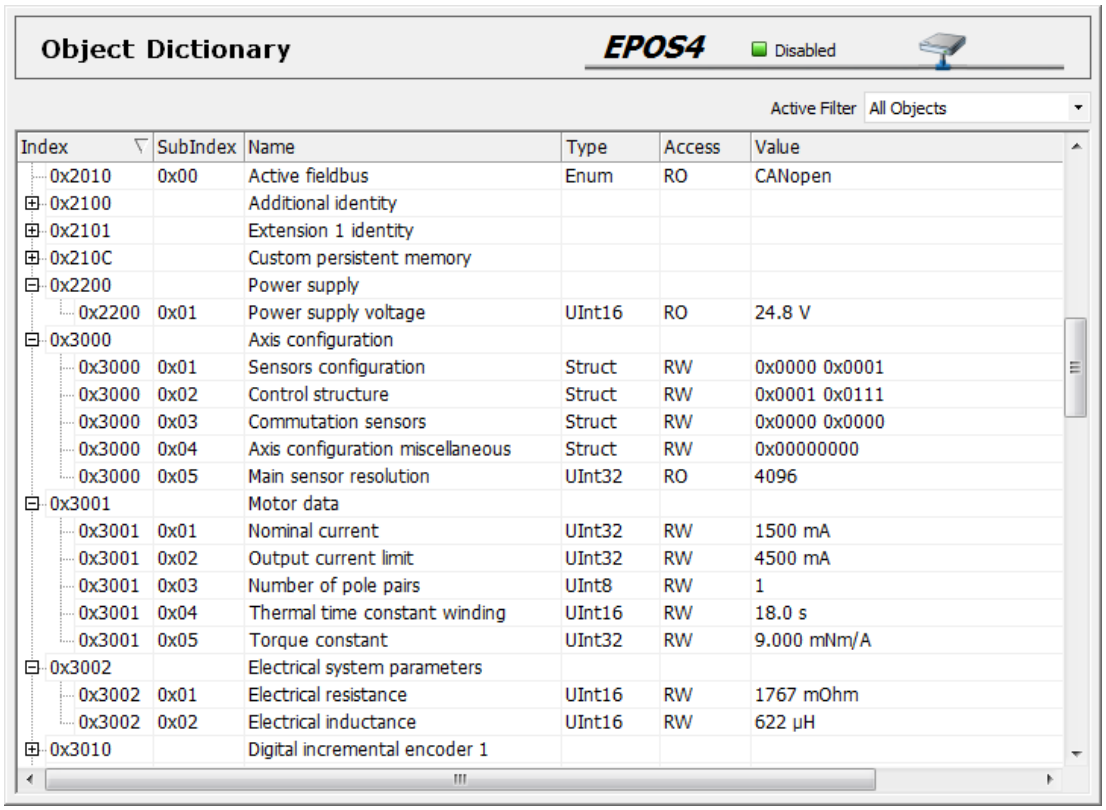
It is important to note that the standard is *open* in the sense that there is a certain degree of freedom which of the objects and functionality must be implemented. For a device to comply with the standard, it must contain certain objects but not all of them. Moreover, the manufacturer is free to add extra objects and functionality as well.

10.4 EPOS4 Object Dictionary tool

Objectives Editing system parameters in the object dictionary.
Generating an object filter.

Let's take, as an example, the object dictionary of the *EPOS4.* The motion control application uses data from the object dictionary (e.g. target position) as inputs to perform a motion and writes the results (e.g. target position reached) to the same object dictionary.

The object dictionary of the *EPOS4* is accessible as a tool in the *EPOS Studio*. Just open the *Object Dictionary* and activate the Object Filter *All Objects* in the upper right corner.



Index	SubIndex	Name	Type	Access	Value
0x2010	0x00	Active fieldbus	Enum	RO	CANopen
0x2100		Additional identity			
0x2101		Extension 1 identity			
0x210C		Custom persistent memory			
0x2200		Power supply			
0x2200	0x01	Power supply voltage	UInt16	RO	24.8 V
0x3000		Axis configuration			
0x3000	0x01	Sensors configuration	Struct	RW	0x0000 0x0001
0x3000	0x02	Control structure	Struct	RW	0x0001 0x0111
0x3000	0x03	Commutation sensors	Struct	RW	0x0000 0x0000
0x3000	0x04	Axis configuration miscellaneous	Struct	RW	0x00000000
0x3000	0x05	Main sensor resolution	UInt32	RO	4096
0x3001		Motor data			
0x3001	0x01	Nominal current	UInt32	RW	1500 mA
0x3001	0x02	Output current limit	UInt32	RW	4500 mA
0x3001	0x03	Number of pole pairs	UInt8	RW	1
0x3001	0x04	Thermal time constant winding	UInt16	RW	18.0 s
0x3001	0x05	Torque constant	UInt32	RW	9.000 mNm/A
0x3002		Electrical system parameters			
0x3002	0x01	Electrical resistance	UInt16	RW	1767 mOhm
0x3002	0x02	Electrical inductance	UInt16	RW	622 µH
0x3010		Digital incremental encoder 1			

