# PLC Commissioning of LinMot Servo Drives

Application Note / How-to

Latest version:
https://shop.linmot.com/E/product/0185-1120-E

Note
The information in this documentation reflects the stage of development at the time of press and is therefore without obligation. NTI AG reserves itself the right to make changes at any time and without notice to reflect further technical advance or product improvement.

## Table of Contents

## Use of This Document

This document is an Application Note / How-To to help the user connecting LinMot drives to different PLC types.

It provides guides when using function block libraries provided by NTI AG / LinMot and guides to integrate LinMot drives using standardized drive profiles.

This document is provided by NTI AG / LinMot free of charge with no warranty for updates.

Also, LinMot accepts no liability for damages that may be caused by using this document or programming examples referred to.

**Drives:** Various

**Classification:**
[x] Application Note
[ ] Installation Guide
[ ] User Manual
[ ] Documentation
[ ] LinMot internally

## Version History

| Version | Date | Author | Description |
|---|---|---|---|
| 1V0 | 31/05/2017 | fj | Initial Release |
| 1V1 | 13/06/2017 | fj | Minor corrections |
| 1V2 | 10/07/2017 | fj | Added chapter 14 Drive Profile: CODESYS SoftMotion, CiA402 |
| 1V3 | 27/03/2019 | fj / mm | Added chapter 9 Drive Profile: Rockwell Automation Motion (CIP Sync)<br>Added chapter 12.4 Encoder Parameter Examples for Linear and Rotary Motors<br>Added chapter 11 Drive Profile: Beckhoff TwinCAT 2/3, CoE DS402<br>Added chapter 13.3.3.3 Motor/Mechanic using a rotary motor<br>Added chapter 16 Drive Profile: B&R NC Motion, SDC interface<br>Updated chapter 10 Drive Profile: Siemens TIA, PROFIdrive<br>Removed chapter OMRON with EC drives (obsolete)<br>Various minor updates and corrections |
| 1V4 | 03/06/2019 | fj | Minor: added library compatibility list: chapter 16.2<br>Minor: fixed screenshot and description in chapter 16.13 |
| 1V5 | 26/02/2020 | fj | Added chapter 13.4.1 Homing using the Schneider template (FB_Home)<br>Added chapter Trademarks<br>Various minor updates and corrections |
| 1V6 | 06/08/2020 | fj / mr | Updated chapter 17.1 Overview of Drive Profile: OMRON Sysmac Studio, CoE DS402<br>Updated chapter 10.4.3 Application based example to home a LinMot Linear Axis and created new example projects<br>Added chapter 8 LinMot Profile: Mitsubishi<br>Removed/Updated contact information on page 2 and the last page |
| 1V7 | 13/07/2021 | fj / pk | Added chapter 6 LinMot Profile: B&R Automation Studio<br>Added chapter 7 LinMot Profile: CODESYS<br>Added chapter 10.7 Telegram 750 / TDB<br>Added chapter 10.8 Using the Closed Loop Force/Torque Control Technology Function<br>Updated chapter 11 Drive Profile: Beckhoff TwinCAT 2/3, CoE DS402 for Firmware 6.10<br>Updated chapter 11.5 Homing: added MC_Home example<br>Added chapter 11.11 Using the Closed Loop Force/Torque Control Technology Function<br>Added chapter 15 Drive Profile: B&R NC Motion, CiA402<br>Updated chapter 17 Drive Profile: OMRON Sysmac Studio, CoE DS402 for FW 6.10<br>Added chapter 18 Drive Profile: TRIO, CiA402<br>Added chapter Appendix II: Basic Closed Loop Torque/Force Control Loop Tuning<br>Small corrections, updates, and improvements over all document |
| 1V8 | 07/09/2021 | fj | Rearranged homing variants in chapter 10.4 Home / Set Reference |
| 1V9 | 01/11/2021 | fj | Updated/Corrected chapter 11.4 |
| 1V10 | 22/09/2022 | fj / pk / mr | Added chapter 1.2 LinMot Drives with multi-interface (MI) Overview / C1250-xx replacement<br>Updated chapter 10.9.1<br>Updated all chapters to support LinMot C125x-MI drives |

## Trademarks

Product and company names and logos referenced may be either trademarks or registered trademarks of their respective companies. The sole purpose of any trademarks in this document is the identification of the corresponding products.

| Company | Trademarks / Registered Trademarks |
|---|---|
| Siemens AG | SIEMENS®, SIMATIC®, TIA Portal®, Step 7® |
| Rockwell Automation, Inc | Rockwell Automation®, Allen-Bradley®, CompactLogix™, ControlLogix®, Studio 5000®, Studio 5000 Logix Designer®, RSLogix 5000®, Stratix®, RSLinx®, |
| Beckhoff Automation GmbH | Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT® |
| Schneider Electric | PacDrive®, SoMachine® |
| Bosch Rexroth | IndraControl, IndraMotion |
| 3S-Smart Software Solutions GmbH | CODESYS |
| B&R Industrial Automation GmbH | B&R Automation Studio®, Ethernet POWERLINK® |
| Omron | SYSMAC®, |
| Profibus and Profinet International (P.I.) | Profibus®, Profinet® |
| Interests Group SERCOS interface e.V | SERCOS interface® |
| ODVA, Inc | EtherNet/IP™, CIP™, CIP Sync™ |
| Trio Motion Technology Ltd | *Motion* Perfect |

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

# 1 General

## 1.1 Abbreviations

### 1.1.1 LinMot Drive System Generations (SG)

| SG | Drives |
|---|---|
| SG3 | Family E1100 (GP, CO, DN, DP) (LC/HC/XC) |
| SG4 | Family B1100 (VF, PP, GP, ML) (LC/HC/XC) |
| SG5 | Family E1200 (GP, DP, DS, EC, IP, PD, PL, PN, SC, SE, LU)<br>Family E1400 (GP, DP, DS, EC, IP, PD, PL, PN, SC, SE, LU) (0S/1S)<br>Family B8000-ML (GP, EC, IP, PL, PN, SC) |
| SG6 | Family C1250 (MI, DS, EC, CM, IP, PD, PL, PN, SC, SE, LU, CC) (0S/1S)<br>Family C1251 (MI) (2S)<br>Family E1400V2 (GP, DP, DS, EC, IP, PD, PL, PN, SC, SE, LU) (0S/1S)<br>Family C1400 (GP, DS, EC, IP, PD, PL, PN, SC, SE, LU) (0S/1S) |
| SG7 | Family A1100<br>Family C1100 (GP, DS, EC, PD, PN, SE) (0S/1S) |

### 1.1.2 LinMot Drive Interface Extensions

| Extension | Interface |
|---|---|
| GP | General Purpose (Depends on system generation and installed FW: LinRS and/or CANopen and/or DeviceNet) |
| CO | CANopen (LinMot Profile) |
| DN | DeviceNet (LinMot Profile) |
| DP | PROFIBUS (LinMot Profile) |
| PN | PROFINET (LinMot Profile) |
| PD | PROFIdrive (PROFIdrive Standard Telegram 5 & 9, SIEMENS Telegram 105) |
| PDS | PROFISAFE (Safety Standard Telegram 30), PROFINET (LinMot Profile) & PROFIdrive (PROFIdrive Standard Telegram 5 & 9, SIEMENS Telegram 105) |
| EC | EtherCAT (LinMot Profile) |
| SE | EtherCAT SoE (Servo Drive Profile over EtherCAT) |
| DS | EtherCAT CoE (DS402/CiA402) |
| PL | Powerlink (DS402/CiA402, LinMot Profile) |
| IP | EtherNet/IP (LinMot Profile) |
| CM | EtherNet/IP (CIP Sync, LinMot Profile) |
| SC | Sercos III (FSP_DRIVE, FSP_IO) |
| CC | CC-Link IE Field Basic (LinMot Profile) |
| LU | LinUDP (LinMot Ethernet UDP protocol) |
| RS | LinRS (LinMot serial protocol for RS232, RS422, RS485) |
| MI | Multi Interface (PN/PD > MIPD, EC/DS > MIDS, CM > MICM, PL > MIPL, SC > MISC, CC > MICC, LU > MILU) |

E.g., C1250-**DS**-XC-xS -> DS = EtherCAT CoE

## 1.2 LinMot Drives with multi-interface (MI) Overview / C1250-xx replacement

The table shown on the next page gives an overview of how and if a C1250 drive with a specific interface (e.g., EC, PN, …) could be replaced by a multi-interface C125x-MI drive. (blue = C1250-MI-XC-xS, yellow = C1251-MI-XC-2S)

The C1250-MI drives can be ordered with a specific interface preinstalled by a defined article number.
Even if there is an interface preinstalled the user is free to install any supported interface during firmware installation.

A generic C1250-MI drive is available too where no interface is preinstalled. Any supported interface can be chosen during firmware installation.

The generic versions have the following ordering information:

| Item | Article no. |
|---|---|
| C1250-MI-XC-1S-000 | 0150-5589 |
| C1250-MI-XC-0S-000 | 0150-5591 |
| | |
| C1250-MI-XC-1S-C00 (Calibrated Measuring Amplifier) | 0150-5590 |
| C1250-MI-XC-0S-C00 (Calibrated Measuring Amplifier) | 0150-5592 |

**Note:**
The generic MI drive supports all interfaces of the C1250 series drives **except SE & IP.**
Supported: PD, PN, DS, EC, CM, PL, SC, LU, CC

**Note:**
For MI drives with preinstalled interface the item name is appended with -0xx / -Cxx where xx is the interface type.

E.g., C1250-MI-XC-1S-000 drive with preinstalled PD/PN interface => C1250-MI-XC-1S-0**PD**

**Note:**
In the PLC the device type of MI drives is shown with the interface abbreviation appended directly to MI.

E.g., C1250-MI-XC-1S-0PD (or …-000) is shown in TIA Portal as C1250-MI**PD**-XC-1S

| | | Drive Family | | | | |
|---|---|---|---|---|---|---|
| **Interface** | | C1250-xx-XC-0S | C1250-xx-XC-1S | **C1250-MI-XC-0S** | **C1250-MI-XC-1S** | **C1251-MI-XC-2S** |
| ***MI (generic)*** | ***Description*** | | | ***C1250-MI-XC-0S-x00*** | ***C1250-MI-XC-1S-x00*** | ***C1251-MI-XC-2S*** |
| | Intf Art no. | | | | | |
| | HW Art no. -000 | | | 0150-5591 (-000) | 0150-5589 (-000) | 0150-2933 |
| | HW Art no. -C00 | | | 0150-5592 (-C00) | 0150-5590 (-C00) | 0150-4185 |
| ***LU*** | ***Description*** | ***C1250-LU-XC-0S*** | ***C1250-LU-XC-1S*** | ***C1250-MI-XC-0S-xLU*** | ***C1250-MI-XC-1S-xLU*** | ***C1251-MILU-XC-2S*** |
| | Intf Art no. | | | 0150-30140 | 0150-30160 | 0150-30000 |
| | HW Art no. -000 | 0150-2491 | 0150-2492 | 0150-5748 (-0LU) | 0150-5734 (-0LU) | 0150-2933 |
| | HW Art no. -C00 | 0150-4129 | 0150-4135 | 0150-5741 (-CLU) | 0150-5727 (-CLU) | 0150-4185 |
| ***EC*** | ***Description*** | ***C1250-EC-XC-0S*** | ***C1250-EC-XC-1S*** | ***C1250-MI-XC-0S-xDS*** | ***C1250-MI-XC-1S-xDS*** | ***C1251-MIDS-XC-2S*** |
| | Intf Art no. | | | 0150-30142 | 0150-30162 | 0150-30002 |
| | HW Art no. -000 | 0150-1884 | 0150-2345 | 0150-5751 (-0DS) | 0150-5737 (-0DS) | 0150-2933 |
| | HW Art no. -C00 | 0150-4125 | 0150-4126 | 0150-5744 (-CDS) | 0150-5730 (-CDS) | 0150-4185 |
| ***DS*** | ***Description*** | ***C1250-DS-XC-0S*** | ***C1250-DS-XC-1S*** | ***C1250-MI-XC-0S-xDS*** | ***C1250-MI-XC-1S-xDS*** | ***C1251-MIDS-XC-2S*** |
| | Intf Art no. | | | 0150-30142 | 0150-30162 | 0150-30002 |
| | HW Art no. -000 | 0150-2415 | 0150-2416 | 0150-5751 (-0DS) | 0150-5737 (-0DS) | 0150-2933 |
| | HW Art no. -C00 | 0150-4123 | 0150-4124 | 0150-5744 (-CDS) | 0150-5730 (-CDS) | 0150-4185 |
| ***SE*** | ***Description*** | ***C1250-SE-XC-0S*** | ***C1250-SE-XC-1S*** | | | |
| | Intf Art no. | | | | | |
| | HW Art no. -000 | 0150-1897 | 0150-2350 | | | |
| | HW Art no. -C00 | 0150-4144 | 0150-4145 | | | |
| ***PL*** | ***Description*** | ***C1250-PL-XC-0S*** | ***C1250-PL-XC-1S*** | ***C1250-MI-XC-0S-xPL*** | ***C1250-MI-XC-1S-xPL*** | ***C1251-MIPL-XC-2S*** |
| | Intf Art no. | | | 0150-30144 | 0150-30164 | 0150-30004 |
| | HW Art no. -000 | 0150-1885 | 0150-2347 | 0150-5749 (-0PL) | 0150-5735 (-0PL) | 0150-2933 |
| | HW Art no. -C00 | 0150-4138 | 0150-4139 | 0150-5742 (-CPL) | 0150-5728 (-CPL) | 0150-4185 |
| ***PD*** | ***Description*** | ***C1250-PD-XC-0S*** | ***C1250-PD-XC-1S*** | ***C1250-MI-XC-0S-xPD*** | ***C1250-MI-XC-1S-xPD*** | ***C1251-MIPD-XC-2S*** |
| | Intf Art no. | | | 0150-30145 | 0150-30165 | 0150-30010 |
| | HW Art no. -000 | 0150-2618 | 0150-2619 | 0150-5746 (-0PD) | 0150-5732 (-0PD) | 0150-2933 |
| | HW Art no. -C00 | 0150-4136 | 0150-4137 | 0150-5739 (-CPD) | 0150-5725 (-CPD) | 0150-4185 |
| ***PN*** | ***Description*** | ***C1250-PN-XC-0S*** | ***C1250-PN-XC-1S*** | ***C1250-MI-XC-0S-xPD*** | ***C1250-MI-XC-1S-xPD*** | ***C1251-MIPD-XC-2S*** |
| | Intf Art no. | | | 0150-30145 | 0150-30165 | 0150-30010 |
| | HW Art no. -000 | 0150-1888 | 0150-2348 | 0150-5746 (-0PD) | 0150-5732 (-0PD) | 0150-2933 |
| | HW Art no. -C00 | 0150-4140 | 0150-4141 | 0150-5739 (-CPD) | 0150-5725 (-CPD) | 0150-4185 |
| ***IP*** | ***Description*** | ***C1250-IP-XC-0S*** | ***C1250-IP-XC-1S*** | | | |
| | Intf Art no. | | | | | |
| | HW Art no. -000 | 0150-1886 | 0150-2346 | | | |
| | HW Art no. -C00 | 0150-4127 | 0150-4128 | | | |
| ***CM*** | ***Description*** | ***C1250-CM-XC-0S*** | ***C1250-CM-XC-1S*** | ***C1250-MI-XC-0S-xCM*** | ***C1250-MI-XC-1S-xCM*** | ***C1251-MICM-XC-2S*** |
| | Intf Art no. | | | 0150-30147 | 0150-30167 | 0150-30008 |
| | HW Art no. -000 | 0150-2900 | 0150-2901 | 0150-5747 (-0CM) | 0150-5733 (-0CM) | 0150-2933 |
| | HW Art no. -C00 | 0150-4121 | 0150-4122 | 0150-5740 (-CCM) | 0150-5726 (-CCM) | 0150-4185 |
| ***SC*** | ***Description*** | ***C1250-SC-XC-0S*** | ***C1250-SC-XC-1S*** | ***C1250-MI-XC-0S-xSC*** | ***C1250-MI-XC-1S-xSC*** | ***C1251-MISC-XC-2S*** |
| | Intf Art no. | | | 0150-30149 | 0150-30169 | 0150-30009 |
| | HW Art no. -000 | 0150-1887 | 0150-2349 | 0150-5750 (-0SC) | 0150-5736 (-0SC) | 0150-2933 |
| | HW Art no. -C00 | 0150-4142 | 0150-4143 | 0150-5743 (-CSC) | 0150-5729 (-CSC) | 0150-4185 |
| ***CC*** | ***Description*** | ***C1250-CC-XC-0S*** | ***C1250-CC-XC-1S*** | ***C1250-MI-XC-0S-xCC*** | ***C1250-MI-XC-1S-xCC*** | ***C1251-MICC-XC-2S*** |
| | Intf Art no. | | | 0150-30150 | 0150-30170 | 0150-30010 |
| | HW Art no. -000 | 0150-4023 | 0150-4024 | 0150-5752 (-0CC) | 0150-5738 (-0CC) | 0150-2933 |
| | HW Art no. -C00 | 0150-4146 | 0150-4147 | 0150-5745 (-CCC) | 0150-5731 (-CCC) | 0150-4185 |
| ***PDS*** | ***Description*** | | | | | ***C1251-MIPD-XC-2S*** |
| | Intf Art no. | | | | | 0150-31005 |
| | HW Art no. -000 | | | | | 0150-2933 |
| | HW Art no. -C00 | | | | | 0150-4185 |

## 1.3 Available LinMot Libraries

LinMot provides several function block libraries for different PLCs and fieldbus systems to allow a fast and comfortable integration of LinMot drives into almost any control concept.

