maxon

EPOS4

Application Notes

epos.maxongroup.com

EPOS4 Positioning Controllers
Application Notes
CCMC | Edition 2021-03 | DocID rel9611

maxon

TABLE OF CONTENTS

READ THIS FIRST

***These instructions are intended for qualified technical personnel. Prior commencing with any activities**…*

- *you must carefully read and understand this manual and*
- *you must follow the instructions given therein.*

***EPOS4** positioning controllers are considered as partly completed machinery according to EU Directive 2006/42/EC, Article 2, Clause (g) and **are intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment**.*

***Therefore, you must not put the device into service**,…*

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

1 ABOUT

1.1 About this Document

1.1.1 Intended Purpose

The purpose of the present document is to provide you specific information to cover particular cases or scenarios that might come in handy during commissioning of your drive system.
Use for other and/or additional purposes is not permitted. maxon, the manufacturer of the equipment described, does not assume any liability for loss or damage that may arise from any other and/or additional use than the intended purpose.

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

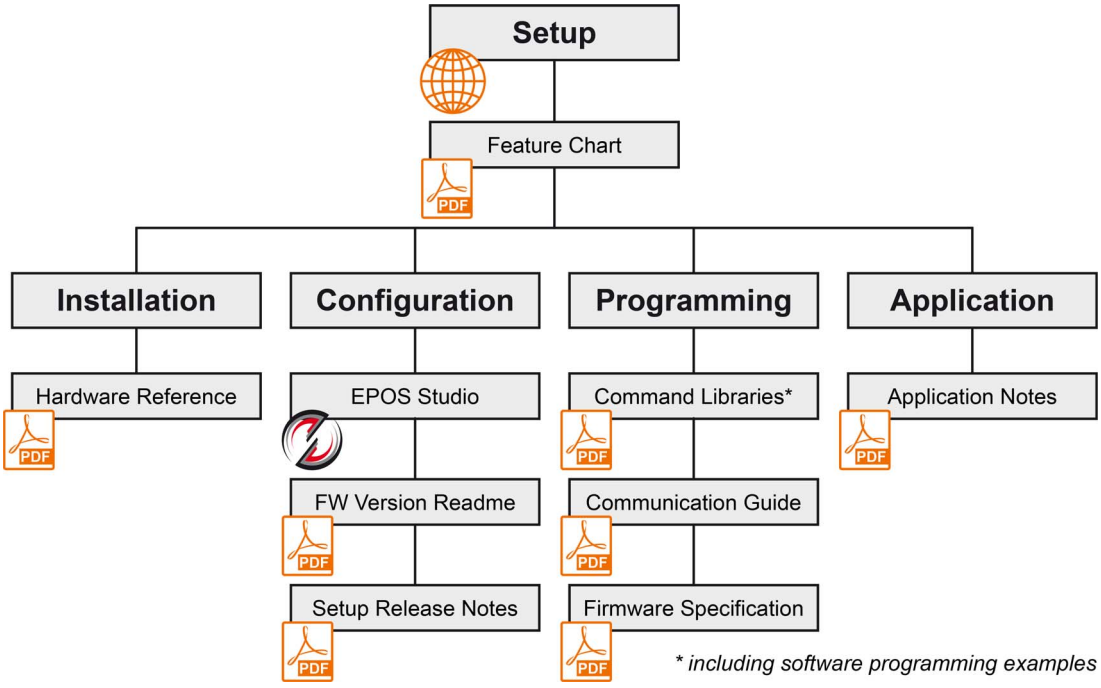


Figure 1-1 Documentation structure

Find the latest edition of the present document as well as additional documentation and software for EPOS4 positioning controllers also on the Internet: ➔http://epos.maxongroup.com

1.1.2 Target Audience

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

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

1.1.3 How to use

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

Notation	Explanation
Compact	referring to any of the EPOS4 Compact versions
Compact CAN	referring to a fully integrated, compact, ready-to-use EPOS4 assembly of plug-in module and CANopen connector board (such as «EPOS4 Compact 50/8 CAN» or «EPOS4 Compact 50/15 CAN»)
Compact EtherCAT	referring to a fully integrated, compact, ready-to-use EPOS4 assembly of plug-in module and EtherCAT connector board (such as «EPOS4 Compact 50/8 EtherCAT» or «EPOS4 Compact 50/15 EtherCAT»)
Disk	referring to any of the EPOS4 Disk versions
Disk CAN	referring to any of the EPOS4 Disk CAN versions
Disk EtherCAT	referring to any of the EPOS4 Disk EtherCAT versions
EPOS4	stands for “EPOS4 Positioning Controller”
Micro	referring to an EPOS4 plug-in module version (such as “EPOS4 Micro 24/5) for use with EPOS4 connector boards or customer-specific motherboards
Module	referring to an EPOS4 plug-in module version (such as “EPOS4 Module 50/8” or “EPOS4 Module 50/15”) for use with EPOS4 connector boards or customer-specific motherboards
«Abcd»	indicating a title or a name (such as of document, product, mode, etc.)
¤Abcd¤	indicating an action to be performed using a software control element (such as folder, menu, drop-down menu, button, check box, etc.) or a hardware element (such as switch, DIP switch, etc.)
(n)	referring to an item (such as order number, list item, etc.)
*****	referring to an internal value
*******	referring to a not yet implemented item
➔	denotes “see”, “see also”, “take note of”, or “go to”

Table 1-1 Notations used

In the later course of the present document, the following abbreviations and acronyms will be used:

Short	Description
STO	Safe Torque Off

Table 1-2 Abbreviations and acronyms used

1.1.4 Symbols and Signs

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

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

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

1.1.5 Trademarks and Brand Names

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

Brand name	Trademark owner
Adobe® Reader®	© Adobe Systems Incorporated, USA-San Jose, CA
APOSS®	© zub machine control AG, CH-Rothenburg
CANopen® CiA®	© CiA CAN in Automation e.V, DE-Nuremberg
EtherCAT®	© EtherCAT Technology Group, DE-Nuremberg, licensed by Beckhoff Automation GmbH, DE-Verl
Sysmac	© OMRON Corporation, JP-Kyoto
TwinCAT®	© Beckhoff Automation GmbH, DE-Verl
Windows®	© Microsoft Corporation, USA-Redmond, WA

Table 1-3 Brand names and trademark owners

1.1.6 Sources for additional Information

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

#	Reference
[1]	IEC/EN 60204-1: Safety of machinery – Electrical equipment of machines
[2]	IEC/EN 61800-5-2: Adjustable speed electrical power drive systems
[3]	CiA 301 CANopen application layer and communication profile www.can-cia.org
[4]	CiA 305 Layer Setting Services (LSS) and protocols www.can-cia.org
[5]	CiA 402 CANopen device profile for drives and motion control www.can-cia.org

Table 1-4 Sources for additional information

1.1.7 Copyright

CCMC | EPOS4 Application Notes | Edition 2021-03 | DocID rel9611

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

1.2 About the Devices

maxon's EPOS4 positioning controllers are small-sized, full digital, smart positioning control units. Their high power density allow flexible use for brushed DC and brushless EC (BLDC) motors with various feedback options, such as Hall sensors, incremental encoders as well as absolute sensors in a multitude of drive applications.

1.3 About the Safety Precautions

IMPORTANT NOTICE: PREREQUISITES FOR PERMISSION TO COMMENCE INSTALLATION

EPOS4 positioning controllers are considered as partly completed machinery according to EU Directive 2006/42/EC, Article 2, Clause (g) and **are intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment**.

WARNING

Risk of Injury
Operating the device without the full compliance of the surrounding system with the EU directive 2006/42/EC may cause serious injuries!
- *Do not operate the device, unless you have made sure that the other machinery fulfills the requirements stated in EU directive!*
- *Do not operate the device, unless the surrounding system fulfills all relevant health and safety aspects!*
- *Do not operate the device, unless all respective interfaces have been established and fulfill the stated requirements!*

SAFETY FIRST!

Keep in mind: Safety first! Always!

- Make sure that you have read and understood the note “READ THIS FIRST” on page A-2!
- Do not engage with any work unless you possess the stated skills (➔chapter “1.1.2 Target Audience” on page 1-5)!
- Refer to ➔chapter “1.1.4 Symbols and Signs” on page 1-7 to understand the subsequently used indicators!
- You must observe any regulation applicable in the country and/or at the site of implementation with regard to health and safety/accident prevention and/or environmental protection!

DANGER

High voltage and/or electrical shock
Touching live wires causes death or serious injuries!
- *Consider any power cable as connected to live power, unless having proven the opposite!*
- *Make sure that neither end of cable is connected to live power!*
- *Make sure that power source cannot be engaged while work is in process!*
- *Obey lock-out/tag-out procedures!*
- *Make sure to securely lock any power engaging equipment against unintentional engagement and tag it with your name!*

Requirements
- *Make sure that all associated devices and components are installed according to local regulations.*
- *Be aware that, by principle, an electronic apparatus cannot be considered fail-safe. Therefore, you must make sure that any machine/apparatus has been fitted with independent monitoring and safety equipment. If the machine/apparatus should break down, if it is operated incorrectly, if the control unit breaks down or if the cables break or get disconnected, etc., the complete drive system must return – and be kept – in a safe operating mode.*
- *Be aware that you are not entitled to perform any repair on components supplied by maxon.*

Continued on next page.

Electrostatic sensitive device (ESD)

- *Wear working cloth and use equipment in compliance with ESD protective measures.*
- *Handle devices with extra care.*

2 CONTROLLER ARCHITECTURE

CONTENT

2.1 In Brief

A wide variety of operating modes permit flexible configuration of drive and automation systems by using positioning, speed and current regulation. The built-in interface allows online commanding by CAN or EtherCAT bus master units as well as networking to multiple axes drives.

Good quality velocity PI control is made possible by the use of algorithms for estimating the motor rotation velocity from the measured rotor position that are based either on a low pass filter or on a velocity observer.

OBJECTIVE

The present application note explains the EPOS4 controller architecture.

In addition to PID position regulation, the functionalities of the built-in acceleration and velocity feedforward are described.

The functionality of the velocity PI controller, the low pass filter, and the observer used for estimating the velocity are described. The benefits of each velocity estimation method are highlighted and illustrated by using practical examples.

maxon

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0100h	Firmware Specification
EPOS4 Disk 60/8 CAN EPOS4 Disk 60/8 EtherCAT	688770 688772	0170h or higher 0170h or higher	
EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC EPOS4 Disk 60/12 EtherCAT EPOS4 Disk 60/12 EtherCAT SSC	688775 709859 688777 709862	0170h or higher 0170h or higher 0170h or higher 0170h or higher	
EPOS4 Module 24/1.5 EPOS4 Compact 24/1.5 CAN EPOS4 Compact 24/1.5 EtherCAT	536630 546714 628092	0110h or higher 0110h or higher 0150h or higher	
EPOS4 Module 50/5 EPOS4 Compact 50/5 CAN EPOS4 Compact 50/5 EtherCAT	534130 541718 628094	0110h or higher 0110h or higher 0150h or higher	
EPOS4 Module 50/8 EPOS4 Compact 50/8 CAN EPOS4 Compact 50/8 EtherCAT	504384 520885 605298	0100h or higher 0100h or higher 0140h or higher	
EPOS4 Module 50/15 EPOS4 Compact 50/15 CAN EPOS4 Compact 50/15 EtherCAT	504383 520886 605299	0100h or higher 0100h or higher 0140h or higher	
EPOS4 50/5	546047	0120h or higher	
EPOS4 70/15	594385	0140h or higher	

Table 2-5 Controller architecture | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.0 or higher

Table 2-6 Controller architecture | Recommended tools

2.2 Overview

The EPOS4 controller architecture contains three built-in control loops.

- Current regulation is used in all modes.
- Position or velocity regulation is only used in position-based or velocity-based modes, respectively.

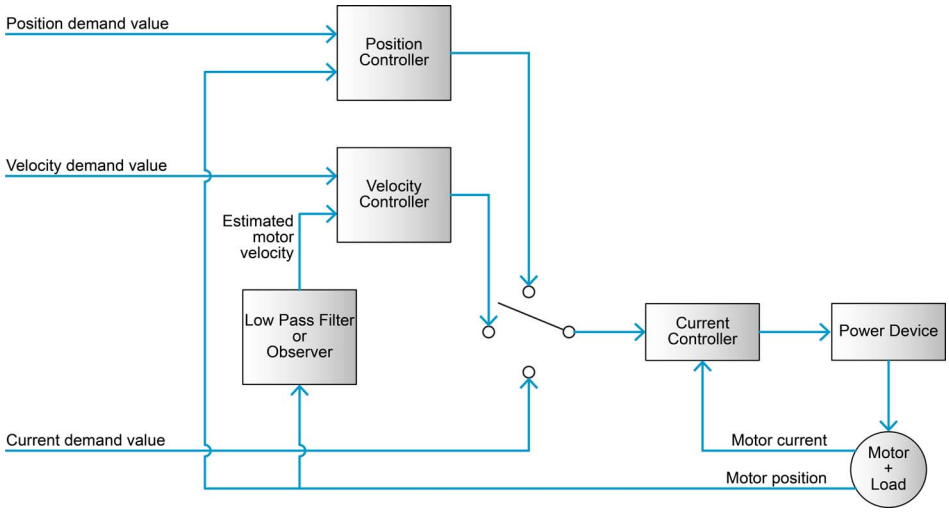


Figure 2-2 Controller architecture | Overview

2.3 Regulation Methods

2.3.1 Current Regulation

During a movement within a drive system, forces and/or torques must be controlled. Therefore, as a principal regulation structure, EPOS4 offers current-based control.

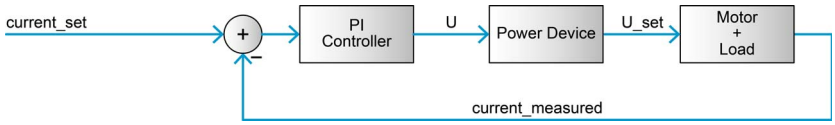


Figure 2-3 Controller architecture | Current regulator

CONSTANTS

Sampling period: $T_s = 0.04ms$

OBJECT DICTIONARY ENTRIES

Symbol	Unit	Name	Index	Subindex
K_{P_EPOS4}	$\frac{mV}{A}$	Current controller P gain	0x30A0	0x01
K_{I_EPOS4}	$\frac{mV}{A \cdot ms}$	Current controller I gain	0x30A0	0x02

Table 2-7 Controller architecture | Current regulation – Object dictionary

CONVERSION OF PI CONTROLLER PARAMETERS (EPOS4 TO SI UNITS)

$$K_{P\_SI} = 0.001 \cdot K_{P\_EPOS4}$$

$$K_{I\_SI} = K_{I\_EPOS4}$$

Current controller parameters in SI units can be used in analytical or numerical simulations via the following transfer function:

$$C_{current}(s) = K_{P\_SI} + \frac{K_{I\_SI}}{s}$$

ANTI-WINDUP

In order to prevent degradation of the control performance when the control input stays at the limit value for long time, an anti-windup algorithm is implemented preventing the integral part of the PI controller to take values larger than the ones bound on the control input.

TRANSPORT DELAY OF THE CONTROL LOOP

Total transport delay of the current regulation loop is always smaller than 0.06 ms.

2.3.2 Velocity Regulation (with Feedforward)

EPOS4 offers velocity regulation based on the subordinated current control.

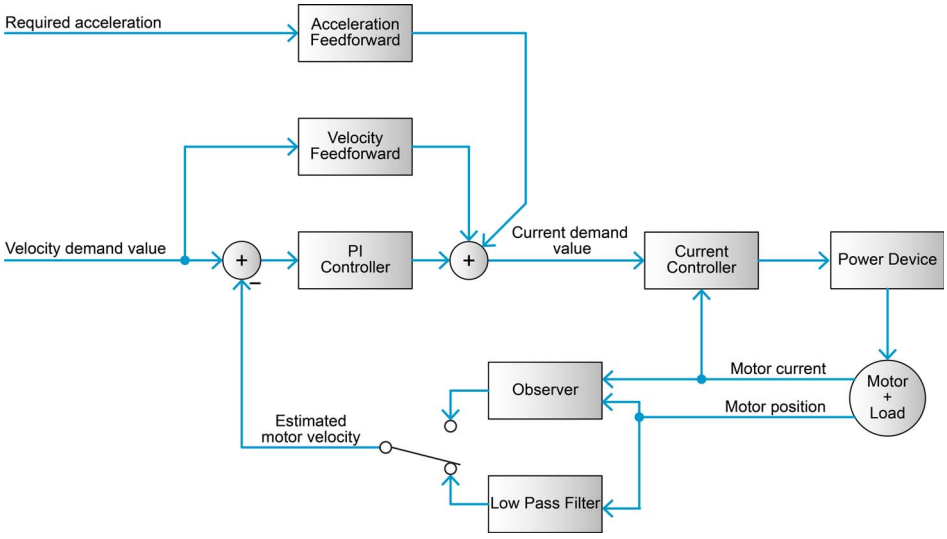


Figure 2-4 Controller architecture | Velocity regulator with feedforward

CONSTANTS

Sampling period: $T_s = 0.4ms$

OBJECT DICTIONARY ENTRIES FOR CONTROLLER

Symbol	Unit	Name	Index	Subindex
$K_{P\omega\_EPOS4}$	$\frac{mA \cdot s}{rad}$	Velocity controller P gain	0x30A2	0x01
$K_{I\omega\_EPOS4}$	$\frac{mA}{rad}$	Velocity controller I gain	0x30A2	0x02
FF_{ω_EPOS4}	$\frac{mA \cdot s}{rad}$	Velocity controller FF velocity gain	0x30A2	0x03
FF_{α_EPOS4}	$\frac{mA \cdot s^2}{rad}$	Velocity controller FF acceleration gain	0x30A2	0x04
f	$\frac{1}{s}$	Velocity controller filter cut-off frequency	0x30A2	0x05

Table 2-8 Controller architecture | Velocity regulation – Object dictionary

CONVERSION OF PI CONTROLLER PARAMETERS (EPOS4 TO SI UNITS)

$$K_{P\omega\_SI} = 0.001 \cdot K_{P\omega\_EPOS4}$$

$$K_{I\omega\_SI} = 0.001 \cdot K_{I\omega\_EPOS4}$$

$$FF_{\omega\_SI} = 0.001 \cdot FF_{\omega\_EPOS4}$$

$$FF_{\alpha\_SI} = 0.001 \cdot FF_{\alpha\_EPOS4}$$

Velocity controller parameters in SI units can be used in analytical or numerical simulations via transfer function for the PI controller:

$$C_{velocity}(s) = K_{P\omega\_SI} + \frac{K_{I\omega\_SI}}{s}$$

ANTI-WINDUP

An anti-windup algorithm is implemented to prevent integration wind-up in PI controller, when the actuators are saturated.

LOW PASS FILTER

The estimation of the motor velocity can be done by using the measured time between consecutive sensor edges, which is low pass filtered in order to eliminate the effects of measurement noise. The transfer function of the low pass filtered estimation functionality that can be used in simulations has the following form:

$$C_{FilterEstimator}(s) = \frac{2 \cdot \pi \cdot f}{s + 2 \cdot \pi \cdot f}$$

OBSERVER

An alternative to the low pass filter is the use of an observer. Thereby, the observed velocity is calculated in two steps. First; prediction of the velocity, position, and external torque, based on the parameters that define the mechanical transfer function of the system. Second; correction of the predicted values based on the newly measured rotor position.

maxon

OBJECT DICTIONARY ENTRIES FOR OBSERVER

Symbol	Unit	Name	Index	Subindex
k_{m_EPOS4}	$\frac{mNm}{A}$	Torque constant	0x3001	0x05
l_{θ_EPOS4}	1	Velocity observer position correction gain	0x30A3	0x01
l_{ω_EPOS4}	Hz	Velocity observer velocity correction gain	0x30A3	0x02
l_{T_EPOS4}	$\frac{mNm}{rad}$	Velocity observer load correction gain	0x30A3	0x03
r_{EPOS4}	$\frac{\mu Nm}{rpm}$	Velocity observer friction	0x30A3	0x04
J_{EPOS4}	$g \cdot cm^2$	Velocity observer inertia	0x30A3	0x05

Table 2-9 Controller architecture | Velocity observer – Object dictionary

All parameters relevant for the observer operation can be entered either manually or can be obtained from the EPOS4 auto tuning procedure. The auto tuning automatically executes the identification experiments, identifies the relevant parameters that characterize the drive train, and calculates the values of the observer correction gains.

CONVERSION OF OBSERVER PARAMETERS (EPOS4 TO SI UNITS)

$$k_{m\_SI} = 0.001 \cdot k_{m\_EPOS4}$$

$$J_{SI} = 0.0000001 \cdot J_{EPOS4}$$

$$r_{SI} = \frac{0.00003}{\pi} \cdot r_{EPOS4}$$

$$l_{\theta\_SI} = l_{\theta\_EPOS4}$$

$$l_{\omega\_SI} = l_{\omega\_EPOS4}$$

$$l_{T\_SI} = 0.001 \cdot l_{T\_EPOS4}$$

The transfer functions characterizing the two steps in the observer calculations and that can be used in numerical simulation of the velocity controller with observer are the following:

PREDICTION STEP

$$\theta_{Observed} = \frac{\omega_{Observed}}{s}$$

$$\omega_{Observed} = \frac{k_{M\_SI} \cdot i_{Measured} - T_{Observed}}{J_{SI} \cdot s + r_{SI}}$$

CORRECTION STEP

$$\theta_{Observed} = \theta_{Observed} + l_{\theta\_SI} \cdot (\theta_{Measured} - \theta_{Observed})$$

$$\omega_{Observed} = \omega_{Observed} + l_{\omega\_SI} \cdot (\theta_{Measured} - \theta_{Observed})$$

$$T_{Observed} = T_{Observed} + l_{T\_SI} \cdot (\theta_{Measured} - \theta_{Observed})$$

WHEN SHOULD THE LOW PASS FILTER BE USED TO ESTIMATE THE VELOCITY?

The estimation of the motor velocity based on measuring the time between consecutive sensor edges and low pass filtering does not rely on any additional information on the mechanical system to which the motor is attached. Therefore, it is suitable in cases when there is no information on the mechanical properties of the system available or when the characteristics of the system change significantly over time.

Typical examples are cases in which the moment of inertia or viscous friction that the motor encounters change significantly during operation.

The solution with the filter gives good results in cases when a high-resolution position sensor is used and when the motor is operated at relatively high velocities (more than 20% of nominal motor speed). However, in cases when the resolution of the position sensor is low and/or the motor operates at low speed, the estimation with the observer results in a better control performance.

WHEN SHOULD THE OBSERVER BE USED TO ESTIMATE THE VELOCITY?

In order to use the observer for estimating the rotational velocity of the motor, parameters, such as inertia and viscous friction coefficient of the drive system, need to be known and must be stable over time and should not change a lot during operation. In EPOS4, there is an option to identify all the required parameters by using the «Auto Tuning Wizard».

The use of the observer brings most advantages when the position feedback sensor has a low resolution. Typical example is the use of Hall sensors for feedback instead of an incremental encoder, or the use of incremental encoders with up to 500 counts per turn. In general, the use of the observer provides a less noisy estimation of the rotor velocity resulting in better regulation and less audible noise especially at low operational velocities.

In addition, the velocity observer can be set stiffer (compared to the case when the filter is used) due to better quality of the estimated feedback signal resulting in a very good dynamical response.

However, when encoders with high resolution (above 500 counts per turn) are used, the performance of the system with observer is similar to its performance in the case when the low pass filter is used.

TRANSPORT DELAY OF THE CONTROL LOOP

Total transport delay of the velocity regulation loop is always smaller than 0.4 ms.

2.3.3 Position Regulation (with Feedforward)

EPOS4 is able to close a positioning control loop based on the subordinated current control.

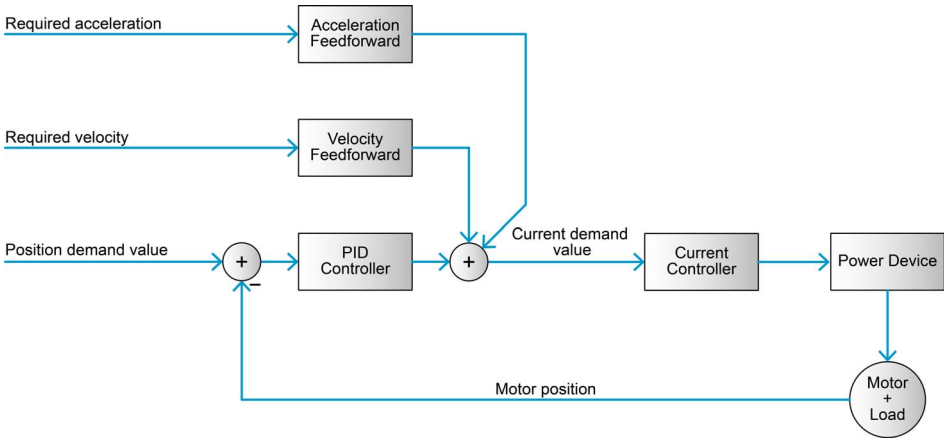


Figure 2-5 Controller architecture | Position regulator with feedforward

CONSTANTS

Sampling period: $T_s = 0.4ms$

OBJECT DICTIONARY ENTRIES

Symbol	Unit	Name	Index	Subindex
K_{PP_EPOS4}	$\frac{mA}{rad}$	Position controller P gain	0x30A1	0x01
K_{IP_EPOS4}	$\frac{mA}{rad \cdot s}$	Position controller I gain	0x30A1	0x02
K_{DP_EPOS4}	$\frac{mA \cdot s}{rad}$	Position controller D gain	0x30A1	0x03
FF_{ω_EPOS4}	$\frac{mA \cdot s}{rad}$	Position controller FF velocity gain	0x30A1	0x04
FF_{α_EPOS4}	$\frac{mA \cdot s^2}{rad}$	Position controller FF acceleration gain	0x30A1	0x05

Table 2-10 Controller architecture | Position regulation – Object dictionary

The position controller is implemented as PID controller. To improve the motion system's setpoint fol- lowing, positioning regulation is supplemented by feedforward control. Thereby, velocity feedforward serves for compensation of speed-proportional friction, whereas acceleration feedforward considers known inertia. In addition, the differential part of the PID Controller signal is low pass filtered before it is added to the proportional and integral part. Low pass filtering is done to prevent negative influence on the control performance by the differentiation of noisy measured motor position.

CONVERSION OF PI CONTROLLER PARAMETERS (EPOS4 TO SI UNITS)

$$K_{PP\_SI} = 0.001 \cdot K_{P\_EPOS4}$$

$$K_{IP\_SI} = 0.001 \cdot K_{I\_EPOS4}$$

$$K_{DP\_SI} = 0.001 \cdot K_{D\_EPOS4}$$

$$FF_{\omega\_SI} = 0.001 \cdot FF_{\omega\_EPOS4}$$

$$FF_{\alpha\_SI} = 0.001 \cdot FF_{\alpha\_EPOS4}$$

Position controller parameters in SI units can be used in analytical or numerical simulations via transfer function:

$$C_{position}(s) = K_{PP\_SI} + \frac{K_{IP\_SI}}{s} + \frac{K_{DP\_SI} \cdot s}{1 + \frac{K_{DP\_SI}}{10 \cdot K_{PP\_SI}} \cdot s}$$

ANTI-WINDUP

The anti-windup method is used to prevent integration wind-up in PID controller when the actuators are saturated.

2.3.4 Operation Modes with Feedforward

Acceleration and velocity feedforward are effective in «Profile Position Mode» (PPM), «Profile Velocity Mode» (PVM), and «Homing Mode» (HMM). All other operating modes are not affected.

PURPOSE OF VELOCITY FEEDFORWARD

Velocity feedforward provides additional current in cases, where the load increases with speed, such as speed-dependent friction. The load is assumed to proportionally increase with speed. The optimal velocity feedforward parameter in SI units is:

$$FF_{\omega\_SI} = \frac{r_{SI}}{k_{m\_SI}}$$

Meaning: With given total friction proportional factor in SI units r_{SI} relative to the motor shaft, and the motor's torque constant also in SI units k_{m_SI}, you ought to adjust the velocity feedforward parameter to:

$$FF_{\omega\_EPOS4} = 1000 \cdot FF_{\omega\_SI} = 1000 \cdot \frac{r_{SI}}{k_{m\_SI}}$$

PURPOSE OF ACCELERATION FEEDFORWARD

Acceleration feedforward provides additional current in cases of high acceleration and/or high load inertias. The optimal acceleration feedforward parameter in SI units is:

$$FF_{a\_SI} = \frac{J_{SI}}{k_{m\_SI}}$$

Meaning: With given total inertia in SI units J_{SI} relative to the motor shaft, and the motor's torque constant in SI units k_{m_SI}, you ought to adjust the acceleration feedforward parameter to:

$$FF_{\alpha\_EPOS4} = 1000 \cdot FF_{\alpha\_SI} = 1000 \cdot \frac{J_{SI}}{k_{m\_SI}}$$

TRANSPORT DELAY OF THE CONTROL LOOP

Total transport delay of the position regulation loop is always smaller than 0.4 ms.

2.4 Regulation Tuning

maxon's «EPOS Studio» features regulation tuning as a powerful wizard allowing to automatically tune all controller, estimator, and feedforward parameters described above for most drive systems within a few minutes.

2.5 Application Examples

Find below “in-practice examples” suitable for daily use.

2.5.1 Example 1: System with High Inertia and Low Friction

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/8 CAN (520885)		
Motor maxon DCX 32 L 110 W 36 V	No load speed (line 2)	n_0 = 7940 rpm
	No load current (line 3)	I_0 = 103 mA
	Nominal current (line 6)	I_n = 2.93 A
	Terminal resistance (line 10)	R = 0.764 Ω
	Terminal inductance (line 11)	L = 0.254 mH
	Torque constant (line 12)	k_m = 42.9 mNm/A
	Rotor inertia (line 16)	J_{motor} = 76.8 g*cm²
Encoder HEDL 5540	Encoder counts per turn	500 pulses/revolution
Mechanical load Disc	Inertia	J_{load} = 1800 g*cm²

Table 2-11 Controller architecture | Example 1: Components

2.5.1.1 Current Regulation – Simulation Part

SIMPLE MODEL OF THE DRIVE PLANT

The following parameters can be deduced:

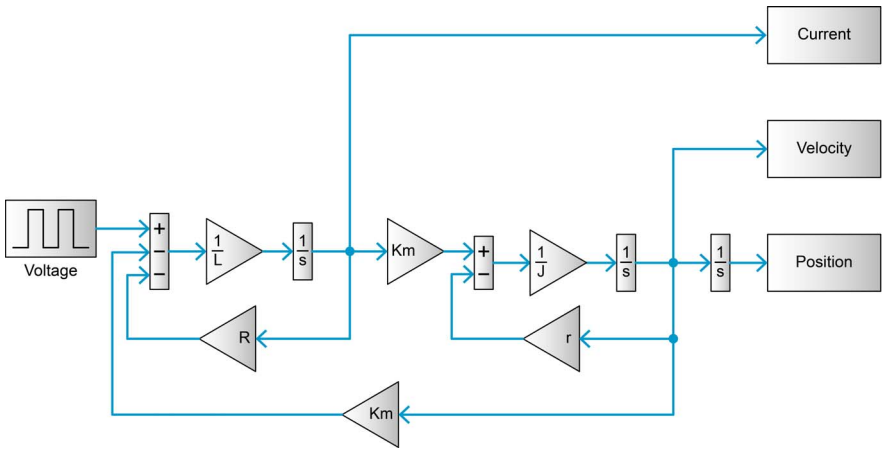


Figure 2-6 Controller architecture | Example 1: Model of the plant

INPUT/OUTPUT PARAMETERS

Input is the voltage at the motor winding.

Outputs are current, velocity, or position.

MODEL PARAMETERS

Resistance	$R = 0.764[\Omega]$
Inductance	$L = 0.254[mH]$
Torque constant	$k_m = 42.9\left[\frac{mNm}{A}\right]$
Mass inertia	$J = J_{motor} + J_{load} = 1876.8[g \cdot cm^2]$
Viscous friction (approximated from the friction at no-load divided by the no-load speed of the motor)	$r = \frac{k_m I_o}{n_o \frac{2\pi rad}{1}\frac{1min}{60s}} = \frac{4.41mNm}{831.45rad/s} = 5.3\left[\frac{\mu Nm}{rad/s}\right]$

Note

- *All model parameters, expect the load inertia (J_{load}), can be taken from the motor data sheet in the maxon catalog.*
- *All parameters (R, L, k_m, I_0, n_0) taken from the motor data sheet are nominal variables, they have tolerances (for more details ➔additional document «Standard Specification No.100»).*

CURRENT CONTROL

The figure below depicts the model of the PI current controller.

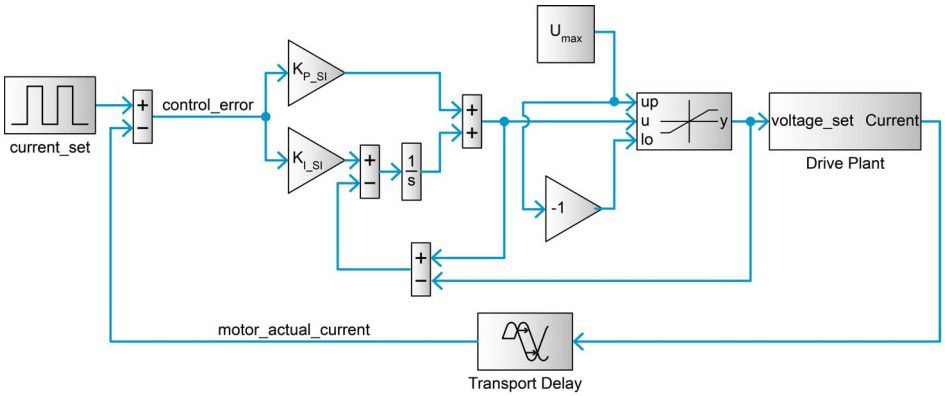


Figure 2-7 Controller architecture | Example 1: Current regulation

MODEL PARAMETERS OF CURRENT CONTROL

- EPOS4 PI current controller gains converted in SI Units.
- Transport delay = 0.060 ms.
- U_{max} corresponds to the nominal voltage of the motor (for details ➔maxon catalog, motor data, line 1).

2.5.1.2 Velocity Regulation with Feedforward - Simulation Part

The figure below displays the model of the PI velocity controller. The PI velocity controller is connected to the current regulation.

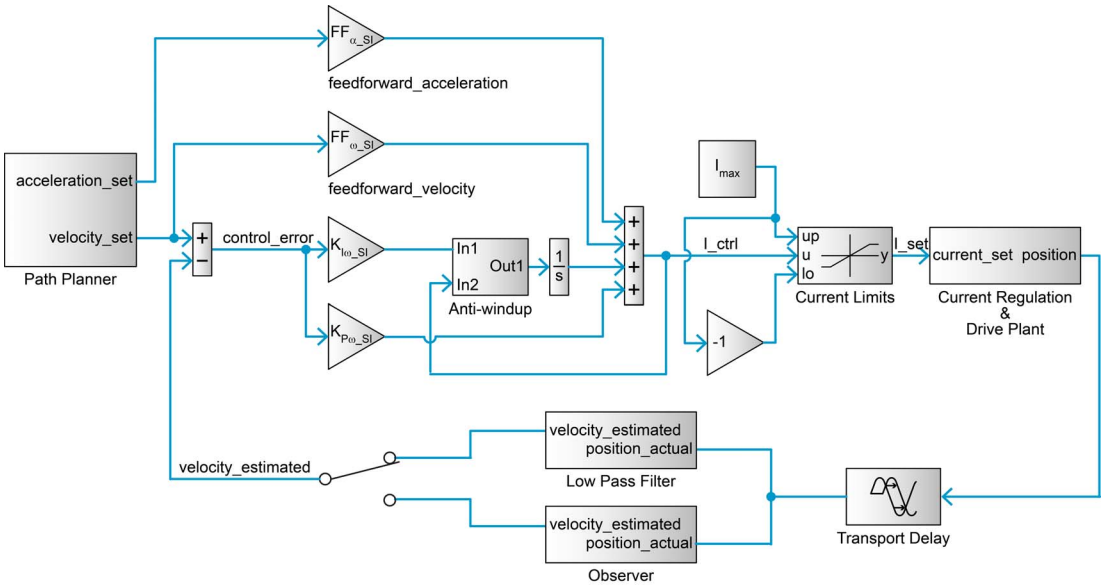


Figure 2-8 Controller architecture | Example 1: Velocity regulation

INPUT/OUTPUT PARAMETERS

- Inputs are the path planner set acceleration and set velocity.
- Outputs are the motor actual position and the motor estimated angular velocity.

MODEL PARAMETERS OF VELOCITY CONTROL

- EPOS4 PI velocity controller gains converted in SI Units.
- EPOS4 feedforward gains converted in SI Units.
- Transport delay = 0.4 ms.
- I_{max} corresponds to the motor's nominal current (for details ➔maxon catalog/motor data/line 6).

INPUT/OUTPUT PARAMETERS OF LOW PASS FILTER / OBSERVER

- Input is the motor actual position.
- Output is the motor estimated angular velocity.

MODEL PARAMETERS OF LOW PASS FILTER

- EPOS4 PI velocity controller gains converted in SI Units.

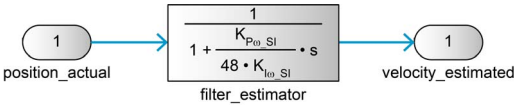


Figure 2-9 Controller architecture | Example 1: Velocity regulation – Low pass filter

MODEL PARAMETERS OF OBSERVER

- The observer is implemented as Matlab function in Simulink.
- The relevant observer parameter, converted in SI Units, are:

Mass inertia	J_{SI}
Motor torque constant	k_{m_SI}
Viscous friction	r_{SI}
Position correction gain	l_{θ_SI}
Velocity correction gain	l_{ω_SI}
Disturbance torque correction gain	l_{T_SI}

POSITION REGULATION WITH FEEDFORWARD – SIMULATION PART

The figure below displays the model of the PID position controller with feedforward. The PID position controller is connected to the current regulation.

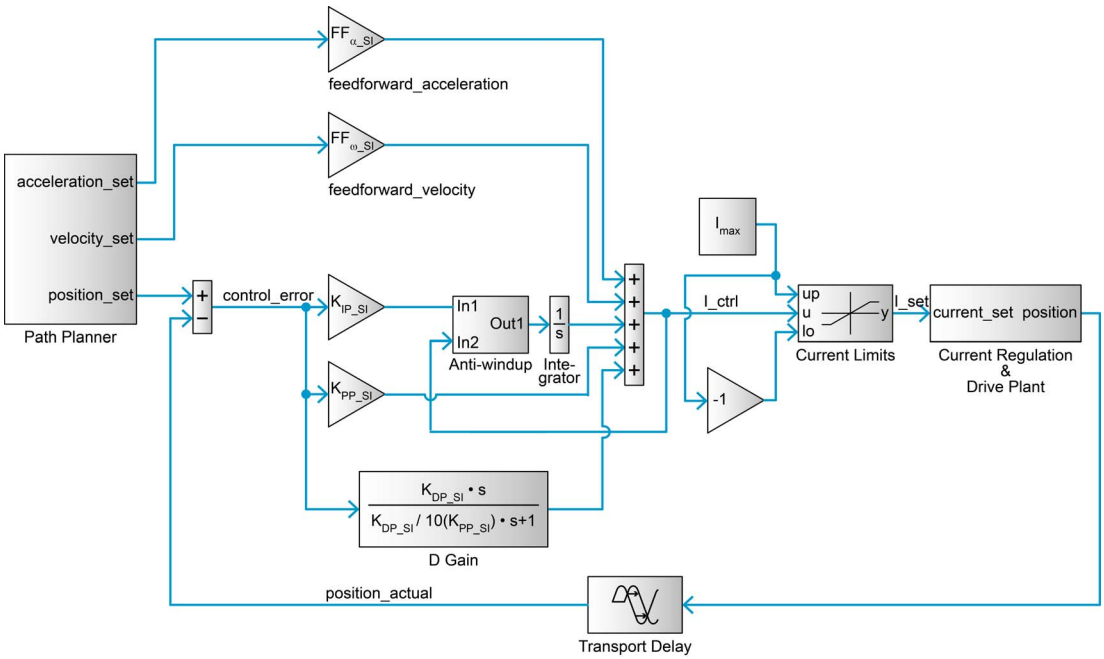


Figure 2-10 Controller architecture | Example 1: Position control with feedforward

INPUT/OUTPUT PARAMETERS

- Inputs are the path planner set acceleration, set velocity and set position.
- Output is the motor actual position.

MODEL PARAMETERS OF POSITION CONTROL

- EPOS4 PID position controller gains converted in SI Units
- EPOS4 Feedforward gains converted in SI Units
- Transport delay = 0.4 ms
- I_{max} corresponds to the motor's nominal current (for details ➔maxon catalog/motor data/line 6)

2.5.2 Example 2: System with Low Inertia and High Friction

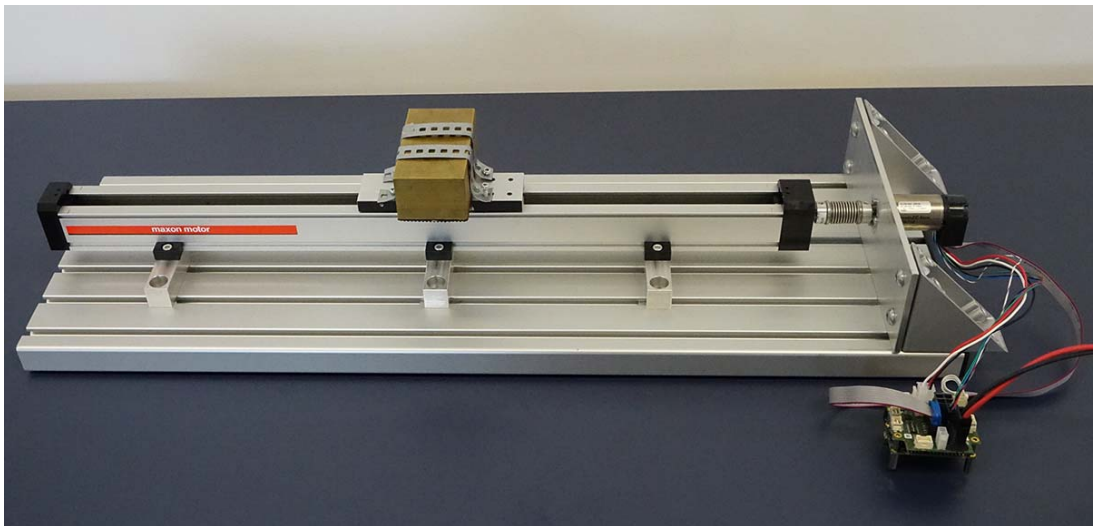


Figure 2-11 Controller architecture | Example 2: System with low inertia/high friction

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/8 CAN (520885)		
Motor maxon EC-4pole 30 (309758)	No load speed (line 2)	n_0 = 17800 rpm
	No load current (line 3)	I_0 = 270 mA
	Nominal current (line 6)	I_n = 2.82 A
	Resistance phase to phase (line 10)	R = 0.836 Ω
	Inductance phase to phase (line 11)	L = 0.118 mH
	Torque constant (line 12)	k_m = 25.5 mNm/A
	Rotor inertia (line 16)	J_{motor} = 18.3 g*cm²
Encoder AEDL-5810 (516208)	Encoder counts per turn	5000 pulses/revolution
Mechanical load Linear drive	Inertia	J_{load} = 170 g*cm²
	Friction $M_r = 6.88mNm \cdot \mathrm{sgn}(\omega) + 445.7 \frac{\mu Nm}{rad/s} \cdot \omega$	

Table 2-12 Controller architecture | Example 2: Components

SIMPLE MODEL OF THE DRIVE PLANT

The following parameters can be deduced:

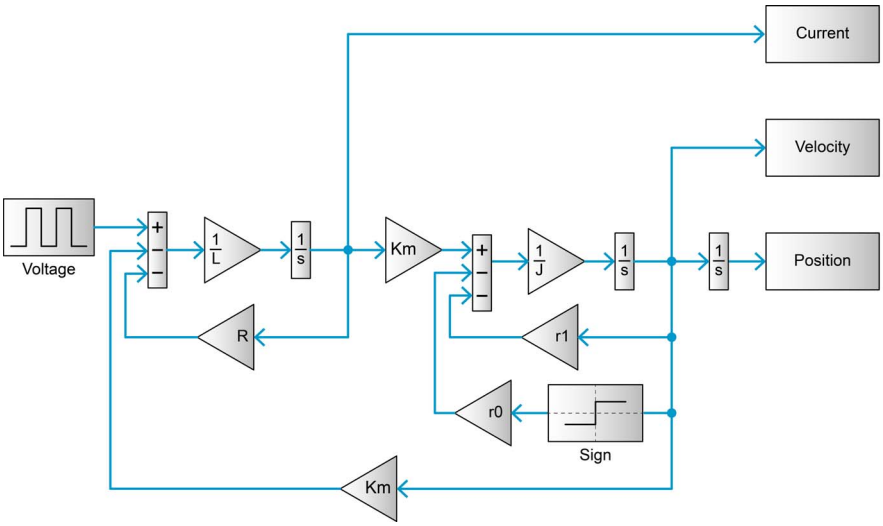


Figure 2-12 Controller architecture | Example 2: Model of the plant

INPUT/OUTPUT PARAMETERS

Input is the voltage at the motor winding.

Outputs are current, velocity, or position.

MODEL PARAMETERS

Resistance	$R = 0.836[\Omega]$
Inductance	$L = 0.118[mH]$
Torque constant	$k_m = 25.5\left[\frac{mNm}{A}\right]$
Mass inertia	$J = J_{motor} + J_{load} = 188.3[g \cdot cm^2]$
Viscous friction (approximated from the friction at no-load divided by the no-load speed of the motor)	$r_1 = \frac{k_m I_o}{n_o \frac{2\pi rad}{1}\frac{1min}{60s}} = 445.7\left[\frac{\mu Nm}{rad/s}\right]$
Static friction	$r_o = k_m I_o = 6.88[mNm]$

2.6 Best Practice Example «Differences in the use of Observer and Filter to estimate Motor Velocity»

Velocity regulation in EPOS4 can be configured by choosing either the low pass filter or the observer for estimating the motor velocity from the position measurement signals. The configuration choice is made in the «Startup Wizard» dialog box under the tab «Regulation» as illustrated in the following figure.

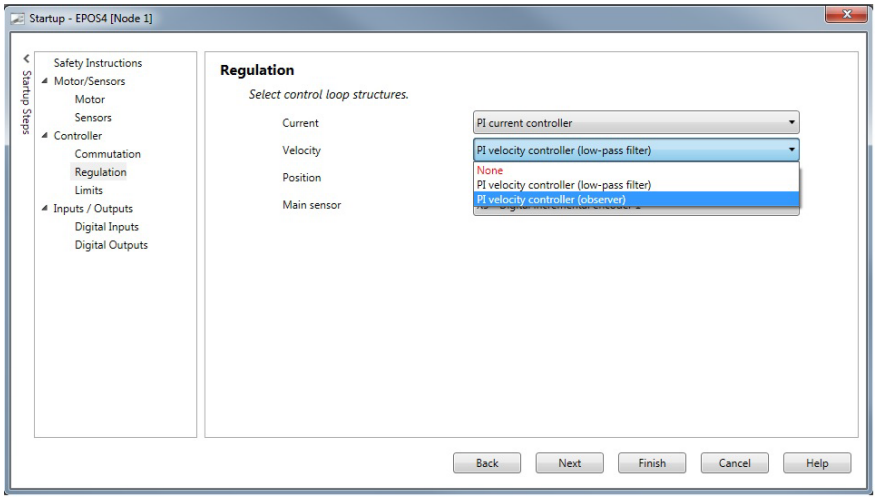


Figure 2-13 Controller architecture | Case 1: Configuration of velocity regulation mechanism

The following examples demonstrate two typical cases, the use of the observer for estimating the rotational velocity of the motor to increase the control performance compared with the case of using the low pass filter. In another example the observer does not bring much advantage and can, in fact, result in reduced control performance if the mechanical characteristics of the system are not well-identified or if they change over time.

2.6.1 Case 1: System with Low-Resolution Incremental Encoder

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/8 CAN (520885)		
Motor maxon DCX 35 L, 80 W, 12 V	No load speed (line 2)	n_0 = 8140 rpm
	No load current (line 3)	I_0 = 321 mA
	Nominal current (line 6)	I_n = 6.0 A
	Terminal resistance (line 10)	R = 0.0792 Ω
	Terminal inductance (line 11)	L = 0.0263 mH
	Torque constant (line 12)	k_m = 13.7 mNm/A
	Rotor inertia (line 16)	J_{motor} = 99.5 g*cm²
Encoder ENX16 EASY	Encoder counts per turn	256 pulses/revolution
Mechanical load Disc	Inertia	J_{load} = 250 g*cm²

Table 2-13 Controller architecture | Case 1: Components

After running the regulation tuning algorithm, the following set of parameters which describe the velocity controller used by EPOS4 are obtained.

Index	Subindex	Name	Value	Unit
0x3001	0x05	Torque constant	14.452	$\frac{mNm}{A}$
0x30A2	0x01	Velocity controller P gain	324.927	$\frac{mA \cdot s}{rad}$
0x30A2	0x02	Velocity controller I gain	2675.916	$\frac{mA}{rad}$
0x30A2	0x03	Velocity controller FF velocity gain	0.000	$\frac{mA \cdot s}{rad}$
0x30A2	0x04	Velocity controller FF acceleration gain	2.466	$\frac{mA \cdot s^2}{rad}$
0x30A2	0x05	Velocity controller filter cut-off frequency	395	$\frac{1}{s}$
0x30A3	0x01	Velocity observer position correction gain	0.600	1
0x30A3	0x02	Velocity observer velocity correction gain	151.120	Hz
0x30A3	0x03	Velocity observer load correction gain	78.971	$\frac{mNm}{rad}$

Continued on next page.

Index	Subindex	Name	Value	Unit
0x30A3	0x04	Velocity observer friction	0.000	$\frac{\mu Nm}{rpm}$
0x30A3	0x05	Velocity observer inertia	357.503	$g \cdot cm^2$

Table 2-14 Controller architecture | Case 1: Velocity regulation with low pass filter parameters, real

Note

The same control parameters are used for both experiments, low pass filter and observer. These parameters were obtained during the auto tuning procedure where the filter in closed loop was selected. The auto tuning algorithm normally gives different parameters when the observer is selected. These parameters correspond to a more aggressive PI controller as it can better utilize the advantages present when the observer is used in closed loop (more about this point is said later on after the presentation of the results).