Figure 53: Object Dictionary of the motion controller EPOS4.
Shown are some entries regarding the maxon axis configuration.

Scrolling through the list we find the following groups of entries (the index and subindex are given as hexadecimal numbers, hence the prefix 0x):

- Index 0x1000 to 0x1FFF CANopen standard communication profile entries
- Index 0x2000 to 0x5FFF maxon EPOS specific entries (not in the CANopen standard)
- Index 0x6000 to 0x9FFF CANopen standard device profile

The maxon specific entries contain - among others - the objects related to non-CANopen communication as well as objects related to the digital and analog inputs and output

configuration.
In the standard device profile entries, you find all the information related to motion control.

The *Object Dictionary* tool gives you access to the full list of objects. You can change the value of any entry provided it is writable. Simply double click on the object or use the context menu (right mouse click on the list).

Refreshing rate of the *Object Dictionary* in the *EPOS Studio*
USB connection is not extremely fast, and it is not real-time, i.e. it does not have the highest priority on your computer and you cannot predict how long it takes for the object dictionary values to refresh. You can accelerate the response time to some extent by minimizing the *Refresh Rate* in the menu bar.

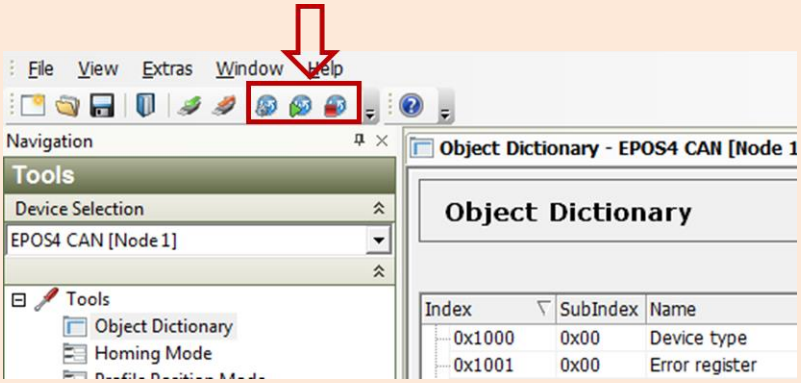


Figure 54: Where to change the Refresh Rate.

Define your own object filter
Looking at the full object dictionary with its many entries can be quite overwhelming. Often you are interested in just a few entries. For this purpose, you can create your own object filter, similar to the predefined *System Parameter* filter.
Here is a recipe for watching the PID gains:

Step 1: Right-click to the dialog window and select *Define Object Filter*.
Step 2: Select *New* to generate a new object filter.
Step 3: Appoint the object filter with *MyRegGains*.
Step 4: *Add* the objects:
- *Current control parameter set*
- *Velocity control parameter set*
- *Position control parameter set*

Step 5: *Save* the object filter as a file (optional) and *Exit*.

Save parameters
Changes in the Object Dictionary tool will only be permanently stored in the EPOS after a *Save All Parameters* command (right mouse click to access the context menu of the Object dictionary).

Basically, there are three sets of *Object Dictionaries*:
- the *actual parameters* in the working memory (RAM) which are lost after a power down.
- the *permanent parameters* saved in the EEPROM which are active after a power up.
- the *default parameters* (i.e. the factory setting) which is always in the EEPROM as a last back-up.

10.5 Parameter Up- and Download

The *Object Dictionary* contains all the parameters describing a CANopen device. It is a fingerprint of the device. The parameter list can be saved in a special electronic data sheets called *Device Configuration File* (parameter export). Thus, it is possible to configure different devices in the same way simply by downloading the corresponding electronic data sheet on the *EPOS4* (parameter import).