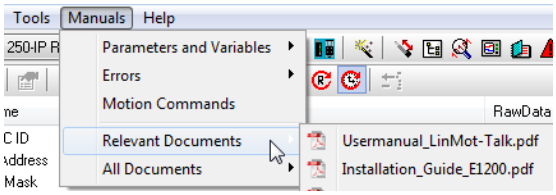
An overview of available LinMot libraries can be downloaded from the LinMot website:
http://www.linmot.com/support/plc-library/

Or search for the document reference (0185-1105) in the LinMot eCatalogue:
http://shop.linmot.com

**Note:** The available libraries can be downloaded from: http://download.linmot.com/plc_lib/libraries
For preliminary libraries that are not available for download please contact the LinMot support team.
support@linmot.com

## 1.4 Documentation / User Manuals / Application Notes

Beside the documentation included in the libraries available for download, reading the following user manuals is essential to understand the communication between the PLC and the LinMot drive. The manuals are included in the LinMot-Talk software (*Menu Manuals → All Documents* or *Relevant Documents* if logged in to a drive) or can be downloaded from the LinMot eCatalogue (search by document reference): http://shop.linmot.com




| User Manuals: General | System Generation | Document Reference |
|---|---|---|
| LinMot-Talk | All | 0185-1059 |
| Motion Control Software | SG3 & SG4 | 0185-1092 |
| Motion Control Software | SG5 - SG7 | 0185-1093 |
| | | |
| **User Manuals: Configuration Over Fieldbus** | **System Generation** | **Document Reference** |
| Drive Configuration Over Fieldbus | SG3 & SG4 | 0185-1072 (SG3), 0185-1073 (SG4) |
| Drive Configuration Over Fieldbus | SG5 - SG7 | 0185-1074 |
| | | |
| **User Manuals: Interfaces** | **System Generation** | **Document Reference** |
| EtherCAT (LinMot Profile) | SG5 - SG7 | 0185-1079 |
| EtherCAT CoE (CiA402, CANopen over EtherCAT) | SG5 - SG7 | 0185-1103 |
| EtherCAT SoE (Servo Drive Profile over EtherCAT) | SG5 - SG7 | 0185-1080 |
| | | |
| PROFIBUS DP | SG3, SG5 & SG6 | 0185-1089 |
| PROFINET IO | SG5 - SG7 | 0185-1090 / 0185-1154 (-MI) |
| PROFINET PROFIdrive | SG6 | 0185-1132 / 0185-1154 (-MI) |
| | | |
| Sercos III (FSP_DRIVE or FSP_IO) | SG5 & SG6 | 0185-1091 |
| | | |
| EtherNet/IP | SG5 & SG6 | 0185-1081 |
| EtherNet/IP CIP Sync | SG6 | 0185-1165 |
| | | |
| Powerlink | SG5 & SG6 | 0185-1088 |
| | | |
| CC-Link IE Field Basic | SG6 (C1250-CC-..) | 0185-1171 |
| | | |
| CANopen | SG3 & SG4 | 0185-1075 |
| CANopen | SG5 - SG7 | 0185-1076 |
| DeviceNet | SG3 & SG4 | 0185-1078 |
| LinRS (LinMot serial protocol) | SG3 - SG7 | 0185-1082 |
| LinUDP (LinMot Ethernet UDP protocol) up to FW6.3 | SG5 & SG6 | 0185-1083 |
| LinUDP V2 (LinMot Ethernet UDP protocol) from FW6.3 | SG5 & SG6 | 0185-1108 |
| | | |
| **Application Notes** | | **Document Reference** |
| HT Config over Realtime | | 0185-0142-E / 0185-0142-D |
| HT Oscilloscope | | 0185-0132-E / 0185-0132-D |
| HT Position Loop Tuning | | 0185-1156-E / 0185-1156-D |
| HT Closed Loop Force/Torque Tuning | | 0185-0144-E / 0185-0144-D |

## 1.5 UPID (Unique Parameter ID)

The value of any parameter of a LinMot drive is stored as an integer value (raw data) in the memory space of the drive. The parameter is identified through its Unique Parameter ID (UPID), which is a 16 Bit integer number.

**Note:** Detailed information about LinMot drive parameters can be found in the user manual *Drive Configuration Over Fieldbus.* See chapter 1.4 Documentation / User Manuals

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

## 1.6 Recent Version of LinMot-Talk and Drive Firmware

The most recent LinMot-Talk version including all drive firmware, device description files and documentation can be downloaded from

either (standard):
https://linmot.com/download/linmot-talk-drive-configuration/

or (alternative link):
http://download.linmot.com/LinMot-Talk/Release/

**Hint:**
LinMot C1250, C1450 & E1450 series drives support configuration over several interfaces (PROFINET, EtherNet/IP, EtherCAT, Sercos III & LinUDP).

This allows to login with LinMot-Talk into the drive and change parameters, monitor the drive, import/export the drive configuration, etc.

The application note “Config over Realtime” covering this topic is available from:
https://shop.linmot.com/E/product/0185-0142-E
https://shop.linmot.com/E/product/0185-0142-D

## 1.7 YouTube Video Tutorials

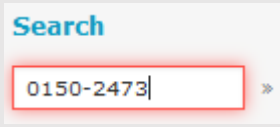
LinMot provides videos for a wide range of topics covering login, firmware installation, motor wizard, integration, and maintenance on the website.

YouTube https://linmot.com/support/linmot-tutorials/drive-configuration/

## 1.8 LinMot Product Information

Datasheets, Installation Guides, Application Notes, drivers, EPlan, CAD Files and so on for LinMot products can be found in the eCataloque: https://shop.linmot.com/

**Note:** To find all relevant information about a product just enter the article number in the search box on https://shop.linmot.com/



# 2 Overview

## 2.1 LinMot Profile (Commissioning using the LinMot Libraries)

The following table shows working combinations of different PLC types and LinMot drives. For these combinations function block libraries and/or example projects are available. Click the link in the *See chapter* column to directly open the according chapter (if available).

| PLC Type | Interface | LinMot I/F Extension | Type | Language | See chapter |
|---|---|---|---|---|---|
| Beckhoff TwinCAT 2&3 | EtherCAT<br>CANopen<br>PROFIBUS | -EC & -MI<br>-GP<br>-DP | Library | ST (Structured Text) | 3 |
| Siemens Step 7 V5.5<br>(S7-300 and higher) | PROFINET<br>PROFIBUS | -PN & -MI<br>-DP | Library | AWL | 4 |
| Siemens TIA<br>(S7-300, S7-1500) | PROFINET<br>PROFIBUS | -PN & -MI<br>-DP | Library | AWL | 4 |
| Siemens TIA<br>(S7-1200, S7-1500) | PROFINET<br>PROFIBUS | -PN & -MI<br>-DP | Library | SCL | 4 |
| Rockwell Automation | EtherNet/IP | -IP<br>-CM & -MI | Addon Instructions (AOI) | LD (Ladder) | 5<br>9 |
| B&R | Powerlink<br>CANopen | -PL & -MI<br>-GP | Library | ST | 6 |
| CODESYS (based) | Various | Various | Library | ST | 7 |
| Mitsubishi | CC-Link IE<br>Field Basic | -CC & -MI | Library | ST | 8 |

## 2.2 Drive Profiles (Direct Integration as Motion Axis using standardized Drive Profiles)

The following table shows the drive profiles supported by LinMot servo drives. Based on your PLC type the correct LinMot drive can be selected. Click the link in the *See chapter* column to directly open the according chapter (if available).

| PLC Type | Interface | LinMot I/F Extension | Integration | Drive Profile | See chapter |
|---|---|---|---|---|---|
| Rockwell Automation | EtherNet/IP (CIP Sync) | -CM & -MI | As motion axis | LinMot PVA Streaming | 9 |
| Siemens (Simotion, S7-1500, …) | PROFIdrive | -PD & -MI | As axis (TO) | Standard Telegram 5<br>SIEMENS Telegram 105 | 10 |
| Beckhoff TwinCAT 2&3 | EtherCAT CoE | -DS & -MI | As axis (NC) | CoE (DS402/CiA402) | 11 |
| Beckhoff TwinCAT 2&3 | EtherCAT SoE | -SE | As axis (NC) | SoE | 12 |
| Schneider Electric (PacDrive 3) | Sercos III | -SC & -MI | As axis | FSP Drive | 13 |
| CODESYS Motion | EtherCAT CoE | -DS & -MI | As axis | CoE (DS402/CiA402) | 14 |
| B&R | Powerlink CiA402 | -PL & -MI | As axis | CoE (DS402/CiA402) | 15 |
| B&R | Powerlink | -PL & -MI | As SDC axis | LinMot PV Streaming | 16 |
| Omron (NJ501-1500) | EtherCAT CoE | -DS & -MI | As axis | CoE (DS402/CiA402) | 17 |
| TRIO (MC4N ECAT, P904) | EtherCAT | -DS & -MI | As axis | CoE (DS402/CiA402) | 18 |
| Bosch Rexroth (IndraControl / IndraMotion) | Sercos III | -SC & -MI | As axis | FSP Drive | Footnote [1] |

[1] See Bosch Rexroth online help for GAT compact. The LinMot Sercos (-SC) drives are integrated as sercosDrive.

# 3 LinMot Profile: Beckhoff TwinCAT 2&3

## 3.1 Overview

This chapter shows how a LinMot drive with *LinMot EtherCAT* interface (e.g., C1250-EC-XC-0S) can be integrated and setup in a Beckhoff TwinCAT environment to be used with the library provided by LinMot. For this example, TwinCAT 2 is used. The steps are generally similar for TwinCAT 3.

**Download:**
The library for TwinCAT 2&3 can be downloaded from:
http://download.linmot.com/plc_lib/libraries/Beckhoff/ (named *Beckhoff_LinMot_Library_...*)

**Note:**
More information can be found in the library documentation (part of the above-mentioned download) and in the user manual EtherCAT (LinMot Profile) (see chapter 1.4 Documentation / User Manuals)

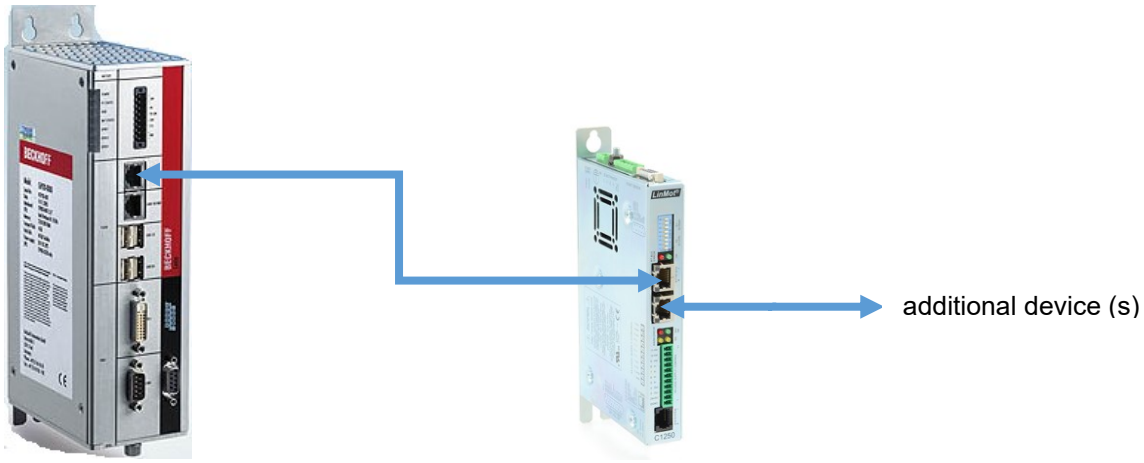



Image Source: http://www.beckhoff.com/

EtherCAT is the open real-time Ethernet network originally developed by Beckhoff. The LinMot acts as Slave in this network and is implemented with the standard ASIC ET1100 from Beckhoff.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

## 3.2 Configuration and Connection of the LinMot Drive

The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 3.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


See Appendix I: Basic Position Control Loop Tuning

### 3.2.2 XML Files

Install the XML file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

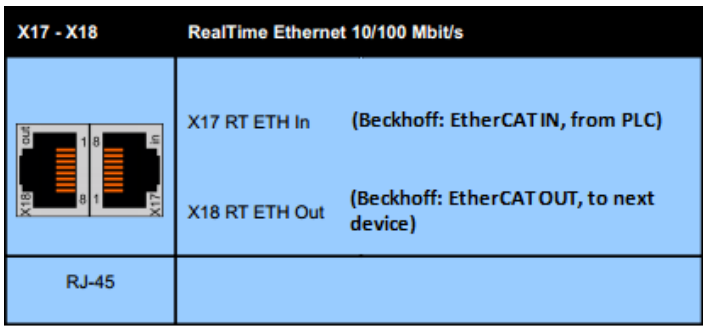
- EtherCAT: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML\
- EtherCAT: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML\ (-MI drives)

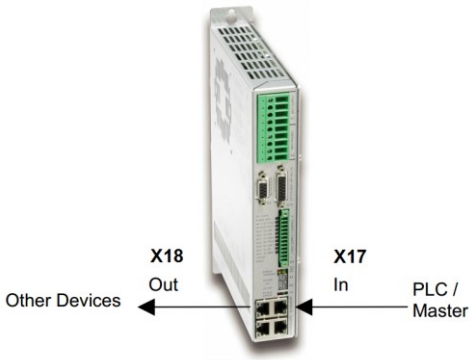
**XML file names:**

| | |
|---|---|
| NTIL_LM_Servos_Vx_xrx.xml | LinMot EC drives SG6-7 |
| NTIL_LM_SG5_Servos_Vx_xrx.xml | LinMot EC drives SG5 |
| NTIL_CiA402_Servos_MI_Vx_xrx.xml | LinMot MI drives SG6 |

### 3.2.3 EtherCAT Connection

The drive is connected to the EtherCAT network using the X17 & X18 connectors.
The below pictures show the ports of an E1250-EC-UC drive. On all other LinMot drives supporting EtherCAT the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.



## 3.3 PLC Setup EtherCAT (E1450-EC-QN, E1250-EC-UC, C1x00-EC-XC-xx)

### 3.3.1 Insert LinMot Drive as Slave

The required XML files can be found in the following folder (default):

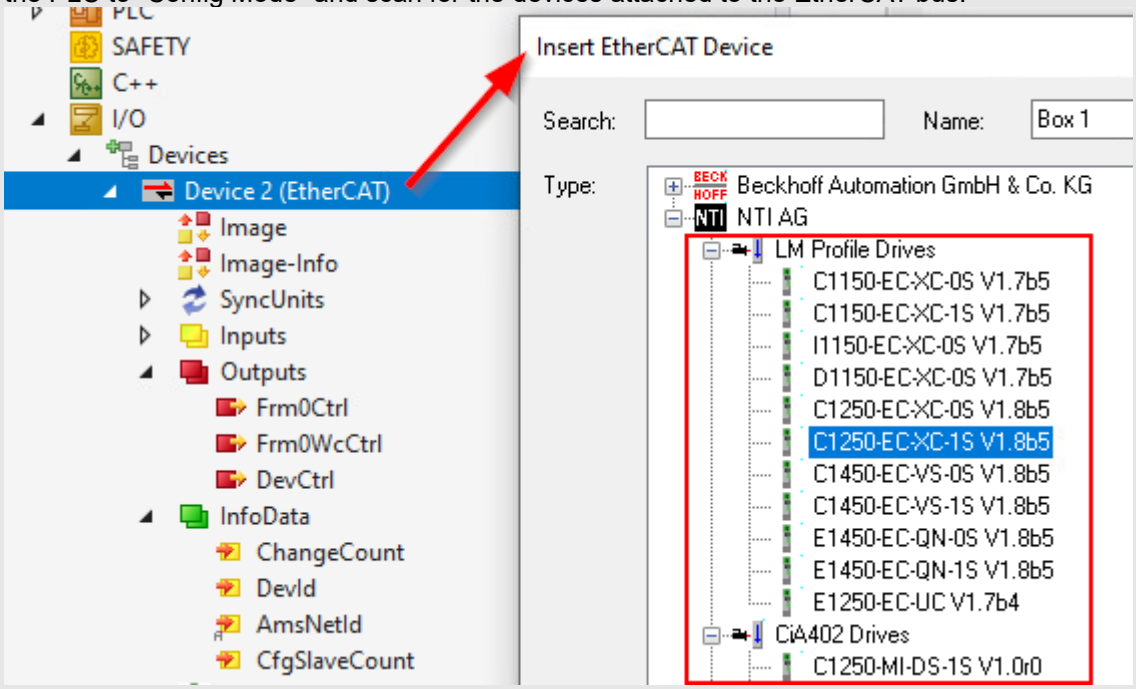
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_NX\XML (-MI Drives)

These files must be copied to the IO folder of TwinCAT:

- C:\TwinCAT\Io\EtherCAT\ (for TwinCAT 2)
- C:\TwinCAT\3.1\Config\Io\EtherCAT\ (for TwinCAT 3)

**Note:** You may have to restart the developing environment to have the new XML file to be recognized.

Insert a new device by right-click on the EtherCAT-Master and *Append Box* (figure below) or restart the PLC to “Config Mode” and scan for the devices attached to the EtherCAT bus.