The comparison of the two different velocity estimation algorithms is done by looking at the step response of the controller for a reference of 1000 rpm and to the ripple in the case of 50 rpm velocity reference.

The measured step response data are given in the following figure.

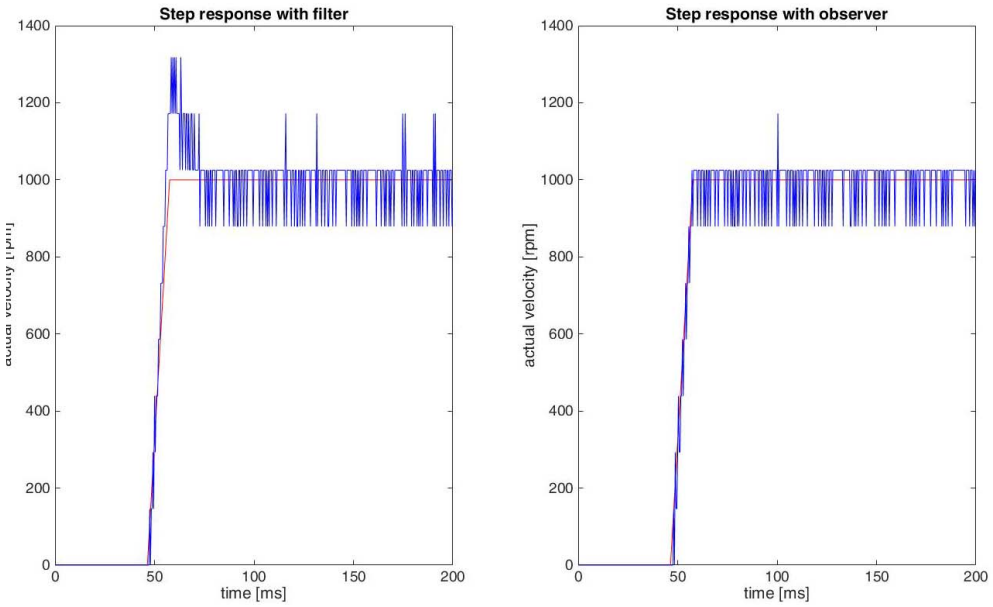


Figure 2-14 Controller architecture | Case 1: Comparison of velocity step responses

These results show the advantage of using the observer instead of the low pass filter. The controller with observer in closed loop results in much smaller overshoot and, hence, tighter reference following. The main reason for this is that the observer introduces much less phase shift in the loop than the filter would.

As a result, the velocity PI controller can be made more aggressive in the case when the observer is used compared to the use of the filter. Thus, using the observer allows much tighter reference tracking. This fact is illustrated by a step response performance comparison for the following set of control parameters which correspond to a more aggressive PI controller than in the first experiment:

Index	Subindex	Name	Value	Unit
0x3001	0x05	Torque constant	14.452	$\frac{mNm}{A}$
0x30A2	0x01	Velocity controller P gain	1127.098	$\frac{mA \cdot s}{rad}$
0x30A2	0x02	Velocity controller I gain	12838.180	$\frac{mA}{rad}$
0x30A2	0x03	Velocity controller FF velocity gain	0.000	$\frac{mA \cdot s}{rad}$
0x30A2	0x04	Velocity controller FF acceleration gain	1.368	$\frac{mA \cdot s^2}{rad}$
0x30A2	0x05	Velocity controller filter cut-off frequency	395	$\frac{1}{s}$
0x30A3	0x01	Velocity observer position correction gain	0.600	1
0x30A3	0x02	Velocity observer velocity correction gain	351.120	Hz
0x30A3	0x03	Velocity observer load correction gain	280.971	$\frac{mNm}{rad}$
0x30A3	0x04	Velocity observer friction	0.000	$\frac{\mu Nm}{rpm}$
0x30A3	0x05	Velocity observer inertia	357.503	$g \cdot cm^2$

Table 2-15 Controller architecture | Case 1: Velocity regulation with observer parameters, real

Comparison of the step responses for the more aggressive controller is given in the following figure. As can be seen, the controller with observer shows a very good performance while the performance of the controller with low pass filter deteriorates and the overshoot becomes extremely high. Additionally, the use of the filter with these high control gains results in significant amplification of audible noise.

In order to exploit this advantage of the observer, the PI controller obtained in auto tuning has higher gains when the observer is used, than when the low pass filter is used.

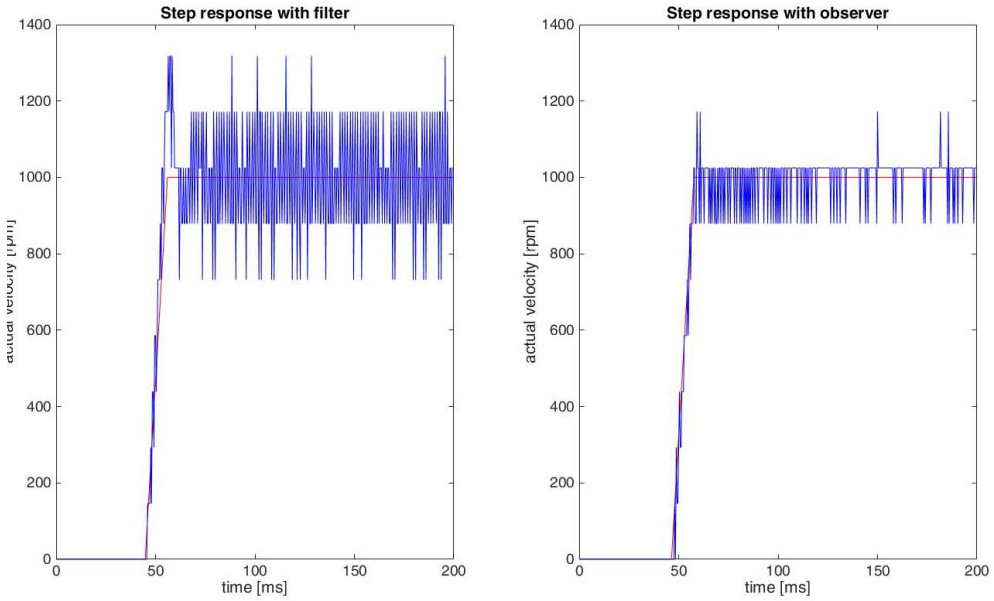


Figure 2-15 Controller architecture | Case 1: Comparison of velocity step responses

In addition to the step response, also the steady state control performance at a low rotational velocity reference value of 50 rpm are compared. The following comparison is given for the tuning parameters in ➔Table 2-15. The averaged values of the measured velocity are shown and compared for the two velocity estimation strategies.

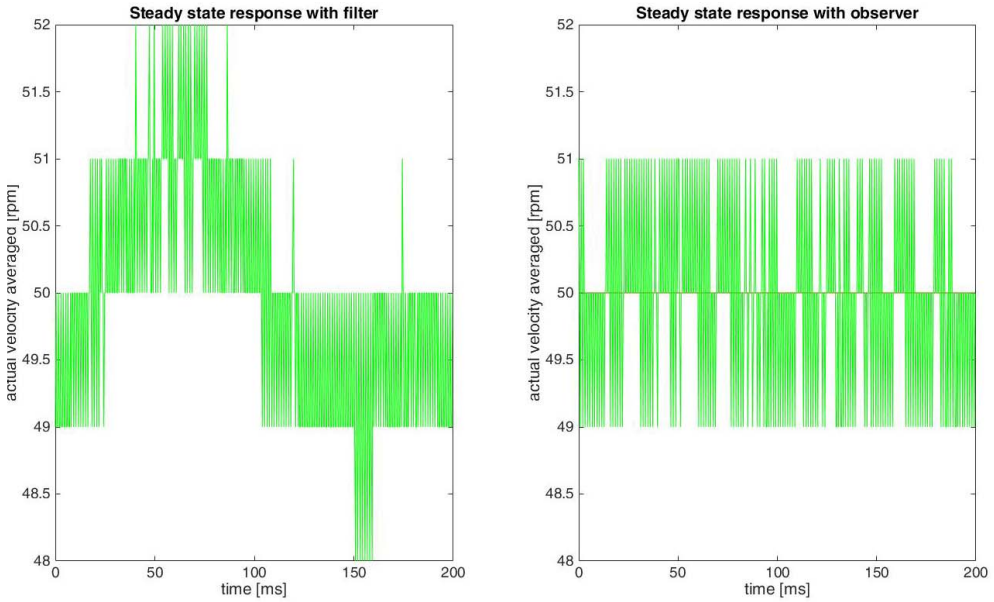


Figure 2-16 Controller architecture | Case 1: Comparison of velocity steady states

At very low velocity, the estimate obtained when the observer is used has higher quality and therefore the overall closed loop results in less ripple at steady state (i.e. more tight velocity reference following), as can be seen in ➔Figure 2-16.

2.6.2 Case 2: System with Hall Sensor

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/8 CAN (520885)		
Motor maxon EC-i 40 (496660) 7 pole pairs	No load speed (line 2)	n_0 = 8000 rpm
	No load current (line 3)	I_0 = 352 mA
	Nominal current (line 6)	I_n = 5.7 A
	Resistance phase to phase (line 10)	R = 0.207 Ω
	Inductance phase to phase (line 11)	L = 0.169 mH
	Torque constant (line 12)	k_m = 37.5 mNm/A
	Rotor inertia (line 16)	J_{motor} = 44 g*cm²
Encoder Built-in Hall sensors	Encoder counts per turn	42 pulses/revolution (7 pole pairs x 6 Hall sensor states)
Mechanical load Two discs coupled to the motor through a belt	Inertia	J_{load} = 324 g*cm²

Table 2-16 Controller architecture | Case 2: Components

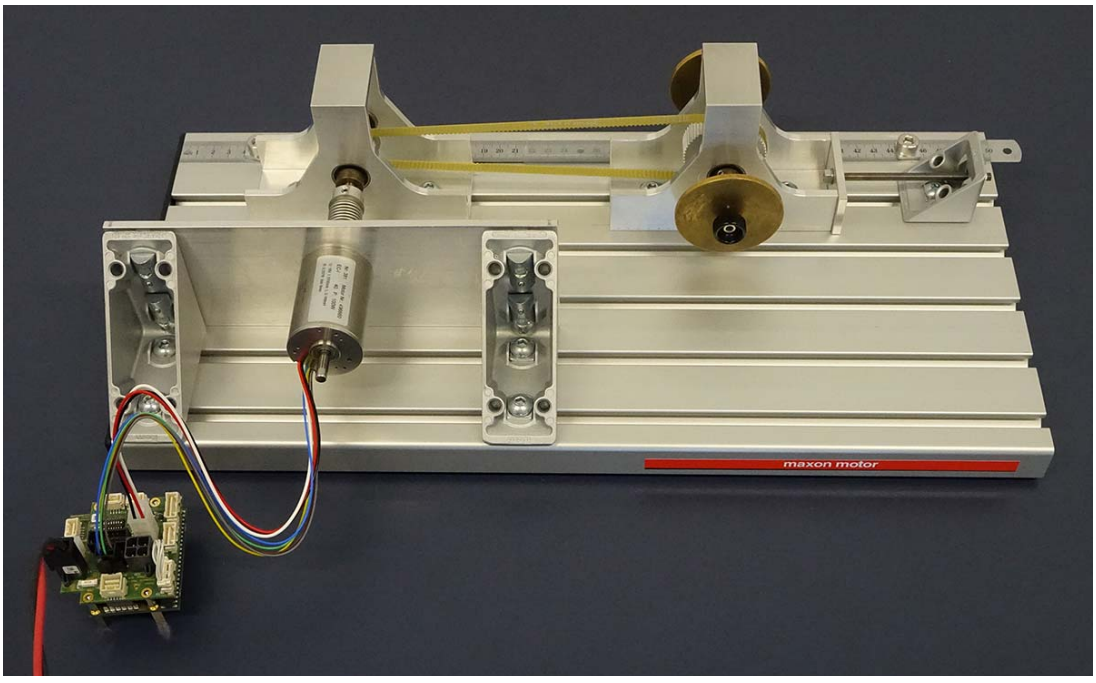


Figure 2-17 Controller architecture | Case 2: Belt drive system

The advantages of using the observer instead of the filter for estimating the motor velocity are best visible in the case when the motor has no incremental encoder, but when a Hall sensor is used for both commutating the motor and estimating its velocity.

The control and observer parameters used in the experiments are the following:

Index	Subindex	Name	Value	Unit
0x3001	0x05	Torque constant	38.120	$\frac{mNm}{A}$
0x30A2	0x01	Velocity controller P gain	138.551	$\frac{mA \cdot s}{rad}$
0x30A2	0x02	Velocity controller I gain	10494.003	$\frac{mA}{rad}$
0x30A2	0x03	Velocity controller FF velocity gain	89.548	$\frac{mA \cdot s}{rad}$
0x30A2	0x04	Velocity controller FF acceleration gain	0.601	$\frac{mA \cdot s^2}{rad}$
0x30A2	0x05	Velocity controller filter cut-off frequency	395	$\frac{1}{s}$
0x30A3	0x01	Velocity observer position correction gain	0.399	1
0x30A3	0x02	Velocity observer velocity correction gain	68.056	Hz
0x30A3	0x03	Velocity observer load correction gain	67.834	$\frac{mNm}{rad}$
0x30A3	0x04	Velocity observer friction	0.366	$\frac{\mu Nm}{rpm}$
0x30A3	0x05	Velocity observer inertia	402.538	$g \cdot cm^2$

Table 2-17 Controller architecture | Case 2: Velocity regulation parameters, real

Compared are the averaged measured motor velocity for a square velocity profile in the cases when the low pass filter and observer are used for estimating the rotor velocity respectively.

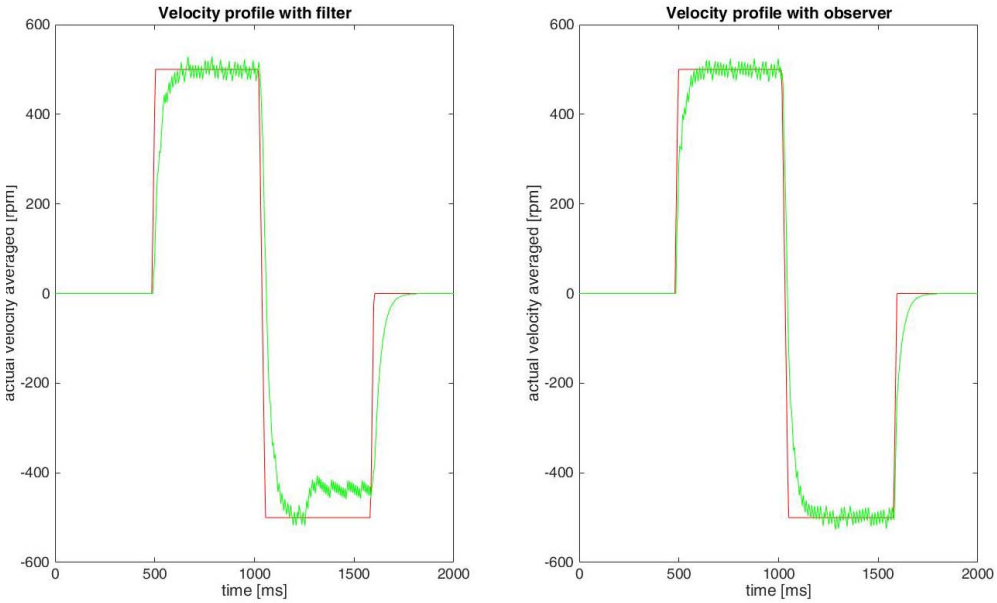


Figure 2-18 Controller architecture | Case 2: Comparison of velocity step responses

As it can be seen, the use of the observer leads to tighter velocity reference tracking. In addition, the controller with the observer in closed loop produces much less audible noise during operation.

2.6.3 Case 3: System with High-Resolution Encoder

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/8 CAN (520885)		
Motor maxon EC 4pole 30 (309758)	No load speed (line 2)	n_0 = 17800 rpm
	No load current (line 3)	I_0 = 270 mA
	Nominal current (line 6)	I_n = 2.82 A
	Resistance phase to phase (line 10)	R = 0.836 Ω
	Inductance phase to phase (line 11)	L = 0.118 mH
	Torque constant (line 12)	k_m = 25.5 mNm/A
	Rotor inertia (line 16)	J_{motor} = 18.3 g*cm²
Encoder AEDL-5810 (516208)	Encoder counts per turn	5000 pulses/revolution
Mechanical load Two discs coupled to the motor through a belt	Inertia	J_{load} = 324 g*cm²

Table 2-18 Controller architecture | Case 3: Components

In this application example, the encoder resolution is very high and therefore the controller with the filter in closed loop has very similar behavior as the closed loop with the observer.

The control parameters, obtained from auto tuning for which the comparison is made, are the following:

Index	Subindex	Name	Value	Unit
0x3001	0x05	Torque constant	26.518	$\frac{mNm}{A}$
0x30A2	0x01	Velocity controller P gain	968.601	$\frac{mA \cdot s}{rad}$
0x30A2	0x02	Velocity controller I gain	16748.581	$\frac{mA}{rad}$
0x30A2	0x03	Velocity controller FF velocity gain	0.000	$\frac{mA \cdot s}{rad}$
0x30A2	0x04	Velocity controller FF acceleration gain	1.313	$\frac{mA \cdot s^2}{rad}$
0x30A2	0x05	Velocity controller filter cut-off frequency	395	$\frac{1}{s}$
0x30A3	0x01	Velocity observer position correction gain	0.650	1
0x30A3	0x02	Velocity observer velocity correction gain	369.465	Hz
0x30A3	0x03	Velocity observer load correction gain	53.370	$\frac{mNm}{rad}$
0x30A3	0x04	Velocity observer friction	0.000	$\frac{\mu Nm}{rpm}$
0x30A3	0x05	Velocity observer inertia	348.148	$g \cdot cm^2$

Table 2-19 Controller architecture | Case 3: Velocity regulation parameters, real

The comparison of the controller transient behavior is done by experiments in which a 1000 rpm step should be followed.

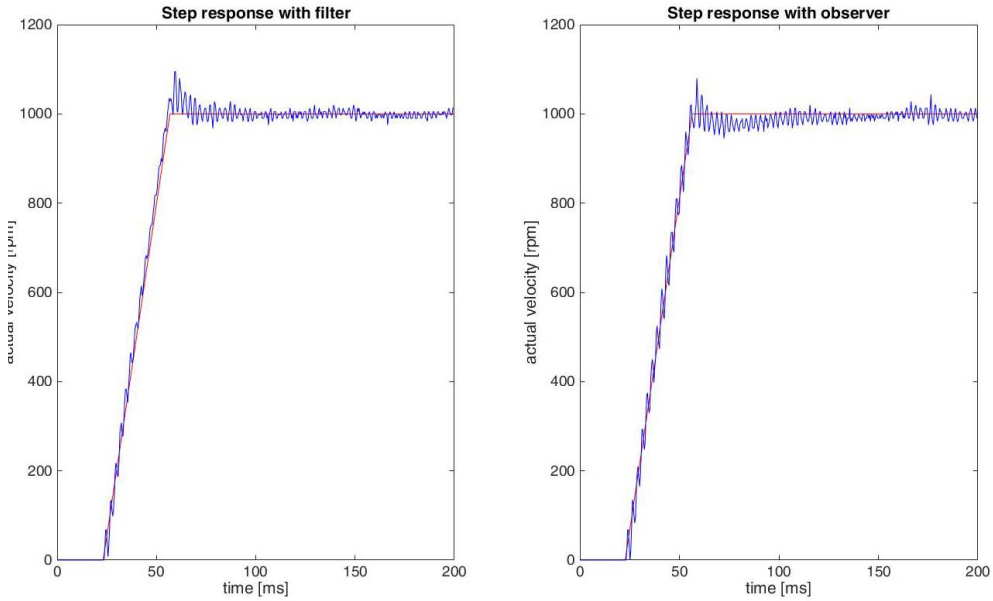


Figure 2-19 Controller architecture | Case 3: Comparison of velocity step responses

In addition, the steady state controller behavior for a constant reference of 20 rpm are compared.

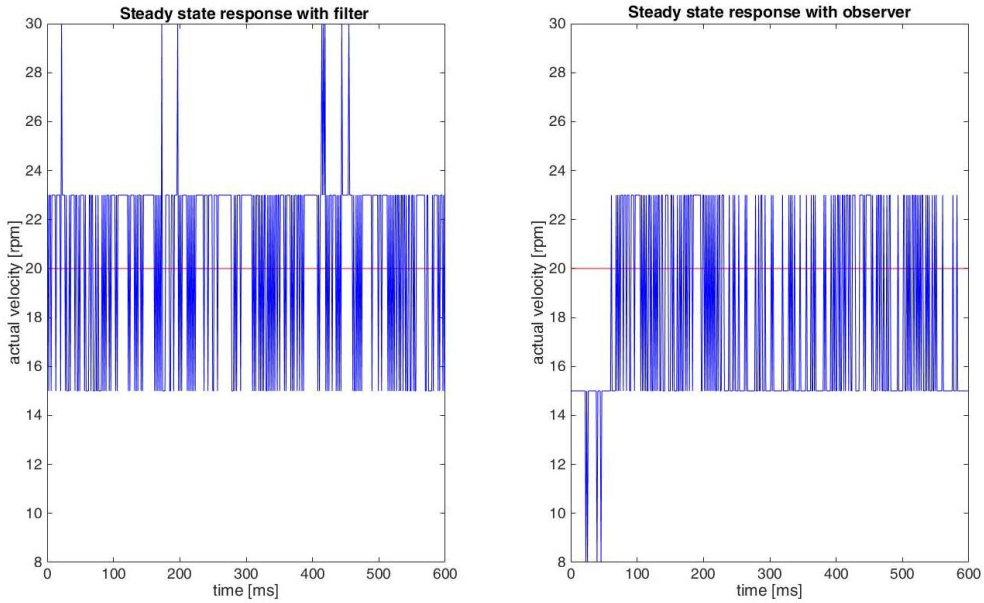


Figure 2-20 Controller architecture | Case 3: Comparison of velocity steady states

As can be seen, there is very little to no difference in the control performance. The reason is that the quality of the position and hence the velocity measurement is very good. Therefore, using the model of the mechanical system to which the motor is attached, as is done when the observer is used does not bring a lot of benefit. On the contrary, in such cases it may happen that the performance of the closed loop with observer becomes worse than the performance with the filter if the mechanical model parameters are not accurate or if they change over time.

2.7 Conclusion

The described application examples show that it makes sense to use the observer for estimating the rotational velocity of the motor in cases when the position sensor has low resolution and when the parameters of the mechanical system are constant and can be well identified. In these cases, the use of the observer results in less ripple at low velocities and allows for more tight dynamic following of the reference signal than in the case when the low pass filter is used. On the other hand, when position sensors with high resolution are used, the use of the observer cannot bring much benefit, but instead could lead to deterioration in control performance if the mechanical model of the system is not accurate. In these cases it is better to use a filter for estimating the rotational velocity.

3 COMPARISON OF MAXON SERIAL PROTOCOLS FOR RS232

3.1 In Brief

With the introduction of the EPOS4 series, the positioning controllers' RS232 transmission protocol has been optimized and is now identical to the USB transmission protocol. This results in higher stability and improved performance of the RS232 serial communication data flow.

3.2 Description

Note
The protocol change has an effect on RS232 communication, only. Hence, USB communication remains unchanged.

The differences between the protocols «maxon V1» and «maxon V2» are as to the following details.

Type of controller	Interface	
	RS232	USB
EPOS2	maxon V1	maxon V2
EPOS4	maxon V2	maxon V2

Table 3-20 maxon serial protocol V1 vs. V2 | Protocol change – Overview

The two protocols feature different RS232 data flow while transmitting and receiving frame for EPOS2 and EPOS4 positioning controllers.

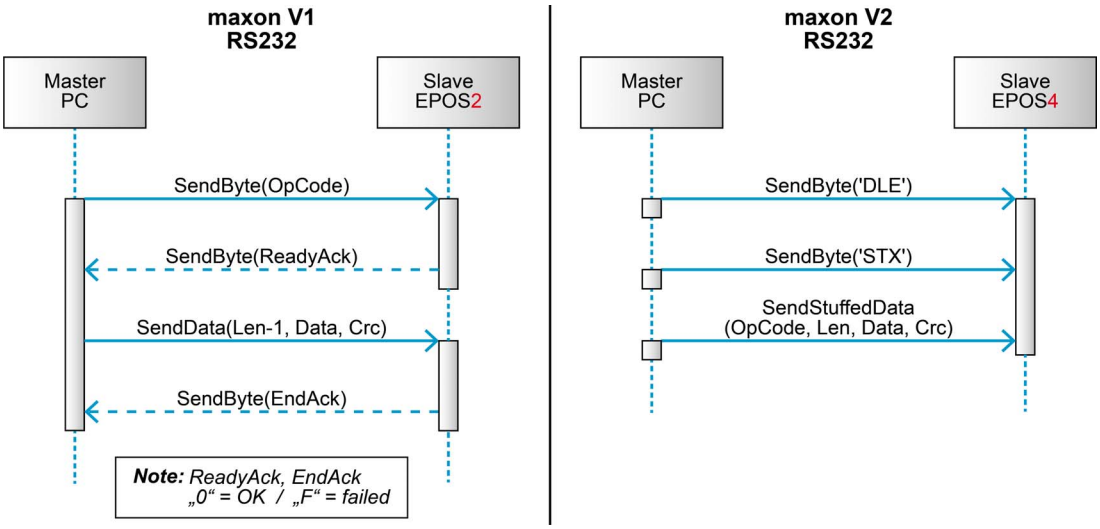


Figure 3-21 maxon serial protocol V1 vs. V2 | RS232 communication – Sending a data frame

3.2.1 maxon Serial V1

The data bytes are sequentially transmitted in frames. After sending the first frame byte (OpCode), the Master needs to wait for the "Ready Acknowledge". A frame composes of…

- header,
- variably long data field, and
- 16-bit long cyclic redundancy check (CRC) for verification of data integrity.

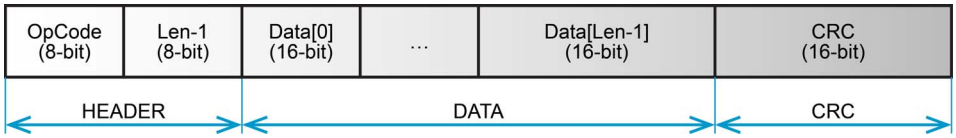


Figure 3-22 maxon serial protocol V1 vs. V2 | maxon serial V1 protocol – Frame structure

3.2.2 maxon Serial V2

The data bytes are sequentially transmitted in frames. The first two bytes (DLE/STX) are used for frame synchronization. Therefore, there is no need to wait for an acknowledge and thus, communication is simplified compared to maxon serial V1 protocol. A frame composes of…

- synchronization characters,
- header with data stuffing,
- variably long data field with data stuffing, and
- 16-bit long cyclic redundancy check (CRC) for verification of data integrity with data stuffing.

Figure 3-23 maxon serial protocol V1 vs. V2 | maxon serial V2 protocol – Frame structure

Note
For further details on commissioning, control possibilities, and command instruction examples for an EPOS2 see separate document ➔«EPOS2 Communication Guide».

4 FIRMWARE UPDATE WITHOUT USE OF «EPOS STUDIO»

CONTENTS

4.1 In Brief

OBJECTIVE

The present application note explains how to carry out a firmware update of an EPOS4 controller including EPOS4 Extensions (such as EtherCAT) without the use of the «EPOS Studio» directly via the existing bus systems. The compatibility of the various versions as well as the necessary implementation sequences for the different communication interfaces are described.

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0100h	Firmware Specification Communication Guide
EPOS4 Disk 60/8 CAN EPOS4 Disk 60/8 EtherCAT	688770 688772	0170h or higher 0170h or higher	
EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC EPOS4 Disk 60/12 EtherCAT EPOS4 Disk 60/12 EtherCAT SSC	688775 709859 688777 709862	0170h or higher 0170h or higher 0170h or higher 0170h or higher	
EPOS4 Module 24/1.5 EPOS4 Compact 24/1.5 CAN EPOS4 Compact 24/1.5 EtherCAT	536630 546714 628092	0110h or higher 0110h or higher 0150h or higher	
EPOS4 Module 50/5 EPOS4 Compact 50/5 CAN EPOS4 Compact 50/5 EtherCAT	534130 541718 628094	0110h or higher 0110h or higher 0150h or higher	
EPOS4 Module 50/8 EPOS4 Compact 50/8 CAN EPOS4 Compact 50/8 EtherCAT	504384 520885 605298	0100h or higher 0100h or higher 0140h or higher	
EPOS4 Module 50/15 EPOS4 Compact 50/15 CAN EPOS4 Compact 50/15 EtherCAT	504383 520886 605299	0100h or higher 0100h or higher 0140h or higher	
EPOS4 50/5	546047	0120h or higher	
EPOS4 70/15	594385	0140h or higher	

Table 4-21 EtherCAT integration | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.4 or higher (required for initial export of Program Data File, only)

Table 4-22 EtherCAT integration | Recommended tools

4.2 Preconditions

SUPPORTED INTERFACES AND EXTENSIONS

The following table shows the relation of a given firmware version on an EPOS4 positioning controller and its support of the firmware update functionality for a specific communication interface or extension.

The firmware update of an extensions (such as EtherCAT) is part of the firmware update of the EPOS4 positioning controller itself (loop through) and cannot be processed independently.

Firmware version	interface				
	USB	RS232	CANopen	EtherCAT	EtherCAT extension
0x0100	✓	—	—	—	—
0x0110	✓	—	—	—	—
0x0120	✓	—	—	—	—
0x0130	✓	—	—	—	✓
0x0140	✓	—	✓	—	✓
0x0150 or higher	✓	—	✓	✓	✓

Table 4-23 Firmware update without «EPOS Studio» | Firmware version vs. interface or extension

4.3 Program Data File

The firmware update sequence requires a «Program Data File» containing the desired firmware version. These files are exported using the «EPOS Studio».

STARTING «EPOS STUDIO»

1) Make sure you installed «EPOS Studio» version 3.4 (or higher) on your PC.
   If not the case, download the latest version here: ➔http://epos.maxongroup.com
2) Start «EPOS Studio» without creating a new project. Thereby, an online connection to an EPOS4 controller is not necessary.

EXPORTING «PROGRAM DATA FILE»

1) In the main menu ¤Extras¤, open the dialog ¤Firmware File Registration¤.
2) Add a firmware file and click the ¤Add File¤ button.

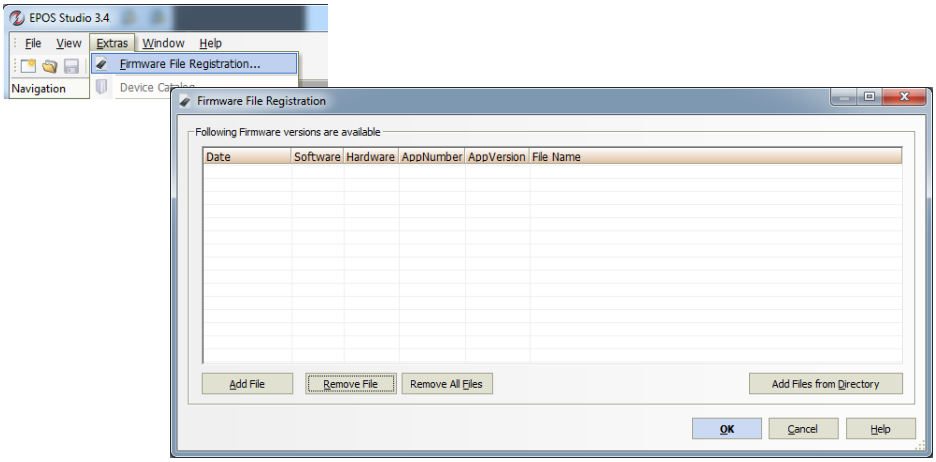


Figure 4-24 Firmware update without «EPOS Studio» | Open firmware file registration dialog

3) Select the firmware and click right to open the context menu, then click ¤Export Program Data File¤.

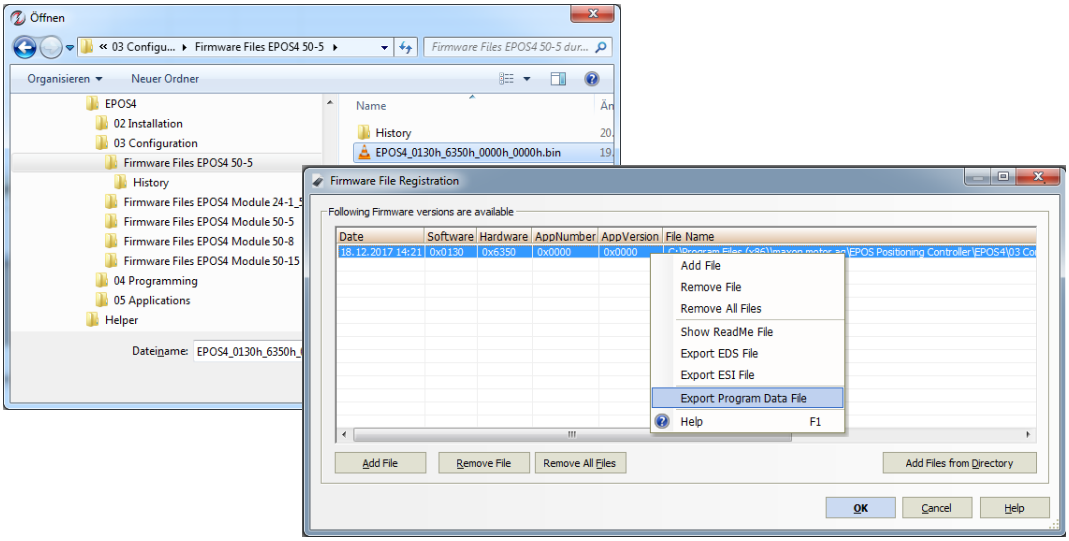


Figure 4-25 Firmware update without «EPOS Studio» | Export program data file

4) Select the directory to export file and click ¤OK¤.

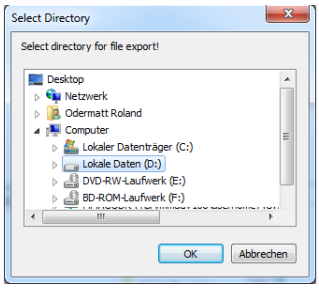


Figure 4-26 Firmware update without «EPOS Studio» | Select export directory

5) Click ¤OK¤ to confirm.

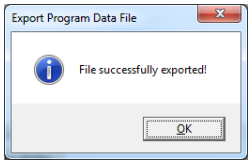


Figure 4-27 Firmware update without «EPOS Studio» | Confirm export directory

6) Check the exported firmware file (*.msdc).

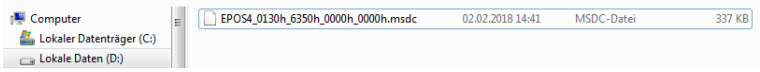


Figure 4-28 Firmware update without «EPOS Studio» | Check firmware file

4.4 Firmware Update via USB

SUPPORTED UPDATE PATHS

The following table shows the compatibility of a given firmware version for direct update via the USB interface.

Note
With «EPOS Studio», an update between all versions can be performed.

OLD firmware version	NEW firmware version					
	0x0100	0x0110	0x0120	0x0130	0x0140	higher
0x0100		✓ 1	✓ 1	n/a	n/a	n/a
0x0110	✓ 1		✓ 1	n/a	n/a	n/a
0x0120	✓ 1	✓ 1		✓ 2	✓ 2	✓ 2
0x0130	✓ 1	✓ 1	✓ 1		✓ 1	✓ 1
0x0140	✓ 1	✓ 1	✓ 1	✓ 1		✓ 1
higher	✓ 1	✓ 1	✓ 1	✓ 1	✓ 1	

✓ 1 supported by Sequence 1

✓ 2 supported by Sequence 2

n/a not supported

Table 4-24 Firmware update without «EPOS Studio» | USB – Old vs. new firmware version

SEQUENCE 1 (STANDARD)

Steps

a) Prepare controller (➔“Prepare Controller” on page 4-45)

b) Download «program data file» (CiA 302-3) “EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc” (➔“Download «Program Data File» (CiA 302-3)” on page 4-46)

c) Check identity (➔“Check Identity” on page 4-47)

SEQUENCE 2 (STANDARD + ETHERCAT EXTENSION)

Steps

a) Prepare controller (➔“Prepare Controller” on page 4-45)

b) Download «program data file» (CiA 302-3) “EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc” (➔“Download «Program Data File» (CiA 302-3)” on page 4-46)

c) If you use an EtherCAT Extension:
– Check existence of «Extension EtherCAT» (➔“Check existence of «Extension EtherCAT»” on page 4-47)
– Re-download program data file (CiA 302-3) “EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc” (➔“Download «Program Data File» (CiA 302-3)” on page 4-46)

d) Check identity (➔“Check Identity” on page 4-47)

4.5 Firmware Update via CANopen

SUPPORTED UPDATE PATHS

The following table shows the compatibility of a given firmware version for direct update via the CANopen interface.

OLD firmware version	NEW firmware version					
	0x0100	0x0110	0x0120	0x0130	0x0140	higher
0x0100		n/a	n/a	n/a	n/a	n/a
0x0110	n/a		n/a	n/a	n/a	n/a
0x0120	n/a	n/a		n/a	n/a	n/a
0x0130	n/a	n/a	n/a		n/a	n/a
0x0140	✓ 1	✓ 1	✓ 1	✓ 1		✓ 1
higher	✓ 1	✓ 1	✓ 1	✓ 1	✓ 1	

✓ 1 supported by Sequence 1

n/a not supported

Table 4-25 Firmware update without «EPOS Studio» | CANopen – Old vs. new firmware version

SEQUENCE 1 (STANDARD)

Steps

a) Prepare controller (➔“Prepare Controller” on page 4-45)

b) Download «program data file» (CiA 302-3) “EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc” (➔“Download «Program Data File» (CiA 302-3)” on page 4-46)

c) Check identity (➔“Check Identity” on page 4-47)

4.6 Firmware Update via RS232

Note

The firmware update functionality for the RS232 interface is available on request.

SEQUENCE 1 (STANDARD)

Steps

a) Prepare controller (➔“Prepare Controller” on page 4-45)

b) Download «program data file» (CiA 302-3) “EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc” (➔“Download «Program Data File» (CiA 302-3)” on page 4-46)

c) Check identity (➔“Check Identity” on page 4-47)

4.7 Firmware Update via EtherCAT

SUPPORTED UPDATE PATHS

The following table shows the compatibility of a given firmware version for direct update via the EtherCAT interface.

OLD firmware version	NEW firmware version	
	0x0150	higher
0x0150		✓
higher	✓	

Table 4-26 Firmware update without «EPOS Studio» | EtherCAT – Old vs. new firmware version

SEQUENCE 1 (STANDARD)

Steps

a) Prepare controller (➔“Prepare Controller” on page 4-45)

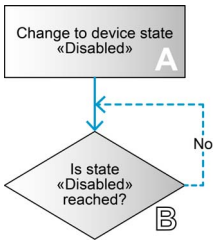
b) Download «program data file» (FoE) “EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc” (➔“Download «Program Data File» (FoE)” on page 4-47)

c) Check identity (➔“Check Identity” on page 4-47)

4.8 Steps: How to…

The following section describes the implementation of the required steps during the different firmware update sequences.

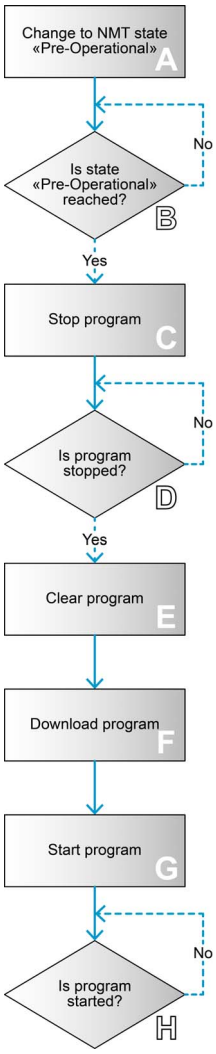
4.8.1 Prepare Controller



#	Step	Description
A	Change to device control state «Disabled»	➔separate document «EPOS4 Firmware Specification»; chapter “Device Control”
B	Check state	

Table 4-27 Firmware update without «EPOS Studio» | How to prepare the controller

4.8.2 Download «Program Data File» (CiA 302-3)



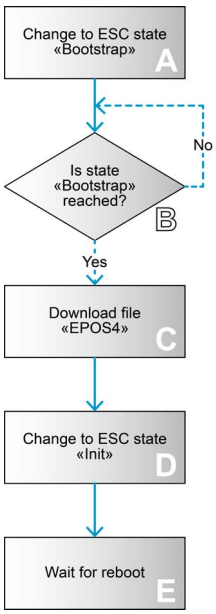
#	Step	Description	
A	Change to device control state «Pre-Operational» [a]	➔separate document «EPOS4 Communication Guide»; chapter “CAN Communication”	
B	Check NMT state [a]		
C	Stop program [b]	Write «Stop» to object «Program Control»	
		Object Value Timeout	0x1F51-01 0x00 (Stop) 10 ms
D	Wait until program is stopped	Read value from object «Program Control»	
		Object Expected value Wait timeout	0x1F51-01 0x00 (Stopped) 10’000 ms
E	Clear program	Write «Clear» to object «Program Control»	
		Object Value Timeout	0x1F51-01 0x03 (Clear) 20’000 ms
F	Download program	Write file content to object «Program Data»	
		Object File Timeout	0x1F50-01 EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc 10’000 ms
G	Start program [b]	Write «Start» to object «Program Control»	
		Object Value Timeout	0x1F51-01 0x01 (start) 10 ms
H	Wait until program is started	Read value from object «Program Control»	
		Object Expected value Wait timeout	0x1F51-01 0x01 (Started) 10’000 ms

[a] only for CANopen interface

[b] During starting or stopping the program, the communication protocol is aborted. The controller does not respond to the received command. Reduce timeout and do not check communication result.

Table 4-28 Firmware update without «EPOS Studio» | How to download the program data file (CiA 302-3)

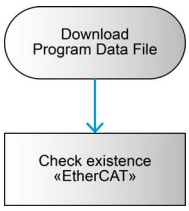
4.8.3 Download «Program Data File» (FoE)



#	Step	Description	
A	Change to ESC state «Bootstrap»		
B	Check ESC state		
C	Download file «EPOS4»	Write file content to file «EPOS4» using FoE protocol	
		Password Source file Target file	0 EPOS4_wwwwh_xxxxh_yyyyh_zzzzh.msdc EPOS4
D	Change to ESC state «Init»		
E	Wait for reboot	Read ESC state	

Table 4-29 Firmware update without «EPOS Studio» | How to download the program data file (FoE)

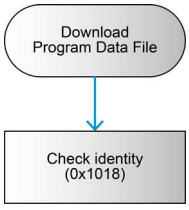
4.8.4 Check existence of «Extension EtherCAT»



Step	Description	
Check «Extension 1 type»	Read value from object «Extension 1 type»	
	Object Expected value	0x2101-05 2 (EtherCAT)

Table 4-30 Firmware update without «EPOS Studio» | How to check existence of «Extension EtherCAT»

4.8.5 Check Identity



Step	Description	
Check «Product Code»	Read value from object «Identity – Product code»	
	Object Expected value	0x1018-02 Hardware version and application number
Check «Revision number»	Read value from object «Identity – Revision number»	
	Object Expected value	0x1018-03 Software version and application version

Table 4-31 Firmware update without «EPOS Studio» | How to check identity

4.9 Object Dictionary

OBJECTS IN «STOPPED» STATE

While the program is stopped, only a few objects are accessible.

Index	Name
0x1000-00	Device type
0x1008-00	Manufacturer device name
0x1018-01	Identity – Vendor-ID
0x1018-02	Identity – Product code
0x1018-03	Identity – Revision number
0x1018-04	Identity – Serial number
0x1F50-00	Program data
0x1F51-00	Program control
0x1F56-00	Program software identification

Table 4-32 Firmware update without «EPOS Studio» | Objects in «Stopped» state

OBJECTS VALUES IN «STOPPED» STATE

While the program is stopped, the displayed values of the following objects differ.

Index	Name	Program started Application active	Program stopped Bootloader active
0x1000-0x00	Device type	0x00020192	0x0000012E
0x1018-0x02	Product code	• High word: Hardware version • Low word: Application number	• High word: Hardware version • Low word: 0x0000
0x1018-0x03	Revision number	• High word: Software version • Low word: Application version	• High word: 0x0000 • Low word: 0x0000

Table 4-33 Firmware update without «EPOS Studio» | Objects values in «Stopped» state

5 CANOPEN BASIC INFORMATION

CONTENTS

5.1 In Brief

A wide variety of operating modes permit flexible configuration of drive and automation systems by using positioning, speed and current regulation. The built-in CANopen interface allows networking to multiple drives as well as online commanding by CAN bus master units.

For fast communication with several EPOS4 devices, we suggest to use the CANopen protocol. The individual devices within the network are commanded by one common CANopen master.

OBJECTIVE

The present Application Note explains the functionality of the CANopen structure and protocol. It also describes the configuration process in a step-by-step procedure.

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0100h	Firmware Specification
EPOS4 Disk 60/8 CAN	688770	0170h or higher	
EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC	688775 709859	0170h or higher 0170h or higher	
EPOS4 Micro 24/5 CAN	638328	0160h or higher	
EPOS4 Module 24/1.5 EPOS4 Compact 24/1.5 CAN	536630 546714	0141h or higher 0110h or higher	
EPOS4 Module 50/5 EPOS4 Compact 50/5 CAN	534130 541718	0110h or higher 0110h or higher	
EPOS4 Module 50/8 EPOS4 Compact 50/8 CAN	504384 520885	0100h or higher 0100h or higher	
EPOS4 Module 50/15 EPOS4 Compact 50/15 CAN	504383 520886	0100h or higher 0100h or higher	
EPOS4 50/5	546047	0120h or higher	
EPOS4 70/15	594385	0140h or higher	
CANopen Network	—	0100h or higher	CiA 301 V4.2 (➔[3]) CiA 402 V3.0 (➔[5])
CANopen Layer Setting Service	—	0160h or higher	CiA 305 V3.0 (➔[4])

Table 5-34 CANopen basic information | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.60 or higher

Table 5-35 CANopen basic information | recommended tools

5.2 Network Structure

maxon EPOS4 drives' CAN interface follows the CANopen specifications CiA 301 V4.2 «CANopen application layer and communication profile» and CiA 402 V3.0 «Drives and motion control device profile».

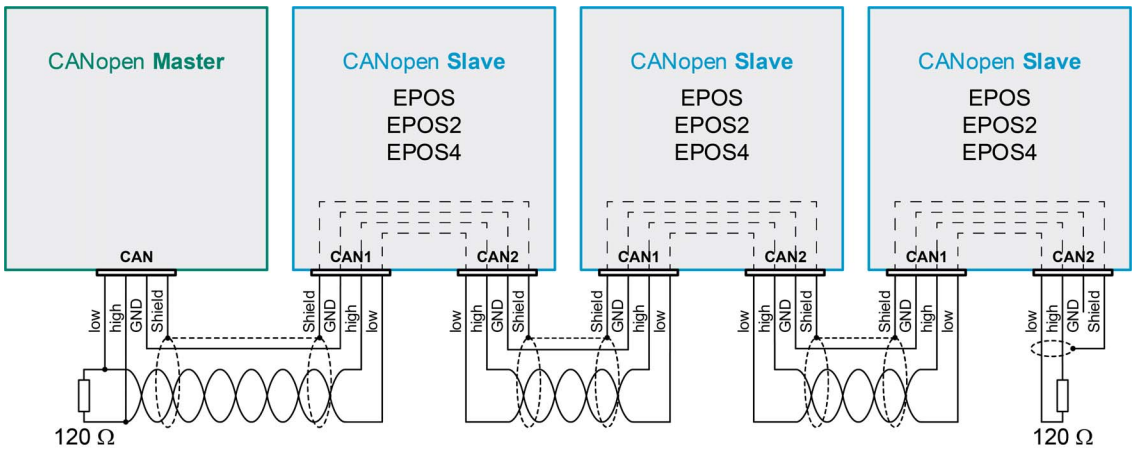


Figure 5-29 CANopen basic information | Topology with external bus termination (example)

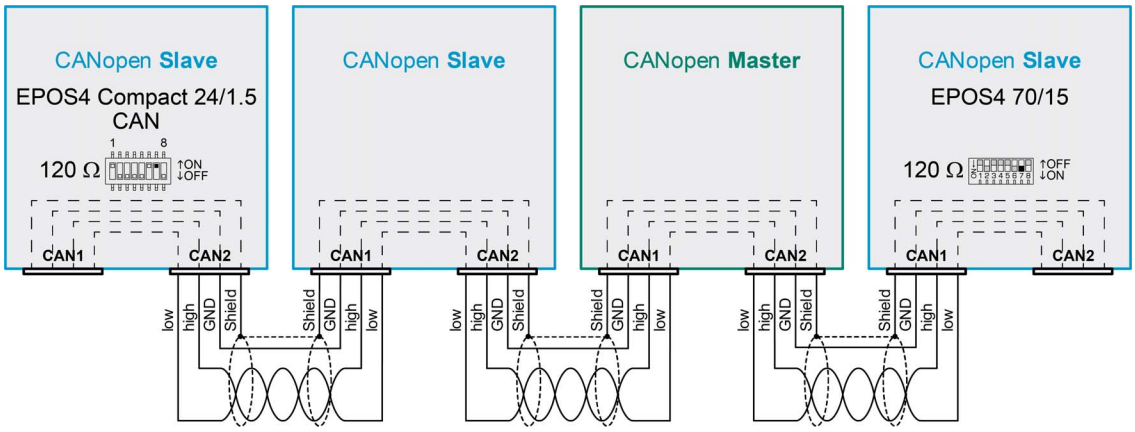


Figure 5-30 CANopen basic information | Topology with internal bus termination (example)

The CAN bus line must be terminated at both ends using a termination resistor of typically 120 Ω. Most EPOS4 positioning controllers are equipped with an internal bus termination feature that can be switched on by a DIP switch.

Module and Micro Versions
DIP switches are only available with EPOS4 Compact CAN and encased housing versions. With EPOS4 Micro and EPOS4 Module, CAN termination must be considered by the motherboard design.

Continued on next page.