The *Parameter Export/Import* wizard in the *EPOS Studio* serves for this purpose. It also allows to *Restore Default Parameters*.

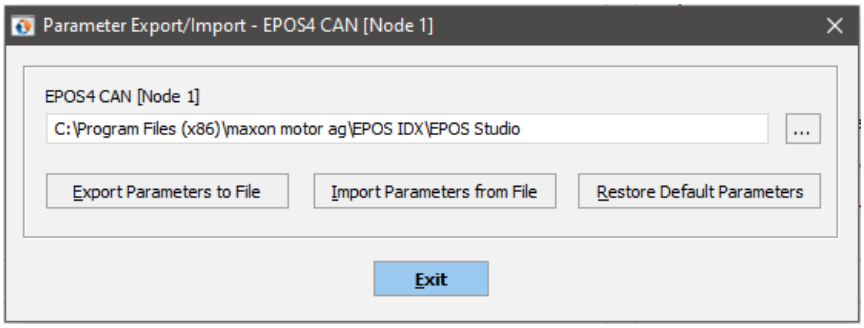


Figure 55: Parameter Export/Import wizard communication window.

10.6 CAN communication

Communication in the basic CAN layers is based on broadcast communication: Each node can send messages that can be picked up by any other node. For avoiding data collision, each telegram has a defined priority. If two messages start at the same time the one with lower priority value wins and is transmitted first.

In CANopen the communication is more specific in the sense that there are predefined communication channels (each with a certain priority level) describing which node is the sender and which node is the receiver of the message (peer-to-peer communication).

Please refer also to the *EPOS 4 Communication Guide* document, chapter 3

Network Management and special messages

The lowest priority values are assigned to messages that ascertain that the network operates correctly, and that the communication is reliable. In this category, we find error control and emergency messages, synchronization and time stamp messages as well as network management messages such as booting and changing the operational state of a node. The exact priority sequence within these categories is not of importance for the purpose of this textbook.

Process Data Objects (PDO)

The next priority is given to *Process Data Objects* (PDO). PDOs are for real-time data exchange of small objects. Typically, these are data that change a lot in the process (sic the name!) and need to be updated frequently as in real time applications.

In order to speed up transmission, the telegrams are kept small by avoiding unnecessary overhead information. To reduce the busload, the correct transmission of a PDO telegram is not confirmed.

The content of each PDO message is defined in advance, as well as the location from where it is sent (which object of which node) and the target location. This configuration is called *mapping*, which also specifies how often a PDO message is sent: periodically or triggered by some event. The PDO mapping can be found in the object dictionary of the devices involved.

The CAN PDO communication is typically used to update the process inputs of the PLC and there can be several PDO communication channels between two devices. Be aware that the cycle time of the PLC task and the update rate of data sent by PDO are two different things. In certain applications, they might need synchronization.

Remark: All the commands sent by the *EPOS Studio* are essentially based on SDO communication.

Service Data Objects (SDO)

The lowest priority – the highest priority values – is given to *Service Data Objects* (SDO). SDO permit the transmission of any data size. Large messages (> 4 bytes) are automatically segmented, i.e. split into several telegrams that are sent consequently.

As in PDO communication, SDO transmission is defined between two specified nodes. There are two communication channels needed, one for each direction. The telegrams contain information which entry in the object dictionary is to be written or read. Half of the useful data content is already consumed by this overhead information.

SDO is a relatively slow communication and it is primarily intended for setting up the device. Since the motion of an individual axis takes typically several hundred milliseconds, the speed of SDO communication is sufficient for many motion control applications. The situation may be different when several axes need to be closely coordinated or synchronized.

PDO and SDO in EPOS systems
PDO channels in EPOS systems are only defined between the slave and the master. There are no predefined communication channels between slaves.

- There is one SDO channel in each direction between the *EPOS4* and its master.
- There are 4 PDO channels to transmit and 4 PDO channels receive process data on the *EPOS4* as can be seen in the *Object Dictionary.*

11 Remarks on EtherCAT

(Information based on https://en.wikipedia.org/wiki/EtherCAT)

On a physical layer, EtherCAT uses standard Ethernet technology.

The main difference to CANopen communication is the much higher performance of EtherCAT, which is up to 100 Mbit/s. That is 100 times faster than the 1Mbit/s maximum bitrate of CANopen. For instance, data for and from 100 servo axes can be updated with up to 10 kHz. Typical network update rates are 1–30 kHz.