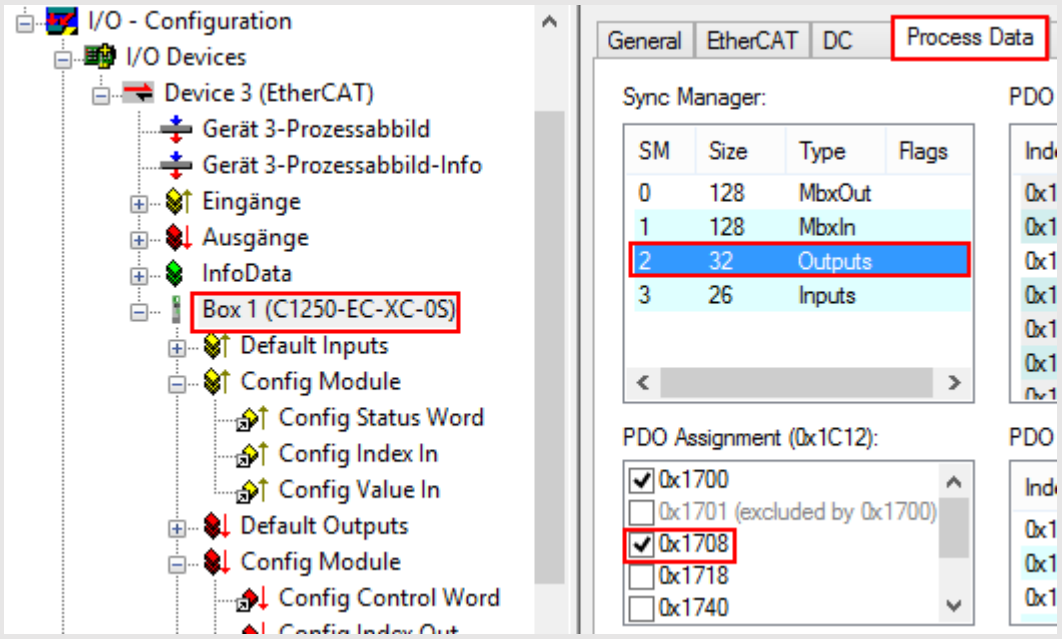


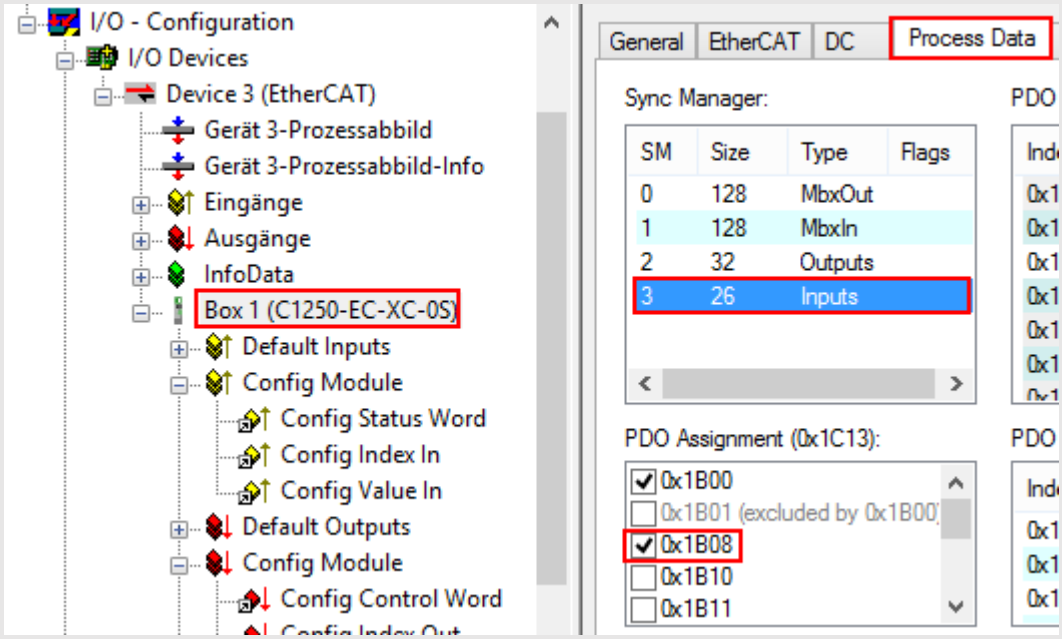

**ATTENTION:**
For **-MI drives** (they show up as CiA402 drives, e.g., *C1250-MIDS-1S V1.0r0*) make sure that **input PDOs 0x1B00 and 0x1B08** as well as **output PDOs 0x1700 and 0x1708** are selected.

### 3.3.2 Config Module

The *Config Module* provides access to parameters, curve profiles, command table and so on of a LinMot drive.

If the config function blocks of the library are used the „Config Module“ inputs and outputs (**0x1B08 & 0x1708**) must be enabled and linked as well.



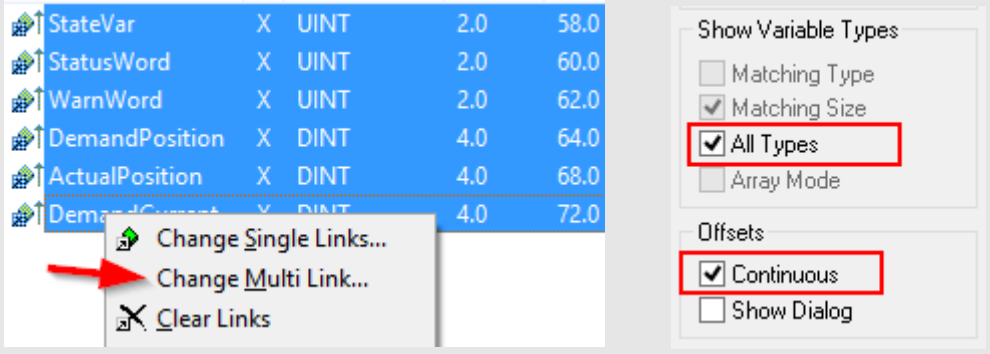
### 3.3.3 Links to PLC Control

To be able to link the input and output data of the LinMot drive you must create an instance of the *tstLM_Axis* data type (part of the library) in the global variables of you PLC project.

The inputs and outputs of the modules must be linked as shown below:

| Name | Linked to |
|---|---|
| StateVar | StateVar . .Axis_A_Axis.DrvToPlc . Inputs . Standard . Li... |
| StatusWord | StatusWord . .Axis_A_Axis.DrvToPlc . Inputs . Standard ... |
| WarnWord | WarnWord . .Axis_A_Axis.DrvToPlc . Inputs . Standard .... |
| DemandPosition | ComDemandPosition . .Axis_A_Axis.DrvToPlc . Inputs .... |
| ActualPosition | ComActualPosition . .Axis_A_Axis.DrvToPlc . Inputs . S... |
| DemandCurrent | ComActualCurrent32 . .Axis_A_Axis.DrvToPlc . Inputs .... |
| Config Status Word | CfgStatusWord . .Axis_A_Axis.DrvToPlc . Inputs . Stand... |
| Config Index In | CfgIndexIn . .Axis_A_Axis.DrvToPlc . Inputs . Standard .... |
| Config Value In | CfgValueIn . .Axis_A_Axis.DrvToPlc . Inputs . Standard .... |
| Control Word | ControlWord . .Axis_A_Axis.PlcToDrv . Outputs . Stand... |
| Motion Command Header | MCHeader . .Axis_A_Axis.PlcToDrv . Outputs . Standar... |
| Motion Command Par 1 | MCParaDWord_00_03 . .Axis_A_Axis.PlcToDrv . Output... |
| Motion Command Par 2 | MCParaDWord_04_07 . .Axis_A_Axis.PlcToDrv . Output... |
| Motion Command Par 3 | MCParaDWord_08_11 . .Axis_A_Axis.PlcToDrv . Output... |
| Motion Command Par 4 | MCParaDWord_12_15 . .Axis_A_Axis.PlcToDrv . Output... |
| Motion Command Par 5 | MCParaDWord_16_19 . .Axis_A_Axis.PlcToDrv . Output... |
| Config Control Word | CfgControlword . .Axis_A_Axis.PlcToDrv . Outputs . St... |
| Config Index Out | CfgIndexOut . .Axis_A_Axis.PlcToDrv . Outputs . Stand... |
| Config Value Out | CfgValueOut . .Axis_A_Axis.PlcToDrv . Outputs . Stand... |

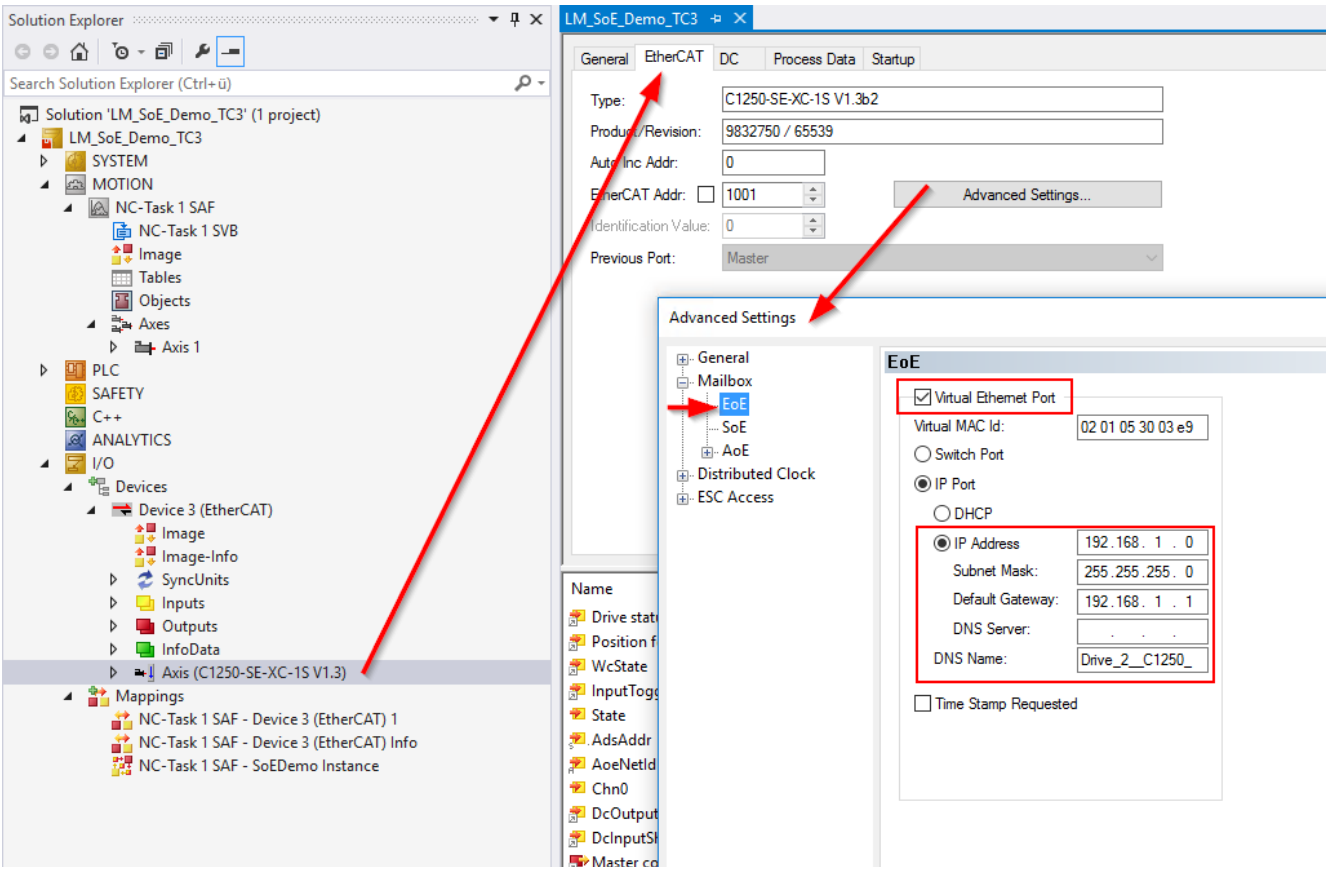
**Hint:**
It is possible to change multiple links at once. Just select the names you want to link > right-click > Change Multi Link. Don't forget to check "All Types" and "Continuous".



### 3.3.4 Check EoE settings

On supported drives (C1250-..., C1450-… & E1450-...) please check the EoE (Ethernet over EtherCAT) settings.

LinMot drives do NOT support the DHCP mode yet. A fix IP must be assigned (select *IP Address*) or the EoE feature must be completely disabled (disable Virtual Ethernet Port).




**Note:**
EoE can be used to login into the drive with LinMot-Talk directly over EtherCAT.
E.g., to change drive settings or for monitoring and tracing.

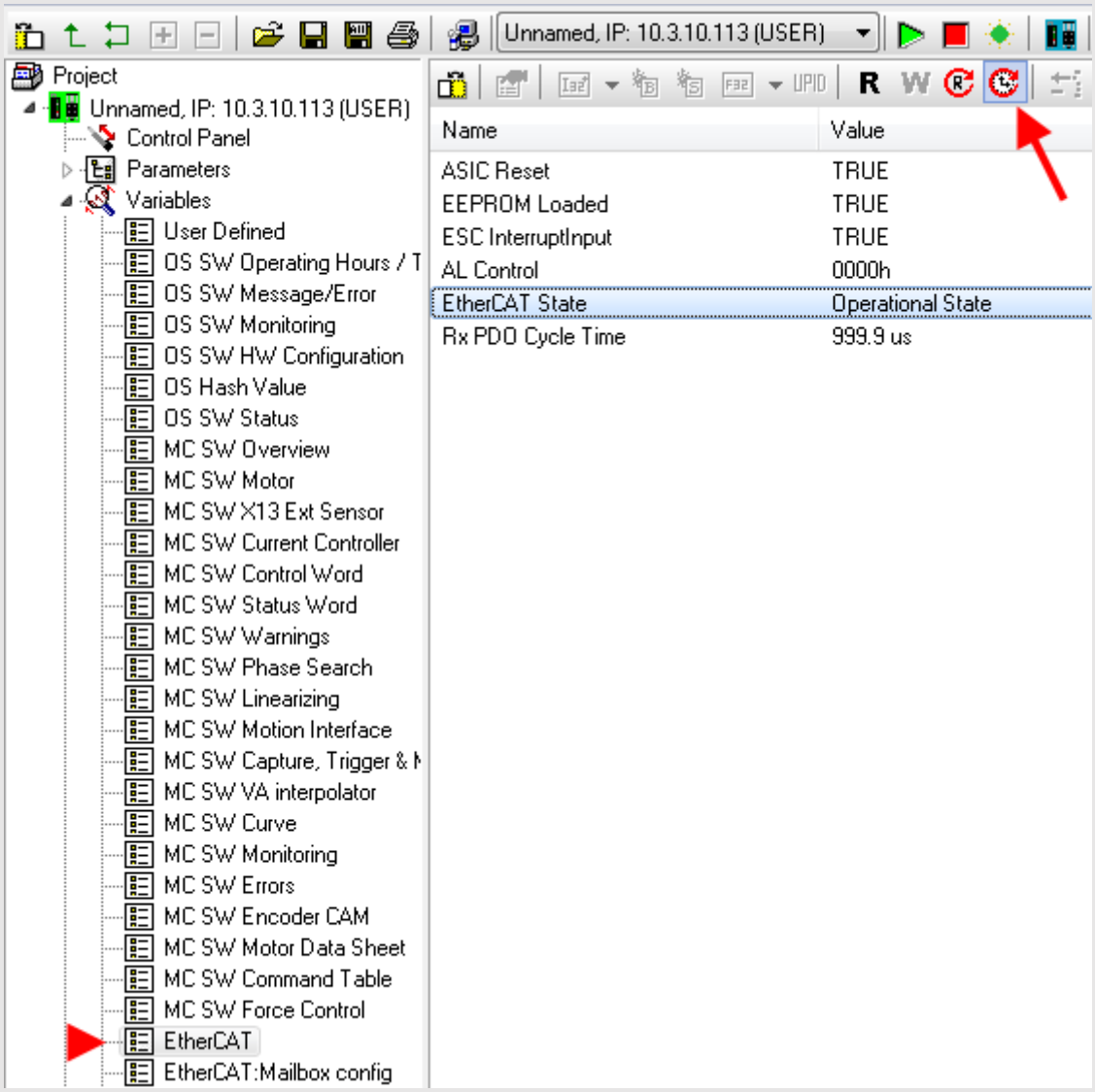
## 3.4 Add Additional Parameters or Variables to the Process Data

To add additional parameters or variables for read and write access to the process data please see chapter *11.10 Add Additional Parameters or Variables to the Process Data*.