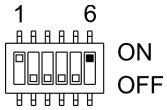
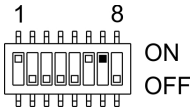
Device	Bus terminated with 120 Ω	DIP switch setting
EPOS4 Disk 60/8 CAN EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC	DIP switch 6 “ON”	
EPOS4 Compact 24/1.5 CAN EPOS4 Compact 50/5 CAN EPOS4 Compact 50/8 CAN EPOS4 Compact 50/15 CAN	DIP switch 7 “ON”	
EPOS4 50/5 EPOS4 70/15	DIP switch 7 “ON”	

Table 5-36 CANopen basic information | DIP switch settings for CAN bus termination

5.3 Configuration

Follow below step-by-step instructions for correct CAN communication setup.

5.3.1 Step 1: CANopen Master

Use one of the PC/CAN interface cards or PLCs listed below. For all of them, motion control libraries, examples and documentation are available on the Internet (for URLs ➔page 1-7).

Recommended Component	Manufacturer / Contact	Supported Product	maxon Motion Control Library [c]
PC/CAN Interface Card [a]	**IXXAT** www.ixxat.de	All offered CANopen cards	Windows 32-Bit/64-Bit DLL Linux 32-Bit/64-Bit (Intel x86) Linux 32-Bit (ARM V7/V8)
	Kvaser www.kvaser.com	All offered CANopen cards	Windows 32-Bit/64-Bit DLL Linux 32-Bit/64-Bit (Intel x86) Linux 32-Bit (ARM V7/V8)
	MTTCAN	nVidia Jetson-TX2	Linux 64-Bit (ARM V8)
	National Instruments www.ni.com/can	All offered CANopen cards	Windows 32-Bit/64-Bit DLL
	piCAN2	Raspberry Pi 2/3	Linux 32-Bit (ARM V7/V8)
	Vector www.vector-informatik.de	All offered CANopen cards	Windows 32-Bit/64-Bit DLL
PLCs [b]	**Beckhoff** www.beckhoff.de	All offered CANopen cards	IEC 61131-3 Beckhoff Library
	Siemens www.siemens.com **Helmholz** www.helmholz.de	S7-300 with Helmholz CAN300 Master	Delivered and supported by Helmholz
Dedicated motion control masters	zub machine control AG www.zub.ch	All offered MACS controllers	Commanding and configured directly by MACS application program

[a] Interface driver of CANopen card must be installed
[b] All CAN products of other manufacturers may also be used. However, no motion control library is available
[c] Find detailed information and specifications on library usage and function calls in separate document ➔«EPOS Command Library» documentation

Table 5-37 CANopen basic information | recommended components

5.3.2 Step 2: CAN Bus Wiring

The two-wire bus line must be terminated at both ends using a termination resistor of 120 Ω. Twisting is recommended, shielding is suggested (depending on EMC requirements).

EPOS4 POSITIONING CONTROLLER

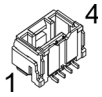
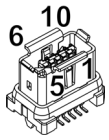
EPOS4			
Compact 24/1.5 CAN (546714) **Compact 50/5 CAN** (541718) **Compact 50/8 CAN** (520885) **Compact 50/15 CAN** (520886) **50/5** (546047) **70/15** (594385)	**Module 24/1.5** (536630) **Module 50/5** (534130) **Module 50/8** (504384) **Module 50/15** (504383)	**Micro 24/5 CAN** (638328)	**Disk 60/8 CAN** (688770) **Disk 60/12 CAN** (688775) **Disk 60/12 CAN SSC** (709859)
Pin 1 “CAN high”	B35 “CAN high”	A79 “CAN high”	Pin 1 “CAN high”
Pin 2 “CAN low”	B36 “CAN low”	A80 “CAN low”	Pin 2 “CAN low”
Pin 3 “CAN GND”	B37 “CAN GND”	A76 “CAN GND”	Pin 6 “CAN V+”
Pin 4 “CAN shield”	–	–	Pin 7 “CAN GND”
 Connector	Pin terminals	Pin terminals	 Connector

Table 5-38 CANopen basic information | CAN bus wiring – Controller

CAN BUS LINE

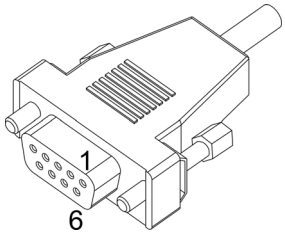
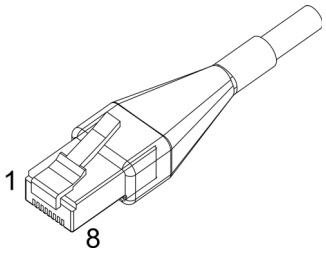
CAN 9 Pin D-Sub (DIN41652) on PLC or PC/CAN Interface	CAN RJ45 on PLC or PC/CAN Interface
Pin 7 “CAN_H” high bus line	Pin 1 “CAN_H” high bus line
Pin 2 “CAN_L” low bus line	Pin 2 “CAN_L” low bus line
Pin 3 “CAN_GND” Ground	Pin 3 “CAN_GND” Ground Pin 7 “CAN_GND” Ground
Pin 5 “CAN_Shield” Cable shield	Pin 6 “CAN_Shield” Cable shield
 D-Sub Connector	 RJ45 Connector

Table 5-39 CANopen basic information | CAN bus wiring – CAN Bus Line

5.3.3 Step 3: CAN Node ID

Generally applicable Rules

- *An unique Node ID must be defined for all devices within the CAN network.*
- *The Node ID results in the summed values of the stated DIP switches set to "1" (ON) or the connected input lines, respectively. The address can be coded using binary code.*
- *By setting all stated DIP switches to "0" (OFF) – or by letting the input lines open, respectively – the Node IDs may be configured by software (changing the object "Node ID"). In this case, the number of addressable nodes is 127.*

5.3.3.1 EPOS4 Compact CAN / EPOS4 50/5 & EPOS4 70/15 (DIP Switch 1…7, Addresses 1…31)

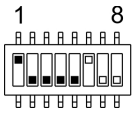
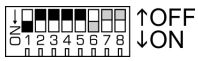
Switch	Binary Code	Valence	DIP Switch	
1	2^0	1	 EPOS4 Compact CAN	 EPOS4 50/5 EPOS4 70/15
2	2^1	2		
3	2^2	4		
4	2^3	8		
5	2^4	16		

Table 5-40 CANopen basic information | Node ID (1)

EXAMPLES

Use the following table as a (non-exhaustive) guide:

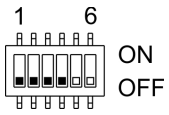
CAN ID/Switch	1	2	3	4	5	
Valence	1	2	4	8	16	
Node ID						**Calculation**
1	1	0	0	0	0	1
2	0	1	0	0	0	2
16	0	0	0	0	1	16
31	1	1	1	1	1	1 + 2 + 4 + 8 + 16

Table 5-41 CANopen basic information | DIP switch 1…5 settings (example)

5.3.3.2 EPOS4 Disk CAN

Setting the ID by DIP switch SW1 and solder pads JP301, JP 302

- *By setting the DIP switch (1…4) address 0 (“OFF”), the ID may be set by software (object 0x2000 «Node-ID», range 1…127).*
- *The ID results in the summed values of DIP switch addresses 1 (“ON”) and shorted solder pads JP301 and JP 302.*

Switch	Binary Code	Valence	DIP Switch
1	2^0	1	 EPOS4 Disk CAN
2	2^1	2	
3	2^2	4	
4	2^3	8	
JP301 closed*	2^4	16	
JP302 closed*	2^5	32	

* the default setting for the solder pads JP301, JP302 is “open”

Table 5-42 CANopen basic information | Node ID (2)

EXAMPLES

Use the following table as a (non-concluding) guide.

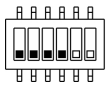
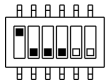
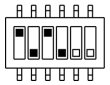
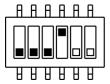
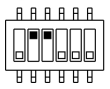
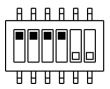
Setting	Switch 1	2	3	4	Solder pad JP301	JP302	ID
	0	0	0	0	0	0	–
	**1**	0	0	0	0	0	1
	**1**	0	**1**	0	0	0	5
	0	0	0	**1**	0	0	8
	0	**1**	**1**	0	**1**	**1**	54
	**1**	**1**	**1**	**1**	**1**	**1**	63
0 = Switch “OFF” / solder pad “open” 1 = Switch “ON” / solder pad “shorted”							

Table 5-43 CANopen basic information | DIP switch 1…4 and solder pad settings (example)

5.3.4 Step 4: CAN Communication

For EPOS4, following CAN bit rates are available:

Object “CAN Bitrate” (Index 0x2001, Subindex 0x00)	Bit rate	Max. Line Length according to CiA 102
0	1 MBit/s	25 m
1	800 kBit/s	50 m
2	500 kBit/s	100 m
3	250 kBit/s	250 m
4	125 kBit/s	500 m
(5)	reserved	–
6	50 kBit/s	1000 m
7	20 kBit/s	2500 m
(8)	not supported (10 kBit/s)	–
9	automatic bit rate detection	–

Table 5-44 CANopen basic information | CAN communication – Bit rates and line lengths

Note

- *All devices within the CAN bus must use the same bit rate.*
- *If “automatic bit rate detection” is in use, at least one CANopen device (e.g. CANopen master) must be present in the network with a fixed defined CAN bit rate configuration.*
- *The CANopen bus’ maximum bit rate depends on the cable length. Use «EPOS Studio» to configure bit rate by writing object “CAN Bit rate” (Index 0x2001, Subindex 0x00).*

5.3.5 Step 5: Activate Changes

Activate changes by saving and resetting the EPOS4 using «EPOS Studio».

1) Execute menu item ¤Save All Parameters¤.
2) Select context menu item ¤Reset Node¤ of the selected node.

5.3.6 Step 6: Communication Test

Use a CAN monitor program (supported by PC's or PLC CAN interface's manufacturer) to check wiring and configuration:

1) Reset all EPOS4 devices in the bus.
2) Upon power on, the EPOS4 will send a boot up message.
3) Make sure that all connected devices send a boot up message. If not, EPOS will produce a «CAN passive mode error» (0x8120).
4) Boot up message:
   COB-ID = 0x700 + Node ID
   Data [0] = 0x00

As an example, the figure below shows the incoming message on the CAN bus (EPOS4 Node ID = 1) displayed by a CAN monitor supplied by IXXAT.

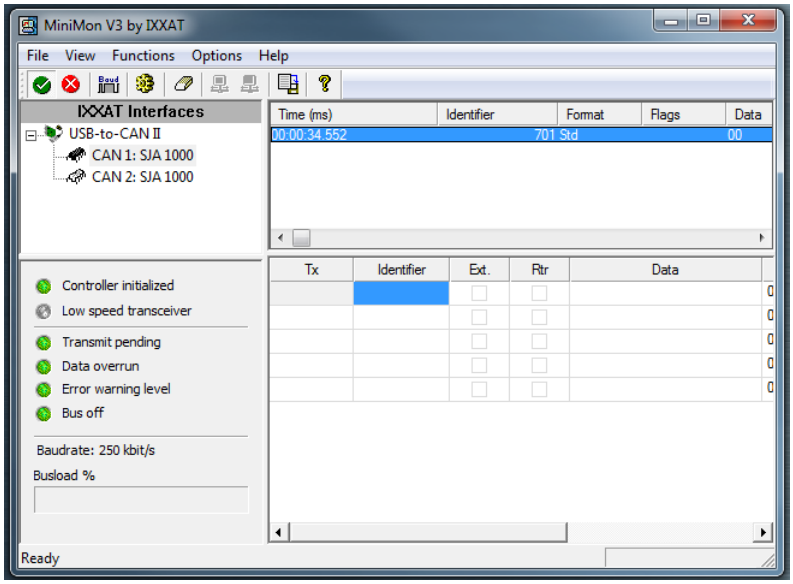


Figure 5-31 CANopen basic information | Example: Boot up message of node 1

5.4 SDO Communication

A **Service Data Object (SDO)** reads from/writes to entries of the Object Dictionary. The SDO transport protocol allows transmission of objects of any size. SDO communication can be used to configure the EPOS4's object.

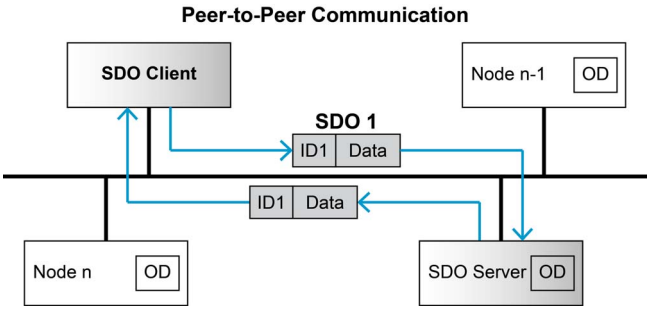


Figure 5-32 CANopen basic information | SDO communication

Two different transfer types are supported:

- Normal transfer: A segmented SDO protocol used to read/write objects larger 4 bytes. This means that the transfer is split into different SDO segments (CAN frames).
- Expedited transfer: A non-segmented SDO protocol, used for objects smaller 4 bytes.

Almost all EPOS4 Object Dictionary entries can be read/written using the non-segmented SDO protocol (expedited transfer). Only the data recorder buffer must be read using the segmented SDO protocol (normal transfer). Thus, only non-segmented SDO protocol will be further explained. For details on the segmented protocol (normal transfer) ➔CANopen specification (CiA 301).

5.4.1 Expedited SDO Protocol

In the subsequent description, the terms will be used as follows:

- «**Client**» refers to the CANopen master reading or writing an object
- «**Server**» refers to the EPOS4 (or any other CANopen slave) which reacts on the request

READING OBJECT (= SDO UPLOAD)

Client => Server

COB-ID	Data [Byte 0]	Data [Byte 1]	Data [Byte 2]	Data [Byte 3]	Data [Byte 4]	Data [Byte 5]	Data [Byte 6]	Data [Byte 7]
0x600 + Node-ID		Index LowByte	Index HighByte	Sub-Index	Reserved			

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	X	X	X	X	X

Server => Client

COB-ID	Data [Byte 0]	Data [Byte 1]	Data [Byte 2]	Data [Byte 3]	Data [Byte 4]	Data [Byte 5]	Data [Byte 6]	Data [Byte 7]
0x580 + Node-ID		Index LowByte	Index HighByte	Sub-Index	Object Byte 0	Object Byte 1	Object Byte 2	Object Byte 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	X	n		e	s

Figure 5-33 CANopen basic information | SDO upload protocol (expedited transfer) – Read

WRITING OBJECT (= SDO DOWNLOAD)

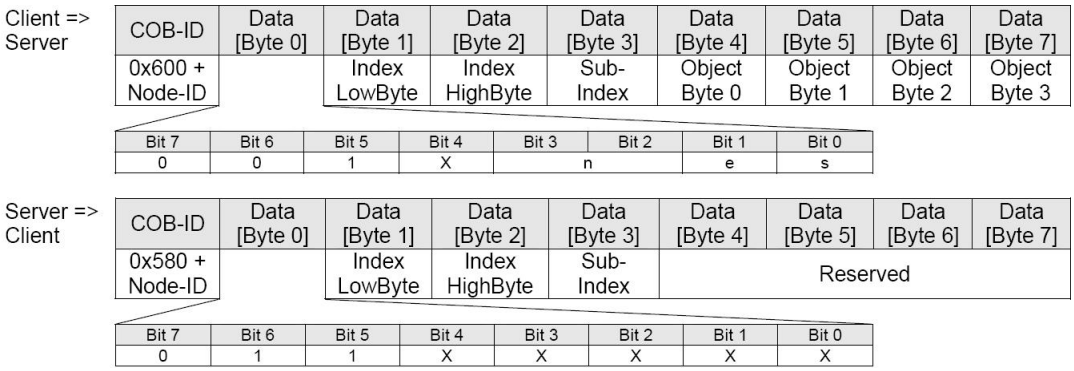


Figure 5-34 CANopen basic information | SDO upload protocol (expedited transfer) – Write

ABORT SDO PROTOCOL (IN CASE OF ERROR)

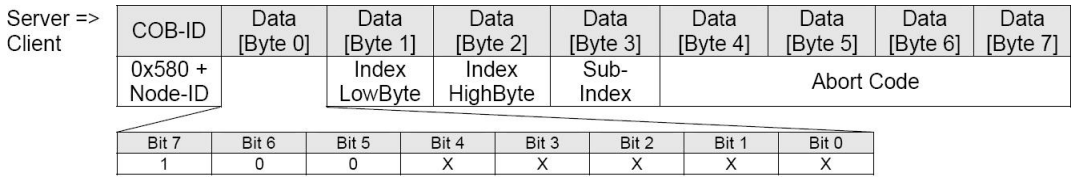


Figure 5-35 CANopen basic information | SDO upload protocol (expedited transfer) – Abort

Note
For detailed descriptions of "Abort Codes" ➔FwSpec.

Legend Data [Byte 0]	
ccs	client command specifier (Bit 7…5) Read Object: ccs = 2 / Write Object: ccs = 1
scs	server command specifier (Bit 7…5) Read Object: scs = 2 / Write Object: scs = 3
cs	command specifier (Bit 7…5) SDO abort transfer: cs = 4
X	not used (always "0")
n	Only valid if e = 1 and s = 1, otherwise 0. If valid, it indicates the number of bytes in Data [Byte 4…7] that do not contain data. Bytes [8 − n, 7] do not contain segment data.
e	Transfer type (0: normal transfer; 1: expedited transfer)
s	Size indicator (0: data set size is not indicated; 1: data set size is indicated)

Table 5-45 CANopen basic information | SDO transfer protocol – Legend

OVERVIEW ON IMPORTANT COMMAND SPECIFIER ([BYTE 0] → BIT 7...5)

Type	Length	Sending Data [Byte 0]	Receiving Data [Byte 0]
Reading Object	1 Byte	40	4F
	2 Byte	40	4B
	3 Byte	40	43
Writing Object	1 Byte	2F (or 22)	60
	2 Byte	2B (or 22)	60
	4 Byte	23 (or 22)	60
	not defined	22	60

Table 5-46 CANopen basic information | Command specifier (overview)

5.4.2 SDO Communication Examples

Read «Statusword» (Index 0x6041, Subindex 0x00) from node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x40	ccs = 2
Data [1]	0x41	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x4B	scs = 2, n = 2, e = 1, s = 1
Data [1]	0x41	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0x08	Data [Byte 0]
Data [5]	0x00	Data [Byte 1]
Data [6]	0x00	reserved
Data [7]	0x00	reserved

Statusword: 0x0008 = 8

Table 5-47 CANopen basic information | Example “Read Statusword”

Write «Controlword» (Index 0x6040, Subindex 0x00: Data 0x000F) to node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x22	ccs = 1, n = 0, e = 1, s = 0
Data [1]	0x40	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0x0F	Data [Byte 0]
Data [5]	0x00	Data [Byte 1]
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x60	scs = 3
Data [1]	0x40	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

Controlword: new value

Table 5-48 CANopen basic information | Example “Write Controlword”

Try to read the content of an object's subindex which does not exist (Index 0x2000, Subindex 0x08) from node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x40	ccs =2
Data [1]	0x00	Index LowByte
Data [2]	0x20	Index HighByte
Data [3]	0x08	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x80	scs = 4
Data [1]	0x00	Index LowByte
Data [2]	0x20	Index HighByte
Data [3]	0x08	Subindex
Data [4]	0x11	Abort Code [Byte 0]
Data [5]	0x00	Abort Code [Byte 1]
Data [6]	0x09	Abort Code [Byte 2]
Data [7]	0x06	Abort Code [Byte 3]

Abort code: 0x06090011 → the last read or write command had a wrong object subindex.

Table 5-49 CANopen basic information | Example “Read non-existent subindex”

Read «Position actual value» (Index 0x6064, Subindex 0x00) from node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x40	ccs = 2
Data [1]	0x64	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x43	scs = 2, n = 0, e = 1, s = 1
Data [1]	0x64	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0xCA	Data [Byte 0]
Data [5]	0x04	Data [Byte 1]
Data [6]	0x00	Data [Byte 2]
Data [7]	0x00	Data [Byte 3]

Position actual value: 0x000004CA = 1226

Table 5-50 CANopen basic information | Example “Read Position actual value”

Write «Target position» (Index 0x607A, Subindex 0x00: Data 0x000008AE → 2222dec) to node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x22	ccs = 1, n = 0, e = 1, s = 0
Data [1]	0x7A	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0xAE	Data [Byte 0]
Data [5]	0x08	Data [Byte 1]
Data [6]	0x00	Data [Byte 2]
Data [7]	0x00	Data [Byte 3]

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x60	scs = 3
Data [1]	0x7A	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex
Data [4]	0x00	Abort Code [Byte 0]
Data [5]	0x00	Abort Code [Byte 1]
Data [6]	0x00	Abort Code [Byte 2]
Data [7]	0x00	Abort Code [Byte 3]

Target position: new value

Table 5-51 CANopen basic information | Example “Write Target position”

5.4.3 EPOS Studio Command Analyzer

If you connect the «EPOS Studio» via CANopen to the EPOS4, it is possible to process a CANopen command using the «EPOS Studio» tool «Command Analyzer». Thereby, a CANopen command can be executed and the sent and received data packets will be recorded.

The «Command Analyzer» is based on maxon's DLL whereby the commands are arranged similarly to the maxon Library. Thereby…

- SDOs can be sent
- PDOs cannot be processed

1) Connect «EPOS Studio» and the EPOS4 via CAN.

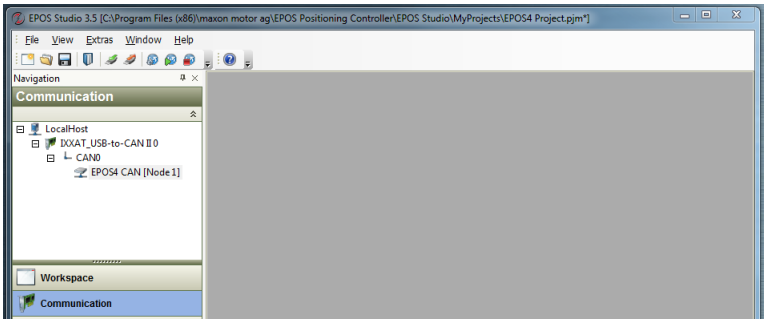


Figure 5-36 CANopen basic information | Connect EPOS4

2) Open the tool ¤Command Analyzer¤.
3) Select ¤Interface¤ from the drop-down list ¤Layer Filter¤.

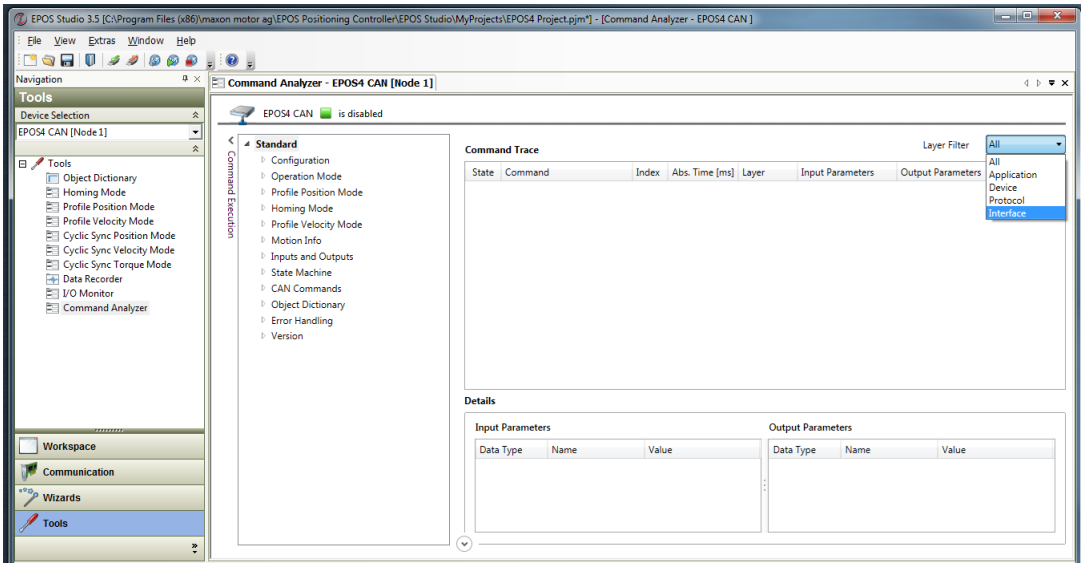


Figure 5-37 CANopen basic information | Select interface layer

COMMAND EXAMPLES

- Read Object GetObject(): Read Position Data 0x6064 0x00 32Bit (4Byte)

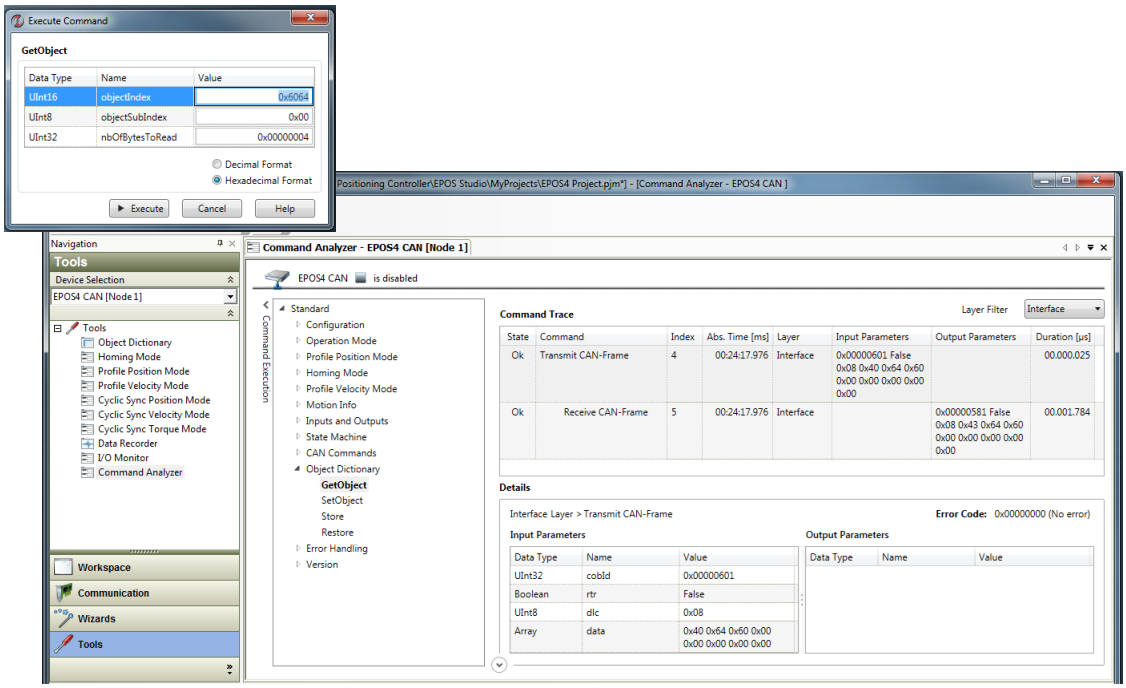


Figure 5-38 CANopen basic information | Command example – Read Object (= GetObject)

- Write Object SetObject(): Write Target Position 0x607A 0x00 32Bit (4Byte) 0x000008AE (2222d)

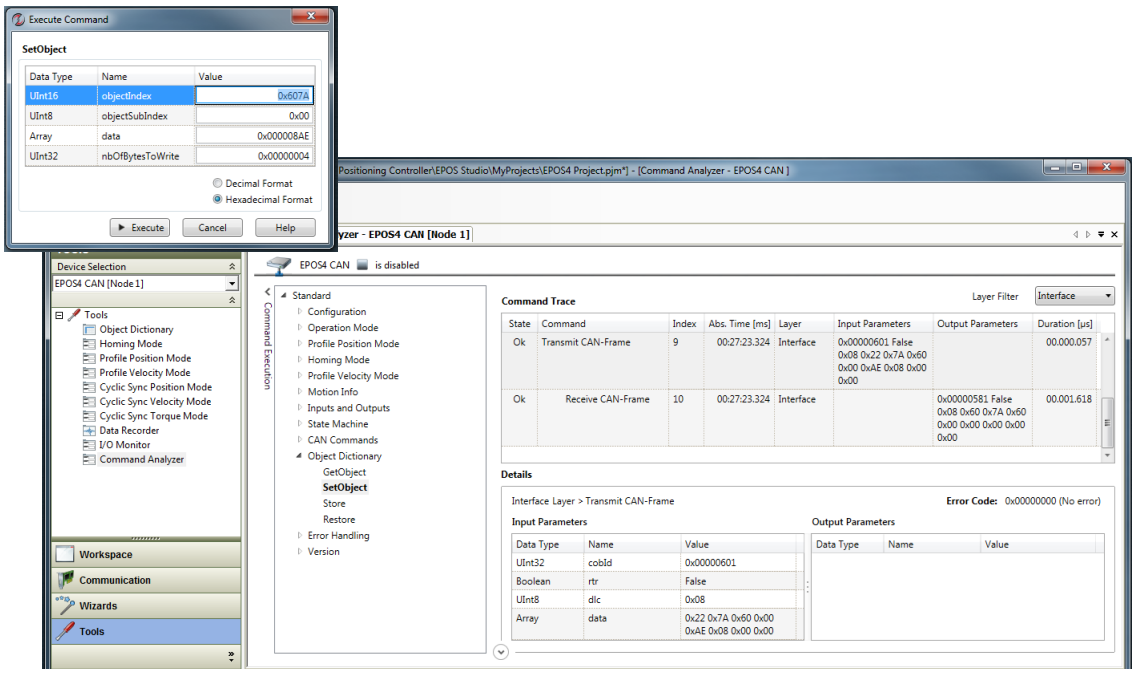


Figure 5-39 CANopen basic information | Command example – Write Object (= SetObject)

5.5 PDO Communication

Process Data Objects (PDOs) – unconfirmed services containing no protocol overhead – are used for fast data transmission (real-time data) with a high priority. Consequently, they represent an extremely fast and flexible method to transmit data from one node to any number of other nodes. PDOs may contain up to 8 data bytes that can be specifically compiled and confirmed to suit own requirements. Each PDO has a unique identifier and is transmitted by only one node, but it can be received by more than one (producer/ consumer communication).

The CANopen network management is node-oriented and follows a master/slave structure. It requires one device in the network, which serves as **NMT (Network Management)** Master. The other nodes are NMT Slaves.

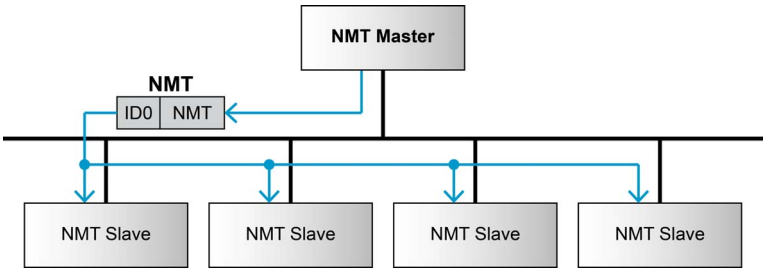


Figure 5-40 CANopen basic information | Network Management (NMT)

The CANopen NMT Slave devices implement a state machine that automatically brings every device to state «Pre-Operational», once powered and initialized. In this state, the node may be configured and parameterized via SDO (e.g. using a configuration tool), PDO communication is not permitted. Thus, to switch from «Pre-Operational» to «Operational», you will need to send the “Start Remote Node Protocol”. For detailed information on NMT Services see separate document ➔«EPOS4 Communication Guide».

Function	COB-ID	CS (Byte 0)	Node ID (Byte 1)	Functionality
Start Remote Node Protocol	0	0x01	0 (all)	All EPOS4 (all CANopen nodes) will enter NMT state «Operational»
	0	0x01	n	The EPOS4 (or CANopen node) with Node ID n will enter NMT state «Operational»
Enter Pre-Operational Protocol	0	0x80	0 (all)	All EPOS4 (all CANopen nodes) will enter NMT state «Pre-Operational»
	0	0x80	n	The EPOS4 (or CANopen node) with Node ID n will enter NMT state «Pre-Operational»
Enter Stopped Protocol	0	0x02	0 (all)	All EPOS4 (all CANopen nodes) will enter NMT state «Stopped»
	0	0x02	n	The EPOS4 (or CANopen node) with Node ID n will enter NMT state «Stopped»
Enter Reset Application Protocol	0	0x81	0 (all)	All EPOS4 (all CANopen nodes) will enter NMT state «Reset Application»
	0	0x81	n	The EPOS4 (or CANopen node) with Node ID n will enter NMT state «Reset Application»
Enter Reset Communi-cation Protocol	0	0x82	0 (all)	All EPOS4 (all CANopen nodes) will enter NMT state «Reset Communication»
	0	0x82	n	The EPOS4 (or CANopen node) with Node ID n will enter NMT state «Reset Communication»

Table 5-52 CANopen basic information | NMT functionality

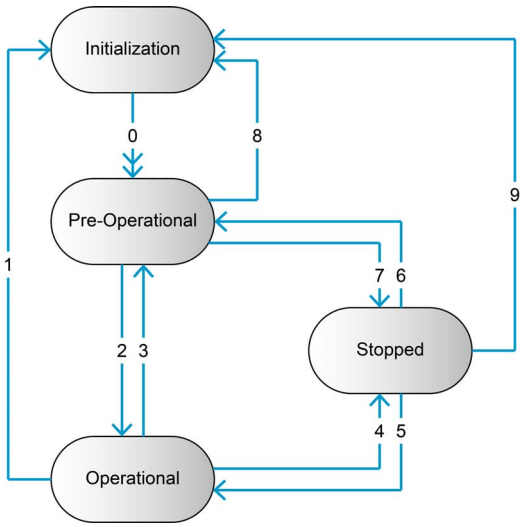


Figure 5-41 CANopen basic information | NMT slave state diagram

5.5.1 PDO Transmissions

PDO transmissions may be driven by remote requests, event triggered and actuated by Sync message received:

- Remotely requested:
  Another device may initiate the transmission of an asynchronous PDO by sending a remote transmission request (remote frame).
- Event triggered (only Transmit PDOs):
  An event of a mapped object (e.g. velocity changed) will cause the transmission of the TxPDO. Subindex 0x03 of object «Transmit PDO X parameter» contains the inhibit time, which represents the minimum interval for PDO transmission. The value is defined as a multiple of 100 us.
- Synchronous transmission:
  In order to initiate simultaneous sampling of input values of all nodes, a periodically transmitted Sync message is required. Synchronous PDO transmission takes place in cyclic and acyclic transmission mode. Cyclic transmission means that the node waits for the Sync message after which it sends its measured values. Its PDO transmission type number (1…240) indicates the Sync rate it listens to (the number of Sync messages the node waits before next transmission of its values). The EPOS supports only Sync rates of 1.

5.5.2 PDO Mapping

Default application objects' mapping as well as the supported transmission mode is described in the Object Dictionary for each PDO. PDO identifiers may have high priority to guarantee short response time. PDO transmission is not confirmed. PDO mapping defines the application objects to be transmitted within a PDO. It describes sequence and length of the mapped application objects. A device supporting variable mapping of PDOs must support this during the state «Pre-Operational». If dynamic mapping during state «Operational» is supported, the SDO Client is responsible for data consistency.

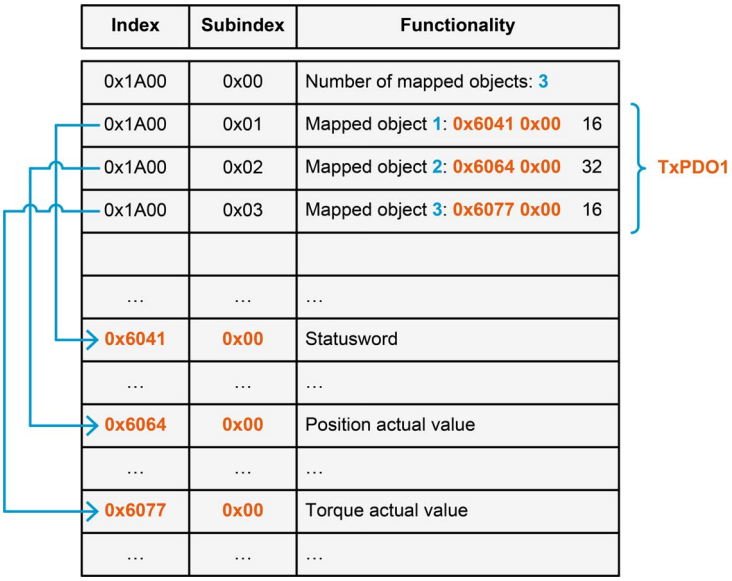


Figure 5-42 CANopen basic information | PDO mapping example

5.5.3 PDO Configuration

For PDO Configuration, the device must be in state «Pre-Operational»!

The following section will explain how a configuration must be implemented step-by-step. Use the CANopen wizard in «EPOS Studio» for PDO configuration as described. For each step, an example quotes "PDO 1" and "Node 1".

5.5.3.1 Step 1: Configure COB-ID

The default value of the COB-ID depends on the Node ID (Default COB-ID = PDO-Offset + Node ID). Otherwise, the COB-ID can be set in a defined range. Below table shows all default COB-IDs and their ranges:

Object	Index	Subindex	Default COB-ID Node 1
TxPDO 1	0x1800	0x01	0x181
TxPDO 2	0x1801	0x01	0x281
TxPDO 3	0x1802	0x01	0x381
TxPDO 4	0x1803	0x01	0x481
RxPDO 1	0x1400	0x01	0x201
RxPDO 2	0x1401	0x01	0x301
RxPDO 3	0x1402	0x01	0x401
RxPDO 4	0x1403	0x01	0x501

Table 5-53 CANopen basic information | COB-IDs – Default values and value range

Changed COB-IDs can be reset by “Restore Default PDO COB-IDs” using context menu of ¤CANopen Object Dictionary¤ view in «EPOS Studio»:

1) Connect «EPOS Studio» using ¤Wizards¤ \ ¤CANopen¤.

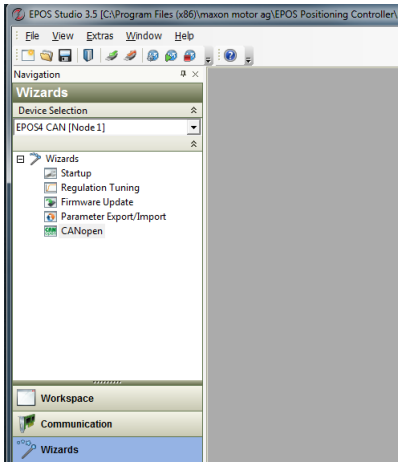


Figure 5-43 CANopen basic information | Start CANopen wizard

2) Select the receive PDOs by right click on ¤Receive PDOs¤ and select ¤Restore default COB-IDs¤.

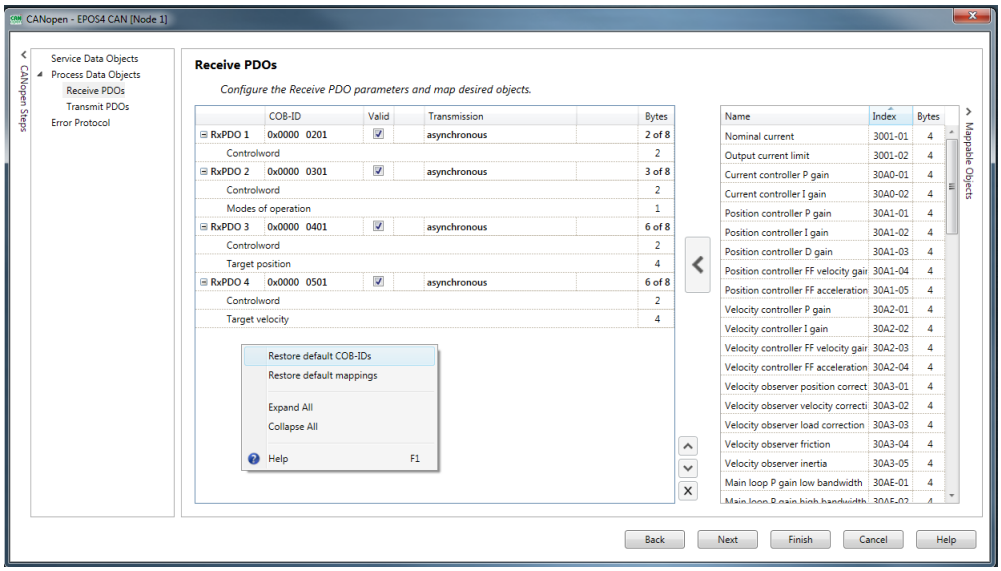


Figure 5-44 CANopen basic information | Receive PDOs: Restore default COB-IDs

3) Select the transmit PDOs by by right click on ¤Transmit PDOs¤ and select ¤Restore default COB-IDs¤.

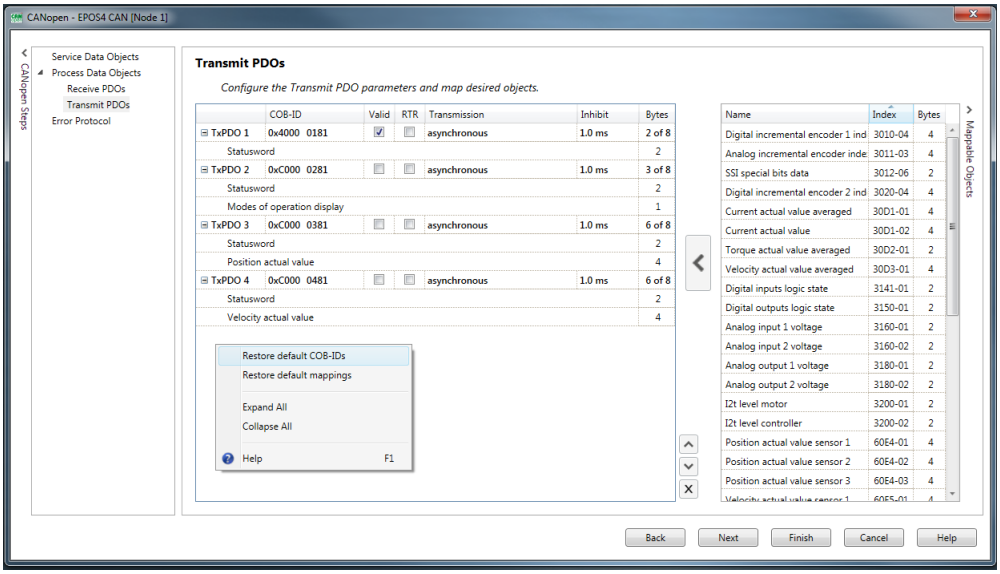


Figure 5-45 CANopen basic information | Transmit PDOs: Restore default COB-IDs

Example: Object → “COB-ID used by RxPDO 1” (Index 0x1400, Subindex 0x01):

Default COB ID RxPDO 1 = 0x200 + Node ID = 0x201

Example: Object → “COB-ID used by TxPDO 1” (Index 0x1800, Subindex 0x01):

Default COB ID TxPDO 1 = 0x180 + Node ID = 0x181

5.5.3.2 Step 2: Set Transmission Type

Type 0x01	TxPDOs	Data is sampled and transmitted after the occurrence of the SYNC.
	RxPDOs	Data is passed on to the EPOS4 and processed after the occurrence of the SYNC.
Type 0xFD	TxPDOs	Data is sampled and transmitted after the occurrence of a remote transmission request (RTR).
Type 0xFF	TxPDOs	Data is sampled and transmitted after one mapped object of the PDO has changed its value and the configured “Inhibit time” has been exceeded.
	RxPDOs	Data is transmitted (by the master to the EPOS4) asynchronously and then directly processed by the EPOS4.

Example: Object → «Transmission type RxPDO 1» (Index 0x1400, Subindex 0x02)
Value = 0x01

5.5.3.3 Step 3: Number of Mapped Application Objects

Disable the PDO by writing a value of “0” (zero) to the subindex 0x00 holding «Number of mapped objects in…».

Example: Object → «Number of mapped objects in RxPDO 1» (Index 0x1600, Subindex 0x00)
Value = 0x00 (i.e. this PDO is disabled)

Example: Object → «Number of mapped objects in TxPDO 1» (Index 0x1A00, Subindex 0x00)
Value = 0x00 (i.e. this PDO is disabled)

5.5.3.4 Step 4: Mapping Objects

Set value from an object.

Example: Object1 → «1st mapped object in RxPDO 1» (Index 0x1600, Subindex 0x01)
Object2 → «2nd mapped object in RxPDO 1» (Index 0x1600, Subindex 0x02)
Object3 → «3rd mapped object in RxPDO 1» (Index 0x1600, Subindex 0x03)

RxPDO 1	#	Mapped Object	
	1	Object_1 = 0x60400010	→ Controlword (16 Bit)
	2	Object_2 = 0x607A0020	→ Target position (32 Bit)
	3	Object_3 = 0x31820110	→ Analog output 1 value (16 Bit)

Note
For details on all mappable objects ➔FwSpec, chapters “Receive PDO… parameter” and “Transmit PDO… parameter”.

5.5.3.5 Step 5: Number of mapped Application Objects

Enable PDO by writing the value of the number of objects in object «Number of mapped objects in…».

Example: Object → «Number of mapped objects in RxPDO 1» (Index 0x1600, Subindex 0x00)

Example: Object → «Number of mapped objects in TxPDO 1» (Index 0x1A00, Subindex 0x00)

5.5.3.6 Step 6: Activate Changes

Changes will directly be activated.

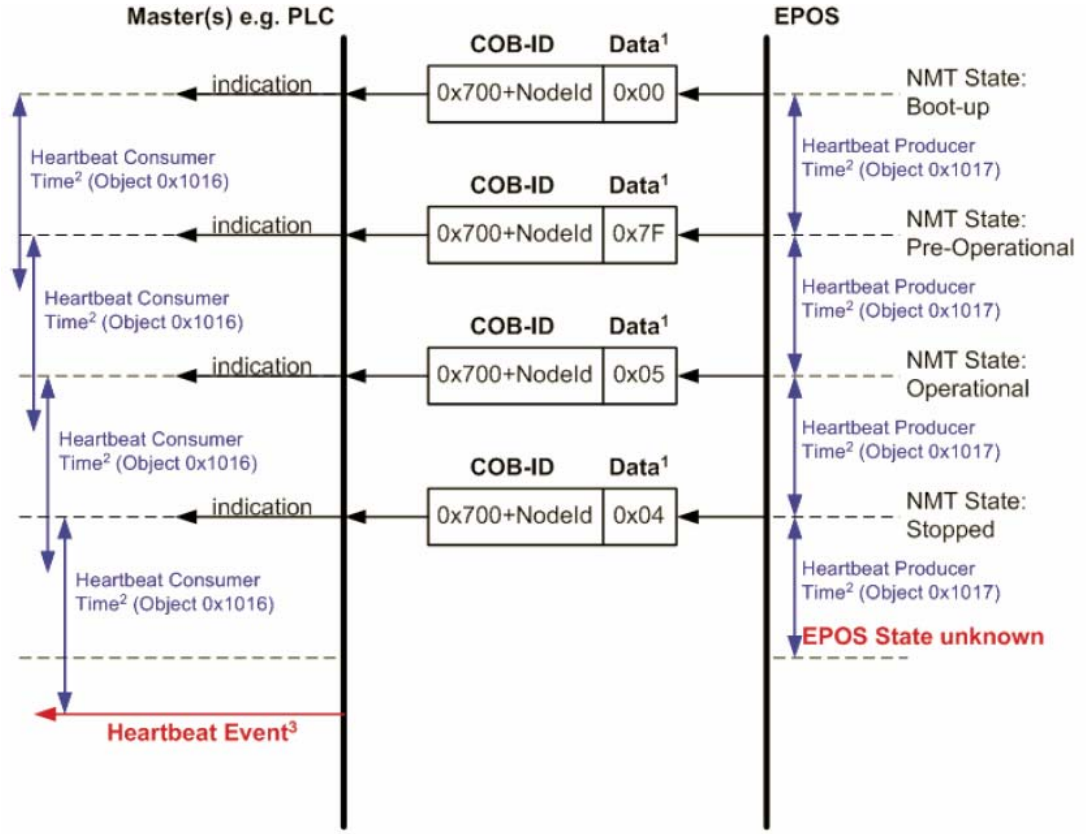
Execute menu item ¤Save All Parameters¤ in the context menu of the used node («EPOS Studio» \ Navigation Window \ Workspace or Communication) or in the context menu in the view “Object Dictionary”.

5.6 Heartbeat Protocol

The EPOS4 transmits a cyclic heartbeat message if the Heartbeat Protocol is enabled (Heartbeat Producer Time 0 = Disabled / greater than 0 = enabled). The Heartbeat Consumer guards receipt of the Heartbeat within the Heartbeat Consumer Time. If the Heartbeat Producer Time is configured in EPOS4, it will start immediately with the Heartbeat Protocol.

Remark

If «Automatic bite rate detection» is activated (Object 0x2001), a couple of frames must be sent first by the Master System for the EPOS4 to synchronize to this bit rate. Only then EPOS4 will start to send the Heartbeat Signal.



Legend: 1) Data Field / 2) Heartbeat Producer and Heartbeat Consumer Time / 3) Hearbeat Event

Figure 5-46 CANopen basic information | Heartbeat protocol – Timing diagram

DATA FIELD

Holds the NMT state. Therefore the following values for the data field are possible:

Value	EPOS4 NMT state
0x00	Bootup
0x04	Stopped
0x05	Operational
0x7F	Pre-Operational

Table 5-54 CANopen basic information | Heartbeat protocol – Data field

HEARTBEAT PRODUCER TIME AND HEARTBEAT CONSUMER TIME

The Heartbeat Consumer Time must be longer than the Heartbeat Producer Time because of generation, sending and indication time ($HeartbeatConsumerTime \geq HeartbeatProducerTime + 20ms$). Each indication of the Master resets the Heartbeat Consumer Time.

HEARTBEAT EVENT

If EPOS4 is in an unknown state (e.g. supply voltage failure), the Heartbeat Protocol cannot be sent to the Master. The Master will recognize this event upon elapsed Heartbeat Consumer Time and will generate a Heartbeat Event.

6 ETHERCAT COMMUNICATION & MASTER INTEGRATION

CONTENTS

6.1 In Brief

OBJECTIVE

The present application note explains the necessary Firewall settings to operate EPOS4 devices with «EPOS Studio» and how to integrate them into different EtherCAT Master Environments.