The EtherCAT master can access all data including name and data types of an EtherCAT slave (e.g. EPOS4). Typically, the EtherCAT master addresses the entire network with just one frame. The EtherCAT slave devices read the data addressed to them while the telegram passes through the device, processing data "on the fly". Similarly, input data are inserted while the telegram passes through.

The EtherCAT option of the EPOS4 just replaces the CAN communication by EtherCAT. The core of the EPOS4 is still a CANopen device with the same *Object Dictionary*. The solution is called CAN application protocol over EtherCAT (CoE). On a hardware level, the EtherCAT versions of the EPOS4 are equipped with a special EtherCAT connector board or with an EtherCAT extension card.

Figure 56: EPOS4 50/8 with CANopen (left) and EtherCAT (right) connector board.

The fast real-time communication CoE brings advantages for challenging applications such as high-speed synchronizing of multiple axes, robotics, pick and place. These applications often require multiple axis path planning and control in the master and use the cyclic synchronous operation modes (CSP, CSV, CST: refer to chapters 8.3 ff).

Part 6: Appendices, References and Index

12 Appendices

12.1 Motor and encoder data sheets

The maxon motor-encoder combination with part number B7723C9C8D10 consists of a DCX 22 S motor with ENX encoder.

Motor data sheet

Values at nominal voltage		
	Nominal voltage	12 V
	No load speed	12400 rpm
	No load current	71.7 mA
	Nominal speed	10700 rpm
	Nominal torque (max. continuous torque)	14.6 mNm
	Nominal current (max. continuous current)	1.65 A
	Stall torque	108 mNm
	Stall current	11.8 A
	Max. efficiency	85 %
Characteristics		
	Terminal resistance	1.02 Ω
	Terminal inductance	0.058 mH
	Torque constant	9.18 mNm/A
	Speed constant	1040 rpm/V
	Speed / torque gradient	116 rpm/mNm
	Mechanical time constant	6.12 ms
	Rotor inertia	5.05 gcm²
Thermal data		
	Thermal resistance housing-ambient	16 K/W
	Thermal resistance winding-housing	7 K/W
	Thermal time constant winding	18.1 s
	Thermal time constant motor	528 s
	Ambient temperature	-40...+100 °C
	Max. winding temperature	+125 °C
Mechanical data		
	Bearing type	ball bearings
	Max. speed	18000 rpm
	Axial play	0 - 0.1 mm
	Radial play	0.02 mm
	Max. axial load (dynamic)	2.5 N
	Max. force for press fits (static)	30 N
	(static, shaft supported)	440 N
	Max. radial load	16 N, 5 mm from flange
Other specifications		
	Number of pole pairs	1
	Number of commutator segments	9
	Number of autoclave cycles	0

Figure 57: Extract from motor data sheet DCX 22 S (GB,KL,12V).

Encoder data sheet

General information

Counts per turn	1024
Number of channels	3
Line Driver	RS422
Max. length of encoder housing	8.5 mm
Max. electrical speed	90000 rpm
Max. mechanical speed	30000 rpm

Technical Data

Supply voltage Vcc	5.0V ± 10.0%
Output signal	Incremental
Driver used logic	Differential, EIA RS 422
Output current per channel	-20...20 mA
Signal rise time	20 ns
Measurement condition for signal rise time	CL=25pF, RL=1kOhm
Signal fall time	20 ns
Measurement condition for signal fall time	CL=25pF, RL=1kOhm
Min. state duration	125 ns
Direction of rotation	A before B CW
Index position	A low & B low
Index synchronized to AB	Yes
Typical current draw at standstill	22 mA
Max. moment of inertia of code wheel	0.05 gcm²
Operating temperature	-40...+100 °C
Orientation of encoder output to motor flange	360 °

Figure 58: Extracts from ENX 16 EASY encoder data sheet.

13 References, Glossary

13.1 List of Figures

13.2 List of Boxes

ESCON Info

EPOS Info

Motion Control Background

Best Practice

13.3 Literature

- Feinmess www.feinmess.de/mt_all/glossar.htm
- John K.-H. John, M. Tiegelkamp “SPS-Programmierung mit IEC 61131-3”, 3rd Edition, Springer Verlag 2000. ISBN 3-540-66445-9
- Wikipedia www.wikipedia.org
- CAN in Automation www.can-cia.org

13.4 Index

L

M

O

P

Q

R

S

T

U

V

W


maxon
academy