## 3.5 Check with LinMot-Talk if the Fieldbus is running

LinMot-Talk shows the status of the fieldbus. Open the variables EtherCAT and check if the bus is running (EtherCAT State = Operational State).

Check fieldbus status:




## 3.6 Next Steps

Now you can start implementing your application using the function blocks and documentation from the download link in chapter **3.1 Overview.**

# 4 LinMot Profile: Siemens TIA

## 4.1 Overview

This chapter shows how a LinMot drive with either *PROFIBUS* or *PROFINET* interface (LinMot Profile) can be connected to a Siemens PLC using TIA Portal (for Step 7 the general steps are similar).

**Download:**
The libraries for TIA / Step 7 can be downloaded from:
http://download.linmot.com/plc_lib/libraries/Siemens/ (named *Siemens_TIA... or Siemens_Step7…*)

**Note:**
More information can be found in the library documentation (part of the above-mentioned download) and in the user manuals PROFINET & PROFIBUS (LinMot Profile) (see chapter 1.4 Documentation / User Manuals)


Image Source: http://automation.siemens.com/

PROFINET is an Industrial Ethernet Standard. The LinMot PROFINET drives act as PROFINET-IO slaves in this network.

PROFIBUS is a fieldbus standardized in IEC 61158. The LinMot PROFIBUS drives are PROFIBUS-DP slaves.

For further information on PROFINET / PROFIBUS please visit:
http://www.PROFIBUS.com

## 4.2 Configuration and Connection of the LinMot Drive

The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 4.2.1 Motor configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


See Appendix I: Basic Position Control Loop Tuning

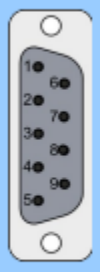
### 4.2.2 GSD / GSDML files

Install the GSD/GSDML file(s) that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- PROFIBUS: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Profibus\GSD
- PROFINET: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\ProfiNet\GSDML
- PROFINET: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\ProfiNet_Nx\GSDML_PD (For -MI drives)

### 4.2.3 PROFIBUS Connection

The drive is connected to the PROFIBUS network using the X9 connector of the LinMot drive.

| X9 | PROFIBUS DP | | |
|---|---|---|---|
|  | 1 | Not connected | |
| | 2 | Not connected | |
| | 3 | RxD/TxD-P | |
| | 4 | CNTR-P | |
| | 5 | GND | (isolated) |
| | 6 | +5V | (isolated) |
| | 7 | Not connected | |
| | 8 | RxD/TxD-N | |
| | 9 | Not connected | |
| | case | Shield | |
| DSUB-9 (f) | Max. Baud rate: 12Mbaud | | |

The only setting that must be done on the drive according to the PROFIBUS interface is setting the node address. By default, it is set with the address selectors *S1 & S2* on the front of the drive.
Alternatively (not recommended) it can be set with the parameter “Node Address Parameter Value” (UPID 2076h). This requires the parameter “Node Address Selection” (UPID 206Ch) to be set to “On”.

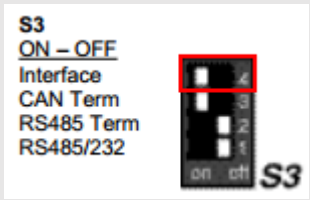
Set the node (Profibus) address using the address selectors *S1 & S2*

| S1 - S2 | Address Selectors | | |
|---|---|---|---|
| E1100<br>E1200 V1 | E1200 V2<br>E1400<br>C1x00 | | |
|  |  | S1 (5 .. 8) | Bus ID High (0 ... F). Bit 5 is LSB, bit 8 MSB. |
| | | S2 (1 .. 4) | Bus ID High (0 ... F). Bit 1 is LSB, bit 4 MSB. |

The rotary switches are hexadecimal.

**Attention:** Do not change any other PROFIBUS interface parameters on the drive!

**Attention:** To activate the PROFIBUS Interface on E1130-DP-xx drives, the Dip-Switch S3.4 “Interface” at the bottom of the drive has to be set to “ON” (factory default)

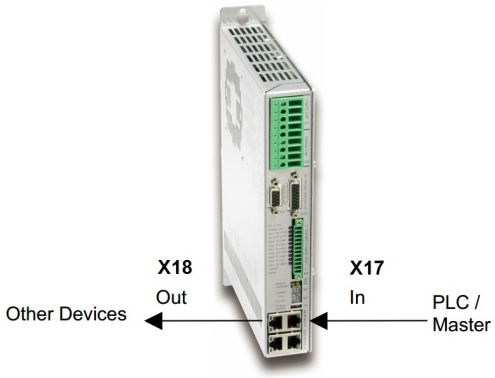



**Attention:**
According to the PROFIBUS standard the **maximal node address is 127** (addresses 126 and 127 are reserved for special purpose and should normally not be used).

### 4.2.4 PROFINET Connection

The drive is connected to the PROFINET network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-PN-UC drive. On all other LinMot drives supporting PROFINET the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.

| X17 - X18 | RealTime Ethernet 10/100 Mbit/s |
|---|---|
|  | X17 RT ETH In (Siemens: Port1, X1.X1)<br>X18 RT ETH Out (Siemens: Port2, X1.X2) |
| RJ-45 | |




**Attention:**
The PROFINET parameters in the drive are left to default values!
The name and IP address are set using the TIA Portal, see chapter 4.4.1

## 4.3 PLC Setup PROFIBUS (E1430-DP-QN, E1230-DP-UC, E1130-DP-xx)

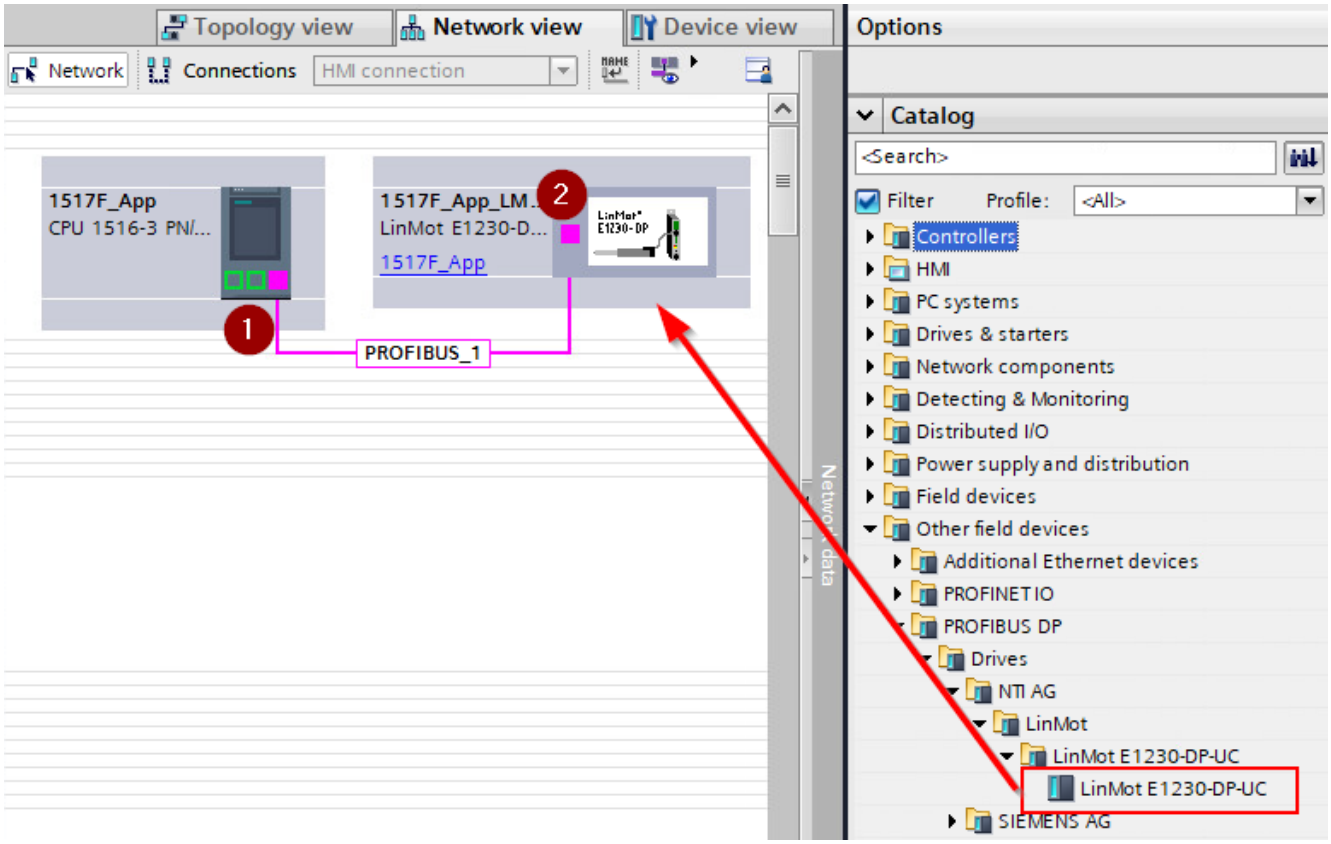
The following procedure can be used for all Siemens PLCs compatible with TIA Portal.

First, install the GSD file for the drive.
(TIA Portal → Options → Install general station description file (GSD))

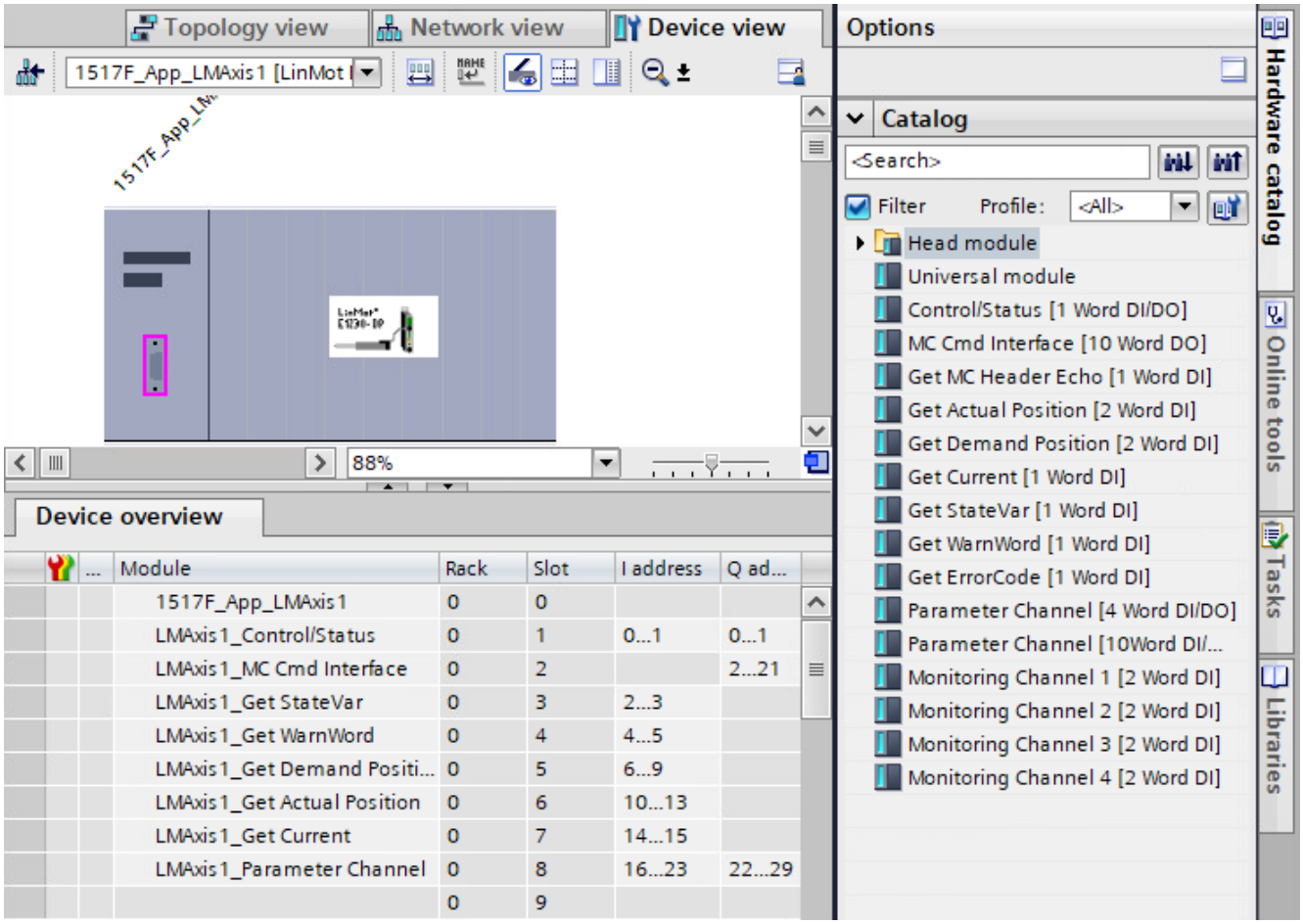
The required GSD file can be found by default in the following folder:

*C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\PROFIBUS\GSD*

Drag and drop the desired drive from the catalogue to the network view and connect it to the Profibus network (1 & 2)



Afterwards insert the required modules into the available slots as shown in the figure below.




**Note:** Set the name and the PROFIBUS address of the device according to the requirements of your application

## 4.4 PLC Setup PROFINET (E1450-PN-QN, E1250-PN-UC, C1250-PN-UC, C1150PN-UC)

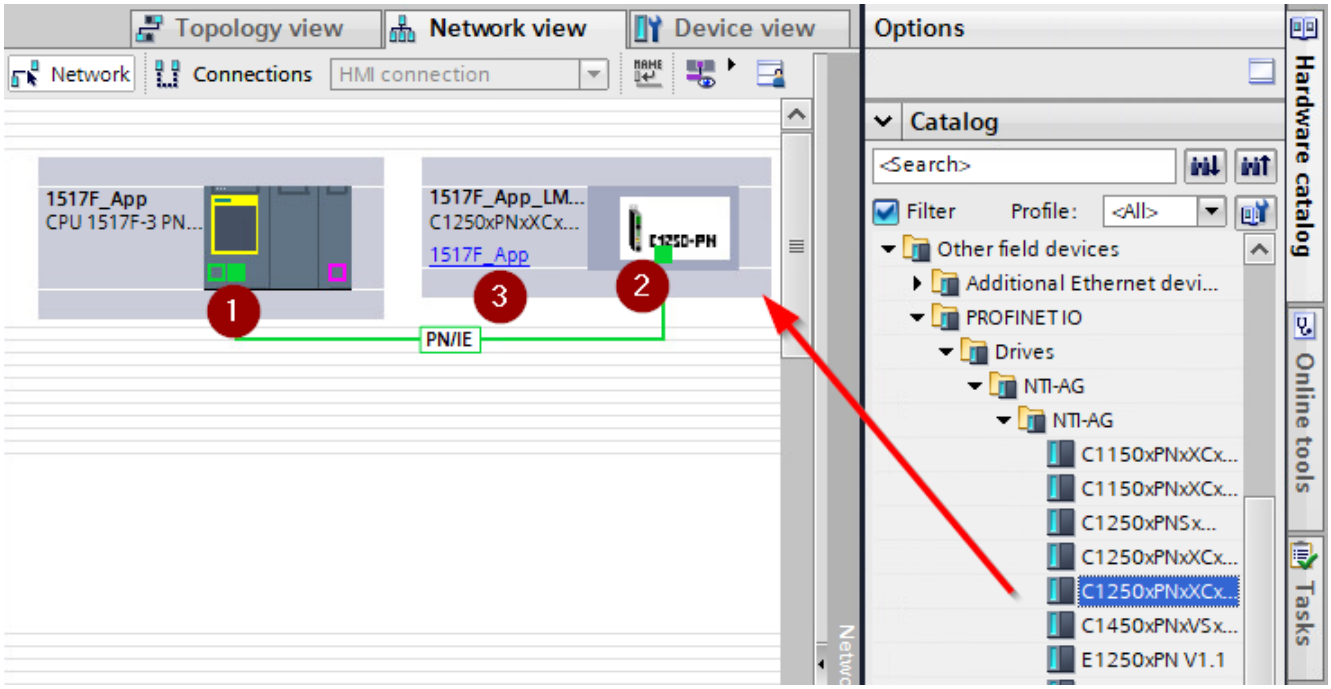
The following procedure can be used for both CPU S7-1200 and CPU S7-1500.