Note

*To operate within an EtherCAT network, some EPOS4 controllers (marked ** in below list) require the optionally available «EPOS4 EtherCAT Card» (581245).*

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0140h	Firmware Specification Communication Guide
EPOS4 Disk 60/8 EtherCAT	688772	0170h or higher	
EPOS4 Disk 60/12 EtherCAT EPOS4 Disk 60/12 EtherCAT SSC	688777 709862	0170h or higher 0170h or higher	
EPOS4 Module 24/1.5** EPOS4 Compact 24/1.5 EtherCAT	536630 628092	0140h or higher 0150h or higher	
EPOS4 Module 50/5** EPOS4 Compact 50/5 EtherCAT	534130 628094	0140h or higher 0150h or higher	
EPOS4 Module 50/8** EPOS4 Compact 50/8 EtherCAT	504384 605298	0140h or higher 0140h or higher	
EPOS4 Module 50/15** EPOS4 Compact 50/15 EtherCAT	504383 605299	0140h or higher 0140h or higher	
EPOS4 50/5 **	546047	0140h or higher	
EPOS4 70/15**	594385	0140h or higher	

Table 6-55 EtherCAT integration | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.4 or higher
	for zub's MACS Multi-Axis EtherCAT Masters ➔"Required Tools" on page 6-96

Table 6-56 EtherCAT integration | Recommended tools

6.2 Setup of Windows Defender Firewall for EtherCAT Communication

This section describes how to setup the Windows Defender Firewall for EtherCAT communication.

For a Windows application to communicate with an EtherCAT network, the «Windows Defender Firewall» needs to be configured to allow sending and receiving network traffic. The required rules for an application to communicate via a EtherCAT network adapter are as follows:

- Allow inbound TCP traffic
- Allow inbound UDP traffic
- Allow outbound TCP traffic
- Alllow outbound UDP traffic

You can manually create the required rules or refer to your IT department if you do not have the required permissions.

6.2.1 Add a new Firewall Rule

1) Open the «Windows Defender Firewall Control Panel».

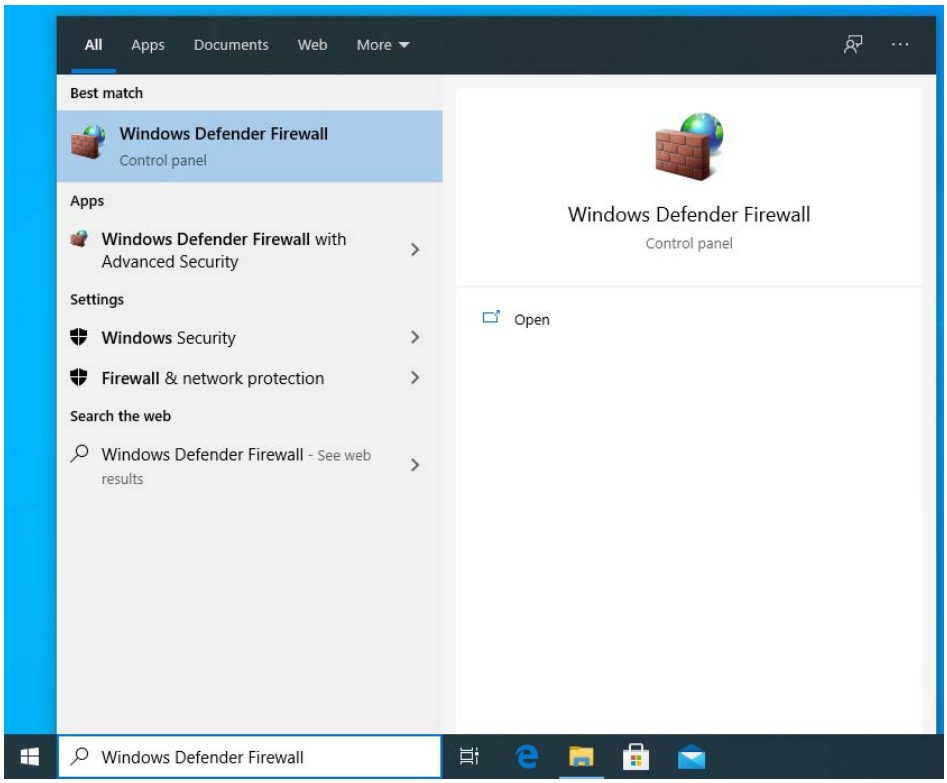


Figure 6-47 EtherCAT integration – Firewall setup | Windows Defender Firewall Control Panel

2) Click ¤Allow an app or feature trough Windows Defender Firewall¤.

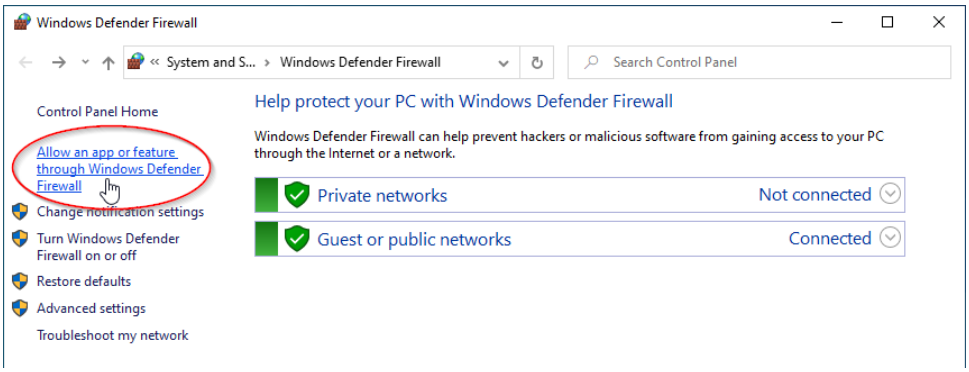


Figure 6-48 EtherCAT integration – Firewall setup | Allow passage

3) Click ¤Change settings¤.

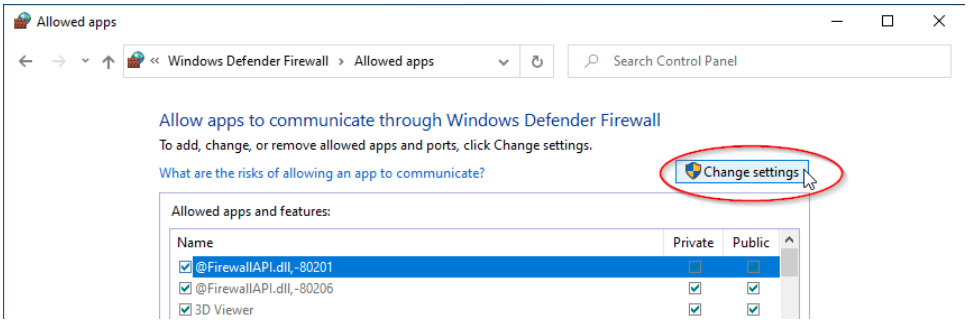


Figure 6-49 EtherCAT integration – Firewall setup | Change settings

4) Click ¤Allow another app…¤.

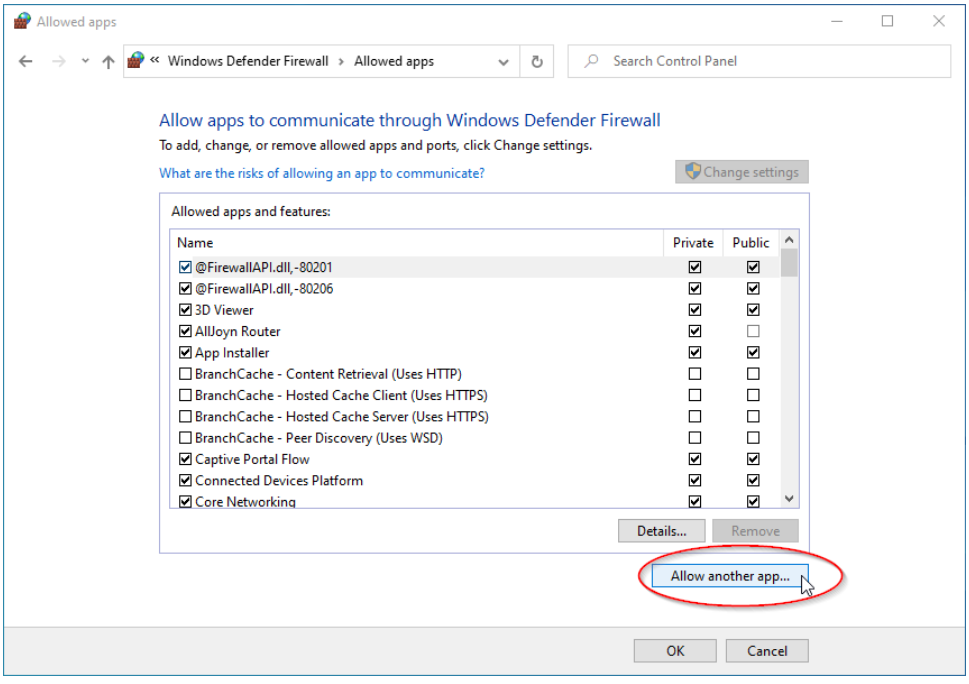


Figure 6-50 EtherCAT integration – Firewall setup | Allow app to communicate

5) Click ¤Browse…¤.

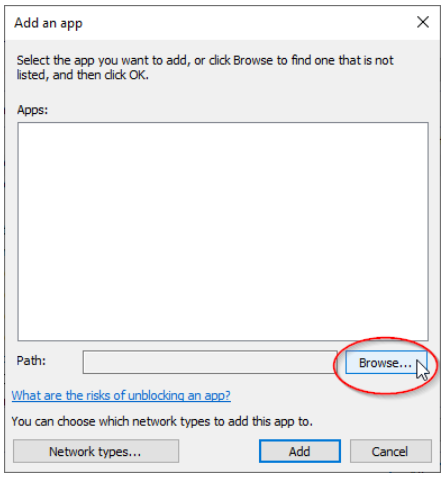


Figure 6-51 EtherCAT integration – Firewall setup | Browse

6) Browse for the application executable (e.g. «EPOS Studio.exe»).
Click ¤Open¤.

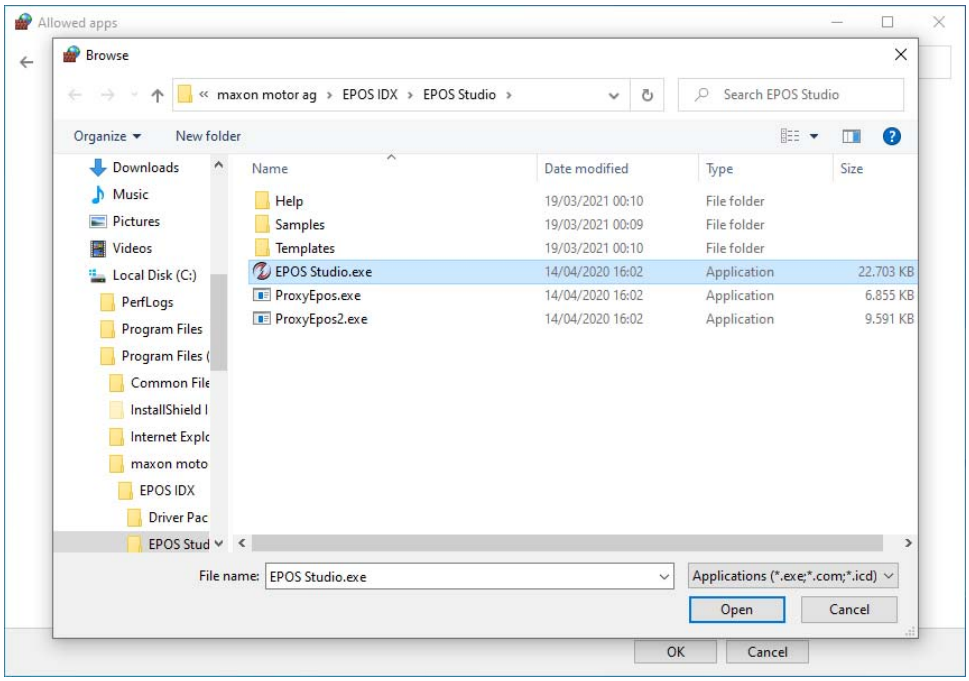


Figure 6-52 EtherCAT integration – Firewall setup | Select executable

7) Click ¤Network types…¤.

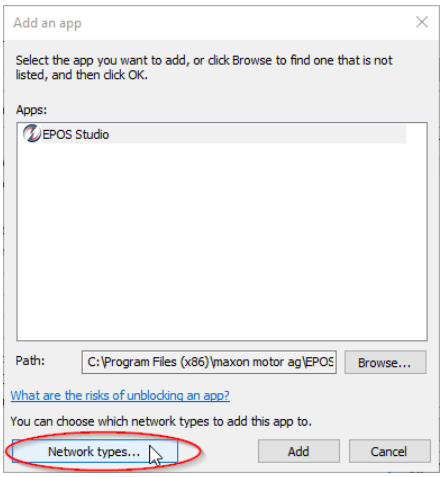


Figure 6-53 EtherCAT integration – Firewall setup | Select network types

8) Check both “Private” and “Public”.
Click ¤OK¤.

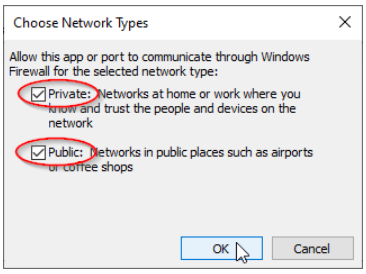


Figure 6-54 EtherCAT integration – Firewall setup | Choose network types

9) Click ¤Add¤.

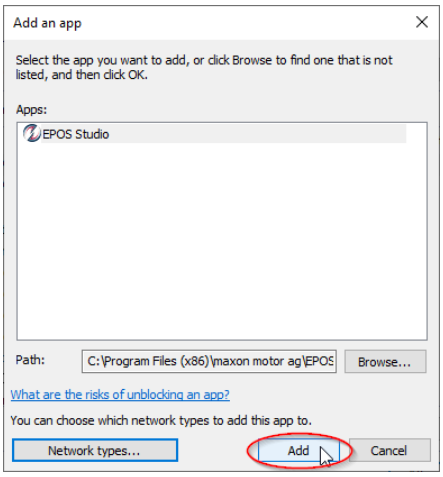


Figure 6-55 EtherCAT integration – Firewall setup | Add network types

10) Verify that your App has been added to the list.
Click ¤OK¤

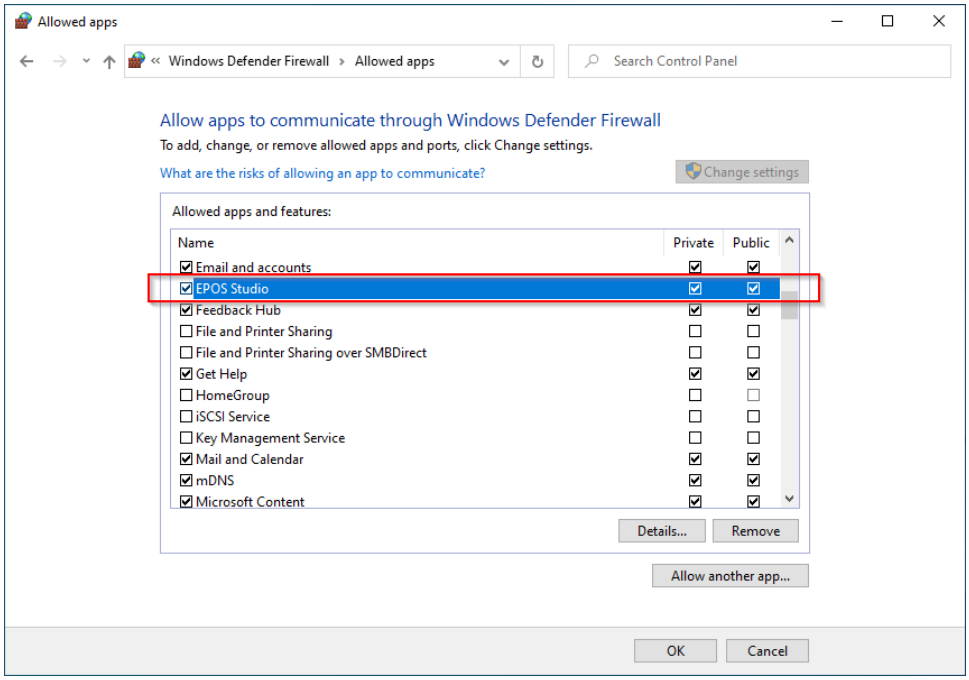


Figure 6-56 EtherCAT integration – Firewall setup | Check app list

6.2.2 Modify an existing Firewall Rule

Any possibly misconfigured firewall rules might prevent the application from communication with the EtherCAT network. Modify existing rules to “allow” traffic on your network adapter’s profile.

11) Open the «Windows Defender Firewall».

12) Click ¤Advanced settings¤

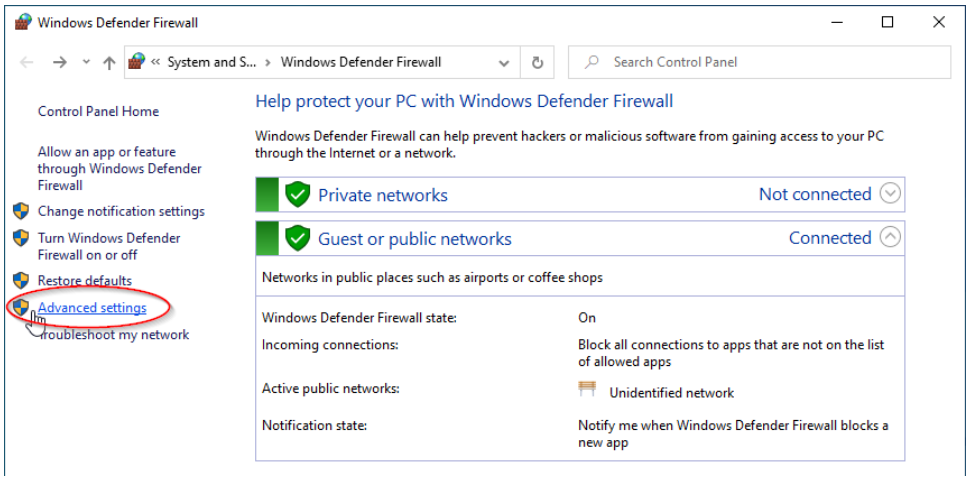


Figure 6-57 EtherCAT integration – Firewall setup | Advanced settings

13) Click ¤Inbound rules¤

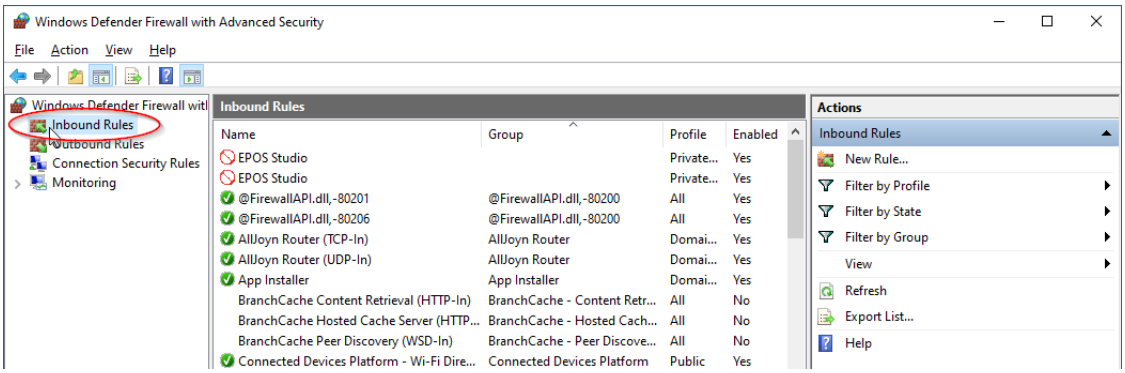


Figure 6-58 EtherCAT integration – Firewall setup | Inbound rules

14) Find any entries for your application (e.g. «EPOS Studio») by verifying the “Program” path and verify their settings.
If the action is set to “Block” or if the profile does not match the EtherCAT network adapter, double-click the rule to modify it.

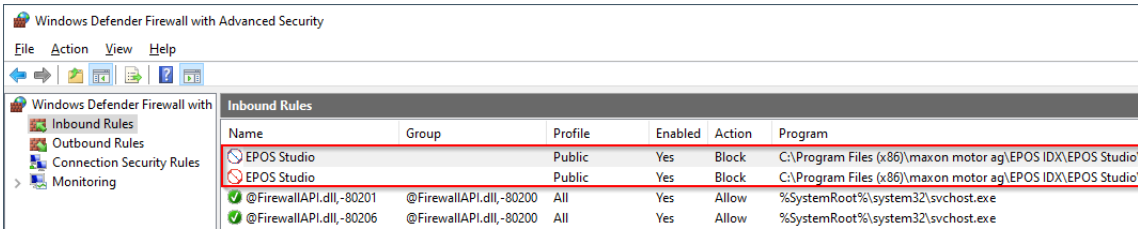


Figure 6-59 EtherCAT integration – Firewall setup | Set rules

15) Select the ¤General¤ tab.
Change the action from “Block the connection” to “Allow the connection”.
Click ¤OK¤.

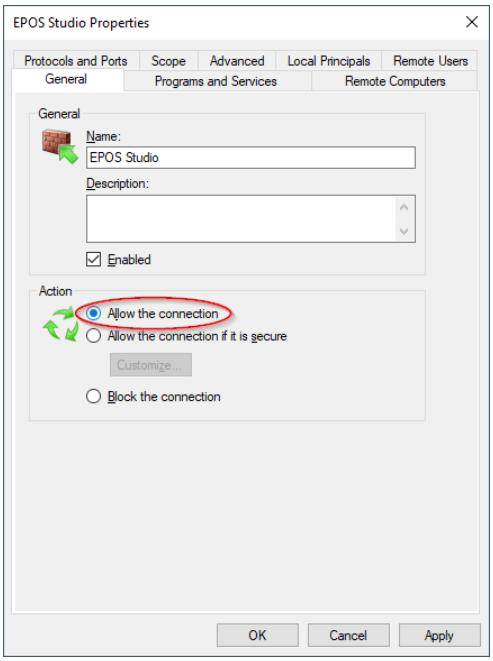


Figure 6-60 EtherCAT integration – Firewall setup | Allow connection

16) Select the ¤Advanced¤ tab.
Check both “Private” and “Public”.
Click ¤OK¤.

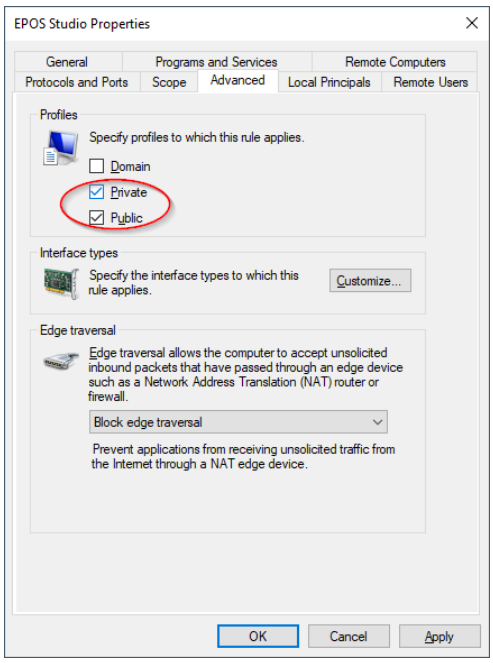


Figure 6-61 EtherCAT integration – Firewall setup | Set rules

17) Check for any blocking outbound rules and correct them if necessary by repeating the above steps 14 through 16.

6.3 Beckhoff TwinCAT Integration

6.3.1 Integrating ESI Files

To integrate an EPOS4 EtherCAT axis into the Beckhoff Master System, copy the ESI (EtherCAT Slave Information) XML file to the following folder. Note that the actual folder designation (***) depends on the TwinCAT version you are using:

- For **TwinCAT XAE** use path “C:\TwinCAT***3.1\Config\Io\EtherCAT\”.
- For **TwinCAT2** use path “C:\TwinCAT\Io\EtherCAT\”.

6.3.2 How to export the ESI File

You can export the ESI file using the «EPOS Studio»:

1) Make sure that the EPOS4 is connected and Online.
2) In the Object Dictionary, click right to open the context menu, then select ¤Export ESI File¤.

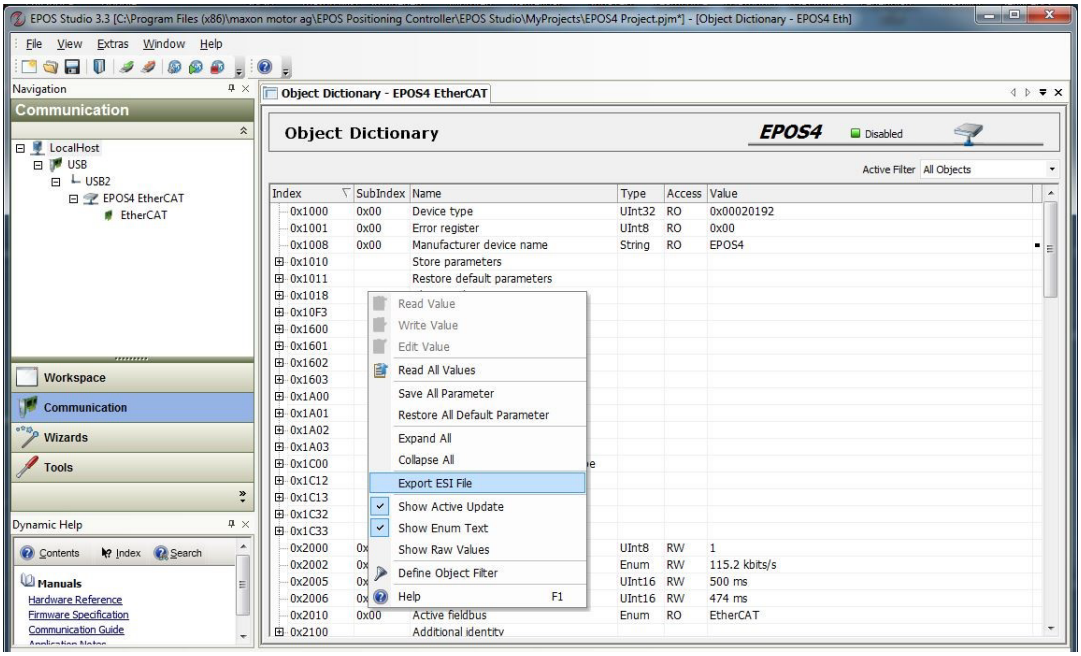


Figure 6-62 EtherCAT integration – Beckhoff TwinCAT | Export ESI file

6.3.3 Scanning the EtherCAT Slave Device

1) Connect the EPOS4 to the EtherCAT Master and turn on power.
2) Open the Beckhoff System Manager and create a new project using menu ¤File¤, then ¤New¤.
3) Open menu ¤Options¤, then select ¤Show Real Time Ethernet Compatible Devices¤.

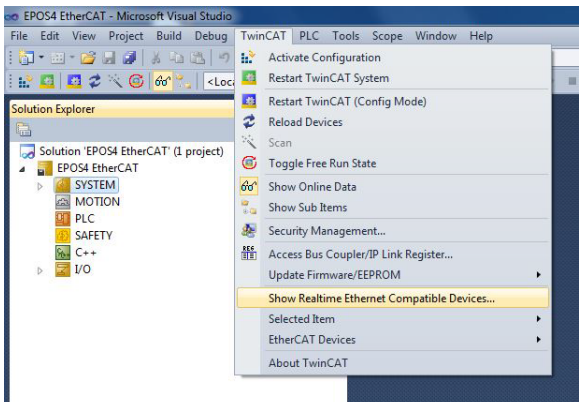


Figure 6-63 EtherCAT integration – Beckhoff TwinCAT | Create new project

4) If “Installed and ready to use devices” does not list a network card, you will need to install the EtherCAT driver for one of the present network cards.

a) Click one of the listed network cards.

b) Click ¤Install¤.

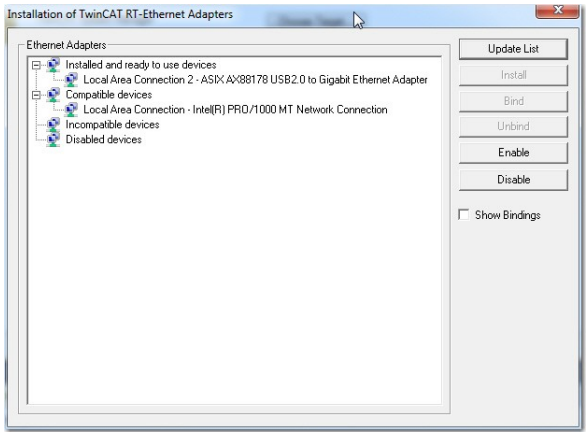


Figure 6-64 EtherCAT integration – Beckhoff TwinCAT | Install Ethernet adapters

5) In the TwinCAT System Manager navigation tree, click right on ¤I/O Devices¤, then select ¤Scan¤.

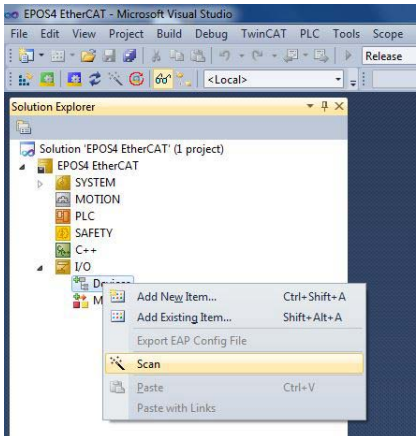


Figure 6-65 EtherCAT integration – Beckhoff TwinCAT | Scan devices

6) Click ¤OK¤ to confirm.

Figure 6-66 EtherCAT integration – Beckhoff TwinCAT | Confirmation

7) All detected E/A devices (network cards) will be listed.
   a) Tick to select the network card to which the EtherCAT devices are connected to and untick all others.
   b) Click ¤OK¤.

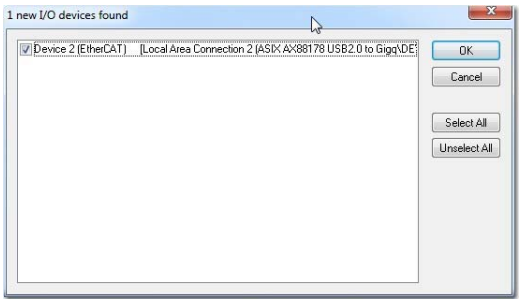


Figure 6-67 EtherCAT integration – Beckhoff TwinCAT | New I/O devices found

8) Click ¤YES¤ to confirm.

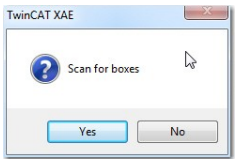


Figure 6-68 EtherCAT integration – Beckhoff TwinCAT | Scan for boxes confirmation

9) The TwinCAT System Manager now searches for connected devices. If one or more controller were found, the following message will appear.
   Click ¤Yes¤.

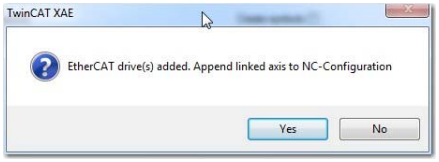


Figure 6-69 EtherCAT integration – Beckhoff TwinCAT | Add drives message

10) Make your selection depending on the intended use:
   • Click ¤Yes¤ if you plan to use the drive as a NC-Configuration
   • Click ¤No¤ if you do not plan to use the drive a NC-Configuration

11) Click ¤Yes¤ to confirm.

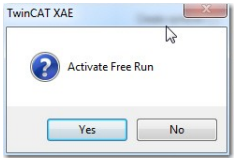


Figure 6-70 EtherCAT integration – Beckhoff TwinCAT | Activate free run message

12) Save the project.

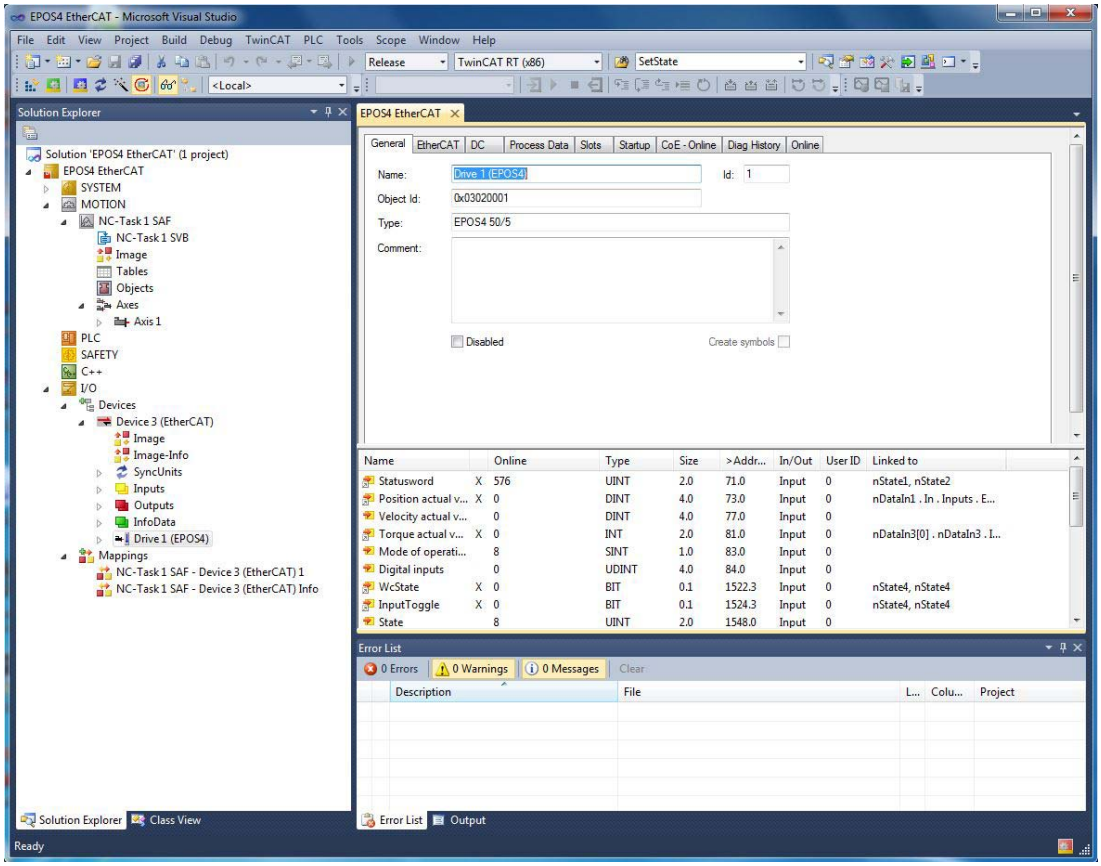


Figure 6-71 EtherCAT integration – Beckhoff TwinCAT | Save project

6.3.4 Configuration for commanding in a Cyclic Synchronous Mode

Via the EtherCAT interface, usually the following operating modes will be used:

- Cyclic Synchronous Position (CSP)
- Cyclic Synchronous Velocity (CSV)
- Cyclic Synchronous Torque (CST)

If you intend to operate the EPOS4 in Cycle Synchronous Mode, you will need to configure PDO Mapping accordingly by defining “Slots”.

Additionally, the following “regular” EPOS4 operating modes may be used:

- Profile Position Mode (PPM)
- Profile Velocity Mode (PVM)

1) Upon recognition of the involved axes, the structure tree will be displayed (example).

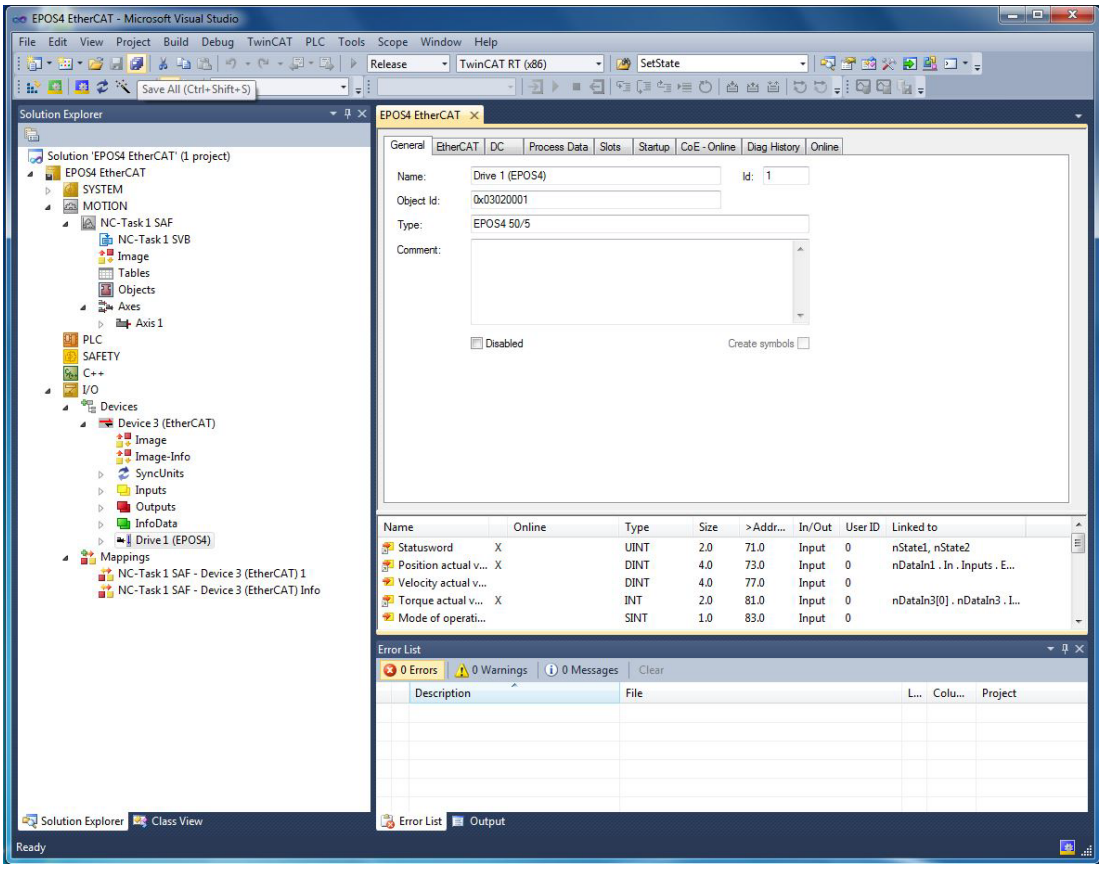


Figure 6-72 EtherCAT integration – Beckhoff TwinCAT | Structure tree

2) Use the tab ¤Slots¤ to allocate the operating mode to be used:

a) Select a ¤Slot¤ from the left pane.

b) Select the desired operating mode from the right pane ¤Module¤.

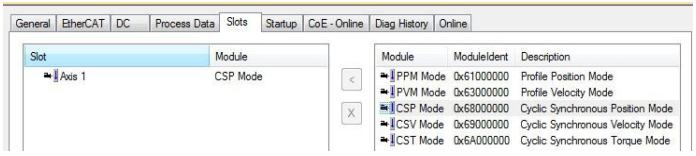


Figure 6-73 EtherCAT integration – Beckhoff TwinCAT | Configure slots

6.3.5 Changing PDO Mapping using Beckhoff TwinCAT

1) Select the device in the Solution Explorer, then click the PDO you wish to edit.

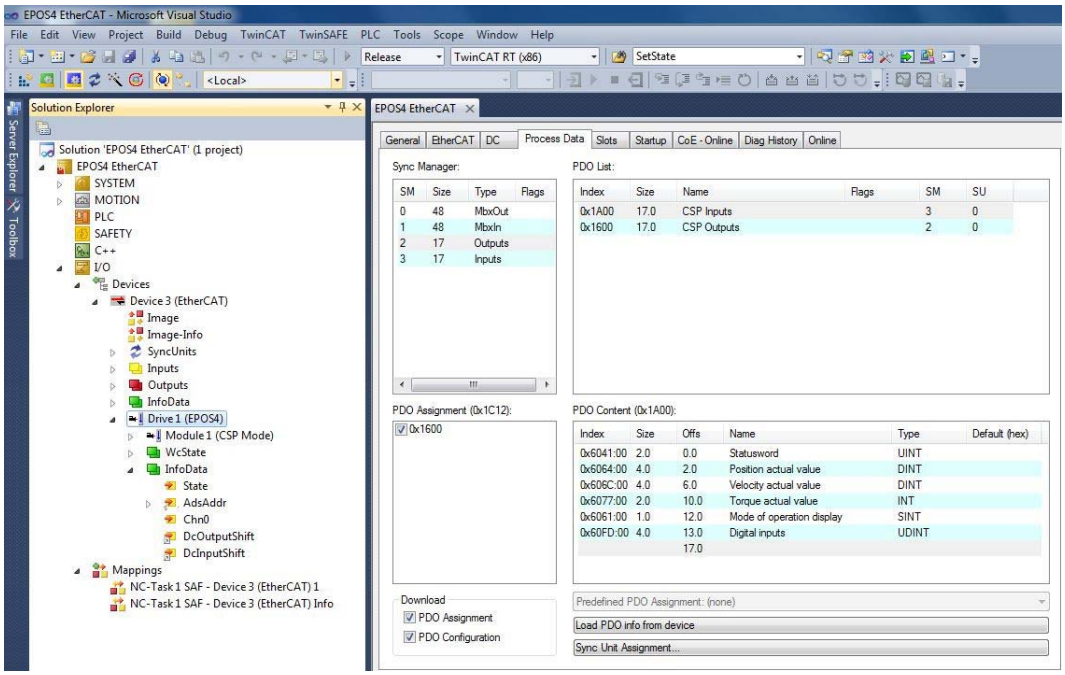


Figure 6-74 EtherCAT integration – Beckhoff TwinCAT | Display process data

2) Click the desired preconfigured PDO mapping from the list. Click right to open the context menu.
3) Choose either ¤Delete¤ to remove an existing variable or ¤Insert¤ to add a new variable.

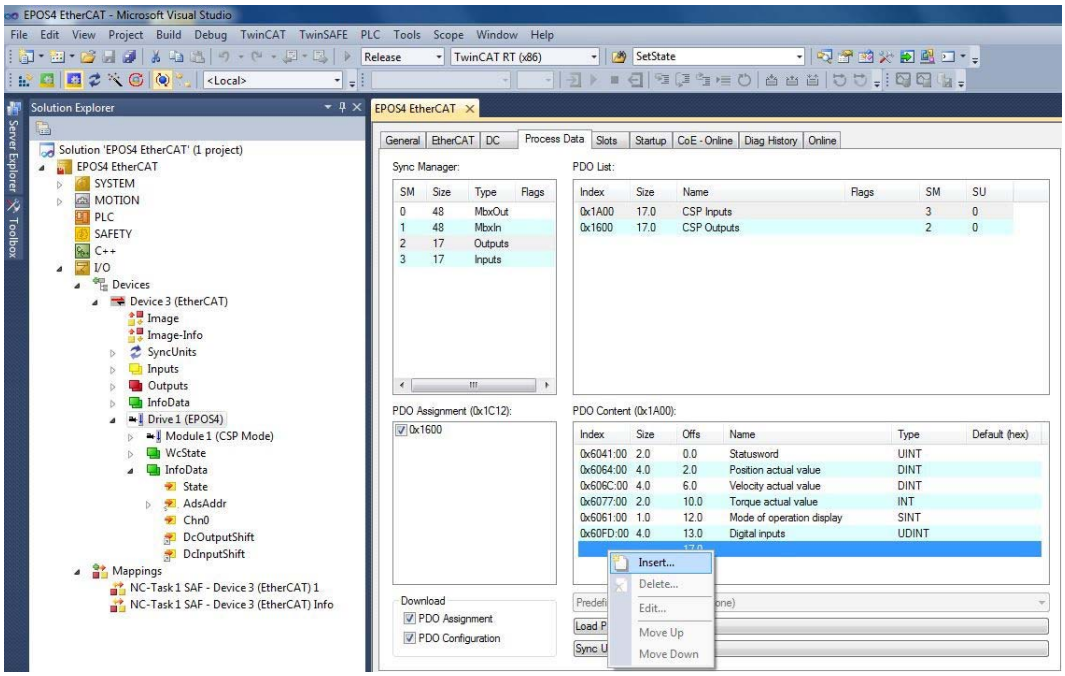


Figure 6-75 EtherCAT integration – Beckhoff TwinCAT | Select PDO

4) Select the object you wish to map and click ¤OK¤.

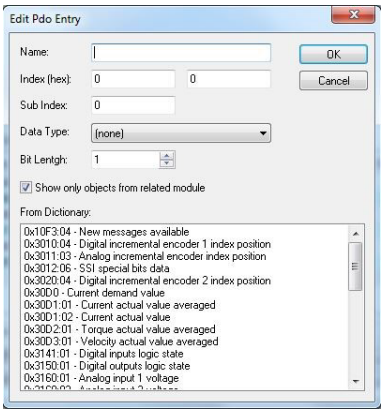


Figure 6-76 EtherCAT integration – Beckhoff TwinCAT | Edit PDO values

VERIFY CSP SETTINGS

5) Enable the Distributed Clock from the EPOS4.

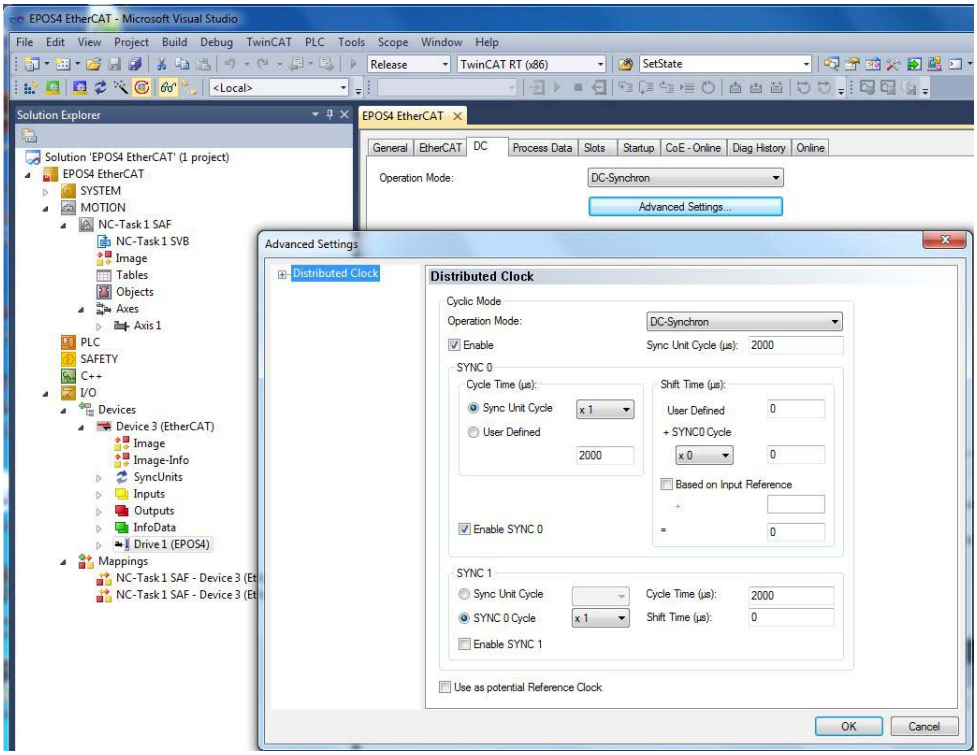


Figure 6-77 EtherCAT integration – Beckhoff TwinCAT | Set distributed clock

6) In the Solution Explorer, click on tree item ¤NC-Task 1 SAF¤, then tab ¤Task¤. Set cycle time to 2 ms.

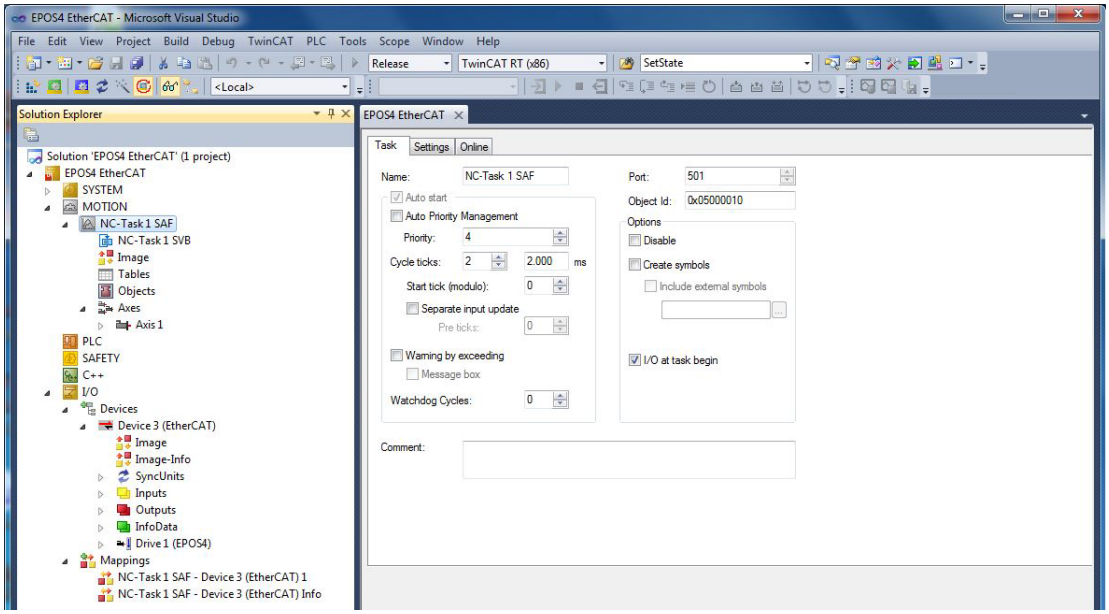
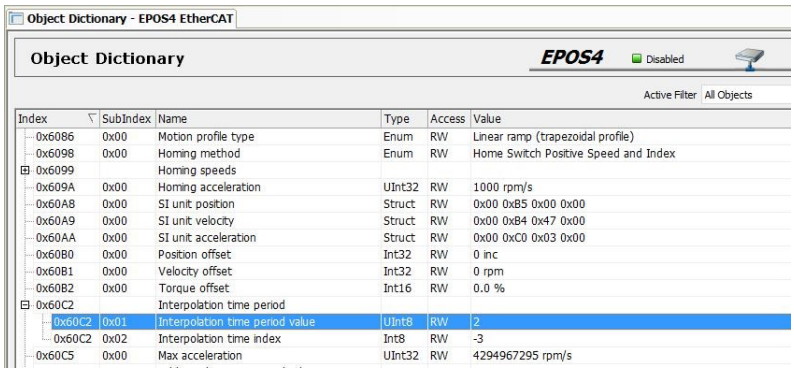


Figure 6-78 EtherCAT integration – Beckhoff TwinCAT | Set cycle ticks 1

7) For **CSP** and **CSV** mode, set object 0x60C2-01 to the same value as the “Cyclic ticks” in ¤NC-Task 1 SAF¤ (➔step 6). For **CST** mode, set it to zero.