First, install the GSDML file for the drive.
(TIA Portal → Options → Manage general station description file (GSD))

The required GSDML file can be found by default in the following folder:

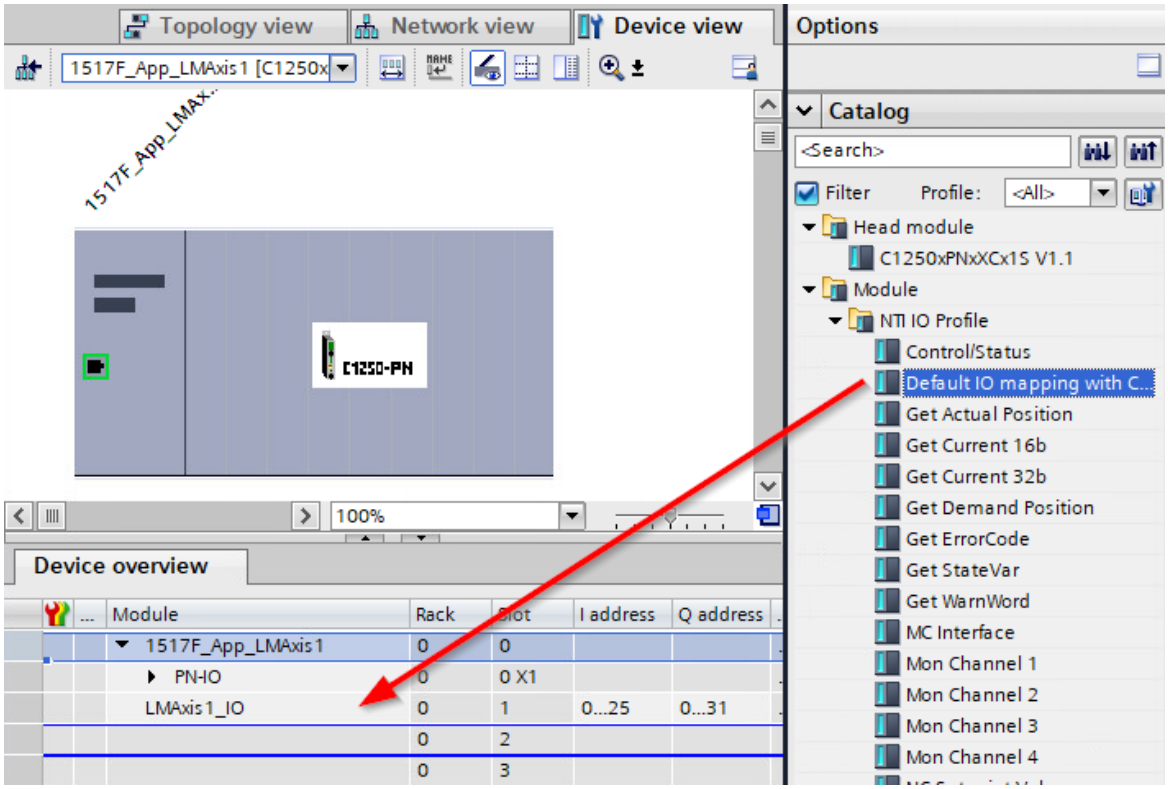
*C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\ProfiNet\GSDML*
*C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\ProfiNet_Nx\GSDML_PD\ (for -MI drives)*

Drag and drop the desired drive from the catalogue to the network view, connect the drive with the PLC (draw line from 1 to 2) and assign the drive to the PLC (3).

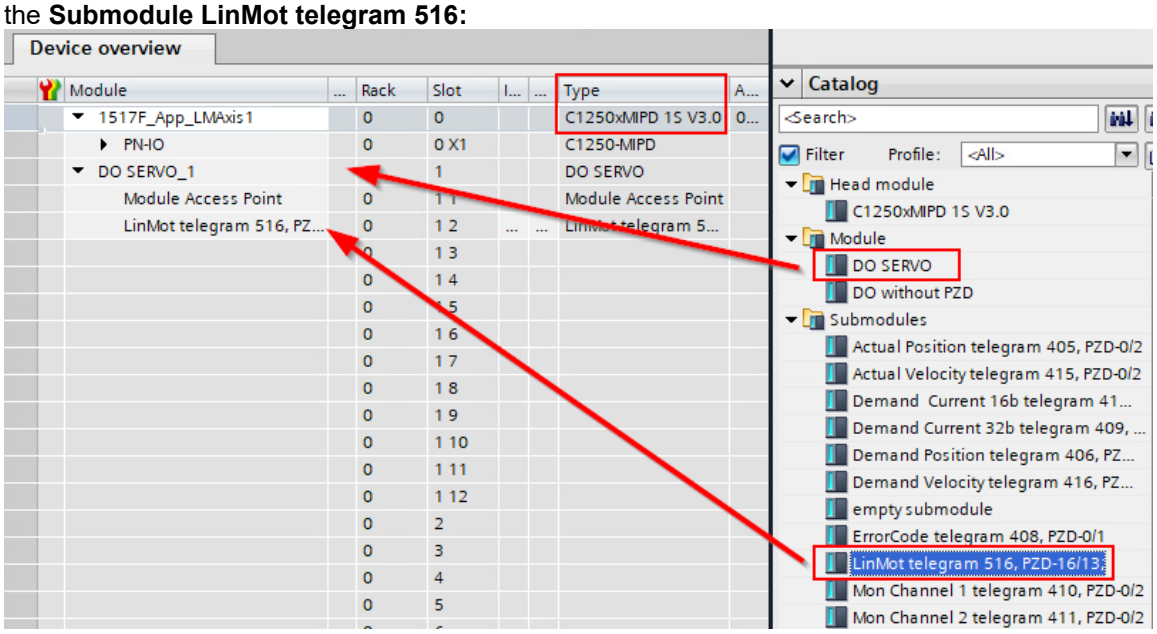


Afterwards in the Device View the module „Default IO mapping with Config“ is dropped into the available slot as shown the figure below.

By default, when adding a new **LinMot -PN drive**, this module is already present in slot 1.




When adding a **C125x-MI drive** (e.g., C1250xMIPD 1S V3.0) insert the **DO SERVO Module** first and then add the **Submodule LinMot telegram 516:**

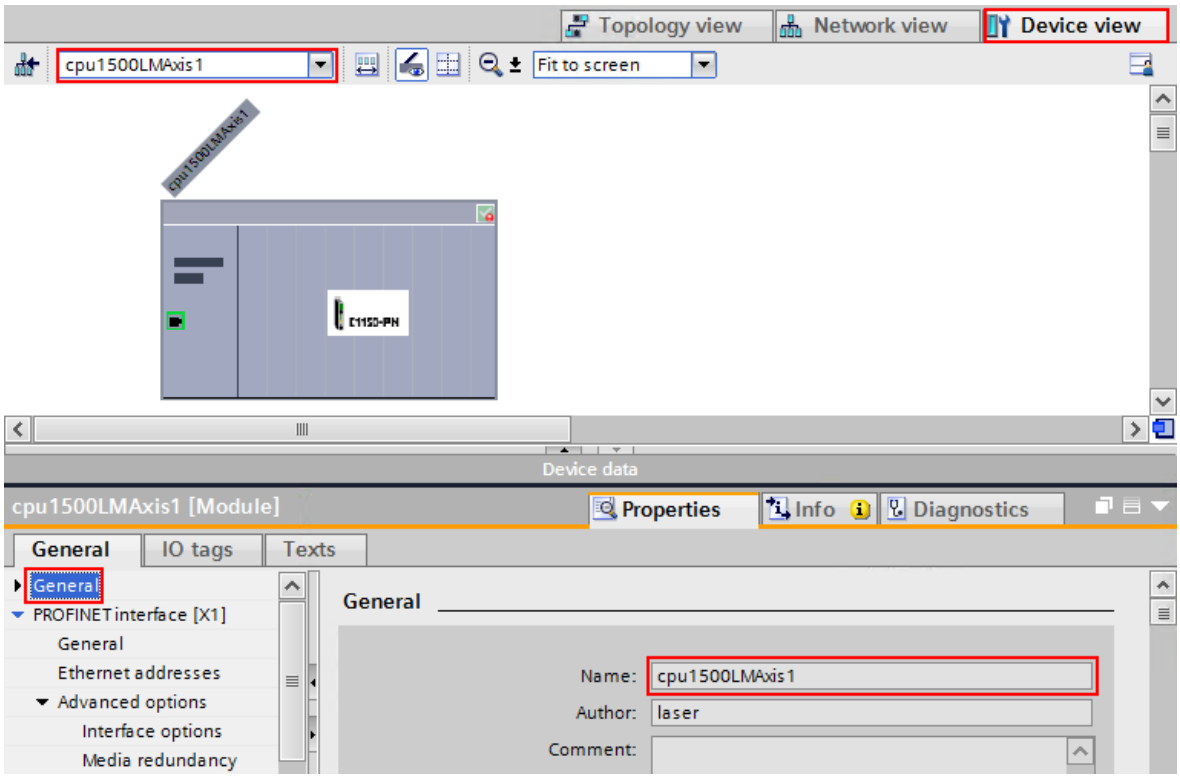


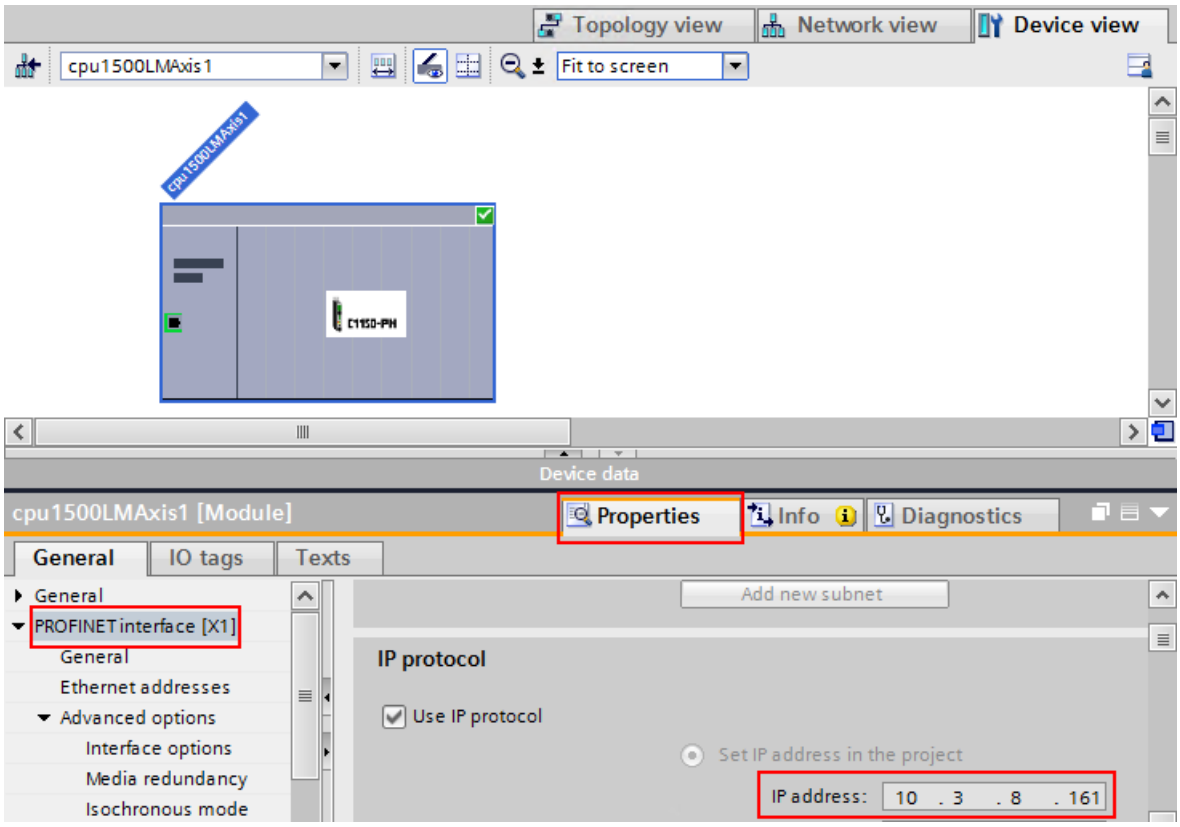

**Tip:**
It is possible to add up to 4 parameter and 4 monitoring channels to write or read specific parameters or variables in the process data.
Please check chapter ***10.6 Add Additional Parameters / Variables to the Real-Time Channel*** for details.

### 4.4.1 Assigning the Device Name and IP Address

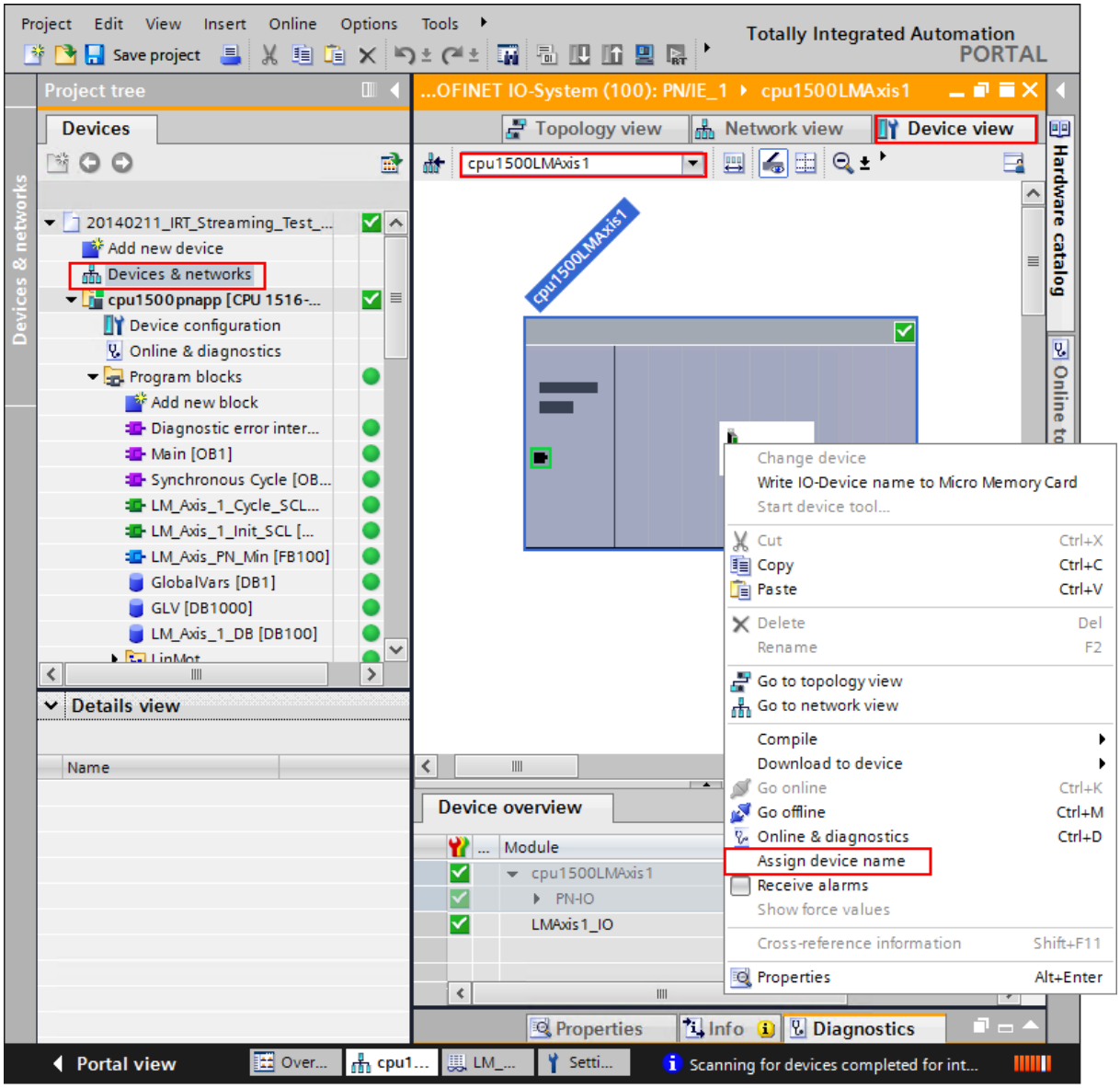
The following three steps show how to assign the PROFINET device name to the LinMot drive using TIA Portal. The IP address is assigned automatically when PROFINET starts.