Index	SubIndex	Name	Type	Access	Value
0x6086	0x00	Motion profile type	Enum	RW	Linear ramp (trapezoidal profile)
0x6098	0x00	Homing method	Enum	RW	Home Switch Positive Speed and Index
0x6099		Homing speeds			
0x609A	0x00	Homing acceleration	UInt32	RW	1000 rpm/s
0x60A8	0x00	SI unit position	Struct	RW	0x00 0xB5 0x00 0x00
0x60A9	0x00	SI unit velocity	Struct	RW	0x00 0xB4 0x47 0x00
0x60AA	0x00	SI unit acceleration	Struct	RW	0x00 0xC0 0x03 0x00
0x60B0	0x00	Position offset	Int32	RW	0 inc
0x60B1	0x00	Velocity offset	Int32	RW	0 rpm
0x60B2	0x00	Torque offset	Int16	RW	0.0 %
0x60C2		Interpolation time period			
0x60C2	0x01	Interpolation time period value	UInt8	RW	2
0x60C2	0x02	Interpolation time index	Int8	RW	-3
0x60C5	0x00	Max acceleration	UInt32	RW	4294967295 rpm/s

Figure 6-79 EtherCAT integration – Beckhoff TwinCAT | Set cycle ticks 2

6.3.6 Configuration of the Axis

1) In the tab ¤Settings¤, verify that ¤Link To I/O…¤ is assigned to the EPOS4 axis (naming is by your choice).

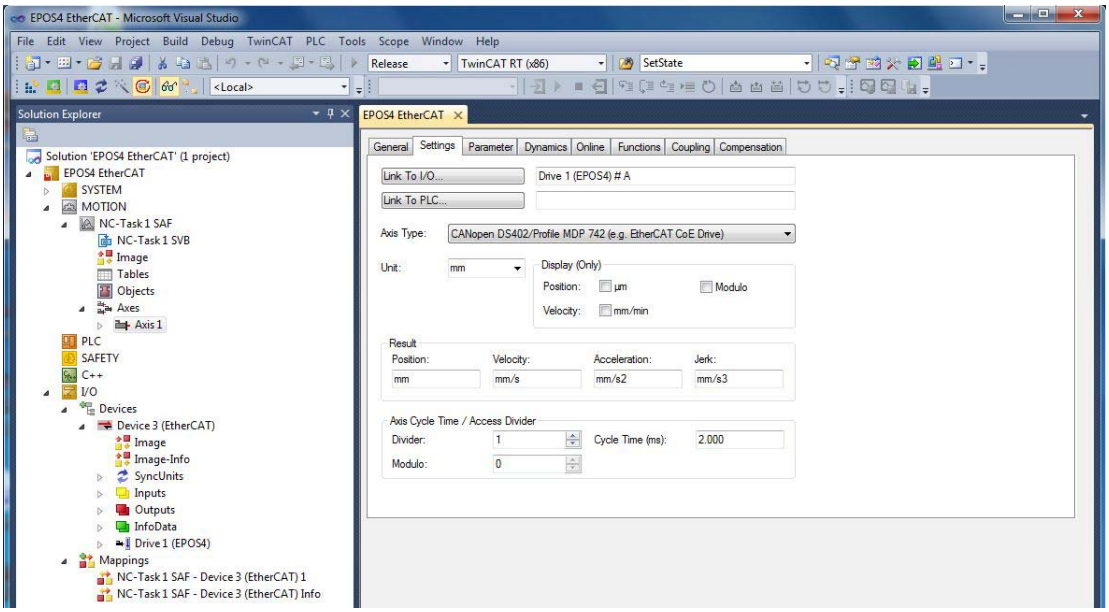


Figure 6-80 EtherCAT integration – Beckhoff TwinCAT | Link axis

2) In the tab ¤Parameter¤, adjust the motor speed settings as to the motor's capability and to the supply voltage.

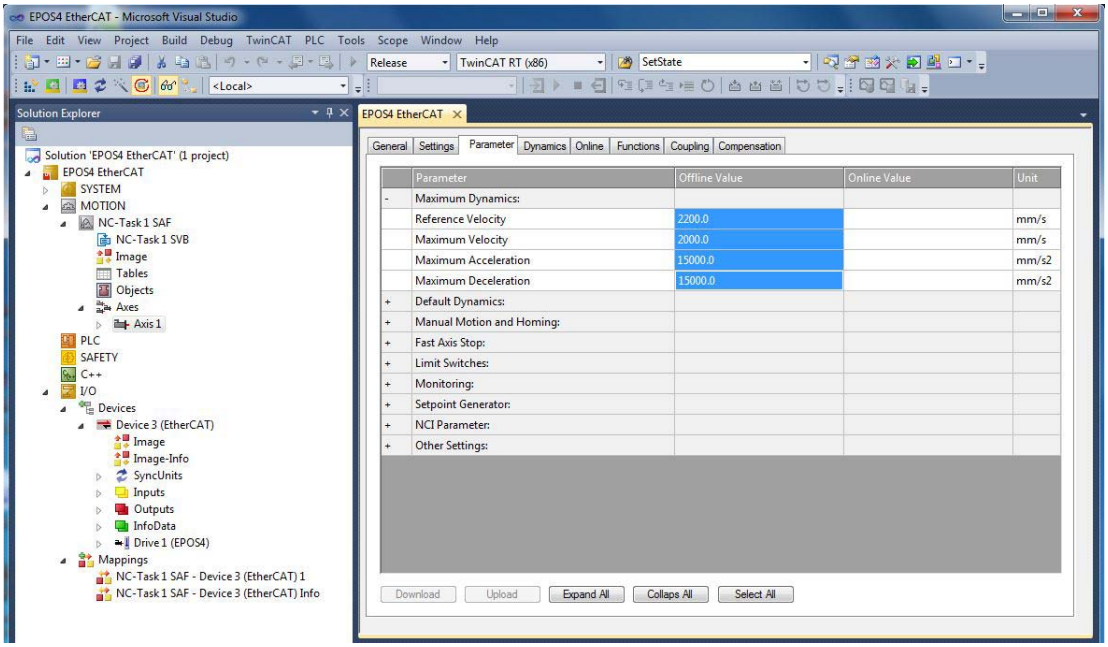


Figure 6-81 EtherCAT integration – Beckhoff TwinCAT | Set speed settings

3) Set Dead Time Compensation to approximately three to four times the set NC-Task SAF Cycle ticks (➔“Verify CSP Settings” on page 6-87; step 5)

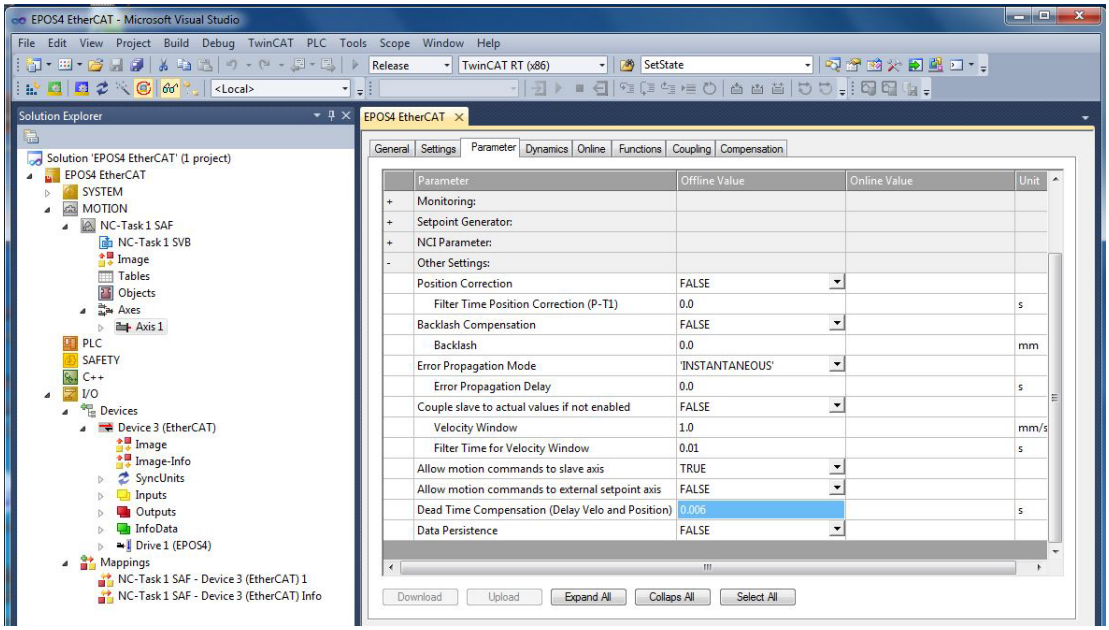


Figure 6-82 EtherCAT integration – Beckhoff TwinCAT | Set dead time compensation

4) Make sure to set the correct encoder resolution.

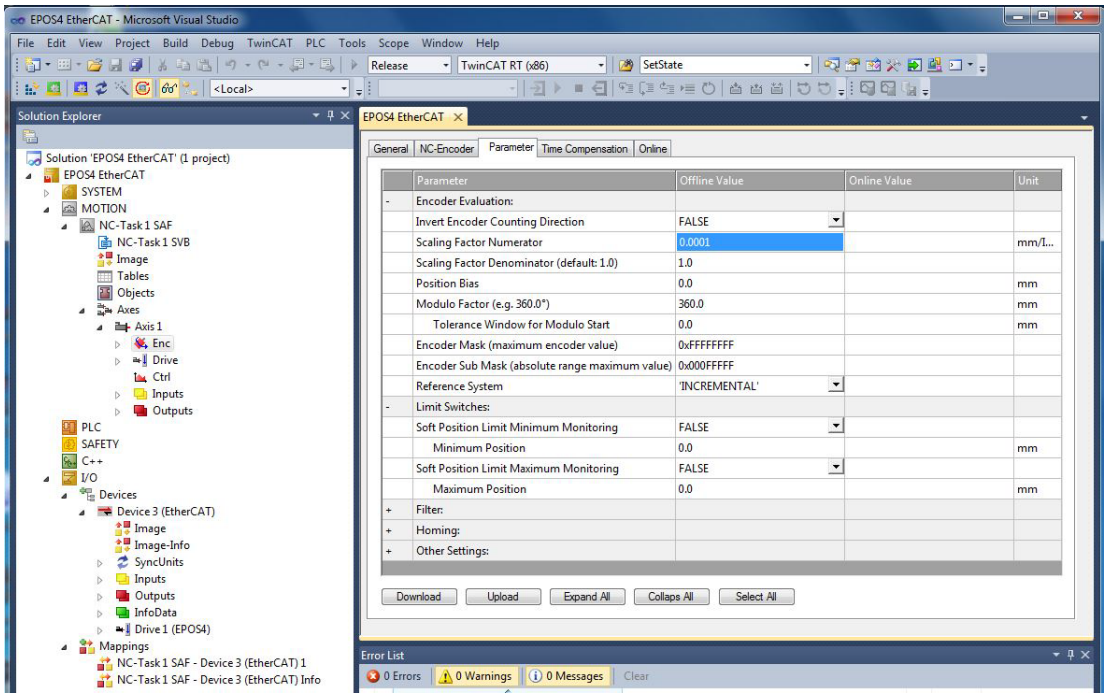


Figure 6-83 EtherCAT integration – Beckhoff TwinCAT | Set encoder settings

5) Configure the modes as follows:

SETTINGS FOR CSP MODE

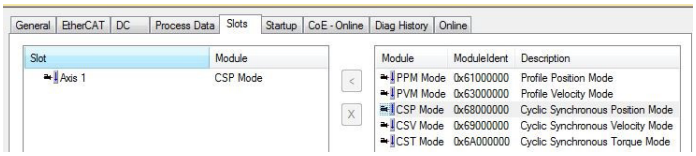


Figure 6-84 EtherCAT integration – Beckhoff TwinCAT | Set CSP settings

Configure the position control loop as follows:

- Position control: Proportional Factor Kv → “0.0”
- Feedforward Velocity: Pre-Control Weighting [0.0…1.0] → “1.0”

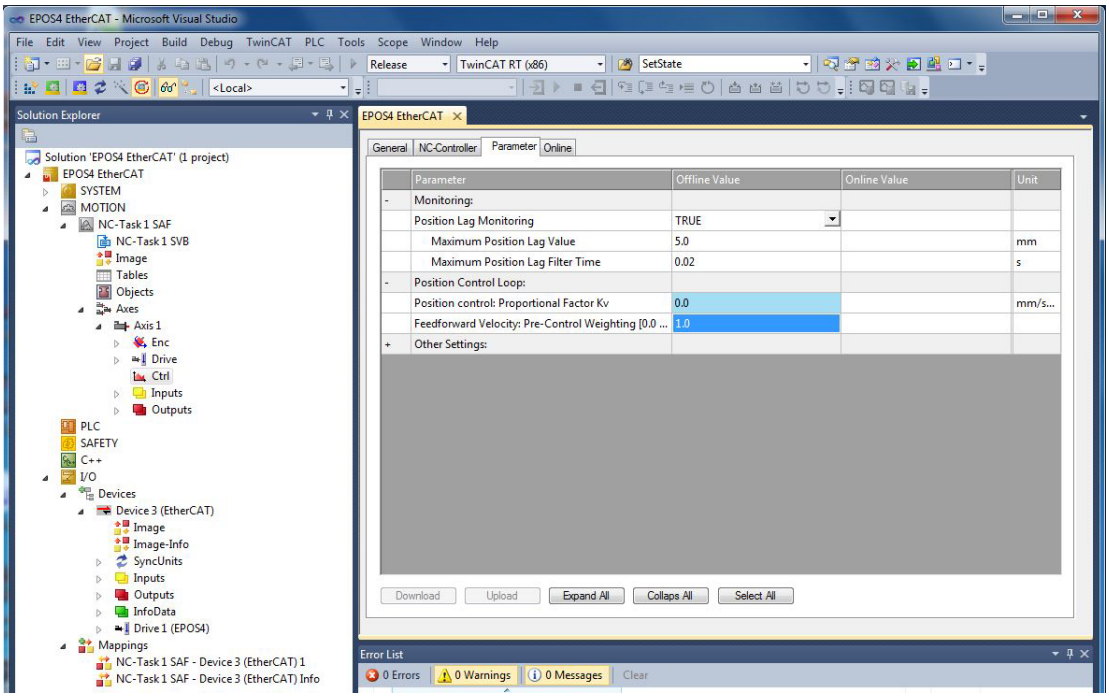


Figure 6-85 EtherCAT integration – Beckhoff TwinCAT | Set position control loop settings

SETTINGS FOR CSV MODE

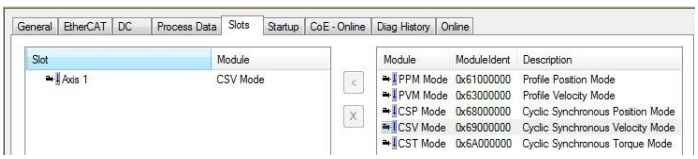


Figure 6-86 EtherCAT integration – Beckhoff TwinCAT | Set CSV settings

Configure the position control loop as follows:

- Position control: Proportional Factor Kv → Select a Kv Factor suitable for your drive system
- Feedforward Velocity: Pre-Control Weighting [0.0…1.0] → “1.0”

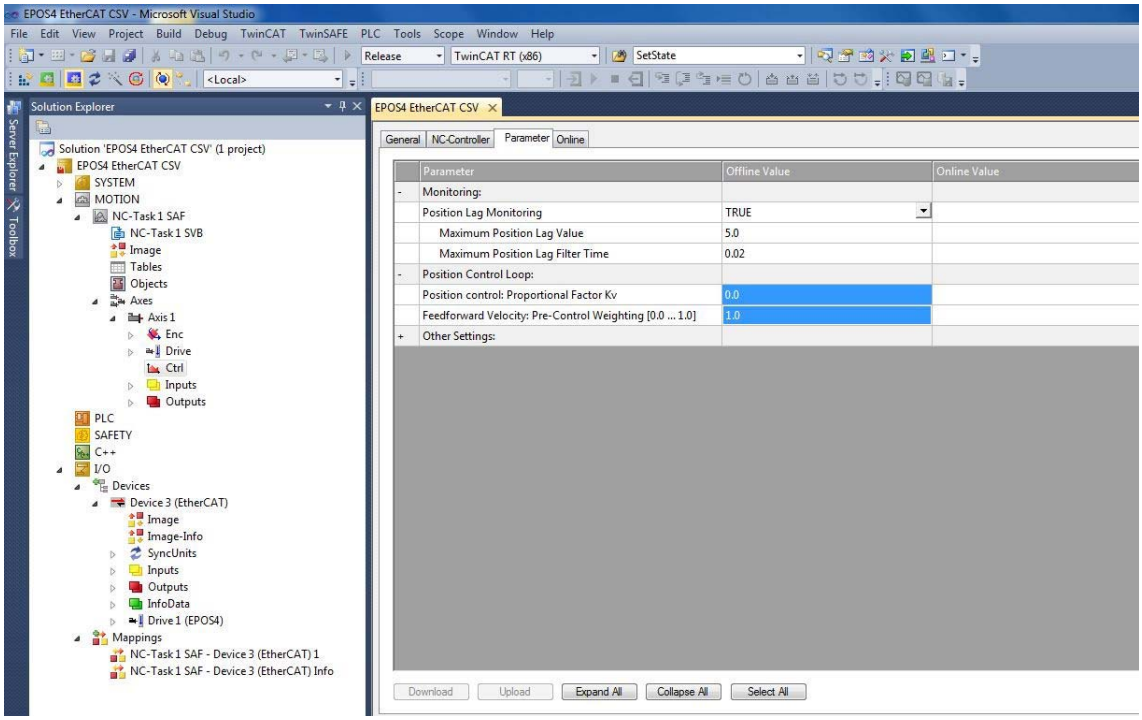


Figure 6-87 EtherCAT integration – Beckhoff TwinCAT | Set position control loop settings

In the tab ¤Parameter¤, set the correct “Output Scaling Factor (Velocity)”. Scaling may be calculated as follows:

Scaling = 7500 / (Encoder count number * 4)

e.g. Encoder with 500 counts per turn: Scaling = 7500 / (500 * 4) = 3.75

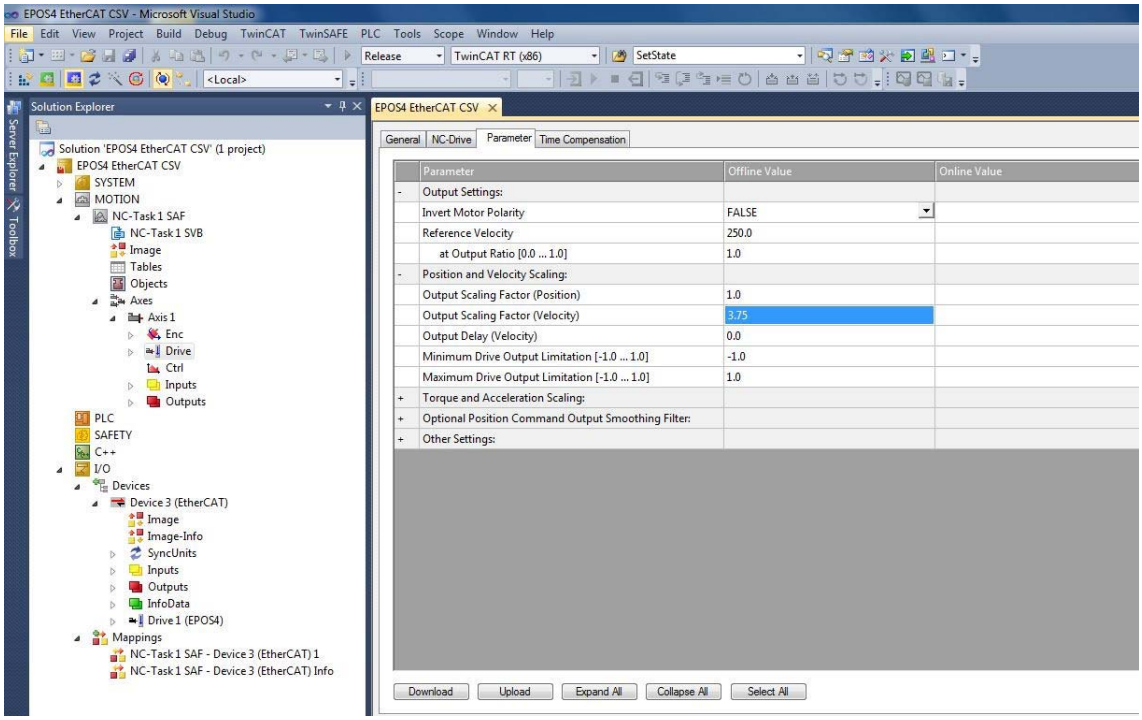


Figure 6-88 EtherCAT integration – Beckhoff TwinCAT | Set output scaling factor

SETTINGS FOR CST MODE

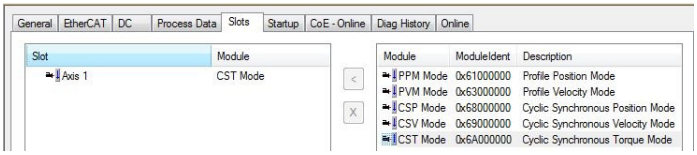


Figure 6-89 EtherCAT integration – Beckhoff TwinCAT | Set CST settings

In the Solution Explorer, select ¤CST Outputs¤ and set the link for the “Target Torque” variable.

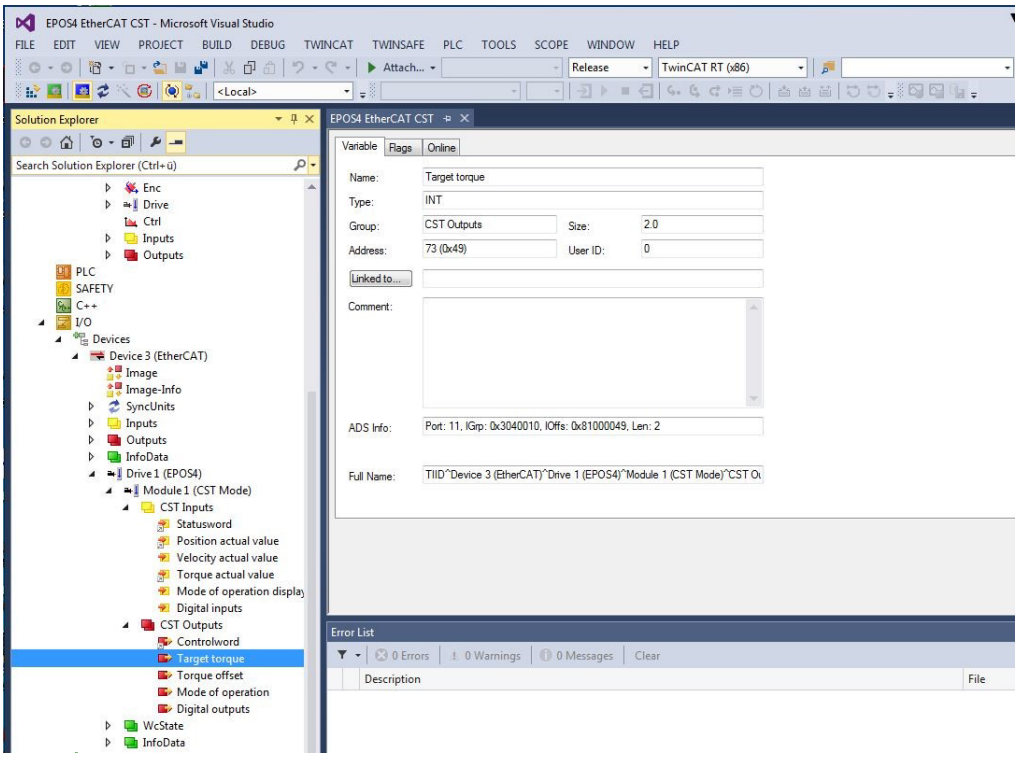


Figure 6-90 EtherCAT integration – Beckhoff TwinCAT | Set target torque

In folder ¤Drive¤ \ ¤Out¤, select “nDataOut2(0)” of Axis 1 as link variable.

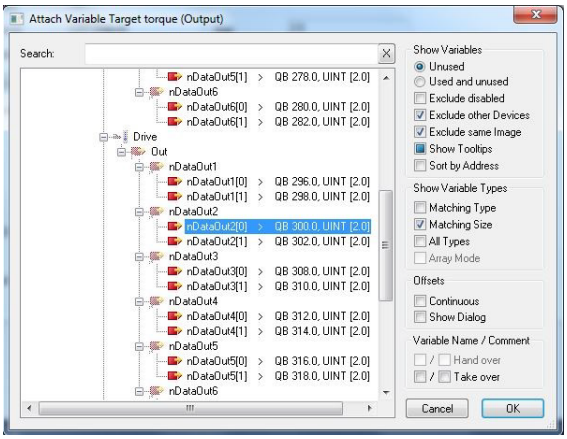


Figure 6-91 EtherCAT integration – Beckhoff TwinCAT | Configure position control loop

Configure the position control loop as follows:

- NC-Controller Type: Position controller PID (with Ka)

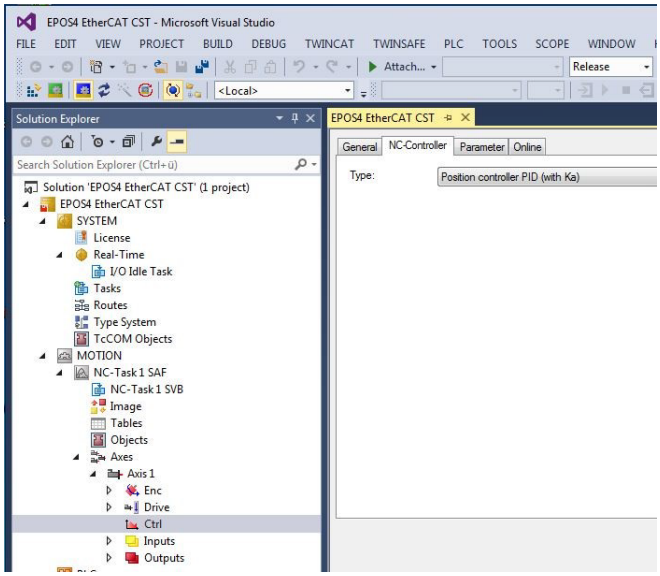


Figure 6-92 EtherCAT integration – Beckhoff TwinCAT | Configure position control type

- Tn = Kp / Ki (EPOS4 object 0x30A1-01 and object 0x30A1-02)
- Tv = Kd / Kp (EPOS4 object 0x30A1-03 and object 0x30A1-01)
- Kv must be termined empirically

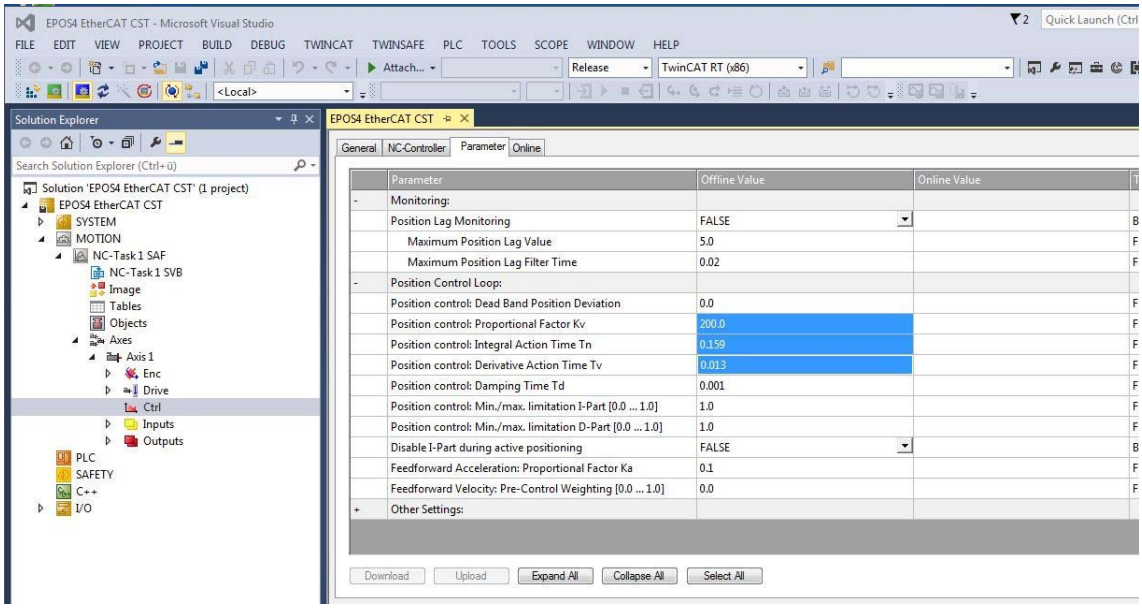


Figure 6-93 EtherCAT integration – Beckhoff TwinCAT | Configure position control parameters

6.4 zub MACS Integration

OBJECTIVE

This chapter explains the required configuration of so-called «MasterMACS» or «MACS5» multi-axis motion controllers to command an EPOS4 by EtherCAT.

BASICS

The MACS master controller product series of the Swiss company zub AG (➔www.zub.ch/en) are freely programmable just as a PLC but are mainly designed for sophisticated coordination and synchronization of multi-axis drive systems. These masters can process the motion of one or multiple axis and command the EPOS4 via CAN or EtherCAT.
As member of the maxon group, the company «zub machine control AG» provides you with industry-proven, high-end solutions that are supported by both maxon motor ag and zub AG.

Available are different types of masters: ➔www.zub.ch/en/products/product-gallery.html

The information given in this chapter refers to the following two product types. They are commanding the EPOS4 in Cyclic Synchronous Position (CSP) mode via EtherCAT:

- MasterMACS
- MACS5

With other MACS product types, with other EPOS4 operating modes (e.g. CSV), or for commanding via CAN (instead of EtherCAT), an adapted configuration will be required.

PRECONDITIONS

The information given in this chapter presumes that there is some level of experience present concerning the functionality and usage of zub's development environment («APOSS») as well as to programming language.

The software and all manuals are free for download from zub's website:
➔www.zub.ch/en/downloads.html

FUNDAMENTALS

Typical PLCs use the *:esi file and a System Manager tool to configure master and slave. The MACS software development environment does not offer such a System Manager tool. The configuration of the communication and data exchange is defined as part of the application's source code. This code section will be typically handled by an include file (*.mi) which can easily be copied into application programs. Thus, configuration can be quite simple but remains still very flexible.

The present chapter provides the code of a typical configuration of the MACS master and the EPOS4 commanded via EtherCAT based on the CSP mode. You may copy/paste the required code from the document into the source code editor of the APOSS development environment in use by MACS controllers.

Take note of the remarks and explanations in the code. They will help you to get a better understanding on the different configuration tasks and the possible additional motor configuration which must be checked or adapted.

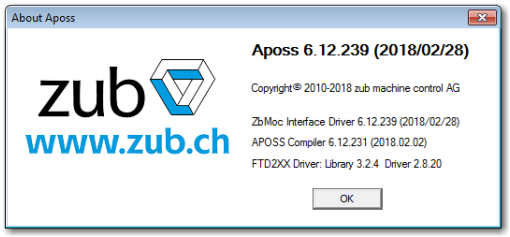
REQUIRED TOOLS

Software Development Environment

APOSS

V 6.12.239 (or higher)

Menu item Help
- About Program



EPOS Studio

V 3.4 Revision 1 (or higher)

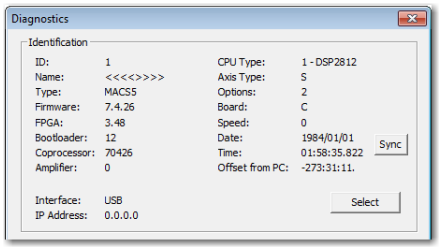
Menu item Help
- About EPOS Studio

Firmware Versions

MACS5 / MasterMACS

7.4.26 (or higher)

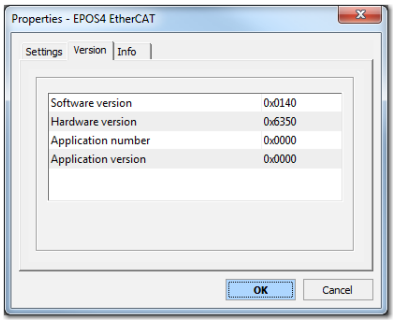
Menu item Controller
- Diagnostics



EPOS4

0x140 (or higher)

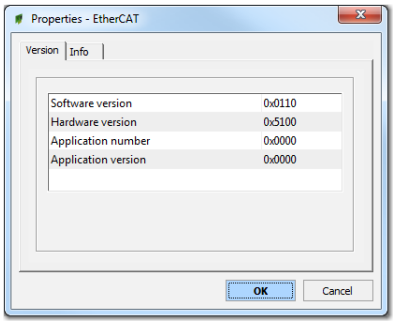
Communication Tree
- EPOS4 icon
- Properties
- Version



EPOS4 EtherCAT Module

0x110 (or higher)

Communication Tree
- EPOS4 icon
- EtherCAT icon
- Properties
- Version



6.4.1 EPOS4: Configuration Tasks

EPOS STUDIO'S "STARTUP WIZARD"

1) Configure motor, sensor, and system data.

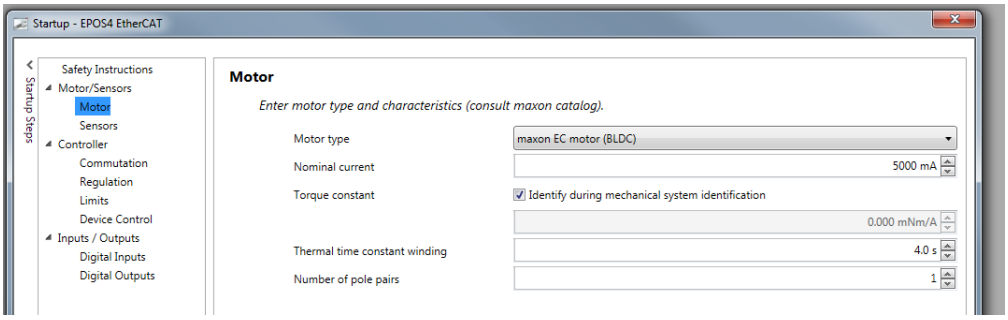


Figure 6-94 EtherCAT integration – zub's MACS Multi-Axis EtherCAT Masters | EPOS Studio "Startup Wizard"

2) Conclude by pressing the ¤Finish¤ button to save all configured data in the EPOS4.

EPOS STUDIO'S "REGULATION TUNING"

1) Tune the current control (with or without load).
Current control parameters do not depend on the load. Therefore, current control tuning can be processed without a load.

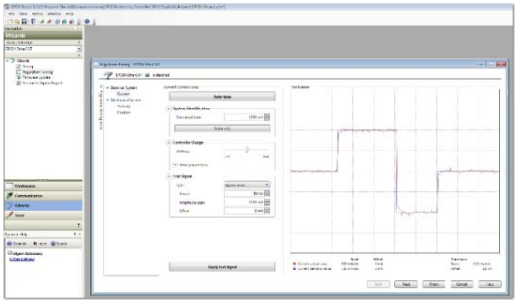


Figure 6-95 EtherCAT integration – zub's MACS Multi-Axis EtherCAT Masters | EPOS Studio "Regulation Tuning" 1

2) Tune the position control loop with load.
Position control parameters depend on the load, particularly if no gear box is present. Therefore, position control tuning should be processed with the attached load.

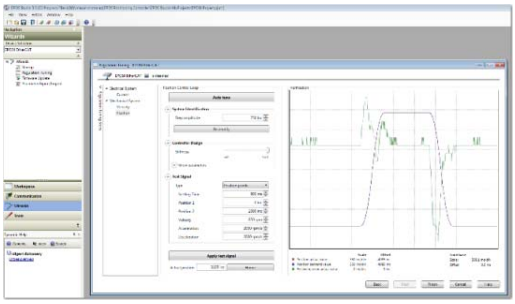


Figure 6-96 EtherCAT integration – zub's MACS Multi-Axis EtherCAT Masters | EPOS Studio "Regulation Tuning" 2

3) Velocity control does not apply in CSP mode. Thus, tuning of the velocity control loop is not mandatory but recommended anyway.
4) Find detailed information on control tuning in the application note ➔“2 Controller Architecture”; chapters “2.4 Regulation Tuning” and “2.5 Application Examples”.
5) Conclude by pressing the ¤Finish¤ buttonto save all configured data in the EPOS4 controller.

6.4.2 MasterMACS / MACS5: Setup Tasks

MACS5 IP MODE CONFIGURATION

With a MACS5 in use, configuration of the IP mode as “EtherCAT Master” is necessary.

- Menu item: Controller
- Parameters
- Global / Axis
- Mark the radio button “EtherCAT Master”

Note
By default, the MasterMACS is configured as “EtherCAT Master”.

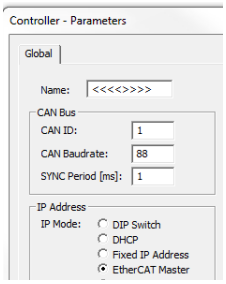


Figure 6-97 EtherCAT integration – zub’s MACS Multi-Axis EtherCAT Masters | MACS5 IP Mode Configuration

MACS & EPOS4: CONFIGURATION OF CONSISTENT PARAMETERS

1) Configure the encoder resolution of the MACS and EPOS4 with matching values.
2) Configure the “Following error window” of the MACS parameter “POSERR” with a higher value than the EPOS4 object 0x6065-00.
3) Ensure that configuration of the MACS’ parameter “EtherCAT SYNC period” (default: 1000 µs) and the EPOS4’s “Interpolation time period value” object 0x60C2-01 correspond to each other.
   – We recommend to keep the MACS’ default setting of 1000 µs and to configure the **EPOS4’s object 0x60C2-01 to a value of 1 ms**.
   – If the settings of this MACS and EPOS4 parameters do not match, electrical noise and/or malfunction in position control may occur.

DEACTIVATION OF MACS POSITION CONTROL

The MACS position control is not required in case of EPOS4’s “Homing” or “CSP” mode. Therefore, position control can be deactivated by setting the MACS position control parameters (P, I, D) to “0” (zero), compare code extract “SetupAxisParam”.

LOCATION OF INCLUDE FILES (*.MI)

If there are include files (*.mi) in use by a program they must be located in the same directory as the application program (*.m). Otherwise, the include file will not be found by the APOSS compiler.

HOMING PROCESSED BY EPOS4

It is recommended to use the “Homing” mode of the EPOS4 before activating CSP and the MACS controller to process path planning and generation of cyclic demand position updates.

- Using EPOS4’s “Homing” mode offers the possibility to use all homing methods of the EPOS4.
- Refer to separate document ➔«EPOS4 Firmware Specification»; chapter “Homing Mode (HMM)” to learn more about the different homing methods.

6.4.3 MACS: Configuration of EtherCAT Communication & Start-up Procedure

The initial configuration of communication and start-up procedure of the MACS can be split into different consecutive steps (functions) which can be integrated in an include file (*.mi) for usage by multiple application programs.

The typical flow of function calls will look as follows:

1) Initial setup tasks
   a) Setup some base axis parameters of the MACS
      ➔source code of function “**SetupAxisParam()**”
   b) Setup CSP: Configure PDOs, SYNC period, and activate CSP mode
      ➔source code of function “**SetupDriveCommandingCSP()**”
2) Start EtherCAT communication
   a) Configure and intitiate EtherCAT communication
      ➔source code of function “**EtherCATMasterStart()**”
3) MACS system setup tasks
   a) Setup the bus module
      ➔source code of function “**SetupBusModule()**”
   b) Setup virtual amplifier
      ➔source code of function “**SetupVirtAmp()**”
   c) Setup virtual counter inputs
      ➔source code of function “**SetupVirtCntInp()**”

The later description provides code samples of all above mentioned functions as well as a simple application program. The source code can be copied from the document and pasted into an include file (such as “MACS-EPOS4-Config.mi”) which then initially can be called up by the application program (such as “MACS-EPOS4-Test.m”).

INITIAL SETUP TASKS: SETUPAXISPARAM(), SETUPPDOMAPPING()

```
#include "sysdef.mi"  // Standardized include file by zub
#pragma NOIMPLICIT

// Setup MACS axis parameters
long SetupAxisParam(long axis, long EncCpt, long MaxRpm, long MaxAcc)
{
// Ensure that this corresponds to the encoder and EPOS4 configuration!
set posencqc x(Axis) (EncCpt*4)  // Set enc. resolution per turn [inc]
set posencrev x(Axis) 1          // Default: 1
set feeddist x(Axis) (EncCpt*4)  // Set feed resolution (= output) per turn [inc]
set feedrev x(Axis) 1            // Default: 1

set velmax  x(Axis) MaxRpm       // Set max. velocity [rpm]
set rampmin x(Axis) MaxAcc       // Set max. acceleration [ms] (0- > MaxRpm)

set kprop x(Axis) 0        // Disable P-Gain of PID-Controller
set kder  x(Axis) 0        // Disable D-Gain of PID-Controller
```

```
set kint  x(Axis) 0        // Disable I-Gain of PID-Controller

set poserr x(Axis) 200000  // Set max following error [inc]
// EPOS4 processes position control,
// => Configuration of EPOS4's "Following Error Window" (0x6065-00) [inc]
}

// Setup CSP: Configure PDOs, SYNC period, activate OP mode
long SetupDriveCommandingCSP(long DriveId)
// DriveId is 1000000 plus the EtherCAT slave position in the bus
{
sdowriten DriveId 0x1C12 0x00 1 0x00     // Disable entry 0x1C12
sdowriten DriveId 0x1C13 0x00 1 0x00     // Disable entry 0x1C13

sdowriten DriveId 0x1A00 0 1 0           // Clear PDO 0x1A00 entries
sdowriten DriveId 0x1A00 1 4 0x60410010  // PDO 0x1A00 entry: Status
sdowriten DriveId 0x1A00 2 4 0x60640020  // PDO 0x1B00 entry: Actual position
sdowriten DriveId 0x1A00 0 1 2           // PDO 0x1A00 entry: Number of entries

sdowriten DriveId 0x1A01 0 1 0           // Clear PDO 0x1A01 entries
sdowriten DriveId 0x1A02 0 1 0           // clear PDO 0x1A02 entries
sdowriten DriveId 0x1A03 0 1 0           // clear PDO 0x1A03 entries

sdowriten DriveId 0x1600 0 1 0           // Clear PDO 0x1600 entries
sdowriten DriveId 0x1600 1 4 0x60400010  // PDO 0x1600 entry: Cmd
sdowriten DriveId 0x1600 2 4 0x607A0020  // PDO 0x1600 entry: Position set point
sdowriten DriveId 0x1600 0 1 2           // PDO 0x1600 entry: Number of entries

sdowriten DriveId 0x1601 0 1 0           // Clear PDO 0x1601 entries
sdowriten DriveId 0x1602 0 1 0           // Clear PDO 0x1602 entries
sdowriten DriveId 0x1603 0 1 0           // Clear PDO 0x1603 entries

sdowriten DriveId 0x1C12 1 2 0x1600      // PDO 0x1C12:01 index
sdowriten DriveId 0x1C12 0 1 1           // PDO 0x1C12 count

sdowriten DriveId 0x1C13 1 2 0x1A00      // PDO 0x1C13:01 index
sdowriten DriveId 0x1C13 0 1 1           // PDO 0x1C13 count

// Ensure that the EtherCAT SYNC and EPOS4 interpolation time period correspond
// It is recommended to use 1 ms (which is also the default setting value of the MACS)
ecatmasterconfig 0x1000 0 1000    // MACS:  EtherCAT master SYNC: 1000 us
sdowriten DriveId 0x60C2 1 0 1    // EPOS4: Interpolation time period value: 1ms

// Set EPOS4 operating mode: CSP
sdowriten DriveId 0x6060 0 1 8   // CSP = mode 8
}
```

START ETHERCAT COMMUNICATION: ETHERCATMASTERSTART()

```
// Starting EtherCAT
long EtherCATMasterStart()
{
ecatmastercommand 0x1000 2    //  Map Input and Output buffers (go to safeop)
ecatmastercommand 0x1000 3    //  Request & wait OP state for all slaves
}
```

MACS SYSTEM SETUP TASKS: SETUPBUSMODULE(), SETUPVIRTAMP(), SETUPVIRTCNTINP()

```
// Setup bus modules
long SetupBusModule(long Axis, long PdoNumber)
{
long busmod
busmod = Axis-1

BUSMOD_PARAM(busmod, BUSMOD_MODE) = 0      // delete existing bus module
BUSMOD_PARAM(busmod, BUSMOD_BUSTYPE) = 2   // EtherCAT Master

BUSMOD_PARAM(busmod, BUSMOD_PISRC_INPUT1) =
VIRTAMP_PROCESS_SRCINDEX(busmod,PO_VIRTAMP_CMDWORD)   // CMD Word
BUSMOD_PARAM(busmod, BUSMOD_PISRC_INPUT2) =
VIRTAMP_PROCESS_SRCINDEX(busmod,PO_VIRTAMP_REFPOS)    // Position setpoint

BUSMOD_PARAM(busmod, BUSMOD_TXMAP_INPUT1) = PdoNumber*0x01000000 + 2*0x00010000 + 0
// pdo; length in bytes; bytes offset of control word
BUSMOD_PARAM(busmod, BUSMOD_TXMAP_INPUT2) = PdoNumber*0x01000000 + 4*0x00010000 + 2
// pdo; length in bytes; bytes offset of target position

BUSMOD_PARAM(busmod, BUSMOD_RXMAP_POVALUE1) = PdoNumber*0x01000000 + 2*0x00010000 + 0
// pdo ; length in bytes; bytes offset of status word
BUSMOD_PARAM(busmod, BUSMOD_RXMAP_POVALUE2) = PdoNumber*0x01000000 + 4*0x00010000 + 2
// pdo ; length in bytes; bytes offset of position actual value

BUSMOD_PARAM(busmod, BUSMOD_MODE) = 2

// Start bus module
ecatmasterconfig (Axis) busmod 0
}

// Setup Virtual Amplifier
long SetupVirtAmp(long Axis)
{
long modno
modno = Axis-1
// virtual amplifiers have a fixed connection to axes number, axe 1 use amp 0
VIRTAMP_PARAM(modno,VIRTAMP_PISRC_CMDWORD)= AXE_PROCESS_SRCINDEX(modno,REG_CNTRLWORD)
VIRTAMP_PARAM(modno,VIRTAMP_PISRC_REFPOS) = AXE_PROCESS_SRCINDEX(modno,REG_COMPOS)
VIRTAMP_PARAM(modno,VIRTAMP_PISRC_REFVEL) = AXE_PROCESS_SRCINDEX(modno,REG_REFERENCE)
VIRTAMP_PARAM(modno,VIRTAMP_PISRC_REFACC) = AXE_PROCESS_SRCINDEX(modno,PID_FFACCPART)
VIRTAMP_PARAM(modno,VIRTAMP_PISRC_STATUS) =
BUSMOD_PROCESS_SRCINDEX(modno,PO_BUSMOD_VALUE1)
VIRTAMP_PARAM(modno,VIRTAMP_CNTRLW_PWROFF)    = 0x06
VIRTAMP_PARAM(modno,VIRTAMP_CNTRLW_PWRONDIS)  = 0x06
VIRTAMP_PARAM(modno,VIRTAMP_CNTRLW_PWRONENP)  = 0x0F
VIRTAMP_PARAM(modno,VIRTAMP_CNTRLW_PWRONENN)  = 0x0F
VIRTAMP_PARAM(modno,VIRTAMP_CNTRLW_QUICKSTOP) = 0x02
VIRTAMP_PARAM(modno,VIRTAMP_CNTRLW_RESET)     = 0x80
VIRTAMP_PARAM(modno,VIRTAMP_STOPDELAY)        = 0x0
VIRTAMP_PARAM(modno,VIRTAMP_ERROR_BITMASK)    = 0x0008
VIRTAMP_PARAM(modno,VIRTAMP_ERROR_POLARITY)   = 1

VIRTAMP_PARAM(modno,VIRTAMP_MODE) = 1
// has to be the last one because it activates all
```

```
}

// Setup Virtual Counter inputs
long SetupVirtCntInp()
{
VIRTCOUNTIN_PARAM(0,VIRTCNTIN_PISRC_COUNTER)=
BUSMOD_PROCESS_SRCINDEX(0,PO_BUSMOD_VALUE2)
VIRTCOUNTIN_PARAM(0,VIRTCNTIN_MODE) = 3    // source is absolute and is taken as it is
}
```

6.4.4 Simple Application Program

The include file “MACS-EPOS-Config.mi“ holds the functions and source code listed on the last pages. This include file must be part of the application program.

```
// Main-Program (.m File)

// Include file providing the required setup functions
#include "MACS-EPOS4-Config.mi"
// Remark: Include files have to be located in the same directory like the *.m file

long Axis, PdoNumber, DriveId, EncCpt, MaxRpm, MaxAcc

Axis = 1
PdoNumber = 1
DriveId = 1000001  // = 1000000 plus the EtherCAT slave position in the bus
EncCpt = 500       // Encoder resolution [cpt]
MaxRpm = 3000      // Max. speed [rpm]
MaxAcc = 100       // Max. acceleration [ms]: 0 -> MaxRpm

errclr  // Clear all error states, if there are any present

ecatmastercommand 0x1000 0    // Disable master
ecatmastercommand 0x1000 1    // Start master

// Initial setup tasks
SetupAxisParam(Axis, EncCpt, MaxRpm, MaxAcc)
SetupDriveCommandingCSP(DriveId)

// Start EtherCAT communication
EtherCATMasterStart()

// MACS system setup tasks
SetupBusModule(Axis,PdoNumber)
SetupVirtAmp(Axis)
SetupVirtCntInp()

// Clear error states, if there are any present
Errclr              // Clear MACS error states
Amperrclr x(Axis)   // Clear EPOS4 error states

delay(20)           // Wait 20 ms
motor on x(Axis)    // Enable the power stage

// Start simple cyclic movement of the motor
while(1) do
   vel x(Axis) 50     // Use 50% of MACS max. velocity
   acc x(Axis) 30     // Use 30% of MACS max. acceleration
   dec x(Axis) 30     // Use 30% of MACS max. deceleration

   // Move absolute 10 turns
   posa x(Axis) (get posencqc)*10
   delay 500        // Wait 500 ms
   posa x(Axis) 0   // Move absolute to position 0
   delay 200        // Wait 200 ms
endwhile
```

6.5 OMRON Sysmac NJ Integration

The below descriptions are based on the following items:

- Sysmac Studio Standard Edition 1.44
- NJ301-1100 (Firmware Version 1.40)

CREATING PROJECT FILE

1) Create a Project File form the Project Window.

ETHERCAT CONFIGURATION

2) In the Multiview Explorer, select ¤Configurations and Setup¤, then ¤EtherCAT¤.

Figure 6-98 EtherCAT integration – OMRON Sysmac NJ | Configuration and Setup

This will open the ¤Edit Pane¤ and will automatically create the master.

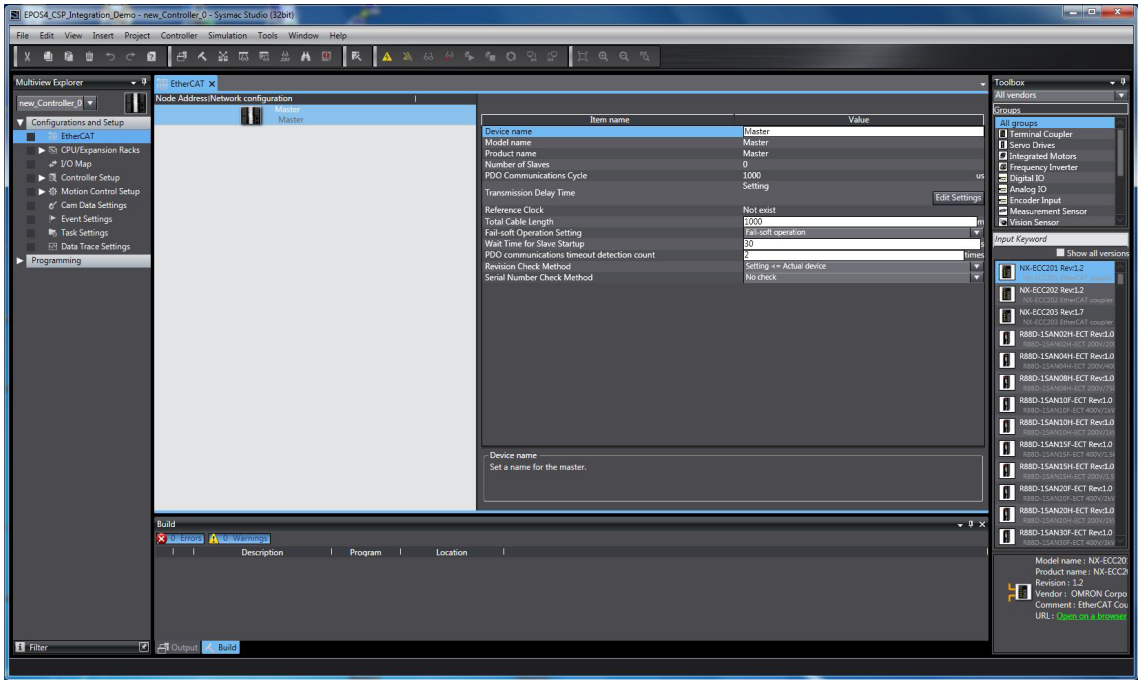


Figure 6-99 EtherCAT integration – OMRON Sysmac NJ | Master

IMPORT ESI LIBRARY

3) In the EtherCAT tab, click right on the master and select ¤Display ESI Library¤.

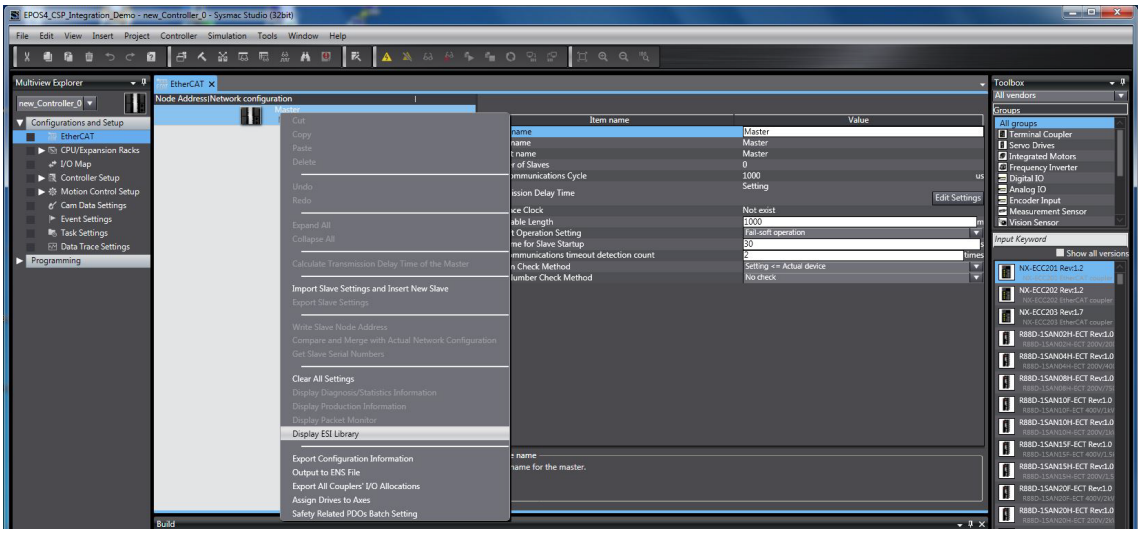


Figure 6-100 EtherCAT integration – OMRON Sysmac NJ | Import of ESI library

4) Click ¤Install (File)¤ to import the EPOS4 ESI file.

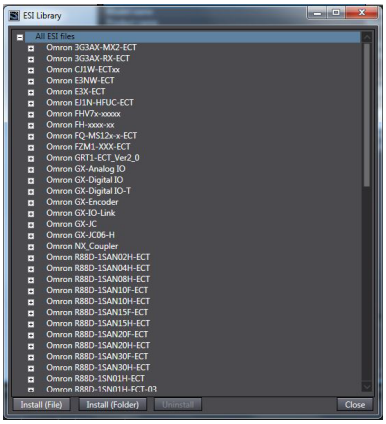


Figure 6-101 EtherCAT integration – OMRON Sysmac NJ | Import of EPOS4 ESI file

5) Store your settings, close and restart the «Sysmac Studio».

6) Select the desired EPOS4 slave(s) from the ¤Toolbox¤ and Drag&Drop it (them) to the ¤Master¤ in the EtherCAT tab.

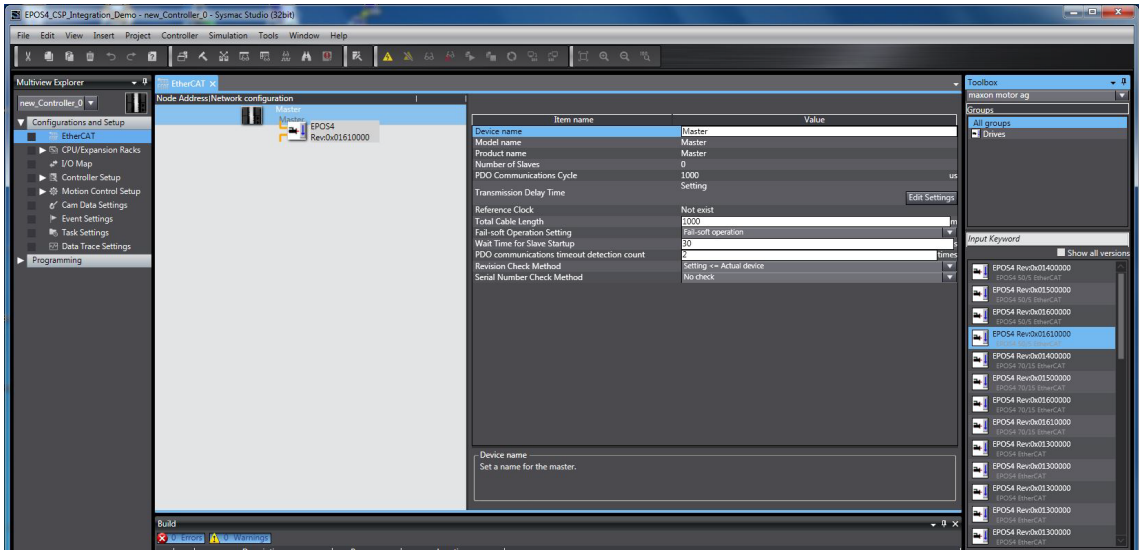


Figure 6-102 EtherCAT integration – OMRON Sysmac NJ | Slave

EPOS4 PARAMETERS

7) In the EtherCAT tab, click right on the slave and select ¤Edit Module Configuration¤.

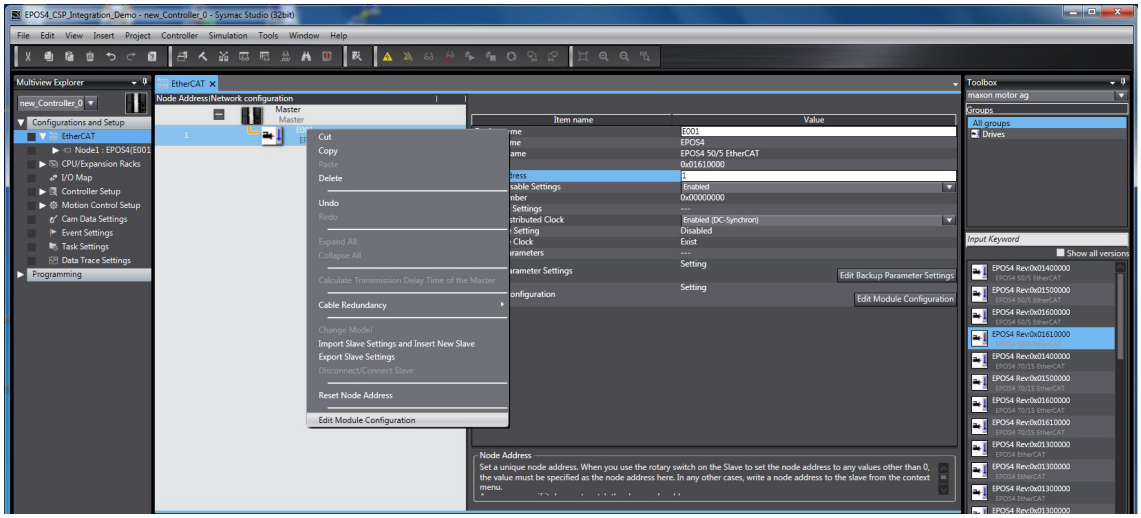


Figure 6-103 EtherCAT integration – OMRON Sysmac NJ | Slave parameters

This will open a new tab named “Node1: EPOS4 (xxx)”.

8) Select the desired operation mode from the ¤Toolbox¤ and Drag&Drop it to the respective axis in the EtherCAT tab.

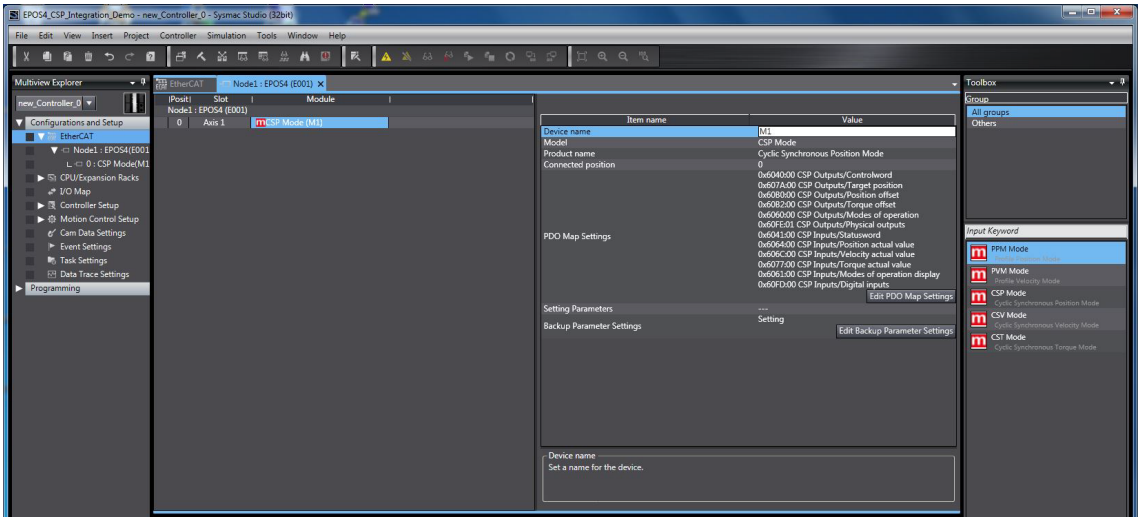


Figure 6-104 EtherCAT integration – OMRON Sysmac NJ | Operation mode

9) Change PDO mapping ¤Edit PDO Map Settings¤.

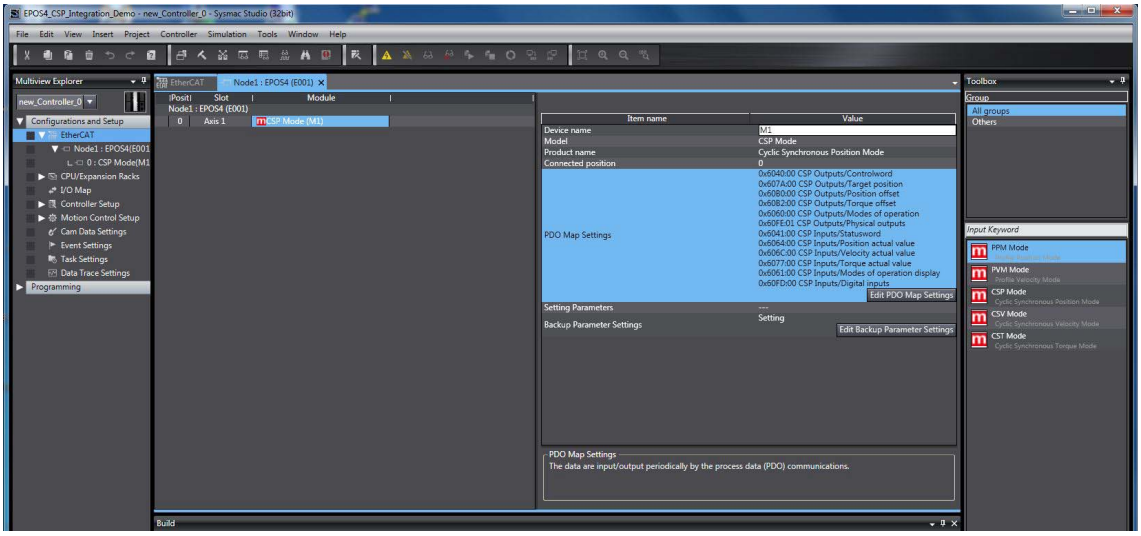


Figure 6-105 EtherCAT integration – OMRON Sysmac NJ | PDO mapping

10) Change the mapping with ¤Add PDO Entry¤ or ¤Delete PDO Entry¤.

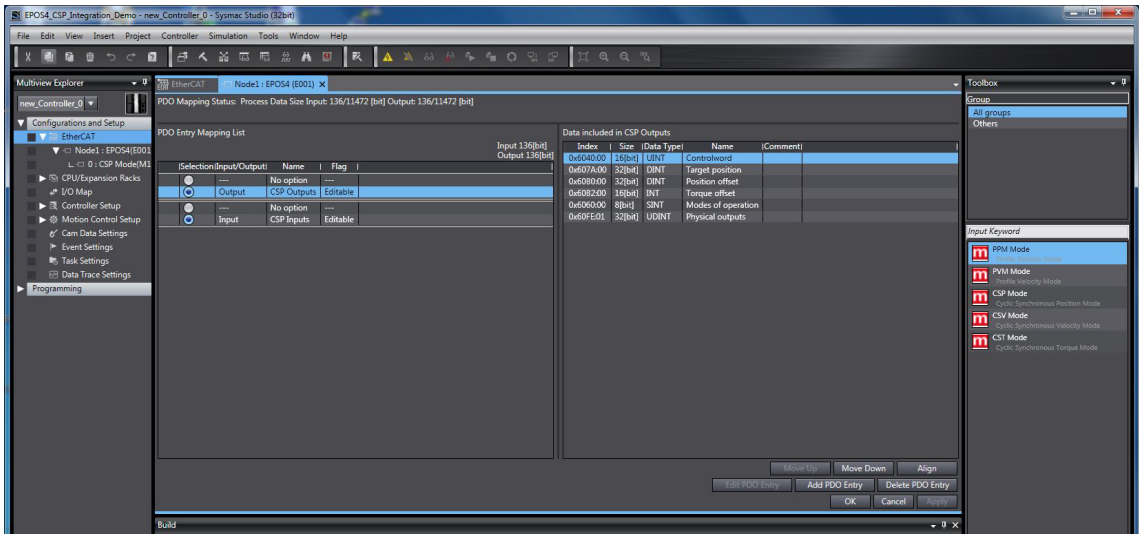


Figure 6-106 EtherCAT integration – OMRON Sysmac NJ | Change PDO mapping

11) Set the EtherCAT Node Address for each EPSO4 by either using the graphic (on the left) or by using the variable list (on the right). Take note that each EtherCAT node requires a unique node address.

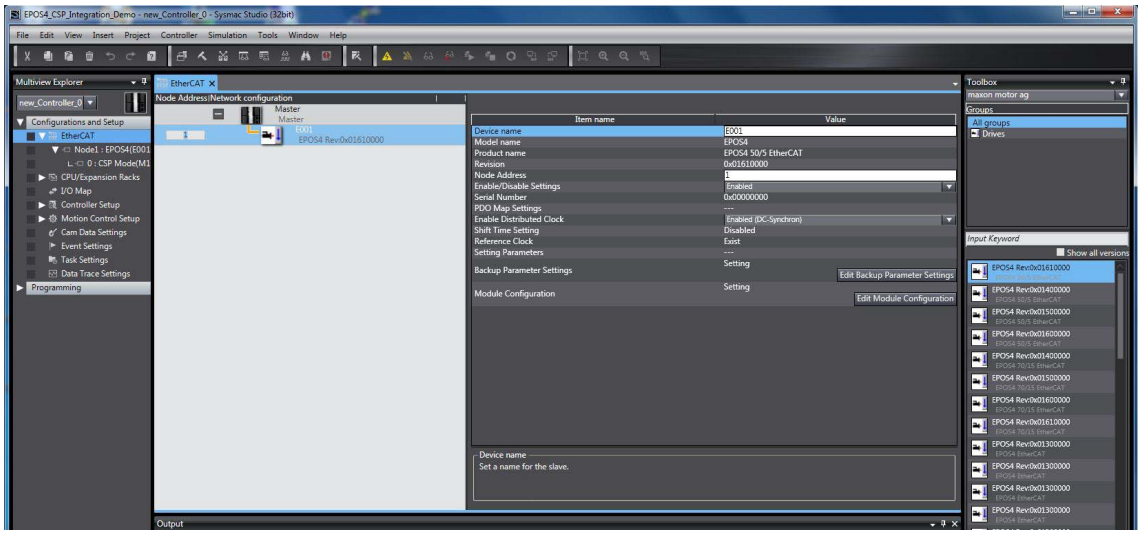


Figure 6-107 EtherCAT integration – OMRON Sysmac NJ | Set EtherCAT node address

12) Go Online to set the connection method (➔OMRON's «Sysmac Studio Operation Manual»).

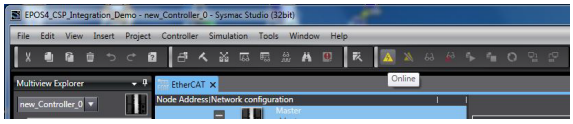


Figure 6-108 EtherCAT integration – OMRON Sysmac NJ | Going Online

13) In the EtherCAT tab, click right on the master and select ¤Write Slave Node Address¤.

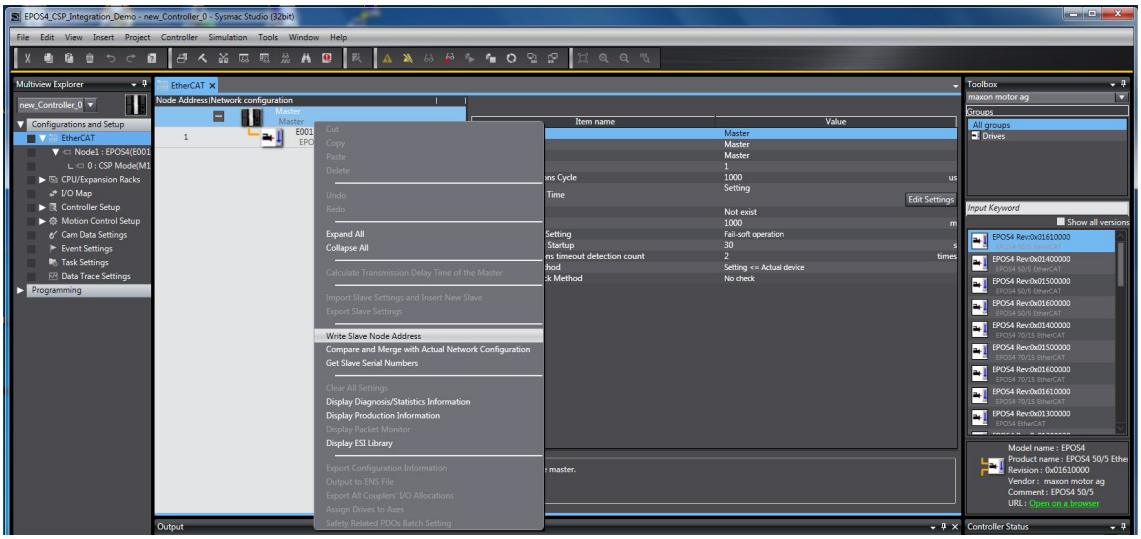


Figure 6-109 EtherCAT integration – OMRON Sysmac NJ | Slave node address

This will display a dialog box.

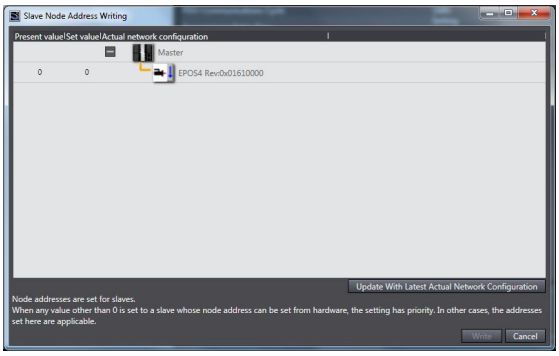


Figure 6-110 EtherCAT integration – OMRON Sysmac NJ | Write slave node address

14) If the node address is set correct, click ¤Cancel¤. Otherwise edit the node address and click ¤Write¤ and power off/power on the EPOS4 to activate the new node address.

15) In the EtherCAT tab, click right on the master and select ¤Compare and Merge with Actual Network Configuration¤.

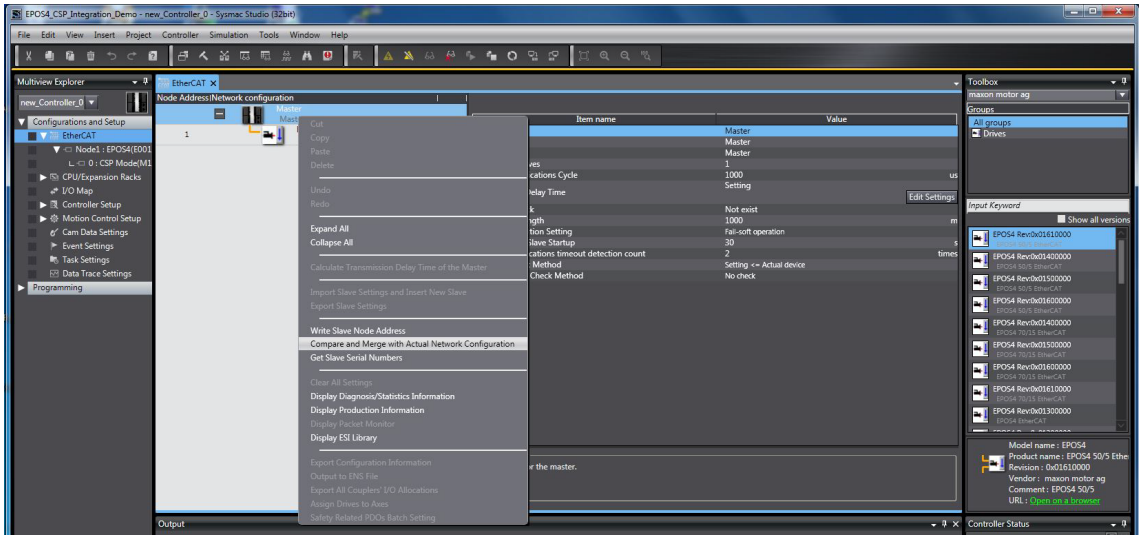


Figure 6-111 EtherCAT integration – OMRON Sysmac NJ | Network configuration

16) Both the actual network and Sysmac Studio configuration will be read and compared. Upon completion, the results are displayed.

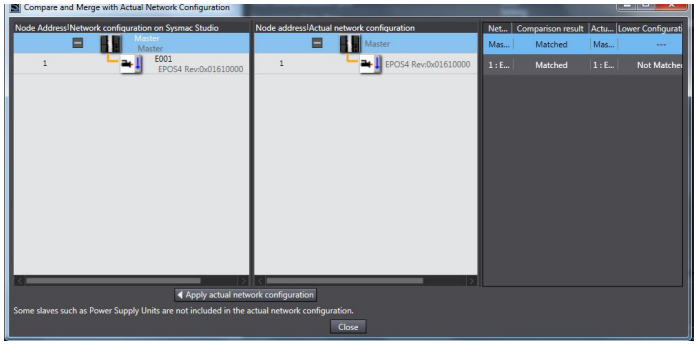


Figure 6-112 EtherCAT integration – OMRON Sysmac NJ | Comparison & Merger

17) Click ¤Apply actual network configuration¤, then click ¤Close¤.
18) Go Offline.
19) In the Multiview Explorer, click right on ¤Axis Settings¤ and select ¤Add¤, then ¤Axis Settings¤.

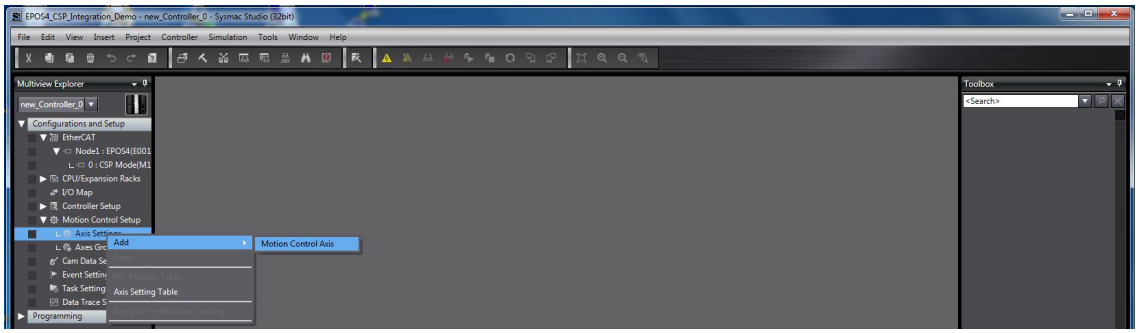


Figure 6-113 EtherCAT integration – OMRON Sysmac NJ | Axis settings

20) Rename the axis as desired.

21) Go to ¤Axis Basic Settings¤ and set the following parameters:
– Axis use = Used axis
– Axis type = Servo axis
– Output device 1" = Node:1, Slot : 0 CSP Mode(M1).

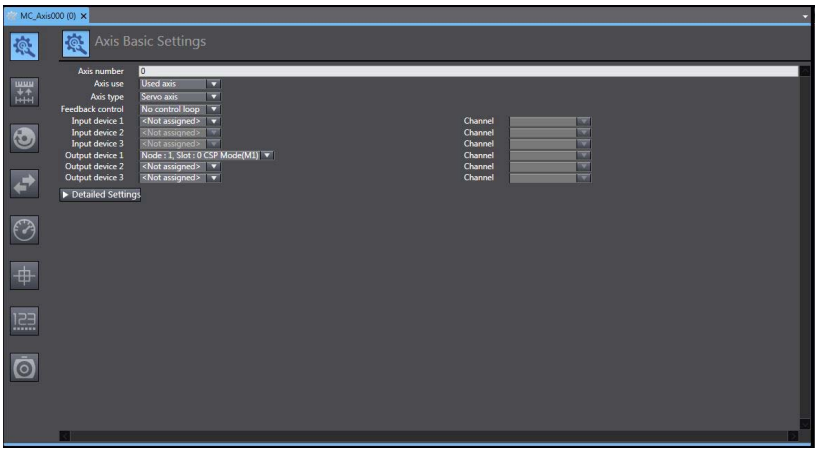


Figure 6-114 EtherCAT integration – OMRON Sysmac NJ | Axis basic settings

22) Expand the Detail Settings pane and set the respective values in the columns ¤Device¤ and ¤Process Data¤.

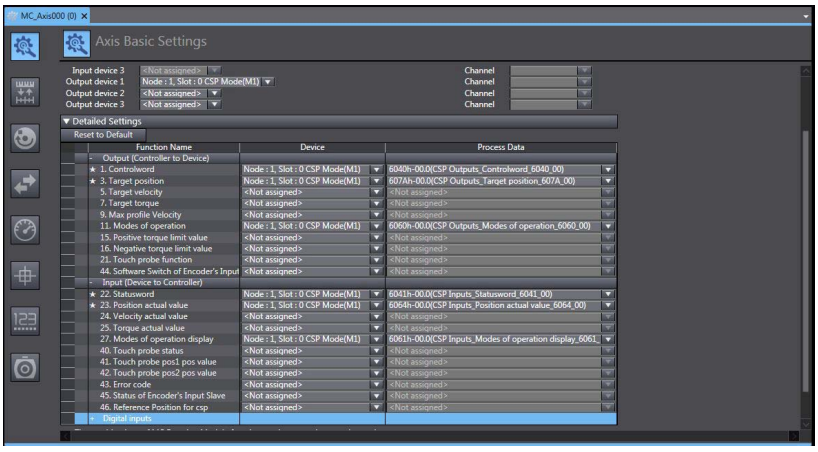


Figure 6-115 EtherCAT integration – OMRON Sysmac NJ | Axis detailed settings

23) Go to ¤Unit Conversion Settings¤ and set the following parameters:
– pulses per motor rotation (e.g. 500 pulse encoder → 2'000 pulse/rev)
– travel distance per motor rotation

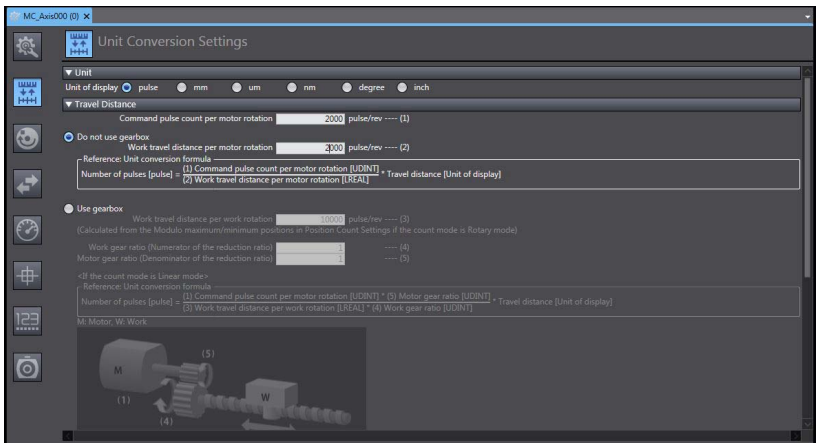


Figure 6-116 EtherCAT integration – OMRON Sysmac NJ | Unit conversion settings

24) Go to ¤ Operation Settings¤ and set the following parameters:
– velocity (converting rpm → pulses/s (e.g. 10'000rpm / 60m*s *2'000 pulses = 333'333 pulses/s)
– acceleration rate
– deceleration rate
– other monitor parameters

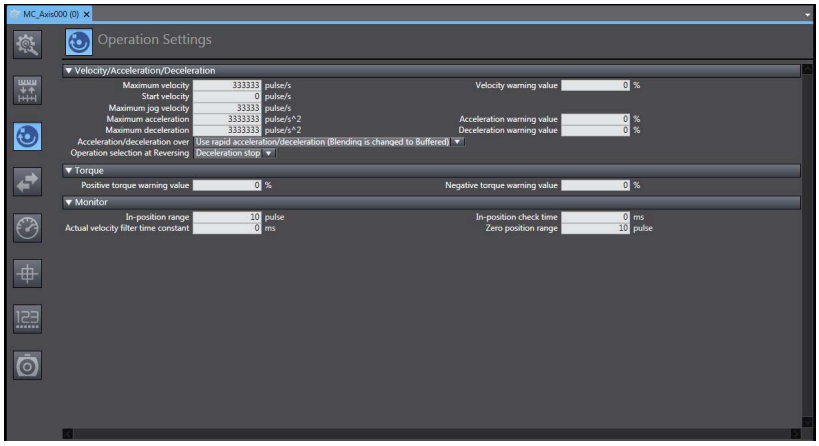


Figure 6-117 EtherCAT integration – OMRON Sysmac NJ | Operation settings

25) Go to ¤ Servo Drive Settings¤ and set the following parameters:
– maximum position setting
– minimum position setting
– main circuit power supply OFF detection to ¤Do not detect¤.

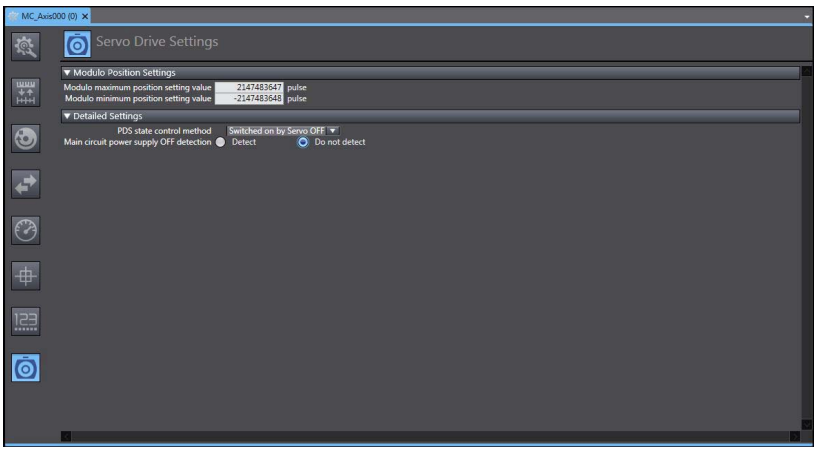


Figure 6-118 EtherCAT integration – OMRON Sysmac NJ | Servo drive settings

PROGRAMMING

For in-depth details see OMRON's operating instruction manual ➔«Sysmac Studio Operation Manual (W504)».

26) In the Multiview Explorer, select ¤Programming¤ \ ¤POUs¤ \ ¤Programs¤ \ ¤Program0¤ \ ¤Section0¤.
Click right on ¤Section0¤ to add a program.

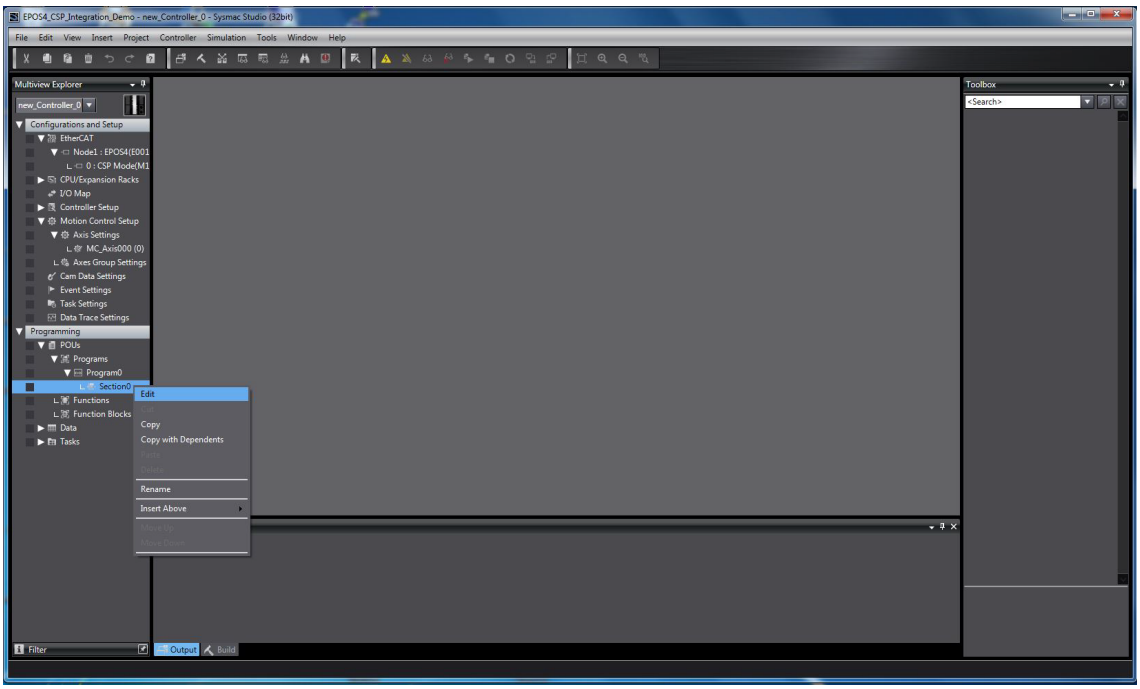


Figure 6-119 EtherCAT integration – OMRON Sysmac NJ | Add program

27) Write a short program as to the following example.

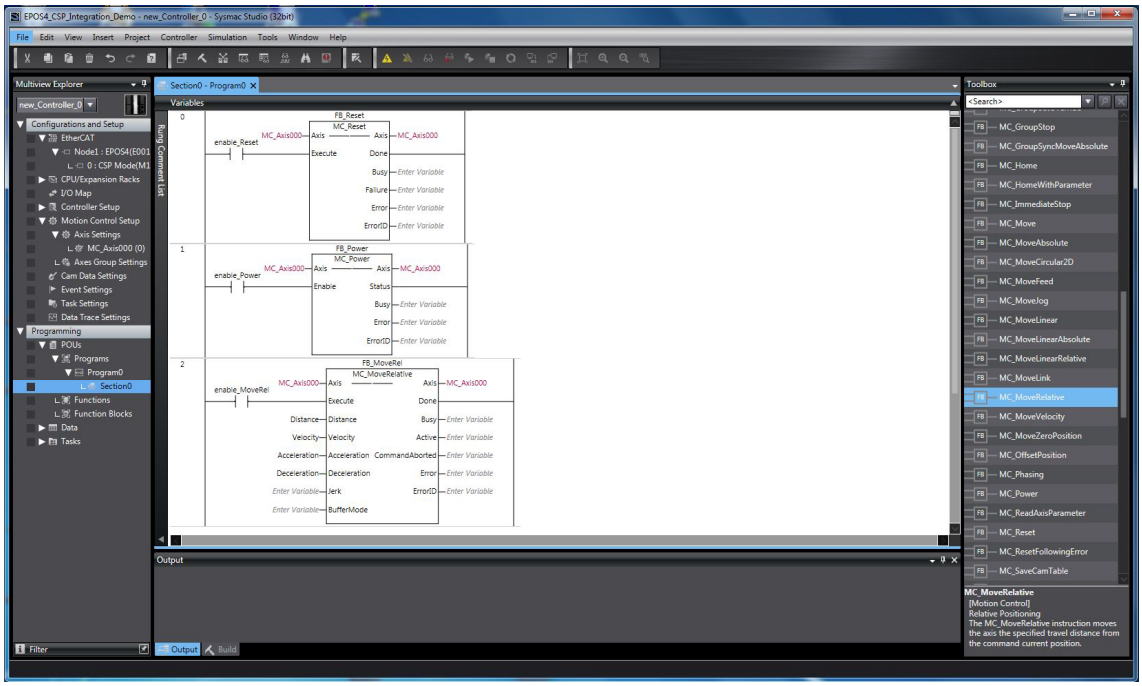


Figure 6-120 EtherCAT integration – OMRON Sysmac NJ | Example program

TASK SETTINGS

28) Go to ¤Program Assignment Settings¤ and assign the application program to the “Primary Task”.

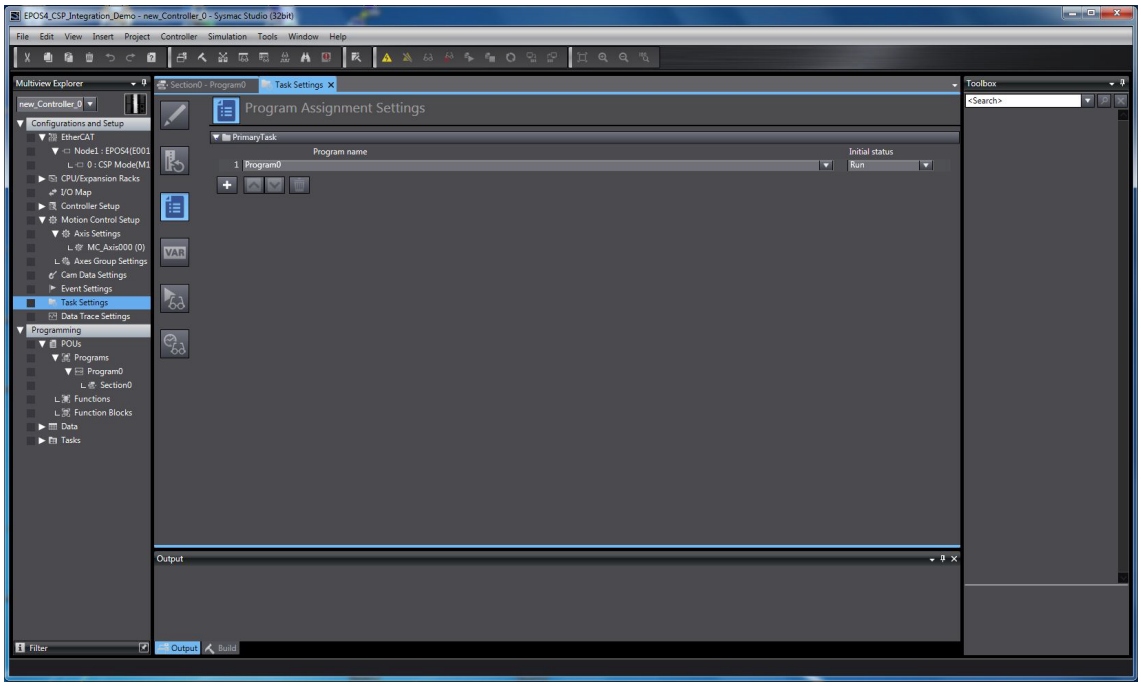


Figure 6-121 EtherCAT integration – OMRON Sysmac NJ | Task settings

29) Go Online and download the program.

30) Click ¤Execute¤ to transfer the program to the controller.

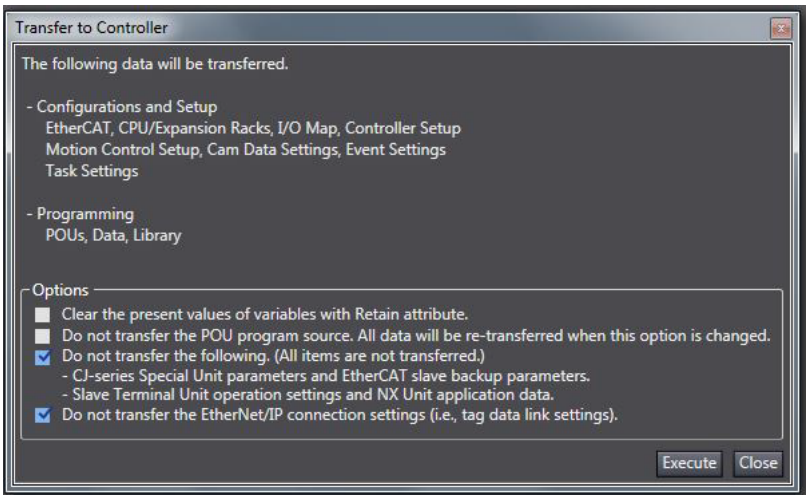


Figure 6-122 EtherCAT integration – OMRON Sysmac NJ | Transfer to controller options

31) Click ¤Yes¤ to confirm.

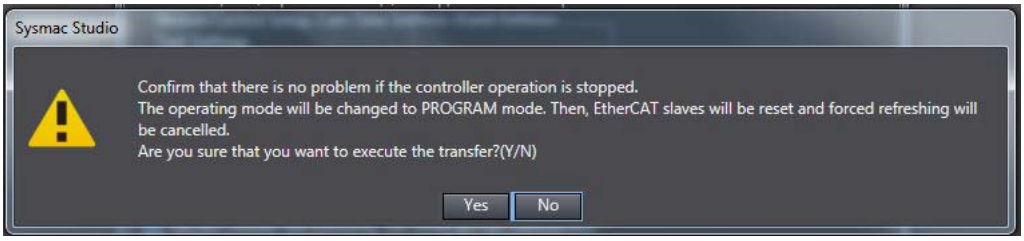


Figure 6-123 EtherCAT integration – OMRON Sysmac NJ | Controller reset

••page intentionally left blank••

7 DEVICE PROGRAMMING

CONTENT

7.1 In Brief

A wide variety of operating modes permit flexible configuration of the drive system by using positioning, velocity, and current regulation. The built-in CANopen interface allows networking to multiple axes drives as well as online commanding by CAN bus master units.

OBJECTIVE

The present application note explains typical commanding sequences for different operating modes based on writing/reading commands to access the Object Dictionary. For detailed information…

- on the objects itself see separate document ➔«EPOS4 Firmware Specification» (subsequently referred to as “FwSpec”),
- on the command structure see separate document ➔«EPOS4 Communication Guide» and ➔«EPOS Studio»; tool “Command Analyzer”.

maxon

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0140h	Firmware Specification Communication Guide
EPOS4 Disk 60/8 CAN EPOS4 Disk 60/8 EtherCAT	688770 688772	0170h or higher 0170h or higher	
EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC EPOS4 Disk 60/12 EtherCAT EPOS4 Disk 60/12 EtherCAT SSC	688775 709859 688777 709862	0170h or higher 0170h or higher 0170h or higher 0170h or higher	
EPOS4 Module 24/1.5 EPOS4 Compact 24/1.5 CAN EPOS4 Compact 24/1.5 EtherCAT	536630 546714 628092	0140h or higher 0140h or higher 0150h or higher	
EPOS4 Module 50/5 EPOS4 Compact 50/5 CAN EPOS4 Compact 50/5 EtherCAT	534130 541718 628094	0140h or higher 0140h or higher 0150h or higher	
EPOS4 Module 50/8 EPOS4 Compact 50/8 CAN EPOS4 Compact 50/8 EtherCAT	504384 520885 605298	0140h or higher 0140h or higher 0140h or higher	
EPOS4 Module 50/15 EPOS4 Compact 50/15 CAN EPOS4 Compact 50/15 EtherCAT	504383 520886 605299	0140h or higher 0140h or higher 0140h or higher	
EPOS4 50/5	546047	0140h or higher	
EPOS4 70/15	594385	0140h or higher	

Table 7-57 Device programming | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.4 or higher

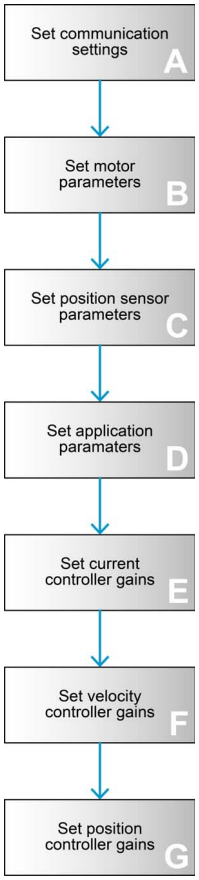
Table 7-58 Device programming | Recommended tools

7.2 First Step

Before the motor will be activated, motor parameters, position sensor parameters, and controller gains must be set. For detailed description ➔FwSpec.

Notes

- *For detailed information on the command structure ➔«EPOS Studio» (command analyzer).*
- *In the later course of the present chapter, the stated units ([inc], [rpm], [rpm/s]) can vary depending on the system units configured by the objects 0x60A8, 0x60A9, and 0x60AA. All stated units correspond to their respective default configuration.*



#	Object name	Object	User value [default value]
A	Node-ID	0x2000-00	User-specific [1]; typically configured by DIP switches
	CAN bit rate	0x2001-00	User-specific [0] (= 1 Mbit/s)
	RS232 bit rate	0x2002-00	User-specific [5] (= 115.2 kBit/s)
B	Motor type	0x6402-00	Motor-specific [10] (= sinusoidal PM BL motor)
	Nominal current	0x3001-01	Motor-specific [mA]
	Output current limit	0x3001-02	User-specific [mA]
	Number of pole pairs	0x3001-03	Motor-specific [1]
	Thermal time constant winding	0x3001-04	Motor-specific [40 x 0.1 s]
	Torque constant	0x3001-05	Motor-specific [µNm/A]
	Max motor speed	0x6080-00	User-specific: Motor or mechanical limits [rpm]
	Max gear input speed	0x3003-03	User-specific: Gear or mechanical limits [rpm]
C	Axis configuration	0x3000-xx	Sensor-specific and system-specific
	Digital incremental encoder 1	0x3010-01 0x3010-02	Encoder-specific: Number of pulses [500 pulses/revolution] Encoder type [0x0001] (= maxon with index)
	SSI absolute encoder	0x3012-xx	SSI encoder-specific
D	Software position limit	0x607D-xx	User-specific [0 inc]
E	Current control parameter set: • Current controller PI gains	0x30A0-xx	Motor-specific and load-specific: Determine optimal parameter using “Regulation Tuning” in «EPOS Studio».
F	Velocity control parameter set: • Velocity controller PI gains • Velocity controller FF gains	0x30A2-xx	
	Velocity observer parameter set	0x30A3-xx	
G	Position control parameter set: • Position controller PID gains • Position controller FF gains	0x30A1-xx	

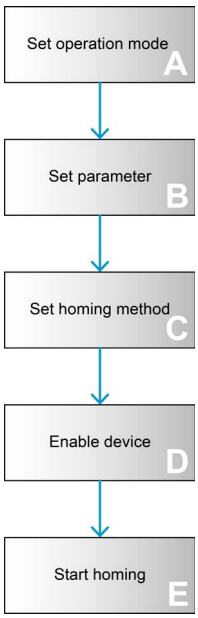
Table 7-59 Device programming | First step

7.3 Homing Mode (HMM)

Note
For details on Controlword bits (0x6040) ➔FwSpec.