1. Set the IP address and the device name in the device properties:

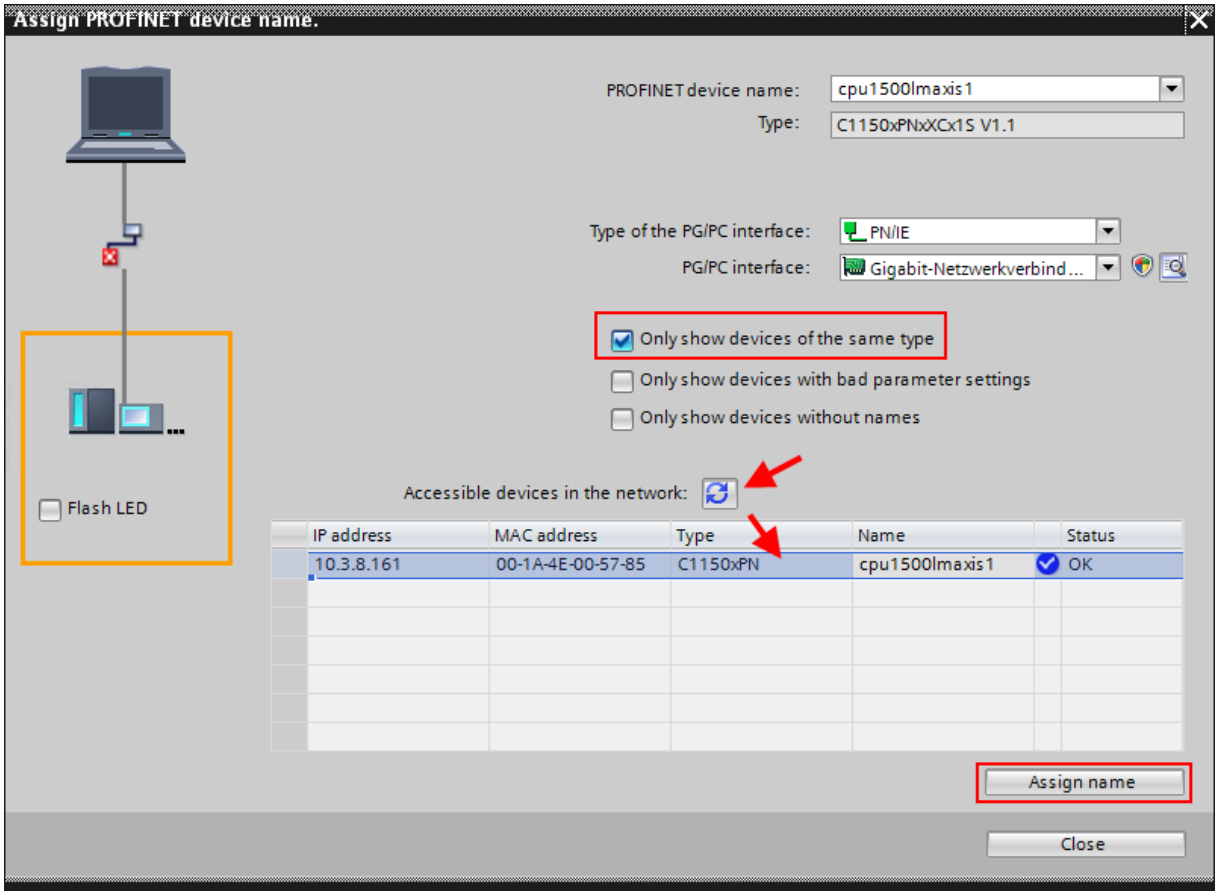


2. Choose the device you want to assign the name to (right-click on the device):



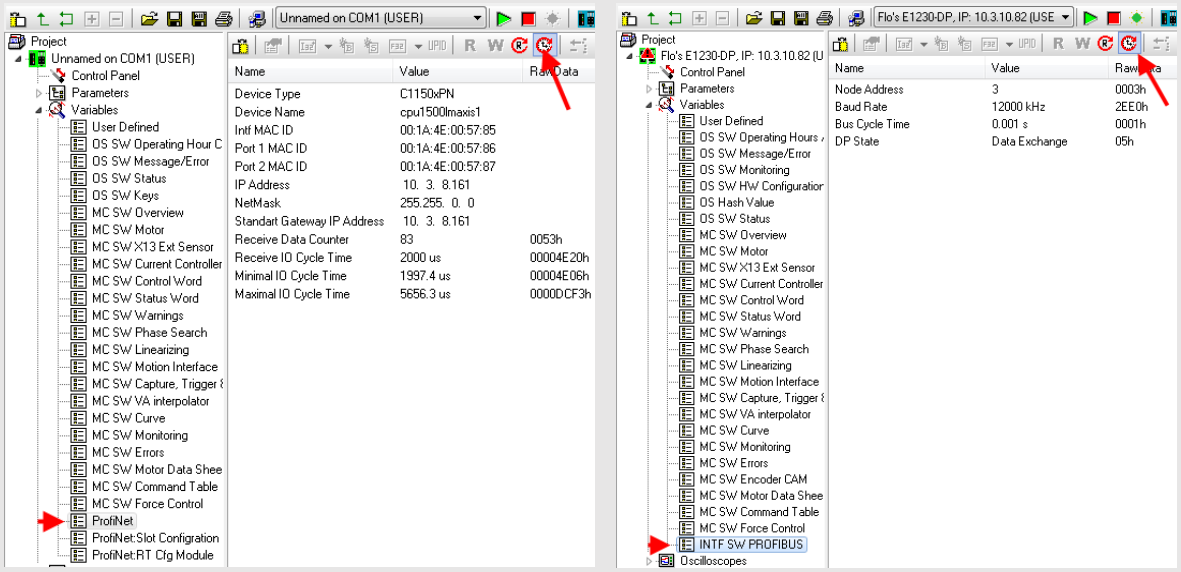
3. Search for the LinMot drive you want to name. The safest way to identify the drive is by either activating the “Flash LED” function in TIA Portal or to verify the MAC address (can be found on the housing of the drive).



## 4.5 Check with LinMot-Talk if the Fieldbus is running

LinMot-Talk shows the status of the fieldbus. Open the variables PROFIBUS / PROFINET and check if the Node Address / IP Address is correct.

Check fieldbus status:




If the “Receive Data Counter” is counting and there is a valid “Receive IO Cycle Time” is shown the bus is properly running.

## 4.6 Next Steps

Now you can start implementing your application using the function blocks and documentation from the download link in chapter **4.1 Overview.**

# 5 LinMot Profile: Rockwell Automation Studio 5000 (Outdated)

## 5.1 Overview

**Update July 2022: LinMot recommends -CM or -MI drives instead of -IP drives for new applications**

The most recent instruction set for LinMot CM and MI drives (CIPSync Motion) now also supports LinMot -IP drives. Please find the instruction set with included example projects and documentation here:
http://download.linmot.com/plc_lib/examples/Rockwell_CM/

This chapter shows how a LinMot drive with *LinMot EtherNet/IP* interface (e.g., C1250-IP-XC-0S) can be integrated and setup in a Rockwell Automation environment to be used with the library provided by LinMot.

**Download:**
The Add-On Instructions for Rockwell Automation controllers can be downloaded from:
http://download.linmot.com/plc_lib/libraries/Rockwell/ (named *Rockwell_LM_AOI_Set_...*)

**Note:**
More information can be found in the library documentation (part of the above-mentioned download) and in the user manual EtherNet/IP (see chapter 1.4 Documentation / User Manuals)

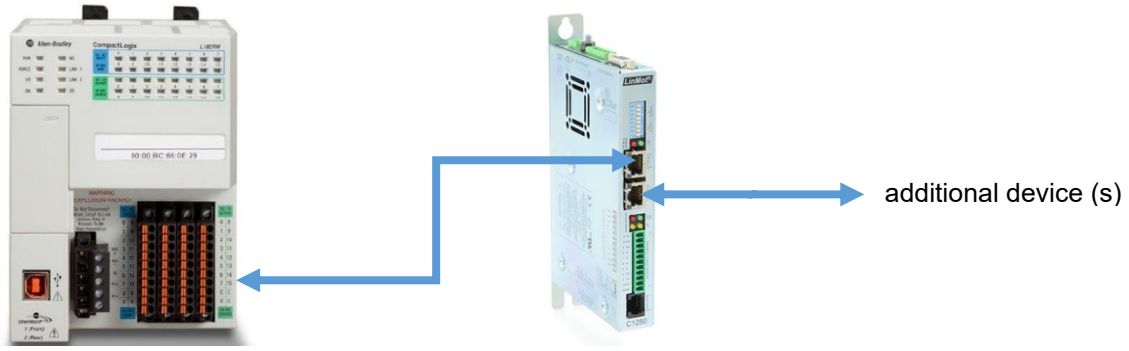



Image Source: http://www.rockwellautomation.com/

EtherNet/IP is an industrial Ethernet network that implements the Common Industrial Protocol (CIP).

For further information on EtherNet/IP please visit:
http://www.odva.org

**YouTube video series** (LinMot Integration – Rockwell Automation EtherNet/IP):
https://youtube.com/playlist?list=PLMSLCScLnf94pElSNCorvUf2-RpGJ8L6R

## 5.2 Minimum Requirements

**Important:**
Use only Allen-Bradley controller firmware revision 18.0 or higher!
For EDS support Allen-Bradley controller firmware revision 20.0 or higher is required!

The AOP for C1250-IP-XC-xS and E1250-IP-UC drives are included in Studio 5000 from V31!
For Studio 5000 versions older than V31 the AOP installation package can be downloaded from:
http://download.linmot.com/plc_lib/libraries/Rockwell/ (LM_DRIVE_Rel15_RELEASE.zip)

**Attention:**
When you have installed the AOP manually, e.g., from the link above, it is highly recommended to uninstall them before installing Studio V31. As they are part of Studio V31 errors may occur during installation.

## 5.3 Configuration and Connection of the LinMot Drive

The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 5.3.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


See Appendix I: Basic Position Control Loop Tuning

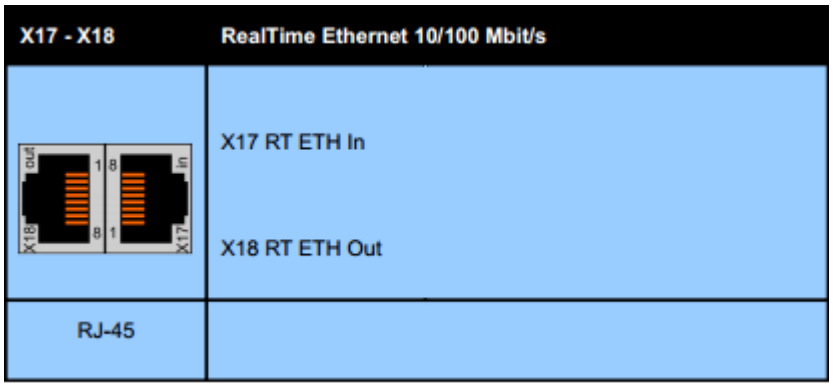
### 5.3.2 EDS Files (Requires Studio 5000 V20 or higher)

Install the EDS file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- EtherNet/IP: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherNetIP\EDS

### 5.3.3 EtherNet/IP Connection

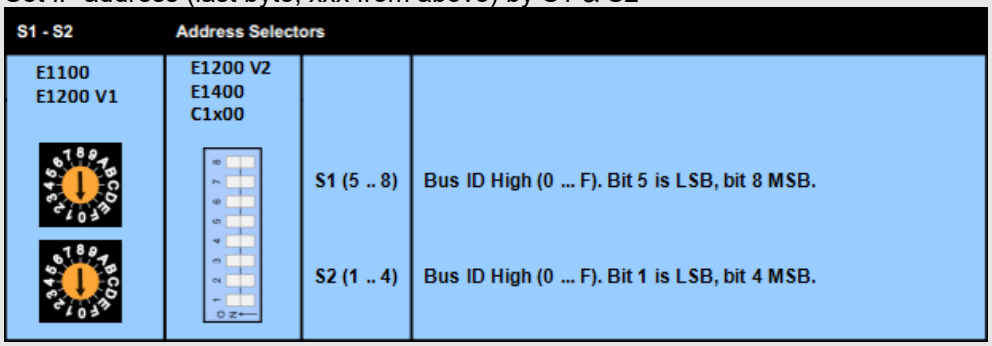
The drive is connected to the EtherNet/IP network using the X17 & X18 connectors.




### 5.3.4 IP Address

The default IP address is 192.168.001.xxx, where the last byte xxx is defined via the two address selectors *S1 & S2*. *S1* sets the high and *S2* the low digit. E.g., S1 = 5, S2 = A -> 5A (hex) = 90 (dec) -> IP = 192.168.1.**90**

Set IP address (last byte, xxx from above) by S1 & S2



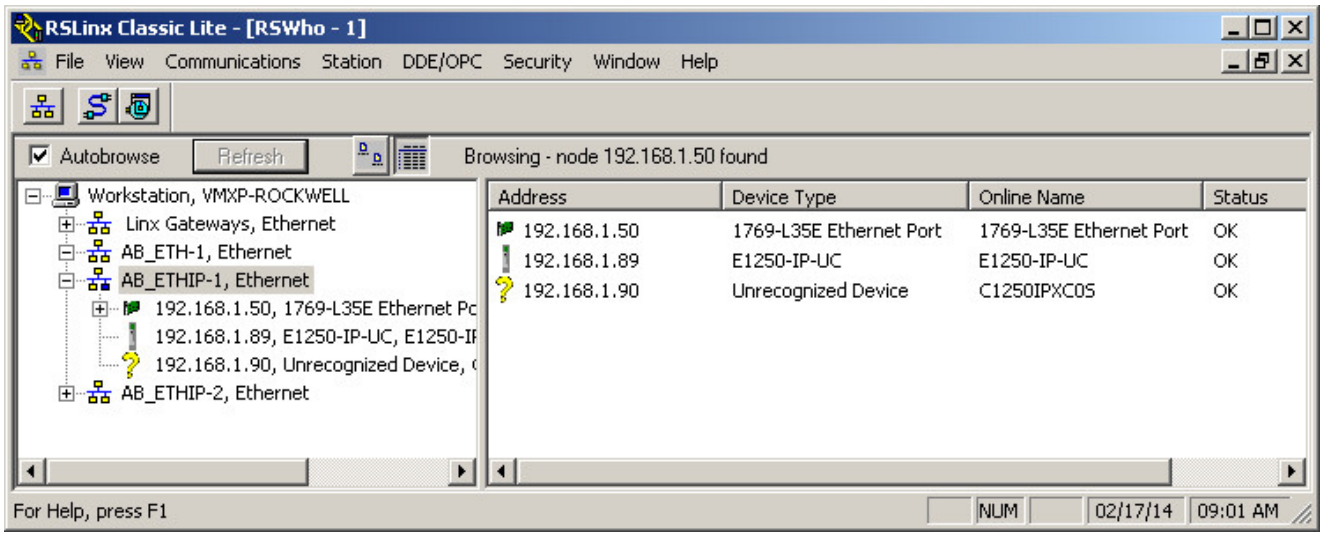

**Attention:**
The switch value **S1 = S2 = 0 (factory default setting)** is a special configuration which acquires the IP address via DHCP (e.g., for use with the BOOTP-DHCP Tool from Rockwell Automation)!

## 5.4 PLC Setup EtherNet/IP

### 5.4.1 RSLinx Classic

After setting the IP address the new LinMot device should appear in RSLinx Classic as “Unrecognized Device” (e.g., IP 192.168.1.90 see below).

If the EDS file is installed, the LinMot drive will appear in RSLinx with its Device Type (e.g., IP 192.168.1.89).




**Important:**
If no AOP or EDS file is used the LinMot must be setup as Generic Ethernet Module.
See chapter 5.4.4

### 5.4.2 Add a LinMot drive using AOP (E1250-IP-xx, C1250-IP-xx)

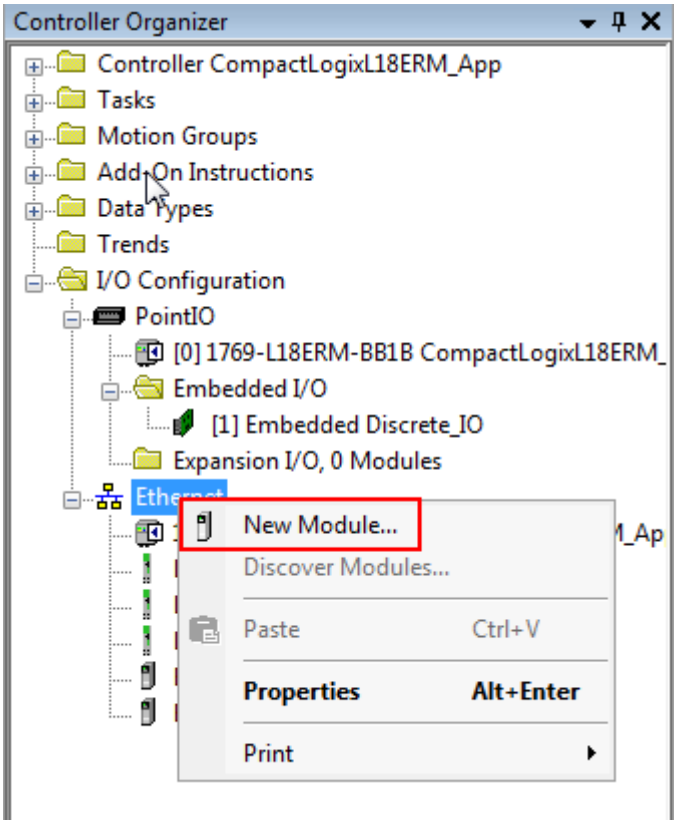
**Important:**
The AOP for C1250-IP-XC-xS and E1250-IP-UC drives are included in Studio 5000 from V31!