START HOMING

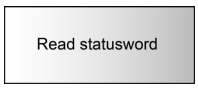
The axis references to an absolute position using the selected homing method.



#	Object name	Object	User value [default value]
A	Modes of operation	0x6060-00	0x06 (Homing Mode)
B	Following error window	0x6065-00	User-specific [2'000 inc]
	Home offset move distance	0x30B1-00	User-specific [0 inc]
	Max profile velocity	0x607F-00	Motor-specific [25'000 rpm]
	Quick stop deceleration	0x6085-00	User-specific [10'000 rpm/s]
	Speed for switch search	0x6099-01	User-specific [100 rpm]
	Speed for zero search	0x6099-02	User-specific [10 rpm]
	Homing acceleration	0x609A-00	User-specific [1'000 rpm/s]
	Home position	0x30B0-00	User-specific [0 inc]
C	Homing method	0x6098-00	Select homing method (for details ➔FwSpec)
D	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
E	Controlword (Start homing)	0x6040-00	0x001F

Table 7-60 Device programming | Homing Mode (Start)

READ STATUS



Object name	Object	User value [default value]
Statusword (Target reached / Homing attained)	0x6041-00	Homing procedure is completed successfully if bit 12 (=“Homing attained”) is set to “1”.

Table 7-61 Device programming | Homing Mode (Read)

STOP HOMING

Object name	Object	User value [default value]
Controlword (Switch on & Enable)	0x6040-00	0x000F
or		
Controlword (Halt homing)	0x6040-00	0x011F
or		
Controlword (Quick stop)	0x6040-00	0x000B

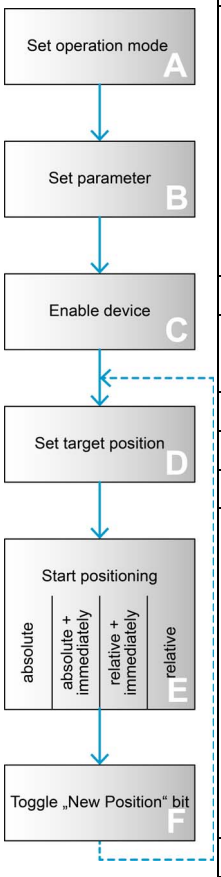
Table 7-62 Device programming | Homing Mode (Stop)

7.4 Profile Position Mode (PPM)

Note
For details on Controlword bits (0x6040) and Statusword bits (0x6041) ➔FwSpec.

SET POSITION

The axis moves to an absolute or relative position using a motion profile.



#	Object name	Object	User value [default value]
A	Modes of operation	0x6060-00	0x01 (Profile Position Mode)
B	Following error window	0x6065-00	User-specific [2'000 inc]
	Max profile velocity	0x607F-00	Motor-specific [50'000 rpm]
	Profile velocity	0x6081-00	Desired velocity [1'000 rpm]
	Profile acceleration	0x6083-00	User-specific [10'000 rpm/s]
	Profile deceleration	0x6084-00	User-specific [10'000 rpm/s]
	Quick stop deceleration	0x6085-00	User-specific [10'000 rpm/s]
	Motion profile type	0x6086-00	User-specific [0]
C	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
D	Target position	0x607A-00	Desired position [inc]
E [a]	Controlword (absolute position)	0x6040-00	0x001F
	or		
	Controlword (absolute position, start immediately)	0x6040-00	0x003F
	or		
	Controlword (relative position, start immediately)	0x6040-00	0x007F
	or		
	Controlword (relative position)	0x6040-00	0x005F
F [b]	Controlword (New Position)	0x6040-00	0x000F (toggle “New Position”)

[a] We recommended to check the status of the drive after an updated Controlword by a request of the Statusword (0x6041). Thereby, a delay of at least 1 ms must be respected before the Statusword is checked. For details on how to check the Statusword consult ➔chapter “7.10 Motion Info” on page 7-128.

[b] Make sure to observe the delay of at least 1 ms in between Controlword commands

Table 7-63 Device programming | Profile Position Mode (Set)

READ STATUS

Read statusword

Object name	Object	User value [default value]
Statusword (Target reached)	0x6041-00	The axis has reached the target position if bit 10 (=”Target reached”) is set to “1” and bit 8 (=”Halt”) of the Controlword (0x6040) was not activated (for details ➔FwSpec).

Table 7-64 Device programming | Profile Position Mode (Read)

STOP POSITIONING

Stop positioning

Object name	Object	User value [default value]
Controlword (Halt profile position mode)	0x6040-00	0x010F
or		
Controlword (Quick stop)	0x6040-00	0x000B

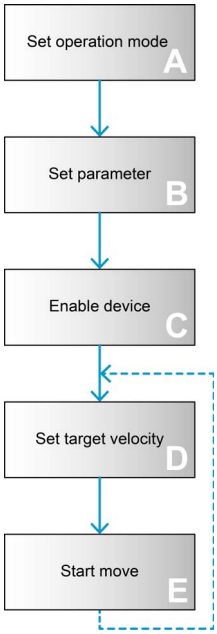
Table 7-65 Device programming | Profile Position Mode (Stop)

7.5 Profile Velocity Mode (PVM)

Note
For details on Controlword bits (0x6040) and Statusword bits (0x6041) ➔FwSpec.

START VELOCITY

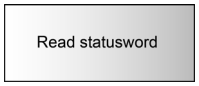
Motor shaft rotates with a certain speed with velocity profile.



#	Object name	Object	User value [default value]
A	Modes of operation	0x6060-00	0x03 (Profile Velocity Mode)
B	Max profile velocity	0x607F-00	Motor-specific [50'000 rpm]
	Profile acceleration	0x6083-00	User-specific [10'000 rpm/s]
	Profile deceleration	0x6084-00	User-specific [10'000 rpm/s]
	Quick stop deceleration	0x6085-00	User-specific [10'000 rpm/s]
	Motion profile type	0x6086-00	User-specific [0]
C	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
D	Target velocity	0x60FF-00	Desired velocity [rpm]
E	Controlword	0x6040-00	0x000F

Table 7-66 Device programming | Profile Velocity Mode (Start)

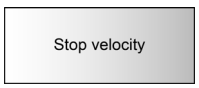
READ STATUS



Object name	Object	User value [default value]
Statusword (Target velocity reached)	0x6041-00	Target velocity is reached if bit 10 is set (for details ➔FwSpec).

Table 7-67 Device programming | Profile Velocity Mode (Read)

STOP VELOCITY



Object name	Object	User value [default value]
Controlword (Halt profile velocity mode)	0x6040-00	0x010F
or		
Controlword (Quick stop)	0x6040-00	0x000B

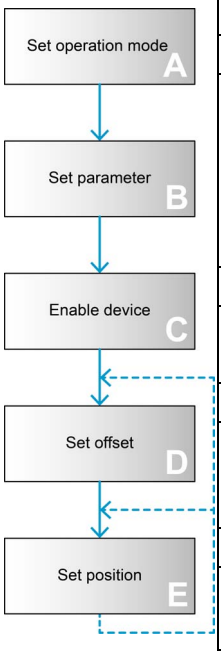
Table 7-68 Device programming | Profile Velocity Mode (Stop)

7.6 Cyclic Synchronous Position Mode (CSP)

Note
For details on Controlword bits (0x6040) ➔FwSpec.

SET POSITION

The axis moves to an absolute or relative position.



#	Object name	Object	User value [default value]
A	Modes of operation	0x6060-00	0x08 (Cyclic Synchronous Position Mode)
B	Max motor speed	0x6080-00	User-specific; motor or mechanical limits [rpm]
	Max gear input speed	0x3003-03	User-specific: Gear or mechanical limits [rpm]
	Profile deceleration [a]	0x6084-00	User-specific [10'000 rpm/s]
	Quick-stop deceleration [a]	0x6085-00	User-specific [10'000 rpm/s]
	Interpolation time period [b]	0x60C2-xx	Master's SYNC period (Cycle ticks) [1 ms]
C	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
D	Torque offset	0x60B2-00	Torque offset [per thousand of "Motor rated torque"; 0x6076]
	Position offset	0x60B0-00	Position offset [0 inc]
E	Target position	0x607A-00	Desired position [inc]

[a] Deceleration values are used for stopping only, they are not used for normal operation

[b] The «Interpolation time period value» must be configured to correspond with the master's synchronized PDO command cycle that updates the CSP set value, respectively the CSV set value.

If a value of "0" (zero) is configured, the EPOS4 immediately takes the new set value and adapts the position (in case of CSP mode), respectively the velocity (in case of CSV mode) to it within the next control cycle (i.e. 0.4 ms). Afterwards it holds this set value until the next set value of the master is received. This results in an interrupted and noisy motion if the master just provides new set values at cycle rates of 1 ms, 2 ms, or even lower.

If the «Interpolation time period value» is configured properly based on the master's PDO cycle time, the EPOS4 interpolates the new set value in between the period. This results in a smooth motion and less noisy control result.

Table 7-69 Device programming | Cyclic Synchronous Position Mode (Set)

STOP MOTION

Object name	Object	User value [default value]
Controlword (Quick stop)	0x6040-00	0x000B

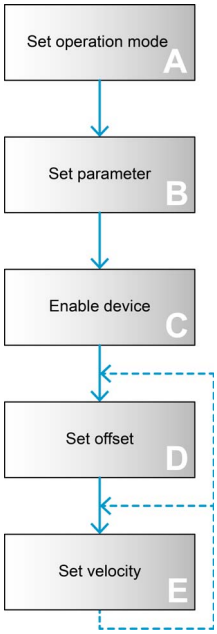
Table 7-70 Device programming | Cyclic Synchronous Position Mode (Stop)

7.7 Cyclic Synchronous Velocity Mode (CSV)

Note
For details on Controlword bits (0x6040) ➔FwSpec.

SET VELOCITY

The axis moves with the commanded velocity.



#	Object name	Object	User value [default value]
A	Modes of operation	0x6060-00	0x09 (Cyclic Synchronous Velocity Mode)
B	Max motor speed	0x6080-00	User-specific; motor or mechanical limits [rpm]
	Max gear input speed	0x3003-03	User-specific: Gear or mechanical limits [rpm]
	Profile deceleration [a]	0x6084-00	User-specific [10'000 rpm/s]
	Quick-stop deceleration [a]	0x6085-00	User-specific [10'000 rpm/s]
	Interpolation time period [b]	0x60C2-xx	Master's SYNC period (Cycle ticks) [1 ms]
C	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
D	Velocity offset	0x60B1-00	Velocity offset [rpm]
E	Target velocity	0x60FF-00	Desired velocity [rpm]

[a] Deceleration values are used for stopping only, they are not used for normal operation

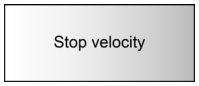
[b] The «Interpolation time period value» must be configured to correspond with the master's synchronized PDO command cycle that updates the CSP set value, respectively the CSV set value.

If a value of "0" (zero) is configured, the EPOS4 immediately takes the new set value and adapts the position (in case of CSP mode), respectively the velocity (in case of CSV mode) to it within the next control cycle (i.e. 0.4 ms). Afterwards it holds this set value until the next set value of the master is received. This results in an interrupted and noisy motion if the master just provides new set values at cycle rates of 1 ms, 2 ms, or even lower.

If the «Interpolation time period value» is configured properly based on the master's PDO cycle time, the EPOS4 interpolates the new set value in between the period. This results in a smooth motion and less noisy control result.

Table 7-71 Device programming | Cyclic Synchronous Velocity Mode (Set)

STOP MOTION



Object name	Object	User value [default value]
Target velocity	0x60FF-00	0x0000
or		
Controlword (Quick stop)	0x6040-00	0x000B

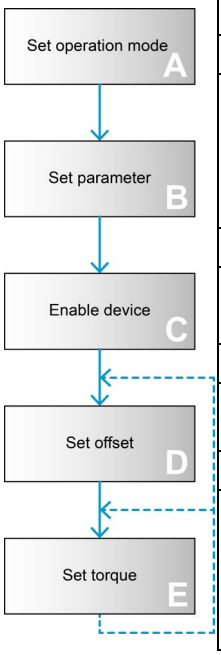
Table 7-72 Device programming | Cyclic Synchronous Velocity Mode (Stop)

7.8 Cyclic Synchronous Torque Mode (CST)

Note
For details on Controlword bits (0x6040) ➔FwSpec.

SET TORQUE

Applies a certain torque (that is: torque = current x torque constant) to the motor winding.



#	Object name	Object	User value [default value]
A	Modes of operation	0x6060-00	0x0A (Cyclic Synchronous Torque Mode)
B	Max motor speed	0x6080-00	User-specific; motor or mechanical limits [rpm]
	Max gear input speed	0x3003-03	User-specific: Gear or mechanical limits [rpm]
	Profile deceleration [a]	0x6084-00	User-specific [10'000 rpm/s]
	Quick-stop deceleration [a]	0x6085-00	User-specific [10'000 rpm/s]
C	Controlword (Shutdown)	0x6040-00	0x0006
	Controlword (Switch on & Enable)	0x6040-00	0x000F
D	Torque offset	0x60B2-00	Torque offset [per thousand of “Motor rated torque”; 0x6076]
E	Target torque	0x6071-00	Desired torque [per thousand of “Motor rated torque”; 0x6076]

[a] Deceleration values are used for stopping only, they are not used for normal operation

Table 7-73 Device programming | Cyclic Synchronous Torque Mode (Set)

STOP MOTION

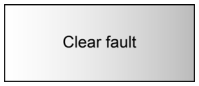
Object name	Object	User value [default value]
Target torque	0x6071-00	0x0000
or		
Controlword (Quick stop)	0x6040-00	0x000B

Table 7-74 Device programming | Cyclic Synchronous Torque Mode (Stop)

7.9 State Machine

CLEAR FAULT

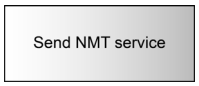
Resetting “Fault” condition sends the Controlword with value 0x0080.



Object name	Object	User value [default value]
Controlword (Fault reset)	0x6040-00	0x0000;0x0080

Table 7-75 Device programming | State machine (clear fault)

SEND NMT SERVICE



Object name	Object	User value [default value]
Node ID (Unique Node ID or “0” (zero) for all nodes)		
Command specifier:	0x01	Start remote node
	0x02	Stop remote node
	0x80	Enter pre-operational
	0x81	Reset node
	0x82	Reset communication

Table 7-76 Device programming | State machine (send NMT service)

7.10 Motion Info

GET MOVEMENT STATE

Read statusword

Object name	Object	User value [default value]
Read statusword	0x6041-00	The bits are partly depending on the operating mode (for details ➔FwSpec)

Table 7-77 Device programming | Motion info (Get movement state)

READ POSITION

Read position

Object name	Object	User value [default value]
Read position	0x6064-00	Position actual value [inc]

Table 7-78 Device programming | Motion info (Read position)

READ VELOCITY

Read velocity

Object name	Object	User value [default value]
Read velocity actual value averaged	0x30D3-01	Velocity actual value averaged [rpm]

Table 7-79 Device programming | Motion info (Read velocity)

READ TORQUE

Read torque

Object name	Object	User value [default value]
Read torque actual value	0x6077-00	Torque actual value [per thousand of “Motor rated torque”; 0x6076]

Table 7-80 Device programming | Motion info (Read torque)

7.11 Utilities

STORE ALL PARAMETERS

Saves all parameters.

Object name	Object	User value [default value]
Save all parameters	0x1010-01	0x65766173 “save”

Table 7-81 Device programming | Utilities (Store all parameters)

RESTORE ALL DEFAULT PARAMETERS

Restores all parameters to factory settings.

Object name	Object	User value [default value]
Restore all default parameters	0x1011-01	0x64616F6C “load”

Table 7-82 Device programming | Utilities (Restore all default parameters)

maxon

••page intentionally left blank••

8 ADJUSTMENT OF SSI COMMUTATION OFFSET VALUE

CONTENT

8.1 In Brief

EPOS4 positioning controllers offer the possibility to use SSI absolute encoders for commutation of BLDC motors without Hall sensor signals.

If you are using a combination of a maxon motor with a maxon SSI absolute encoder, the “zero” position of the encoder is factory-aligned with the rotor position. Hence, the SSI commutation offset value is “0” (zero) and no further actions will be required.

If you are using third party products, the encoder’s “zero” position is not necessarily aligned with the rotor position. Thus, manual adjustment of the SSI commutation offset value will be required during commissioning.

The present application note will guide you through the necessary steps.

Important

- *The described adjustment is only valid for SSI absolute encoders in combination with EPOS4 positioning controllers.*
- *If a gear is present in the system, the mounting position of the SSI encoder shall be on the motor axis. Otherwise, the encoder cannot be used for commutation.*
- *Certain SSI absolute encoders offer the possibility of programming an “Offset” or “Addition” value to the encoder itself. If this case, make sure that no value is stored for your encoder in use.*
- *Do not execute any homing procedure with the EPOS4 positioning controller prior having completed the described adjustment. If you accidentally did execute homing already, execute a «Restore all default parameters» command (object 0x1011), first.*
- *Follow the instructions in given order.*

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0120h	Firmware Specification
EPOS4 Disk 60/8 CAN EPOS4 Disk 60/8 EtherCAT	688770 688772	0170h or higher 0170h or higher	
EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC EPOS4 Disk 60/12 EtherCAT EPOS4 Disk 60/12 EtherCAT SSC	688775 709859 688777 709862	0170h or higher 0170h or higher 0170h or higher 0170h or higher	
EPOS4 Module 24/1.5 EPOS4 Compact 24/1.5 CAN EPOS4 Compact 24/1.5 EtherCAT	536630 546714 628092	0120h or higher 0120h or higher 0150h or higher	
EPOS4 Module 50/5 EPOS4 Compact 50/5 CAN EPOS4 Compact 50/5 EtherCAT	534130 541718 628094	0120h or higher 0120h or higher 0150h or higher	
EPOS4 Module 50/8 EPOS4 Compact 50/8 CAN EPOS4 Compact 50/8 EtherCAT	504384 520885 605298	0120h or higher 0120h or higher 0140h or higher	
EPOS4 Module 50/15 EPOS4 Compact 50/15 CAN EPOS4 Compact 50/15 EtherCAT	504383 520886 605299	0120h or higher 0120h or higher 0140h or higher	
EPOS4 50/5	546047	0120h or higher	
EPOS4 70/15	594385	0140h or higher	

Table 8-83 Adjustment of SSI commutation offset value | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.4 or higher

Table 8-84 Adjustment of SSI commutation offset value | Recommended tools

8.2 Preconditions

EPOS Studio

1) Make sure you installed **«EPOS Studio» version 3.4 (or later)** on your PC.
   If not the case, download the latest version here: ➔http://epos.maxongroup.com
2) Connect «EPOS Studio» with your EPOS4 via USB, RS232, or CANopen.

EPOS4 Positioning Controller

3) Make sure you installed **«EPOS4 Firmware» version 0x0140 (or later)** on your EPOS4.
   If not the case, download the latest version here: ➔http://epos.maxongroup.com
4) Make sure that your EPOS4 is **powered with the power supply voltage**.

Motor and SSI absolute encoder

5) Make sure that **motor and encoder are correctly wired to the EPOS4**.
6) If you are not quite sure what parameters are set, execute the command «Restore all default parameters» (➔Figure 8-124) before continuing.

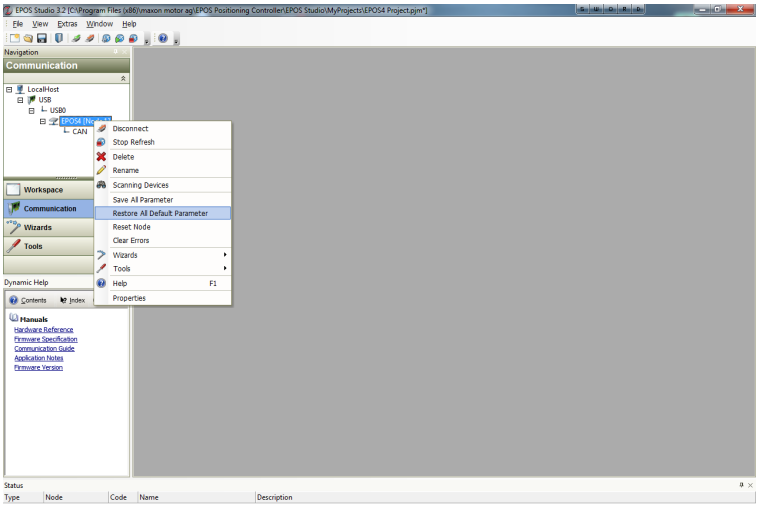


Figure 8-124 Adjustment of SSI commutation offset value | Restore all default parameters

7) Configure the data for motor and SSI absolute encoder using «EPOS Studio» \ ¤Startup¤ wizard \ ¤Motor/Sensors¤ \ ¤Motor¤ or ¤Sensors¤ respectively (➔Figure 8-125).

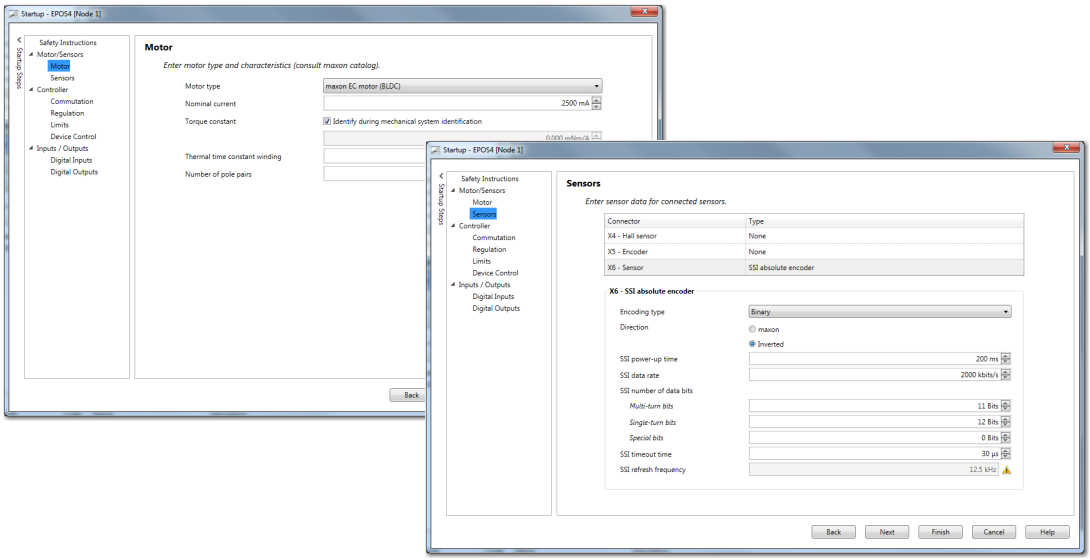


Figure 8-125 Adjustment of SSI commutation offset value | Set motor and sensor data

Incorrect configuration can cause damage to the motor
Be aware that incorrect configuration settings can permanently damage the motor during the subsequent alignment process.
Previously to the next step, make sure that you have configured the motor data (including «Nominal current») for the exact type of motor you are using.

8) Make sure that the motor can run freely and check that the brake, if any, does not engage.
9) Verify the SSI absolute encoder direction:
   a) Open «EPOS Studio» and select the tool ¤Profile Position Mode¤.
   b) Turn the motor shaft by hand counterclockwise (CCW) as seen towards the motor's mounting flange (➔Figure 8-126).

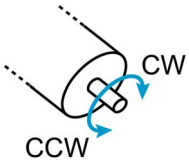


Figure 8-126 Adjustment of SSI commutation offset value | Check sense of rotation

c) The indication «Position actual value» (➔Figure 8-127) must increase.

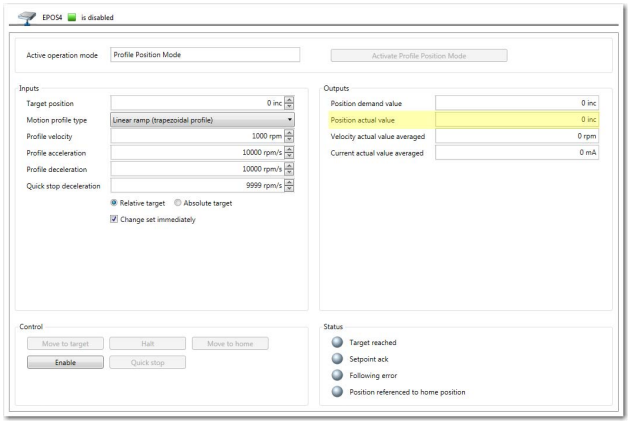


Figure 8-127 Adjustment of SSI commutation offset value | Check sense of rotation

d) If the indication decreases, open the ¤Startup¤ wizard \ ¤Motor/Sensors¤ and select ¤Sensors¤. Toggle the sense of rotation by changing the active checkbox ¤Direction¤ (➔Figure 8-128).
Click ¤Finish¤ to close the wizard and verify the correct setting as described above.

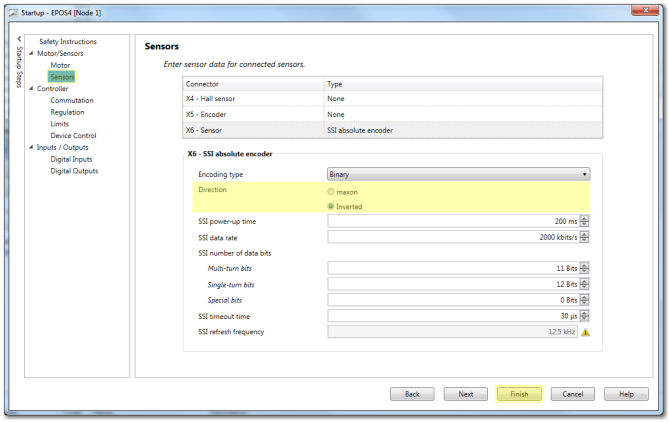


Figure 8-128 Adjustment of SSI commutation offset value | Toggle sense of rotation

No indication or odd counting behavior
If the indication «Position actual value» does not change at all, or in case of an odd counting behavior: Check the SSI absolute encoder for correct wiring and verify the configuration settings.

8.3 Determination of the «SSI commutation offset value»

To determine the «SSI commutation offset value», consider the following criteria:

- Number of pole pairs (0x3001-03)
- SSI single-turn bits (part of configuration for 0x3012-02; Bit 8…15)
- SSI encoder direction (part of configuration for 0x3012-03; Bit 4)
- SSI position raw value (0x3012-09)

1) Open the ¤Startup¤ wizard \ ¤Motor/Sensors¤ and select ¤Motor¤.
Change the Motor type to ¤maxon DC motor¤.
Click ¤Finish¤ to close the wizard.

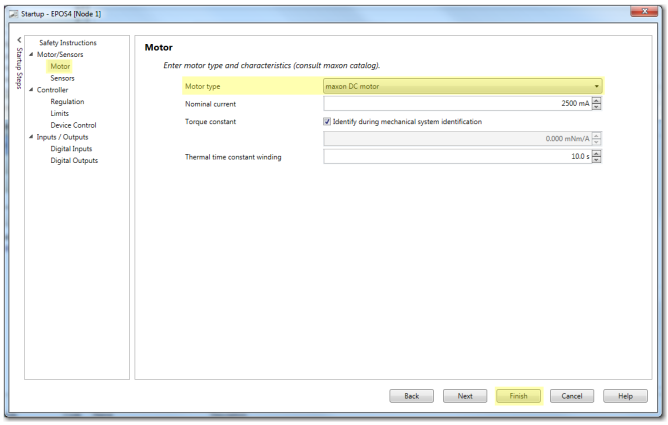


Figure 8-129 Adjustment of SSI commutation offset value | Select motor type maxon DC motor

2) Open the tool ¤Cyclic Sync Torque Mode¤.
Select ¤Activate Cyclic Synchronous Torque Mode¤ and click ¤Enable¤.

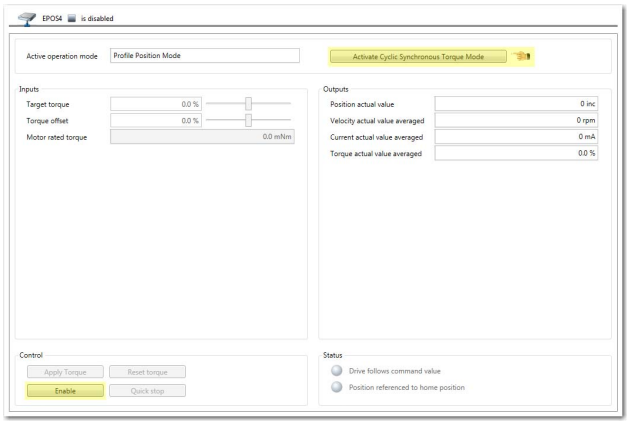


Figure 8-130 Adjustment of SSI commutation offset value | Activate CST

3) Set the value «Target torque» to 30.0% and click ¤Apply Torque¤.
The motor will now be aligned but it will not continually rotate.
Verify that the indication of «Current actual value averaged» shows 30% of the configured «Nominal current» for the motor you are using.

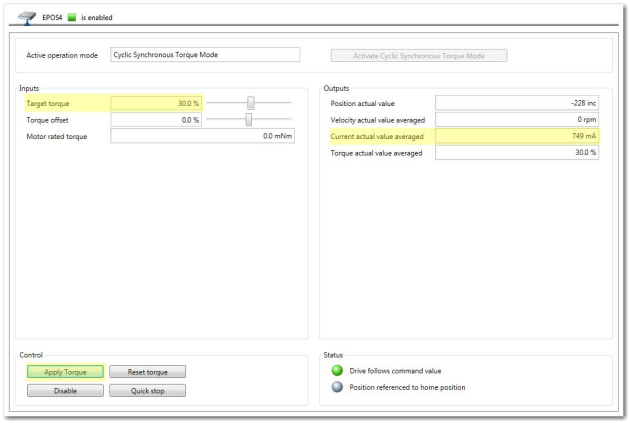


Figure 8-131 Adjustment of SSI commutation offset value | Apply target torque

Best Practice

If the motor is connected to a heavy load or to a system with high mechanical friction, more than 30% «Target torque» might be needed for motor alignment.

4) Read object 0x3012-09 «SSI position raw value» from the Object Dictionary and **note down the value for future use**.
Click ¤Disable¤.

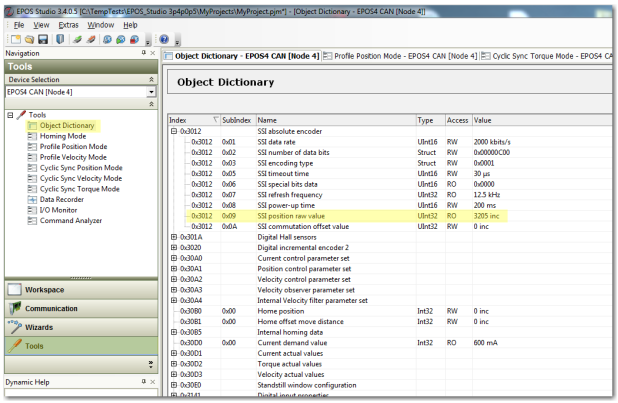


Figure 8-132 Adjustment of SSI commutation offset value | Read SSI position raw value

5) Determine the «SSI commutation offset value» as follows (for detailed information you might wish to also check on the ➔“Calculation Example” on page 8-141):

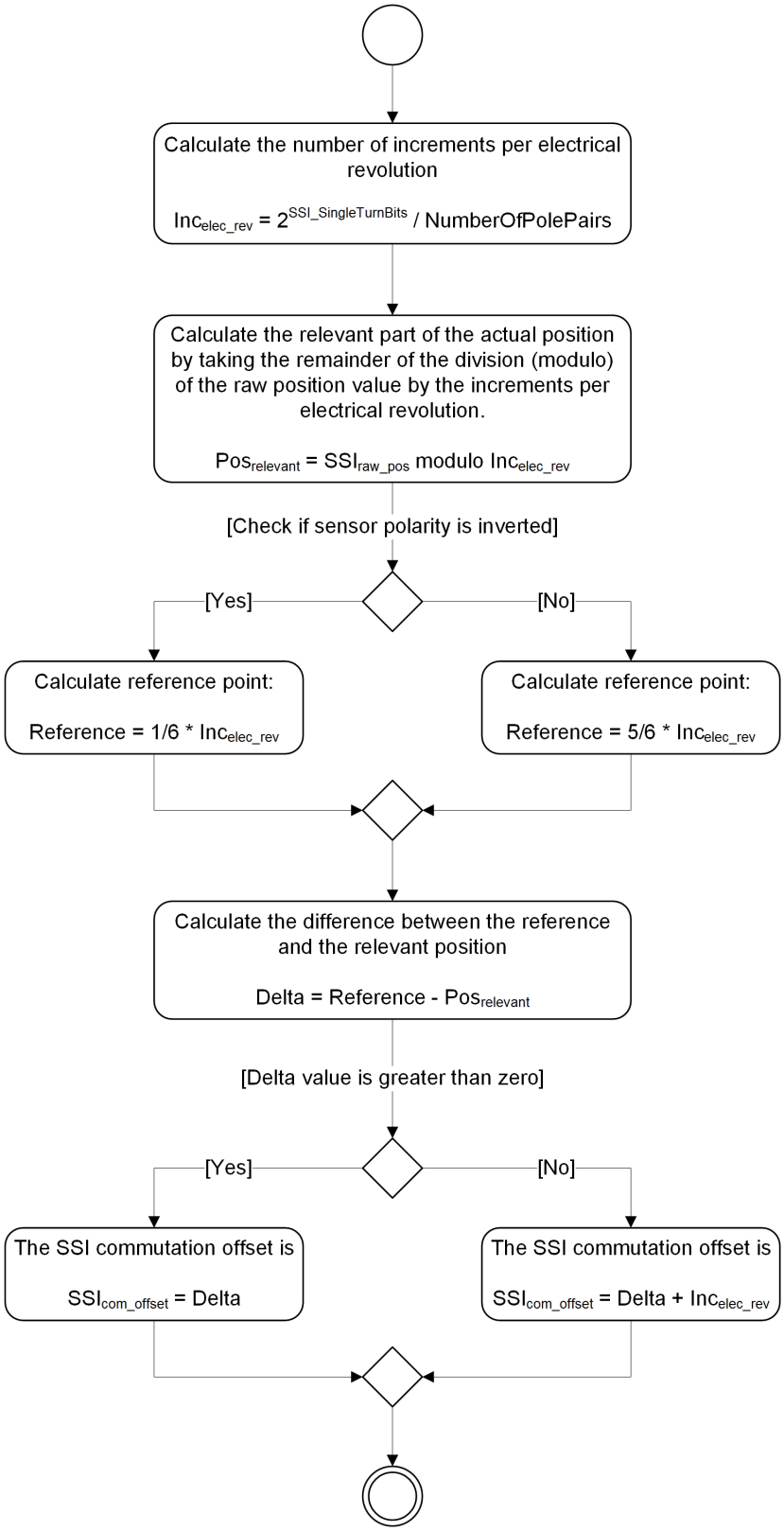


Figure 8-133 Adjustment of SSI commutation offset value | Determine SSI commutation offset value

6) Open the ¤Startup¤ wizard \ ¤Motor/Sensors¤ and select ¤Motor¤.
Change the Motor type to ¤maxon EC motor (BLDC)¤.

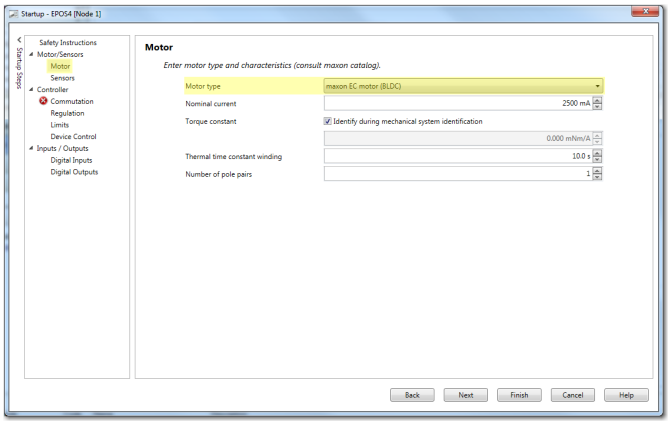


Figure 8-134 Adjustment of SSI commutation offset value | Select motor type maxon EC motor

7) Switch to ¤Commutation¤ and insert the above calculated value to «SSI commutation offset value».
Verify all parameters.
Click ¤Finish¤ to close the wizard.

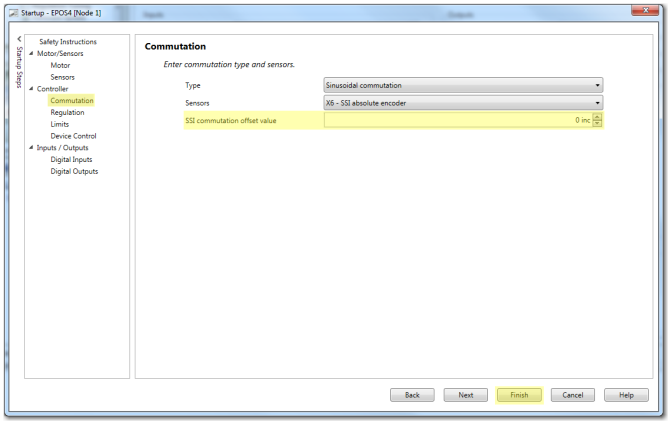


Figure 8-135 Adjustment of SSI commutation offset value | Set SSI commutation offset value

8) Open the ¤Regulation Tuning¤ wizard and complete all tuning steps.

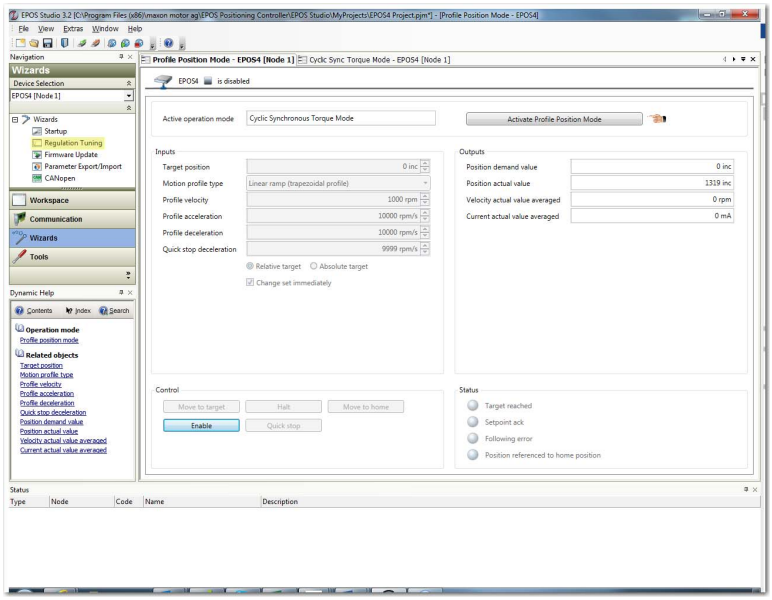


Figure 8-136 Adjustment of SSI commutation offset value | Identify parameters

9) Run the motor in «Profile Velocity Mode».
If «Current actual value averaged» shows an unusual high current, or if the velocity is not according the value set, readjustment the «SSI commutation offset value» and repeat above described procedure.

8.4 Calculation Example

The following values are given as an example:

- Number of pole pairs (0x3001-03) — 7
- SSI single-turn bits (part of configuration for 0x3012-02; Bit 8…15) — 12 bit
- SSI encoder direction (part of configuration for 0x3012-03; Bit 4) — maxon (0x0001h)
- SSI position raw value (0x3012-09) — 3'205 inc

$$Inc_{\text{elec\_rev}} = \frac{2^{\text{SSI\_SingleTurnBits}}}{NumberOfPolePairs} = \frac{2^{12}}{7}$$

$$Inc_{\text{elec\_rev}} = 585.1inc$$

$$Pos_{\text{relevant}} = 3205\text{inc Modulo } 585.1\text{inc} = 279.5inc$$

Note: Modulo finds the remainder after an integer division.

Example:

$$\frac{3205inc}{585.1inc} = 5.4777$$

For this follows: The integer value of the division is 5, therefore…

$$5 \cdot 585.1 = 2925.5inc$$

For this follows: The remainder of the integer division is…

$$3205inc - 2925.5inc = 279.5inc$$

0x3012-03; Bit4 = 0
(0 = maxon (not inverted) / 1 = inverted)

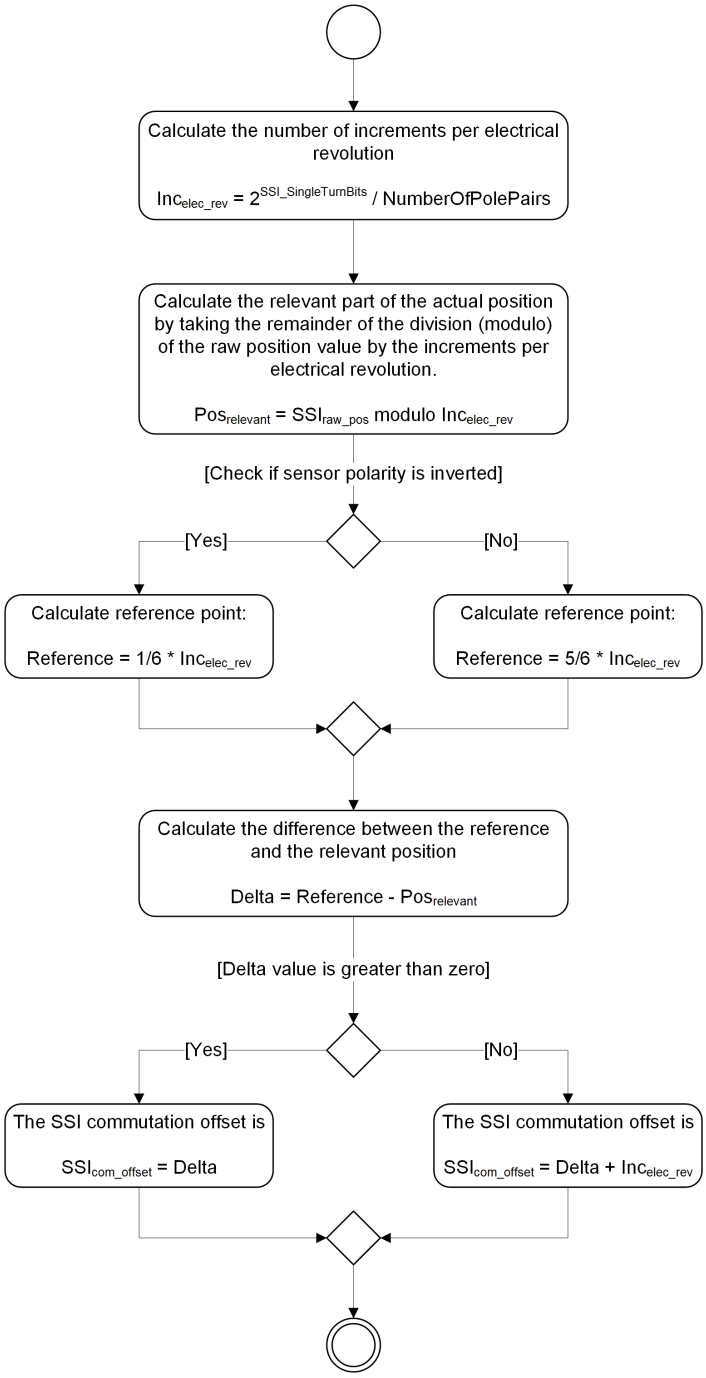
$$Reference = \frac{5}{6} \cdot 585.1inc = 487.6inc$$

$$Delta = Reference - Pos_{relevant}$$

$$Delta = 487.6inc - 279.5inc = 208inc$$

208inc > 0:

$$SSI_{\text{COM\_offset}} = Delta = 208inc$$



••page intentionally left blank••

9 SAFE TORQUE OFF (STO) FUNCTIONALITY; NOT CERTIFIED

CONTENT

9.1 In Brief

The EPOS4 offers the Safe Torque Off (STO) safety feature based on IEC/EN 61800-5-2.

The present application note explains how to setup and configure the EPOS4 controller for the STO functionality.

Pin numbering in the diagrams shown is related to EPOS4 controllers that feature connectors.

Non-certified STO functionality
The implemented STO functionality will not be certified.

9.2 Precautionary Measures

WARNING

Risk of Injury
Operating the device without the full compliance of all relevant safety regulations and/or neglecting the basic working principle of Safe Torque Off (STO) may cause serious injuries!

- *Carry out a comprehensive and thorough risk assessment covering the entire safety system and all safety-relevant aspects to ensure that the STO function will fulfill all relevant safety requirements of the application.*
- *The STO function **does not** cut the power supply to the drive and **does not** provide electrical isolation.*
- *The STO function **can prevent** unexpected motor rotation of an electronically commutated motor (EC motor, BLDC motor, brushless DC motor) in a save manner. Even in error condition with one or more short-circuited power stage transistors, an electronically commutated motor will not be able to generate torque over a relevant rotation angle.*
- *Vice versa, the STO function **cannot prevent** unexpected motor rotation of a mechanically commutated motor (DC motor, brushed motor) in a safe manner. Despite of the STO functionality, an error condition of short-circuited power stage transistors may lead to unexpected motor rotation.*

9.3 Overview

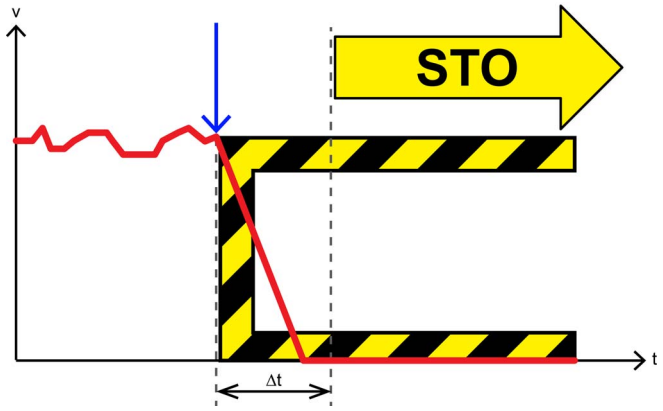


Figure 9-137 Safe Torque Off (STO) | Working principle

The STO function is the most common and basic drive-integrated safety function. It ensures that no torque-generating energy can continue to act on a motor and prevents unintentional starting.

STO has the immediate effect that the drive can no longer supply any torque-generating energy. STO can be used whenever the drive will be brought to a standstill in a sufficiently short time by load torque or friction, or if coasting down of the drive is not relevant to safety. STO enables safe working when, for example, the protective door is open (restart interlock) and has a wide range of uses in machinery with moving axes (such as handling or conveyor systems).

Mechanical brakes must be used if output shafts of motors or gearboxes are affected by forces that could trigger a movement once the motor has been shut down. Possible applications are vertical axes or motors with high inertia.

The STO function can be utilized to perform a safe stop according to IEC/EN 60204-1, stop category 0 (uncontrolled stop by immediate shut-down of the power supply to the actuators).

9.4 Functional Diagram

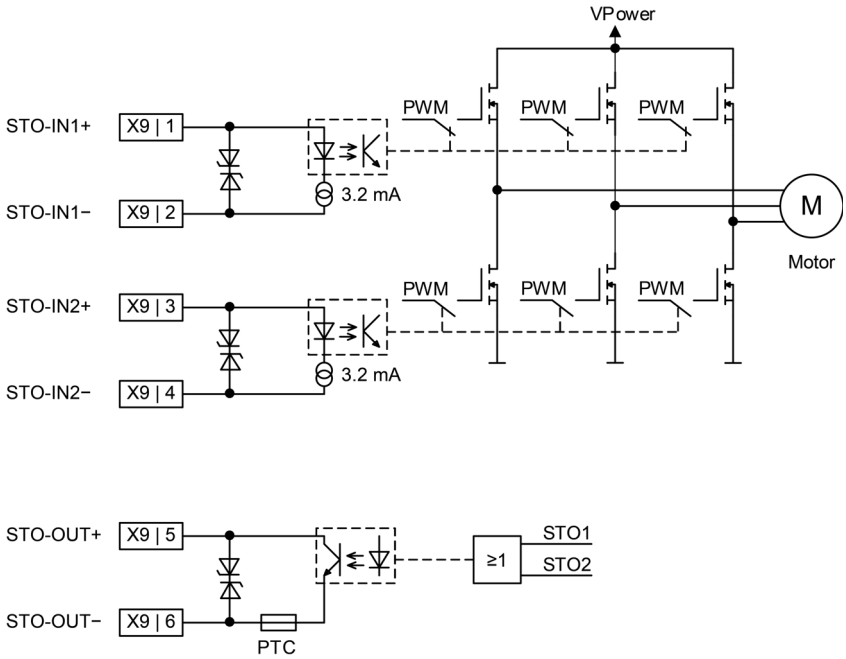


Figure 9-138 Safe Torque Off (STO) | Functional diagram

Interrupting the current to either STO1 or STO2 input will disable the drive output. Thus, the power supply to the motor is cut by stopping the switching process of the output transistors in a safe way.

The STO output is activated when either STO1 or STO2 input is powered. For details on the STO logic states ➔Table 9-87.

9.5 STO Idle Connector

In order to activate the power stage, **either** both STO inputs must be powered **or** the «STO Idle Connector» (520860) must be plugged.

Do not use the activation voltage V_{STO} (+5 VDC) for any other purpose.

The «STO Idle Connector» is included with every EPOS4 controller that features connectors.