For Studio 5000 version <V31 the AOP installation package for manual installation can be downloaded from:
http://download.linmot.com/plc_lib/libraries/Rockwell/ (LM_DRIVE_Rel15_RELEASE.zip)

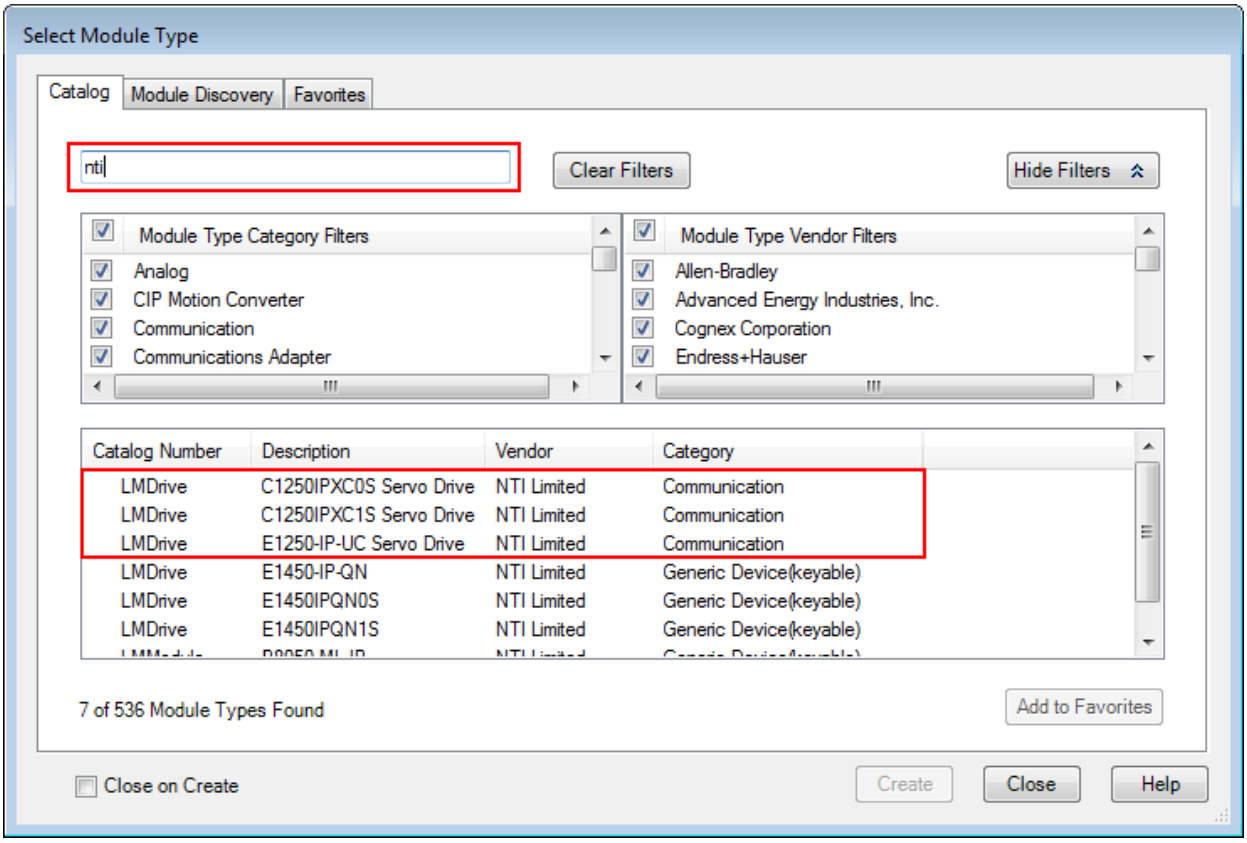
**Attention:**
When you have installed the AOP manually it is highly recommended to uninstall them before installing Studio V31.

How to uninstall AOP:
1) You need to have the installation files for the version of the AOP you are currently running unzipped and available.
2) Go to a DOS (command) window. Then navigate to the folder where you have the installation files for the version of the AOP you wish to uninstall.
3) Execute this command: MPSETUP/CU
4) Follow the prompts and make sure that you check the UNINSTALL checkbox.
5) Click "Next". Expand the tree in the left pane to see choices for which AOP(s) you wish to uninstall. Select which AOPs to uninstall.
6) Follow the prompt to finish the uninstall

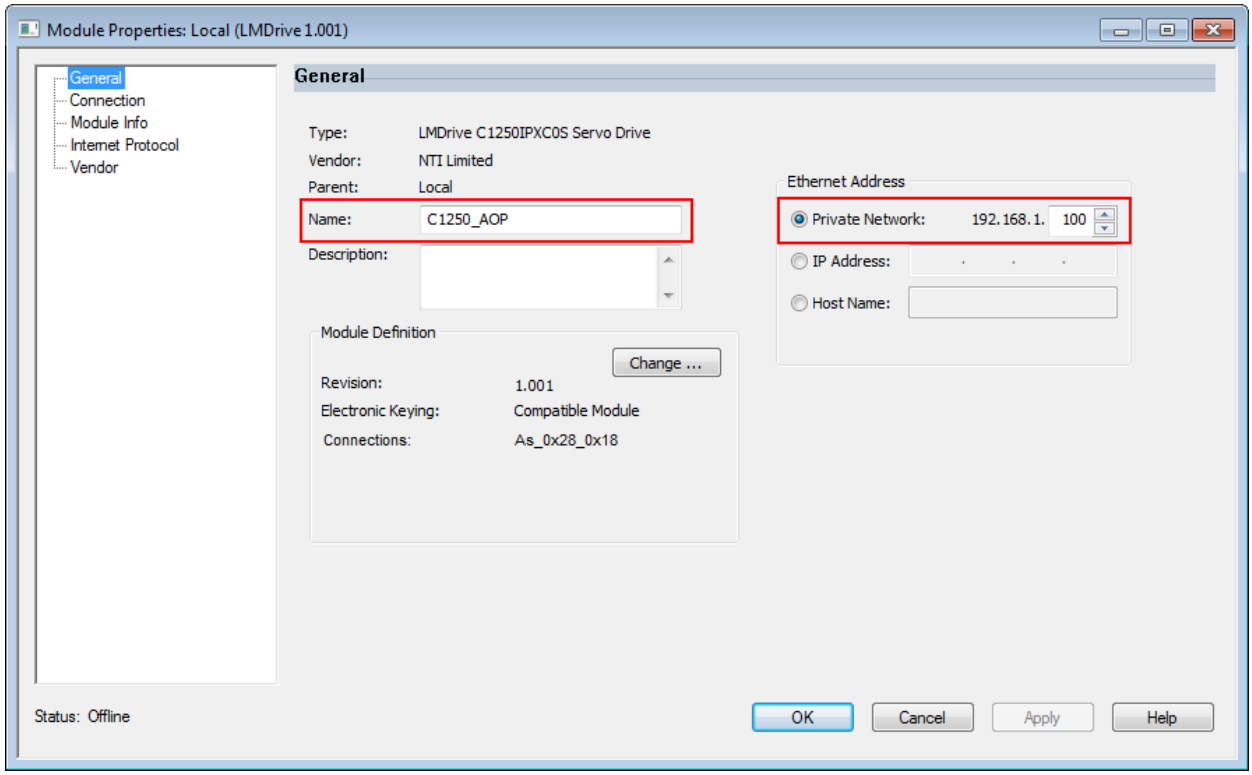
1. Add a new module by right-clicking on Ethernet in the I/O configuration:



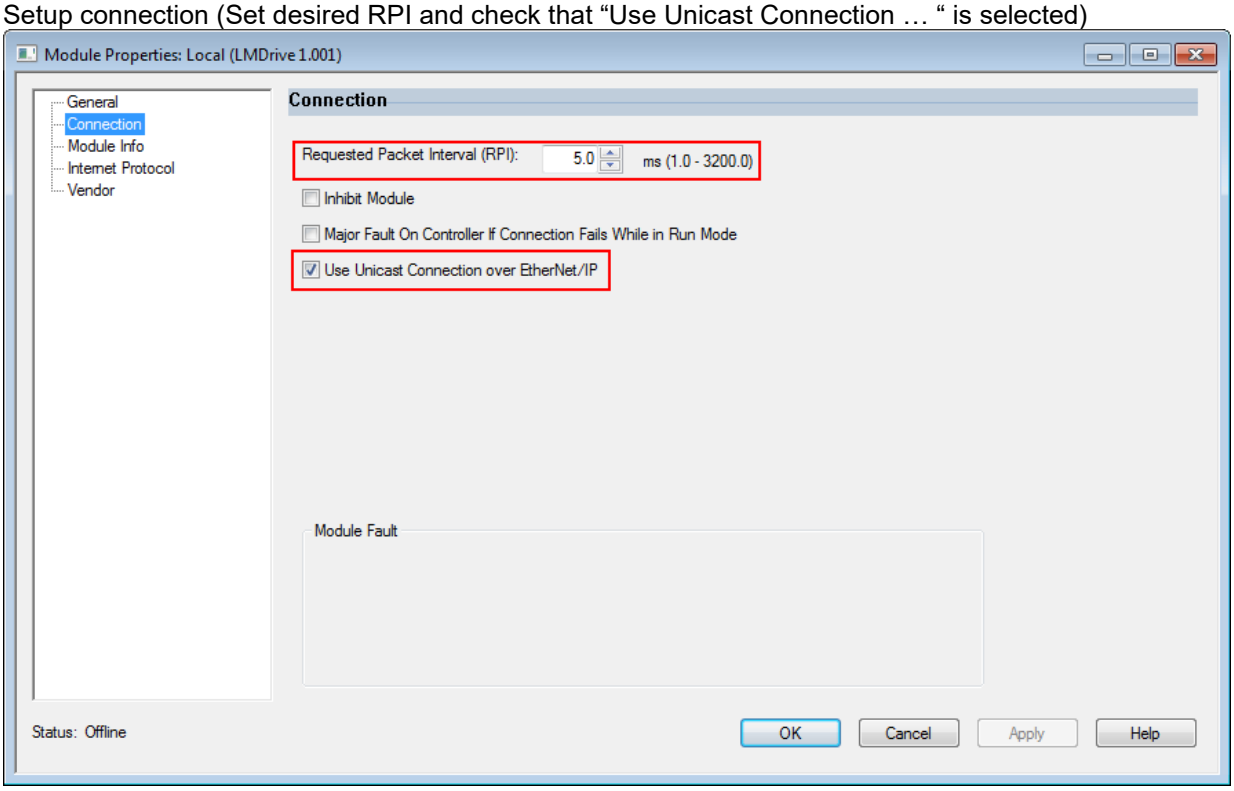
2. Select module type (LMDrive):



3. Set name and IP address:



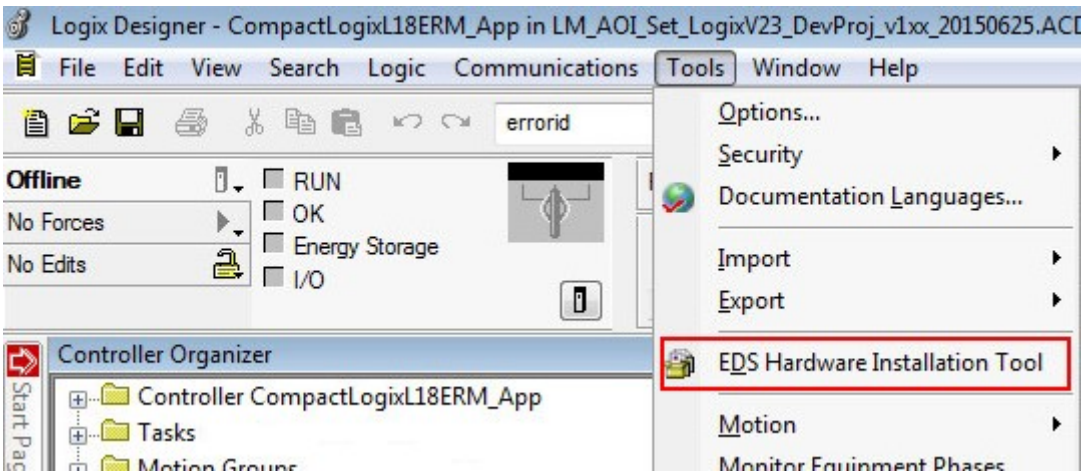
4. Setup connection (Set desired RPI and check that “Use Unicast Connection … “ is selected)




Now you can download your project to the controller and check if the connection is running, as described in chapter 5.5.

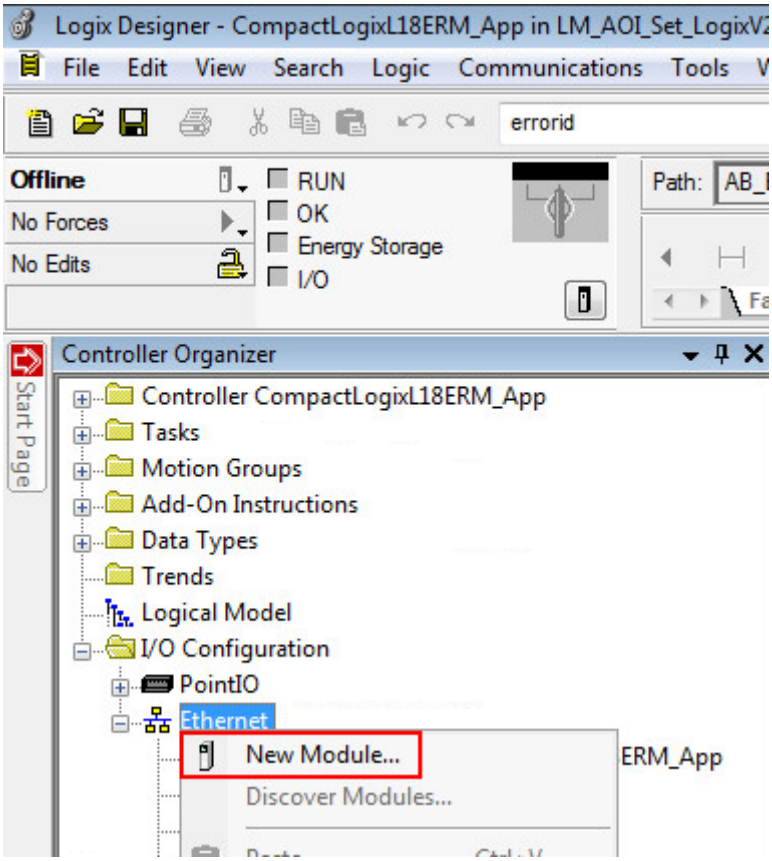
### 5.4.3 Add a LinMot drive using the EDS file (recommended for Studio <V31)

1. The EDS files can be installed using the EDS Hardware Installation Tool:

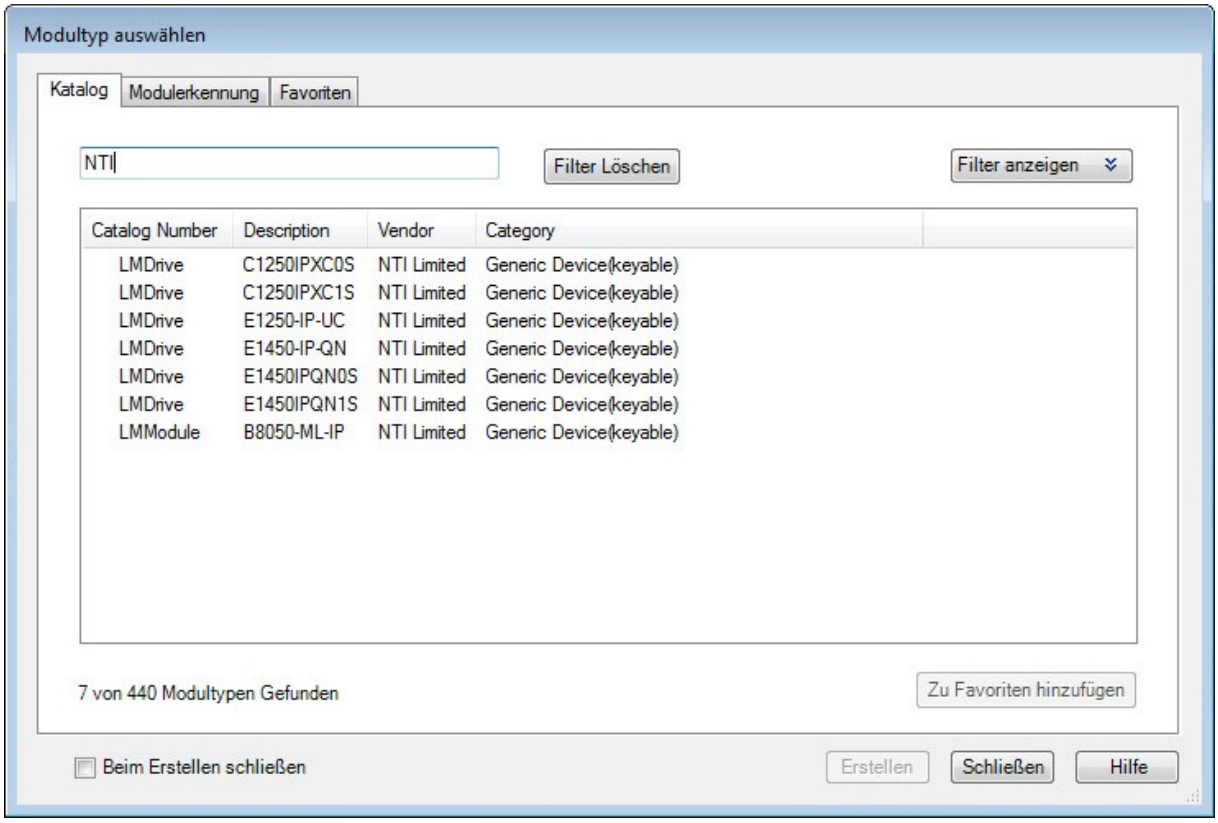



Follow the wizard and install the EDS files from the location mentioned in chapter 5.3.2.