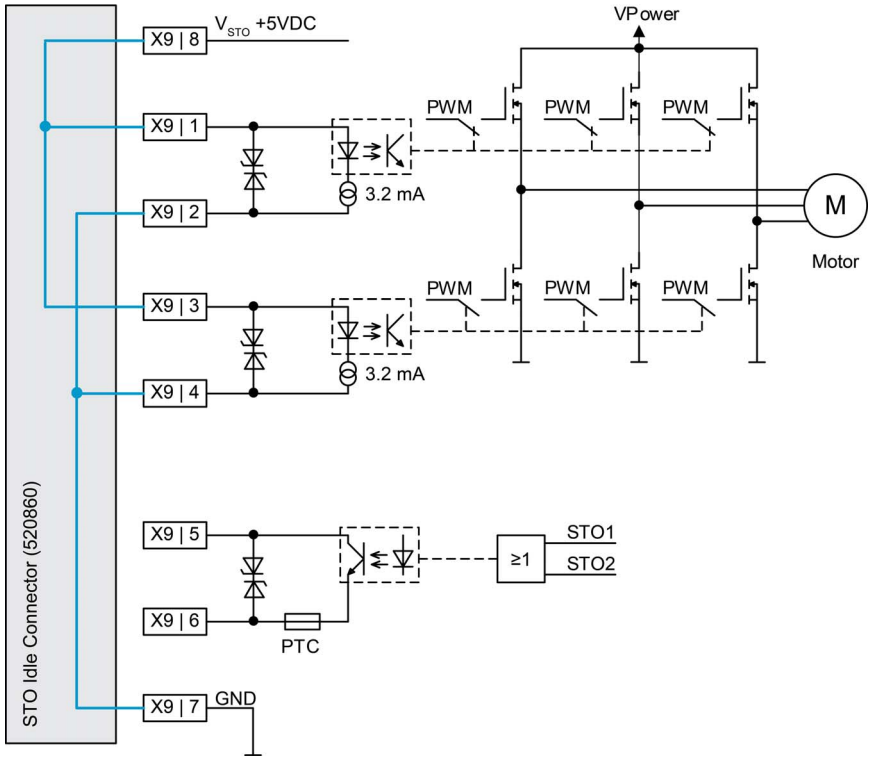


Figure 9-139 Safe Torque Off (STO) | STO Idle Connector

9.6 STO Inputs 1 & 2

9.6.1 Specifications

Safe Torque Off inputs 1…2	
Circuit type	Optically isolated input
Input voltage	0…+30 VDC
Max. input voltage	±30 VDC
Logic 0	<1.0 VDC
Logic 1	>4.5 VDC
Input current at logic 1	>2 mA @ 5 VDC typically 3.2 mA @ 24 VDC
Reaction time	<25 ms

Table 9-85 STO input specification

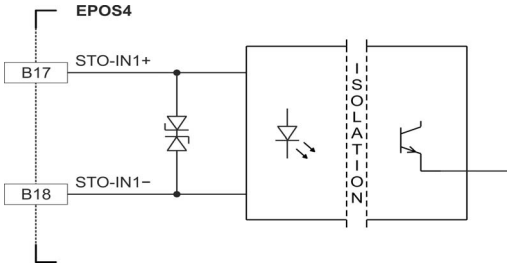


Figure 9-140 Safe Torque Off (STO) | STO-IN1 circuit (analogously valid for STO-IN2)

9.6.2 Test Pulses

The STO1 and STO2 inputs are designed for use with fail-safe output terminals with test pulses.

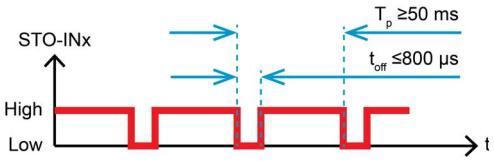


Figure 9-141 Safe Torque Off (STO) | Test pulses

Test pulses that do not fulfill the stated specifications for T_p and t_{off} can have a negative impact on the power stage gate control and can lead to unpredictable behavior.

9.6.3 Input Current

To achieve a fail-safe current measurement supervision on the output terminal, the current threshold must be lower than the typical STO input current (3.2 mA @ 24 VDC).

9.7 STO Output

9.7.1 Specifications

Safe Torque Off output	
Circuit type	Optically isolated output with self-resetting short-circuit protection
Max. input voltage	±30 VDC
Max. load current	15 mA
Leakage current	<10 µA @ +30 VDC
Max. voltage drop	1.3 V @ 2 mA 2.5 V @ 15 mA

Table 9-86 STO output specification

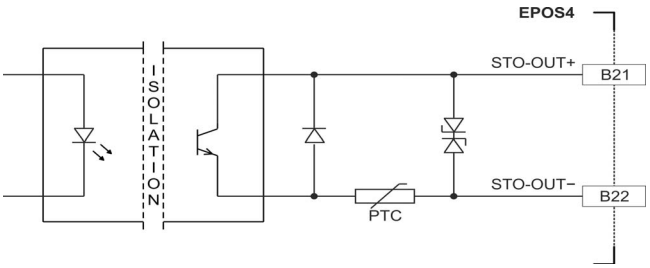


Figure 9-142 Safe Torque Off (STO) | STO-OUT circuit

9.7.2 Diagnostics

The STO output is used for proof test of the EPOS4's internal STO functionality. Thereby, the proof test must be triggered by an external logic.

Proof test is essential to reveal any dangerous, undetected failure after a given period of time.

STO logic state			
STO-IN1	**STO-IN2**	**STO-OUT**	**Power Stage**
0	0	open	inactive
1	0	closed	inactive
0	1	closed	inactive
1	1	closed	active

Table 9-87 Safe Torque Off (STO) | Logic state

For diagnostics, maintain the reaction time of <25 ms between STO input state change and the STO output state change.

••page intentionally left blank••

10 DUAL LOOP CONTROL

CONTENT

10.1 In Brief

In applications where gearheads (subsequently called “gears”), spindles, or belt systems are used to transmit the motor rotation to the load, imperfections of these transmission elements do have an influence on the performance of the load position control. If very accurate load positioning is required, it is common practice to add an additional encoder at the load side to measure the exact load position. Such drive systems contain two encoders; one on the motor shaft (so-called “auxiliary encoder”) and one on the load side (so-called “main encoder”).

SCOPE

Hardware	Order #	Firmware version	Reference
EPOS4		0150h	Firmware Specification
EPOS4 Disk 60/8 CAN EPOS4 Disk 60/8 EtherCAT	688770 688772	0170h or higher 0170h or higher	
EPOS4 Disk 60/12 CAN EPOS4 Disk 60/12 CAN SSC EPOS4 Disk 60/12 EtherCAT EPOS4 Disk 60/12 EtherCAT SSC	688775 709859 688777 709862	0170h or higher 0170h or higher 0170h or higher 0170h or higher	
EPOS4 Module 24/1.5 EPOS4 Compact 24/1.5 CAN EPOS4 Compact 24/1.5 EtherCAT	536630 546714 628092	0150h or higher 0150h or higher 0150h or higher	
EPOS4 Module 50/5 EPOS4 Compact 50/5 CAN EPOS4 Compact 50/5 EtherCAT	534130 541718 628094	0150h or higher 0150h or higher 0150h or higher	
EPOS4 Module 50/8 EPOS4 Compact 50/8 CAN EPOS4 Compact 50/8 EtherCAT	504384 520885 605298	0150h or higher 0150h or higher 0150h or higher	
EPOS4 Module 50/15 EPOS4 Compact 50/15 CAN EPOS4 Compact 50/15 EtherCAT	504383 520886 605299	0150h or higher 0150h or higher 0150h or higher	
EPOS4 50/5	546047	0150h or higher	
EPOS4 70/15	594385	0150h or higher	

Table 10-88 Dual loop control | Covered hardware and required documents

TOOLS

Tools	Description
Software	«EPOS Studio» Version 3.5 or higher

Table 10-89 Dual loop control | Recommended tools

10.2 Overview

With two encoders present in the system both position measurements can be used for feedback control in order to improve performance of the load position control. This control method, the use of the two feedbacks of both motor and load side, is called dual loop control. The structure of the EPOS4 dual loop control is shown in ➔Figure 10-143.

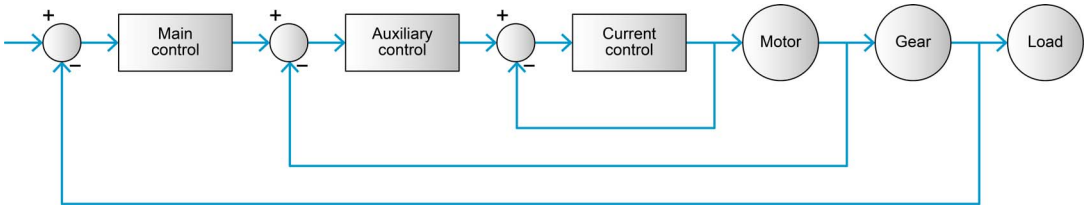


Figure 10-143 Dual loop control | Control structure

The EPOS4 dual loop control consists of two distinct control loops:

- **Auxiliary control loop** – controls the velocity of the motor and gives the motor current reference as its output
- **Main control loop** – controls the position of the load and gives the motor velocity and acceleration references as its outputs

A more detailed structure of EPOS4 dual loop control is shown in ➔Figure 10-144.

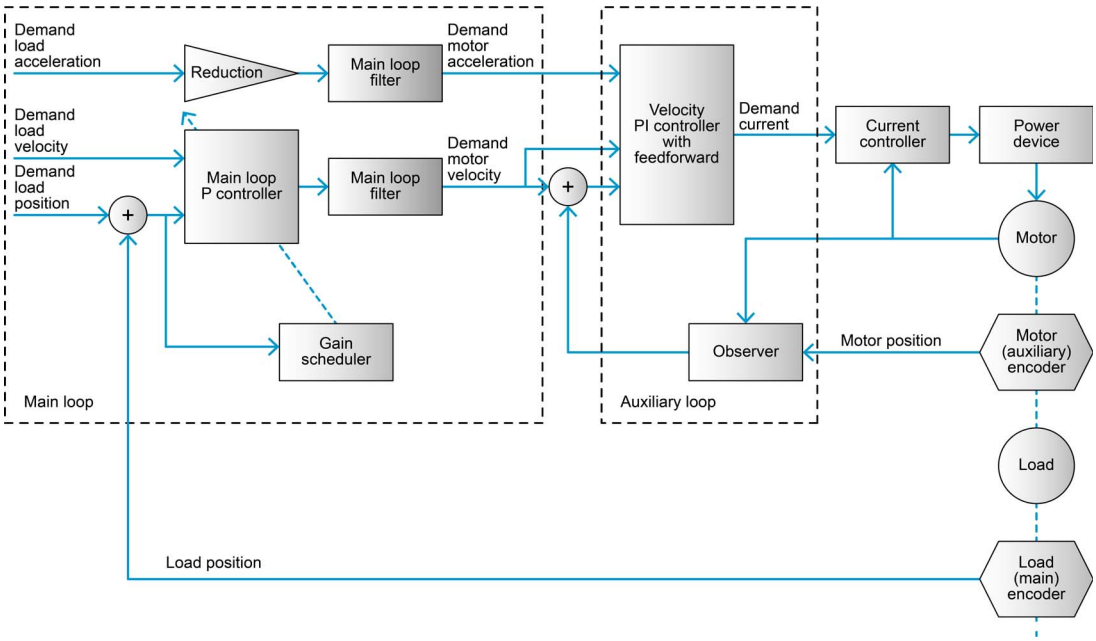


Figure 10-144 Dual loop control | Control structure in detail

The two loops are implemented with the following sampling periods:

$$T_{Auxiliary} = 0.4ms$$

$$T_{Main} = 1.2ms$$

10.3 Auxiliary Control Loop

The inputs of the auxiliary control loop are the desired motor velocity and acceleration (which are gathered from the main control loop) and the measured motor position from the auxiliary encoder. These inputs serve as basis to calculate the desired motor current that is then sent to the current controller. The auxiliary control loop consists of a PI speed controller with feedforward and an observer for motor velocity estimation. Find a detailed description on their structure in ➔chapter “2.3.2 Velocity Regulation (with Feedforward)” on page 2-14. The object dictionary entries relevant to the auxiliary control loop are given in ➔Table 10-90.

Symbol	Unit	Name	Index	Subindex
$K_{P\omega\_Auxiliary}$	$\frac{mA \cdot s}{rad}$	Auxiliary loop P gain	0x30AE	0x20
$K_{I\omega\_Auxiliary}$	$\frac{mA}{rad}$	Auxiliary loop I gain	0x30AE	0x21
$FF_{\omega\_Auxiliary}$	$\frac{mA \cdot s}{rad}$	Auxiliary loop FF velocity gain	0x30AE	0x22
$FF_{\alpha\_Auxiliary}$	$\frac{mA \cdot s^2}{rad}$	Auxiliary loop FF acceleration gain	0x30AE	0x23
$l_{\theta\_Auxiliary}$	1	Auxiliary loop observer position correction gain	0x30AE	0x30
$l_{\omega\_Auxiliary}$	Hz	Auxiliary loop observer velocity correction gain	0x30AE	0x31
$l_{T\_Auxiliary}$	$\frac{mNm}{rad}$	Auxiliary loop observer load correction gain	0x30AE	0x32
$r_{Auxiliary}$	$\frac{\mu Nm}{rpm}$	Auxiliary loop observer friction	0x30AE	0x33
$J_{Auxiliary}$	$g \cdot cm^2$	Auxiliary loop observer inertia	0x30AE	0x34

Table 10-90 Dual loop control | Auxiliary control loop parameters – Object dictionary entries

10.4 Main Control Loop

The inputs of the main control loop are the desired load position, velocity and acceleration (which are gathered from the path planner) and the measured load position from the main sensor. The outputs of the main control loop are the desired motor velocity and acceleration.

The key elements of the main control loop are the proportional controller, gain scheduler, and the main loop filter which are subsequently described in detail. The respective object dictionary entries are given in ➔Table 10-91.

Symbol	Unit	Name	Index	Subindex
$K_{LowMain}$	$\frac{1}{s}$	Main loop P gain low bandwidth	0x30AE	0x01
$K_{HighMain}$	$\frac{1}{s}$	Main loop P gain high bandwidth	0x30AE	0x02
H	1	Main loop gain scheduling weight	0x30AE	0x03
FC_A	1	Main loop filter coefficient a	0x30AE	0x10
FC_B	1	Main loop filter coefficient b	0x30AE	0x11
FC_C	1	Main loop filter coefficient c	0x30AE	0x12
FC_D	1	Main loop filter coefficient d	0x30AE	0x13
FC_E	1	Main loop filter coefficient e	0x30AE	0x14
$F_{\text{on/off}}$	1	Dual loop configuration miscellaneous	0x30AE	0x40

Table 10-91 Dual loop control | Main control loop parameters object dictionary entries

10.4.1 Proportional Controller

The velocity reference for the auxiliary control loop is calculated as sum of the multiplication of the tracking error (as seen by the main encoder) with the main loop proportional gain, and the load velocity reference scaled by the drivetrain amplification ratio. The resulting reference is limited to the maximal permitted motor speed.

$$\omega_{ReferenceAuxiliary} = K_{Main} \cdot (\theta_{Desired} - \theta_{Measured}) + GearRatio \cdot \omega_{ReferenceMain}$$

10.4.2 Gain Scheduler

The main loop proportional gain does not have a fixed value. Its value is continuously modified by the gain scheduler based on the load position tracking error value. Particularly, with a large tracking error, this block sets a high control gain in order that the load position quickly approaches the desired reference. On the other hand, when the tracking error is small (as the actual position almost reaches the reference), this block reduces the gain value in order to have less aggressive control and to avoid negative effects caused by backlash and other nonlinear phenomena. The governing equation of the gain scheduler is as follows:

$$K_{Main} = K_{HighMain} + (K_{LowMain} - K_{HighMain}) e^{-H|(\theta_{Measured} - \theta_{Desired})}$$

The parameters K_{Main}, $K_{HighMain}$, and H are the tuning parameters that can either be manually set or they can be automatically calculated in an auto tuning procedure. The shape of the gain scheduling function for different values of the gain scheduling weight H is shown in ➔Figure 10-145.

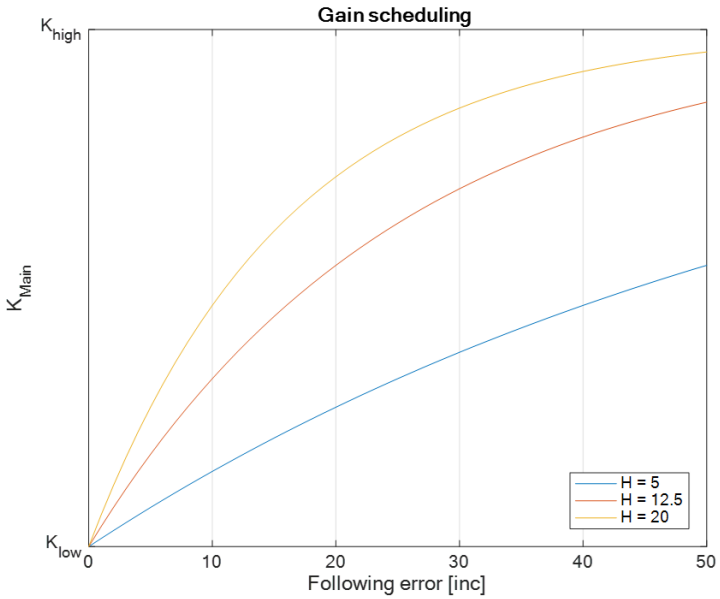


Figure 10-145 Dual loop control | Gain scheduling function for different values of main loop gain scheduling weight

10.4.3 Main Loop Filter

The main loop filter is used to filter out the output of the proportional controller in the main control loop. An identical filter is applied to the acceleration feedforward signal before it is passed to the auxiliary control loop. The purpose of this filter is to remove the part of the frequency spectrum that may cause strong oscillations in the system (resonant oscillations caused by the elasticity of the drivetrain) from the control loop. The transfer function of the second order filter is as follows:

$$G_{Filter}(s) = \frac{FC_A \cdot s^2 + FC_B \cdot s + FC_C}{s^2 + FC_D + FC_E}$$

All filter coefficients can be estimated automatically during the auto tuning procedure. The filter can also be turned off by changing the $F_{\text{on/off}}$ value. With $F_{\text{on/off}} = 0$, the filter is turned off, with $F_{\text{on/off}} = 2$, the filter is turned on. To reactivate the turned off filter, the parameter $F_{\text{on/off}} = 0$ must be first set to $F_{\text{on/off}} = 1$, then to $F_{\text{on/off}} = 2$ for correct calculation of the filter parameters.

WHEN SHOULD THE MAIN LOOP FILTER BE USED?

Dual loop control can be configured such that the use of the filter in the main control loop is disabled. Using the filter brings the most benefits with an elastic drivetrain that causes resonant oscillation in a certain frequency range. In such case, the main loop filter may be used to eliminate these frequencies from the closed loop control and prevent undesirable oscillations. However, if the drive train is rigid, the filter is of not much use. However, if a high resolution main encoder is used, the main loop filter may also be turned off without any influence on the control performance.

10.4.4 Transport Delay of the Control Loop

The total transport delay of the dual loop controller is always smaller than

$$T_{Auxiliary} + T_{Main} = 1.6 \text{ ms}$$

10.5 Proper Use of Dual Loop Control

Dual loop control should be employed when a gear with backlash is used and/or when the drivetrain between motor and load has dynamic properties which influence the performance of the load position control (for example, with an elastic shaft connecting motor and load).

The dual loop control structure has two degrees of freedom, as it uses both position measurements of the load and the motor. This allows reduction or even elimination of any negative effects of the drivetrain, such as backlash and elasticity. This leads to fast and accurate control of the load despite the fact that a dynamically complex drivetrain is used. Of course, the limitations of dual loop control are given by the limitations of the drive system as a whole. If the motor and drivetrain are setup in a way that a certain velocity and precision cannot be reached, dual loop control may not be able to change this fact. In addition, for a good performance of the dual loop controller, the resolution of the main encoder must not be less than the resolution of the auxiliary encoder.

If the drivetrain between motor and load does not feature any dynamic or nonlinear effects, the advantages of using dual loop control are minor. In this case and due to the fact that it is simpler to tune and parametrize, the use of standard PID position control with either an auxiliary or a main encoder is recommended instead.

10.6 Best Practice Examples

The load position control can be configured by choosing either PID position controller (for detailed description ➔chapter “2.3.3 Position Regulation (with Feedforward)” on page 2-18) or dual loop position controller. If the PID position controller is used, the main sensor can be located either on the motor or on the load (if both exist in the system). If dual loop position controller is selected, both the main encoder (on the load side) and the auxiliary encoder (on the motor side) must be present in the system and be properly configured. ➔Figure 10-146 illustrates the correct configuration of dual loop control.

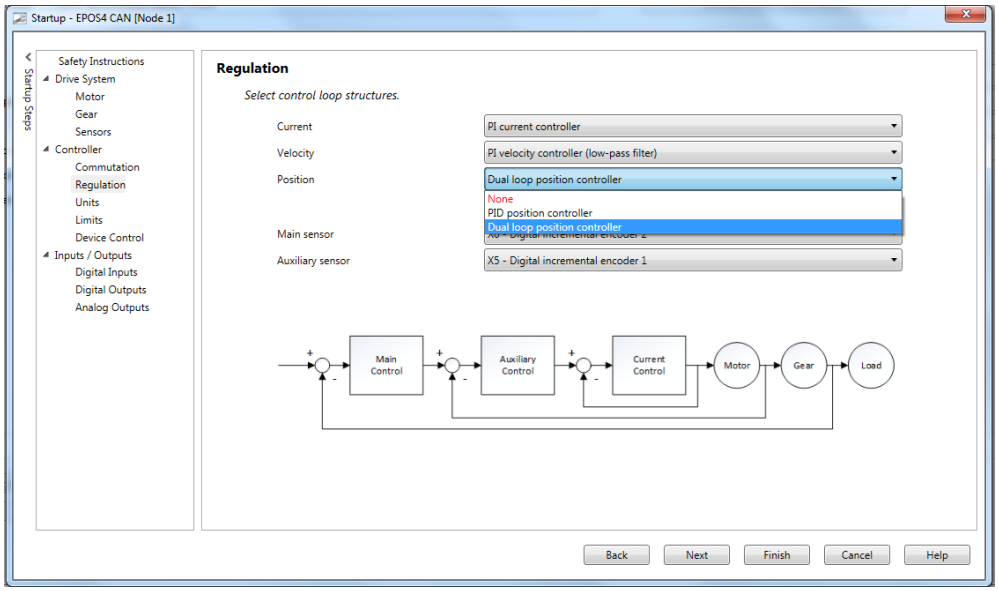


Figure 10-146 Dual loop control | Configuration

The following examples demonstrate the advantages of using dual loop control in eliminating the negative effects of backlash and elastic motor load coupling. They also illustrate how the auto tuning procedure in the dual loop control structure works.

Systems used for these illustrations are rotative systems comprising a motor with attached encoder and a load with a respectively attached encoder. Motor and load are coupled with a gear and a shaft that can be rigid or elastic.

10.6.1 Use Case 1: System with high Reduction Gear, moderate Backlash, and no Coupling Elasticity

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/8 CAN (520885)		
Motor maxon EC-4pole 30 brushless, 100 W (309755)	No load speed (line 2)	n_0 = 17500 rpm
	No load current (line 3)	I_0 = 505 mA
	Nominal current (line 6)	I_n = 7.74 A
	Terminal resistance (line 10)	R = 0.135 Ω
	Terminal inductance (line 11)	L = 0.017 mH
	Torque constant (line 12)	k_m = 9.8 mNm/A
	Rotor inertia (line 16)	J_{motor} = 18.3 gcm^2
Auxiliary encoder HEDL 5540 500 impulse, 3 channel, with Line Driver (110514)	Encoder counts per turn	500 pulses/revolution
Main encoder HEDL 5540 500 impulse, 3 channel, with Line Driver (110514)	Encoder counts per turn	500 pulses/revolution
Gear Planetary gear GP 32 HP Ø 32 mm, 4.0…8.0 Nm (324946)	Gear ratio	411:1 359424/875
Mechanical load connected via rigid coupling	Load inertia	J_{load} = 95.3 gcm^2

Table 10-92 Dual loop control | Use case 1: System components

In order to illustrate the advantage of using dual loop control for this configuration, regular PID position control is considered first. Thereby, the sensor on the load side is configured as the main sensor. After running the auto tuning procedure, the parameters of the PID position controller are automatically calculated and have the values listed in ➔Table 10-93.

Index	Subindex	Name	Value	Unit
0x30A1	0x01	Position controller P gain	95159.615	$\frac{mA}{rad}$
0x30A1	0x02	Position controller I gain	170609.607	$\frac{mA}{rad \cdot s}$
0x30A1	0x03	Position controller D gain	4247.317	$\frac{mA \cdot s}{rad}$
0x30A1	0x04	Position controller FF velocity gain	175.721	$\frac{mA \cdot s}{rad}$
0x30A1	0x05	Position controller FF acceleration gain	68.528	$\frac{mA \cdot s^2}{rad}$

Table 10-93 Dual loop control | Use case 1: PID position controller parameters

Using the path planner and executing the trajectory with the maximal velocity of 1 rpm, an acceleration of $1000\frac{rpm}{s}$, and a position change of 25 increments, the result obtained is as in ➔Figure 10-147.

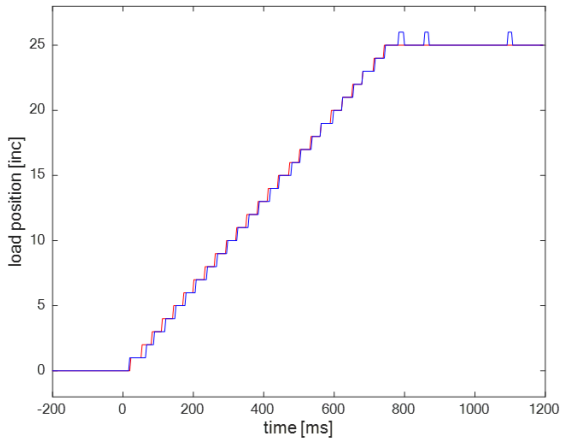


Figure 10-147 Dual loop control | Use case 1: Load position control with PID position controller

As seen, the reference tracking is generally good. However, after 700 ms (when the system should stop moving and keep the load position fixed), several short changes in the actual load position may be observed. These changes are caused by the backlash effects of the gear. Namely, the teeth of the gear are imperfect to their theoretical shape and they have some play between them. Whenever the system reverses the direction of motion or changes from movement to standstill, the use of a single loop controller may result in oscillations of the load position due to nonlinear effects introduced by the backlash.

Now, the use of dual loop control in the application will be considered to illustrate how the negative effects caused by the backlash can be eliminated.

The dual loop controller can also be tuned automatically by using the «Regulation Tuning Wizard». Before starting dual loop tuning, the current controller must be properly tuned. Auto tuning of the dual loop controller itself is done in two steps:

First, the auxiliary loop will be tuned using the respective dialog box given in ➔Figure 10-148.
Upon pressing the ¤Auto tune¤ button, an identification experiment is conducted in which all the mechanical parameters of the system relevant for the auxiliary loop are identified. In the tuning dialog box, stiffness and damping of the feedforward PI controller and the bandwidth of the observer used to calculate motor velocity can be adjusted. Moving the sliders to the right results in faster and more aggressive control in the auxiliary loop. By applying the test signal, a visual representation of the auxiliary loop performance is being produced. The auxiliary loop should be fast and aggressive without a lot of overshoot in the signal and it should be adjusted with the sliders until the desired performance is observed in the resulting test signal.

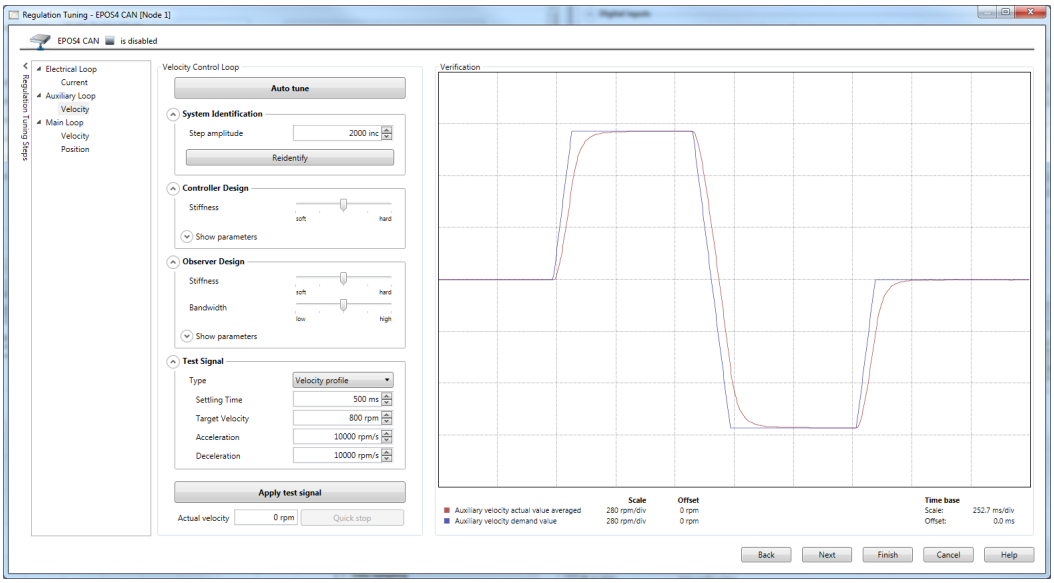


Figure 10-148 Dual loop control | Use case 1: Auxiliary loop tuning dialog box

Then the main loop can be tuned by using the main loop tuning dialog box. Upon pressing the ¤Auto tune¤ button, an identification experiment is executed in which the transfer function of the system drivetrain together with the auxiliary loop control is identified.

CONSIDERATIONS

- Running the experiment for tuning of the main loop makes sense only if the auxiliary loop has been tuned before. The auxiliary loop controller is used as a part of the main loop identification experiment and its parameters influence the calculation of the parameters in the main control loop.
- The Bode plot of the experimentally determined transfer function and its fit are displayed in the ¤Identification¤ tab. Based on this Bode plot, an impression on the characteristics of the system to be controlled can be gained. For example, if the Bode plot is relatively straight, it means that the motor/load coupling is rigid. On the other hand if a resonant peak (a lob in the Bode plot) can be observed, it means that the coupling is elastic and that the controller will try to handle this elasticity by the use of the main loop filter.

- The ¤Verification¤ tab in the dialog box contains the test signal of the resulting controller. The «Bandwidth» of the controller can be adjusted using a slider. Higher bandwidth means faster and more aggressive control behavior. However, it should be noted that for each system there are limitations on the achievable closed loop bandwidth and that the auto tuning algorithm will consider these limitations when calculating the main loop parameters. The other two sliders may be used to set the «Bandwidth ratio» and the «Gain scheduling weight». These parameters influence the elimination of the backlash effects and should be adjusted such that no backlash effects (as shown in ➔Figure 10-147) are visible in the verification signal. Increasing the «Gain scheduling weight» (moving the slider to the right) and decreasing the «Bandwidth ratio» (moving the slider to the left) has a positive effect on eliminating the negative effects caused by the backlash. The main loop tuning dialog box is shown in ➔Figure 10-149 and ➔Figure 10-150.

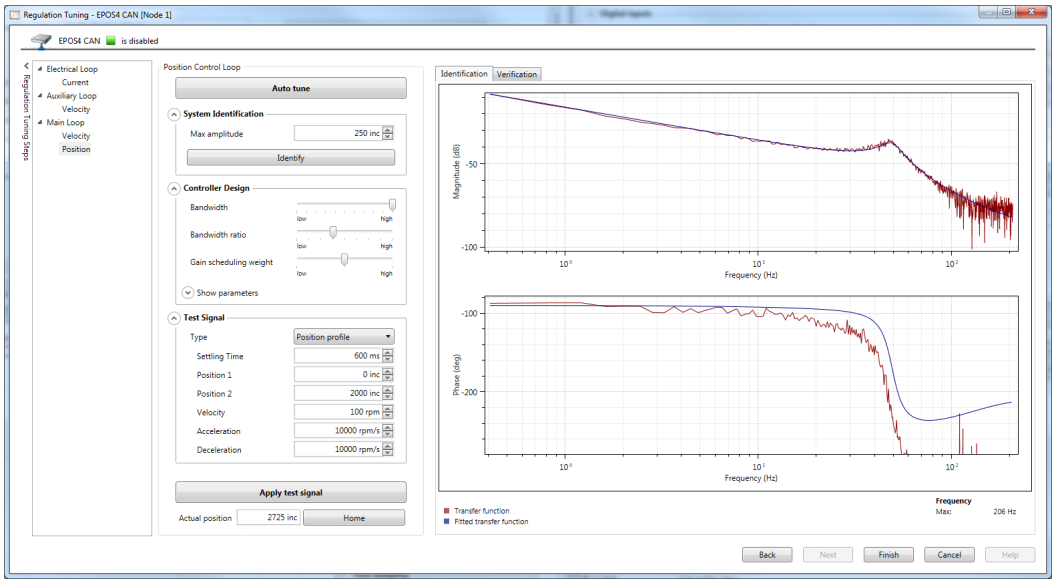


Figure 10-149 Dual loop control | Use case 1: Main loop tuning dialog box with the «Identification» tab active

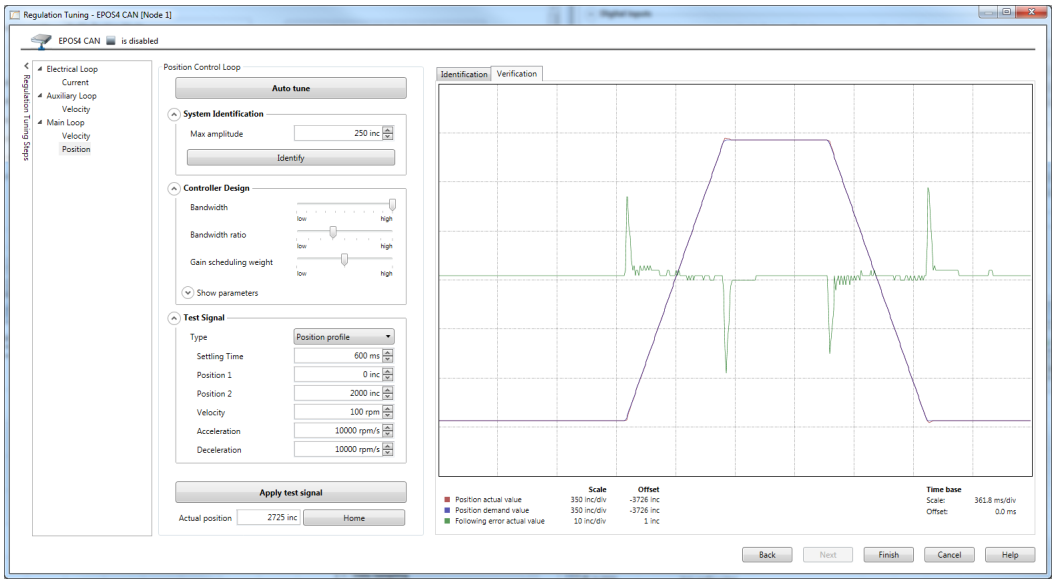


Figure 10-150 Dual loop control | Use case 1: Main loop tuning dialog box with the «Verification» tab active

Dual loop controller parameters obtained after the auto tuning procedure for the considered application example are given in ➔Table 10-94.

Index	Subindex	Name	Value	Unit
0x30AE	0x01	Main loop P gain low bandwidth	221.34	$\frac{1}{s}$
0x30AE	0x02	Main loop P gain high bandwidth	44.268	$\frac{1}{s}$
0x30AE	0x03	Main loop gain scheduling weight	10.00	1
0x30AE	0x10	Main loop filter coefficient a	0	1
0x30AE	0x11	Main loop filter coefficient b	0	1
0x30AE	0x12	Main loop filter coefficient c	35530.0	1
0x30AE	0x13	Main loop filter coefficient d	376.0	1
0x30AE	0x14	Main loop filter coefficient e	35530.0	1
0x30AE	0x20	Auxiliary loop P gain	138.825	$\frac{mA \cdot s}{rad}$
0x30AE	0x21	Auxiliary loop I gain	2470.774	$\frac{mA}{rad}$
0x30AE	0x22	Auxiliary loop FF velocity gain	0.151	$\frac{mA \cdot s}{rad}$
0x30AE	0x23	Auxiliary loop FF acceleration gain	0.173	$\frac{mA \cdot s^2}{rad}$
0x30AE	0x30	Auxiliary loop observer position correction gain	0.950	1
0x30AE	0x31	Auxiliary loop observer velocity correction gain	751.283	Hz
0x30AE	0x32	Auxiliary loop observer load correction gain	0.030	$\frac{mNm}{rad}$
0x30AE	0x33	Auxiliary loop observer friction	0	$\frac{\mu Nm}{rad}$
0x30AE	0x34	Auxiliary loop observer inertia	20.48	$g \cdot cm^2$
0x30AE	0x40	Dual loop configuration miscellaneous	2	1

Table 10-94 Dual loop control | Use case 1: Dual loop load position controller parameters

The use of dual loop control results in load position control shown in ➔Figure 10-151. As can be seen, negative effects caused by backlash are fully eliminated.

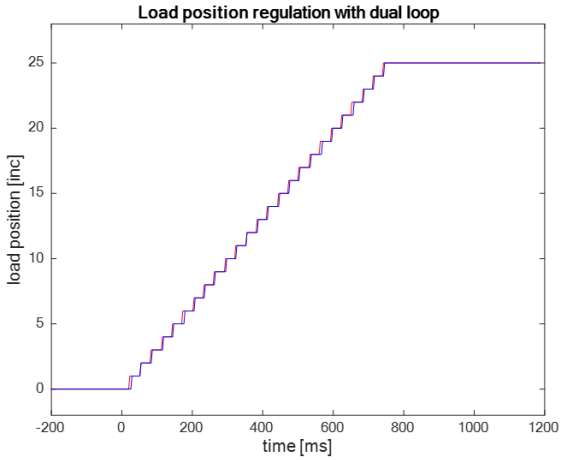


Figure 10-151 Dual loop control | Use case 1: Position control with dual loop controller

10.6.2 Use Case 2: System with considerable Backlash and with Coupling Elasticity

SYSTEM COMPONENTS

Item	Description	Setting
Controller EPOS4 Compact 50/5 CAN (541718)		
Motor maxon EC-4pole 30 brushless, 100 W (309756)	No load speed (line 2)	n_0 = 17500 rpm
	No load current (line 3)	I_0 = 379 mA
	Nominal current (line 6)	I_n = 5.56 A
	Terminal resistance (line 10)	R = 0.248 Ω
	Terminal inductance (line 11)	L = 0.03 mH
	Torque constant (line 12)	k_m = 13.1 mNm/A
	Rotor inertia (line 16)	J_{motor} = 18.3 gcm^2
Auxiliary encoder HEDL 5540 500 impulse, 3 channel, with Line Driver (110514)	Encoder counts per turn	500 pulses/revolution
Main encoder HEDL 5540 500 impulse, 3 channel, with Line Driver (110514)	Encoder counts per turn	500 pulses/revolution
Gear Planetary gear GP 32 HP Ø 32 mm, 4.0…8.0 Nm (320247)	Gear ratio	14:1 676/49
Mechanical load connected via spring coupling FKZS 1225	Load inertia	J_{load} = 51.2 gcm^2

Table 10-95 Dual loop control | Use case 2: System components

The first steps in properly configuring the dual loop controller for this application example is to run the current controller auto tuning followed by the auxiliary loop auto tuning. Then, the main loop may be tuned. The identification experiment that is run as part of the main loop tuning identifies the transfer function of the system as seen by the main loop. The Bode plot of the identified transfer function is given in ➔Figure 10-152.

The Bode plot indicates that the drivetrain has a resonant frequency at 48 Hz. At this frequency, the Bode plot has a significant magnitude increase. The resonance is a direct consequence of the elastic coupling between the gear and the load.

In order to achieve good control with the dual loop controller, the resonance must be taken into account when designing the controller. In EPOS4 dual loop architecture this is done with the main loop filter, which is automatically designed to neutralize the negative effects of the resonance. The transfer function obtained by multiplying the automatically calculated main loop filter with the identified transfer function is shown in ➔Figure 10-152.

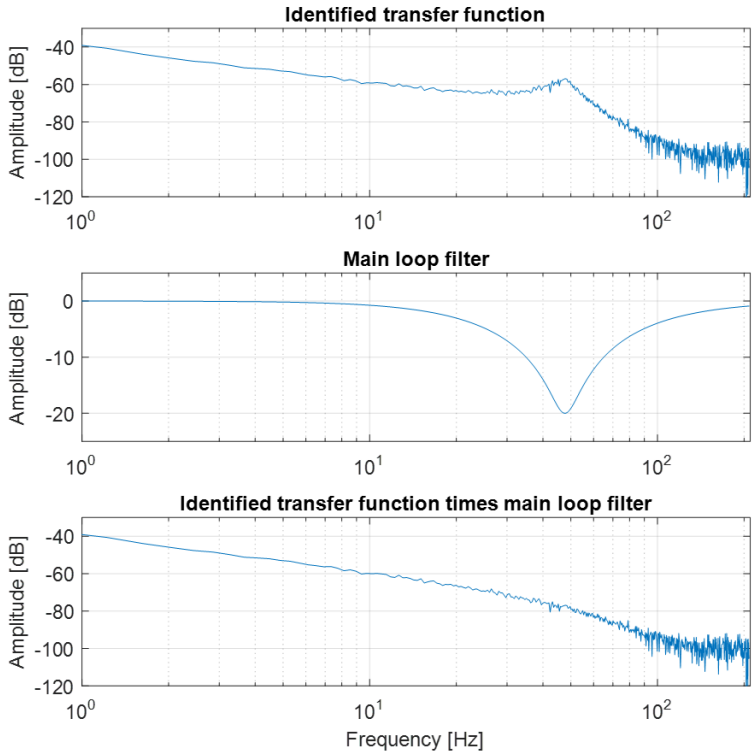


Figure 10-152 Dual loop control | Use case 2: Comparison of Bode plots

For cases with complex shapes of the identified Bode plot and if the automatic calculation of the main loop filter does not give satisfactory performance, the main identification parameters may be saved. To do so click right in the system identification area of the Regulation Tuning Wizard's main loop tuning dialog (➔Figure 10-153). Thereby, three files with the extension ".csv" are created. They contain the raw recorded experimental data, fitted zeros and poles, and the experimental transfer function estimate respectively. The saved values may be imported by an advanced calculation software and as such may be used to manually configure the filter parameters. Note that such manual parameter configuration requires expert knowledge.

After running main loop tuning, the dual loop controller parameters as to ➔Table 10-96 are obtained.

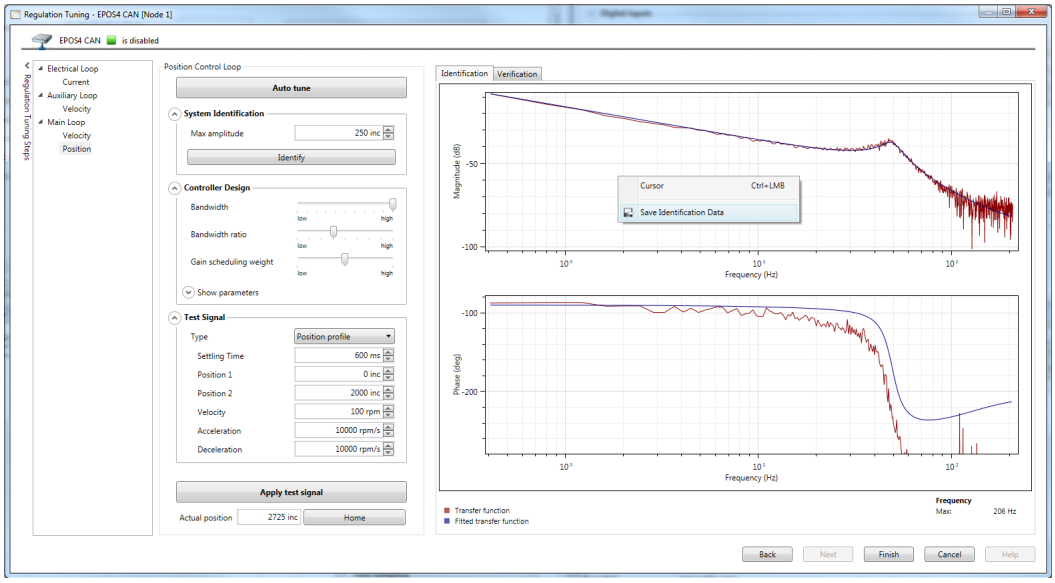


Figure 10-153 Dual loop control | Use case 2: Saving of the identification data

Index	Subindex	Name	Value	Unit
0x30AE	0x01	Main loop P gain low bandwidth	4.321	$\frac{1}{s}$
0x30AE	0x02	Main loop P gain high bandwidth	120.12	$\frac{1}{s}$
0x30AE	0x03	Main loop gain scheduling weight	10	1
0x30AE	0x10	Main loop filter coefficient a	1	1
0x30AE	0x11	Main loop filter coefficient b	42.846	1
0x30AE	0x12	Main loop filter coefficient c	93660.611	1
0x30AE	0x13	Main loop filter coefficient d	612.080	1
0x30AE	0x14	Main loop filter coefficient e	93660.611	1
0x30AE	0x20	Auxiliary loop P gain	112.32	$\frac{mA \cdot s}{rad}$
0x30AE	0x21	Auxiliary loop I gain	1109.41	$\frac{mA}{rad}$
0x30AE	0x22	Auxiliary loop FF velocity gain	0	$\frac{mA \cdot s}{rad}$
0x30AE	0x23	Auxiliary loop FF acceleration gain	0.18	$\frac{mA \cdot s^2}{rad}$

Continued on next page.

Index	Subindex	Name	Value	Unit
0x30AE	0x30	Auxiliary loop observer position correction gain	0.8	1
0x30AE	0x31	Auxiliary loop observer velocity correction gain	502.74	Hz
0x30AE	0x32	Auxiliary loop observer load correction gain	11.21	$\frac{mNm}{rad}$
0x30AE	0x33	Auxiliary loop observer friction	0	$\frac{\mu Nm}{rad}$
0x30AE	0x34	Auxiliary loop observer inertia	20.6	$g \cdot cm^2$
0x30AE	0x40	Dual loop configuration miscellaneous	2	1

Table 10-96 Dual loop control | Use case 2: Dual loop load position controller parameters

The main loop filter is now used to eliminate the resonant effects. Therefore, turning the filter off may significantly reduce the controller performance, or even lead to controller instability as illustrated in ➔Figure 10-154.

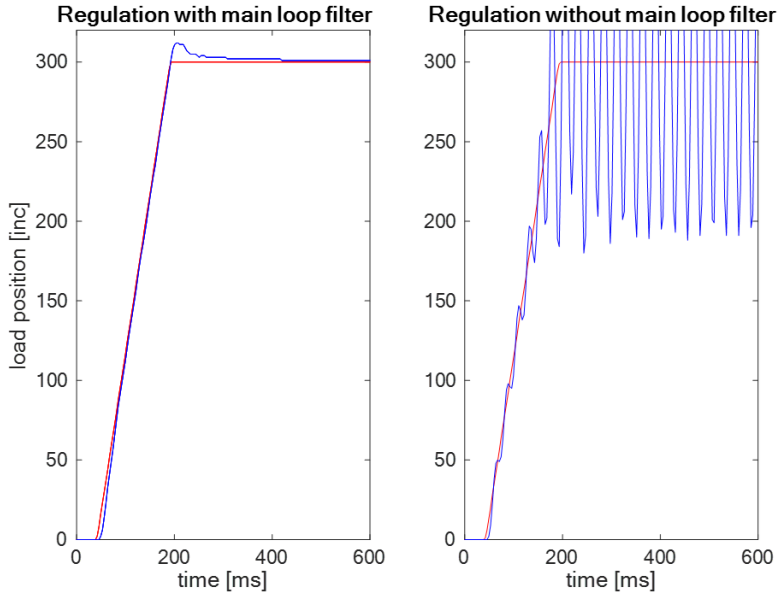


Figure 10-154 Dual loop control | Use case 2: Control performance with/without main loop filter

The reason for the controller to become instable when the filter is turned off lies in the fact that the system has a resonant peak. In order for the controller to remain stable even in the case when the main loop filter is not used, the controller bandwidth must be significantly reduced. However, in this case the oscillation at resonant frequency of the system remains.

Due to strong backlash, the performance of the dual loop controller obtained by the default settings in the main loop auto tuning is not ideal. This is particularly the case in the phase where the load position is close to reaching the target. This can be improved by adjusting the sliders for «Bandwidth ratio» and «Gain scheduling weight». Thereby, moving the «Bandwidth ratio» slider to the left and the «Gain scheduling» slider to the right in order that the «Main loop P gain low bandwidth» and the «Main loop gain scheduling weight» are set to 15.32 1⁄s and 20 respectively will result in improved tracking performance (➔Figure 10-155).

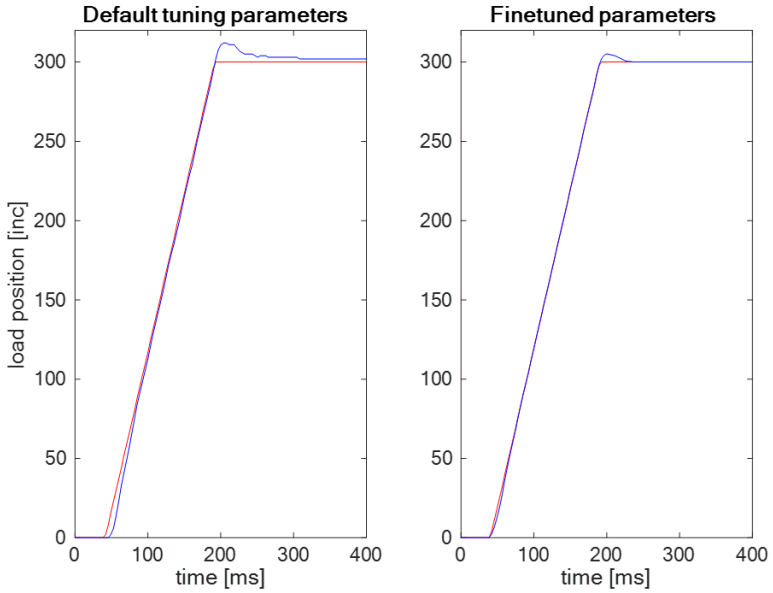


Figure 10-155 Dual loop control | Use case 2: Control performance with default/fine-tuned main loop parameters

10.7 Conclusion

If a gear with backlash is involved and/or if the coupling between the motor and the load is elastic, the described application examples show that using dual loop control results in better control performance than using a PID position controller. In addition, using the main loop filter in the dual loop controller structure is suggested when the drivetrain has a clearly prominent resonant frequency. If not the case, the main loop filter can be turned off especially with a high resolution main encoder.

LIST OF FIGURES

maxon

maxon

LIST OF TABLES

maxon

maxon

INDEX

H

I

L

M

N

O

P

R

S

T

U

V

Z

maxon

EtherCAT

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

CCMC | EPOS4 Application Notes | Edition 2021-03 | DocID rel9611

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