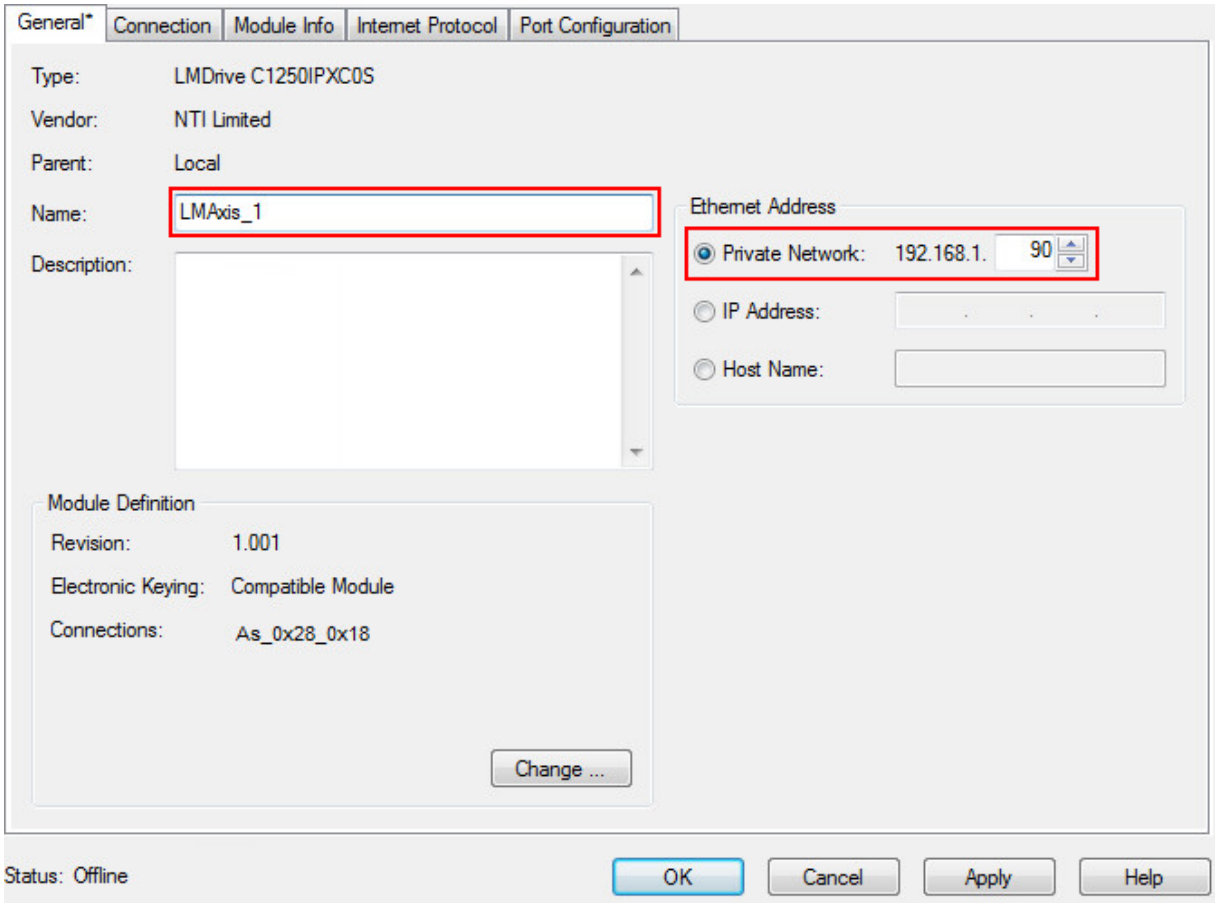
2. Add a new module by right-click on Ethernet in the I/O configuration:



3. Select Module Type:

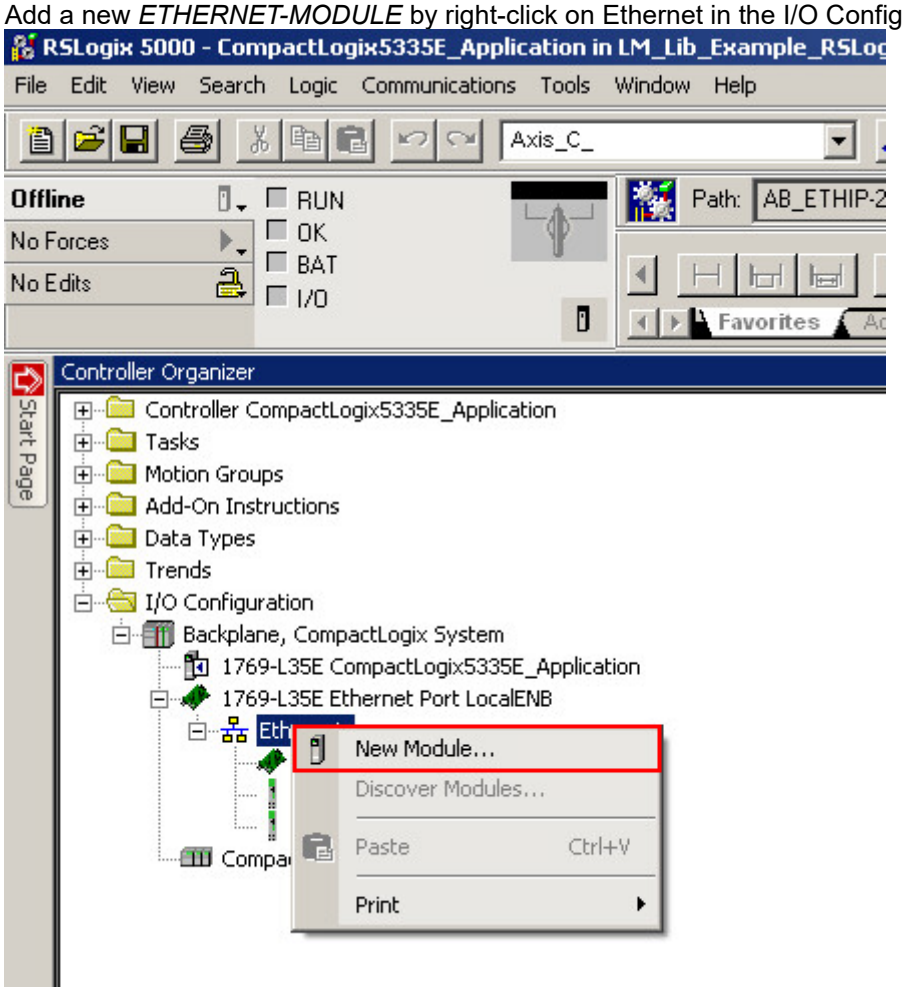


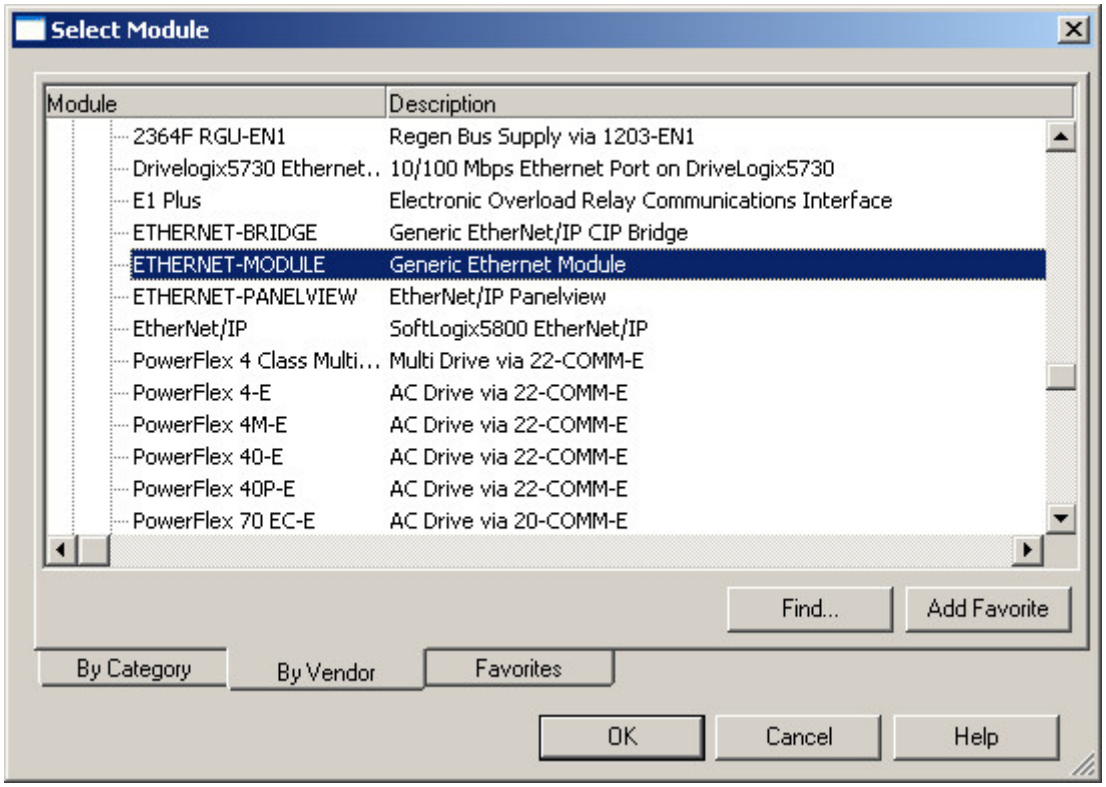
4. Set Name and IP Address



### 5.4.4 Add a LinMot drive as Generic Ethernet Module (ETHERNET-MODULE)

1. Add a new *ETHERNET-MODULE* by right-click on Ethernet in the I/O Configuration:

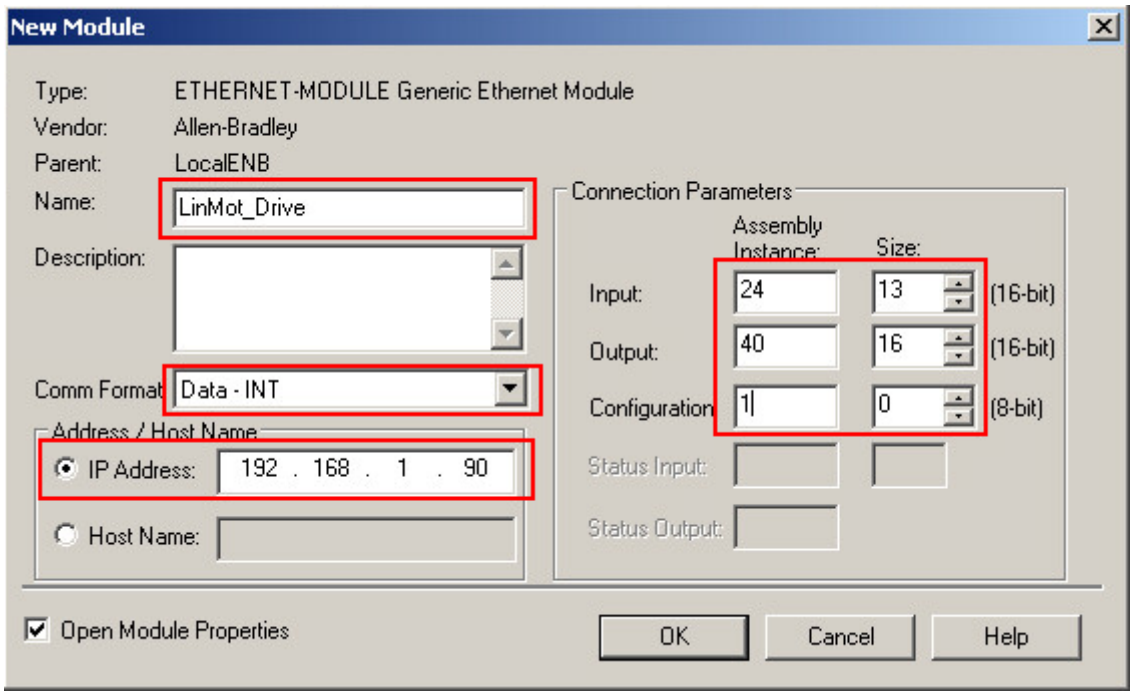


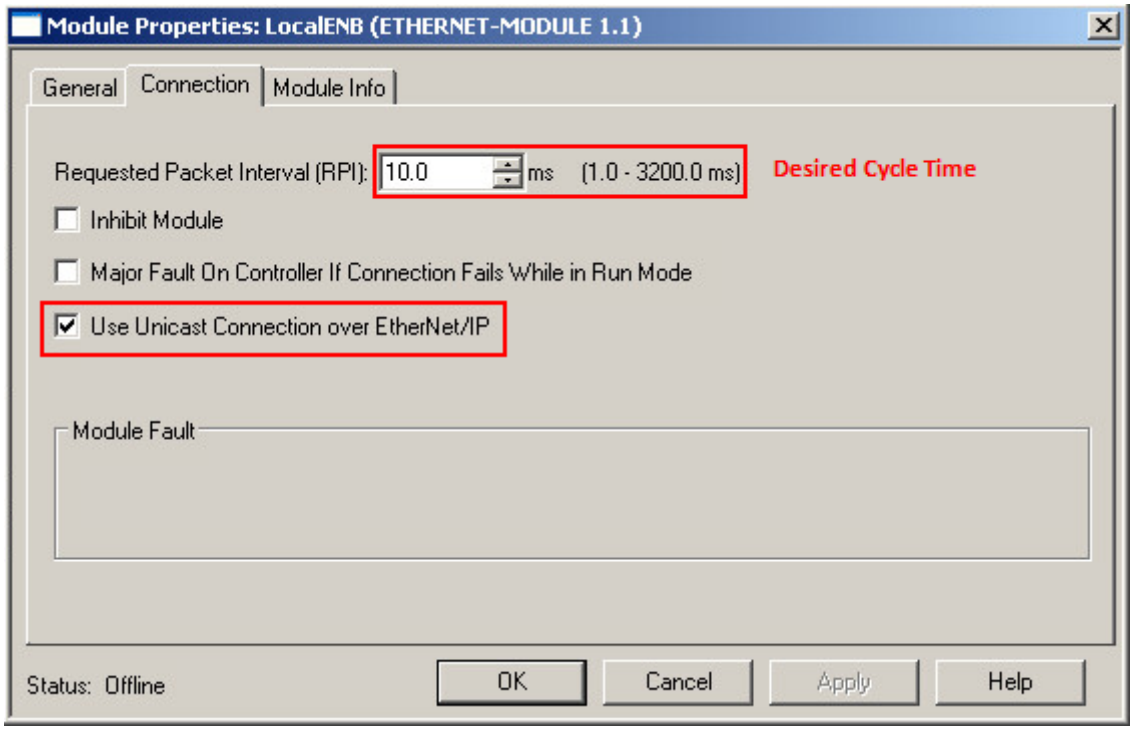



2. Setup of the Module Properties:
   - Name
   - Comm Format
   - IP Address
   - Input and output assembly size

**Attention:**
Be careful when defining these parameters, because only a correct setting will run in the EtherNet/IP network. Only the name and the RPI (Requested Packet Interval) can be defined freely.

The IP must be set according to the IP settings of the LinMot drive or vice-versa.

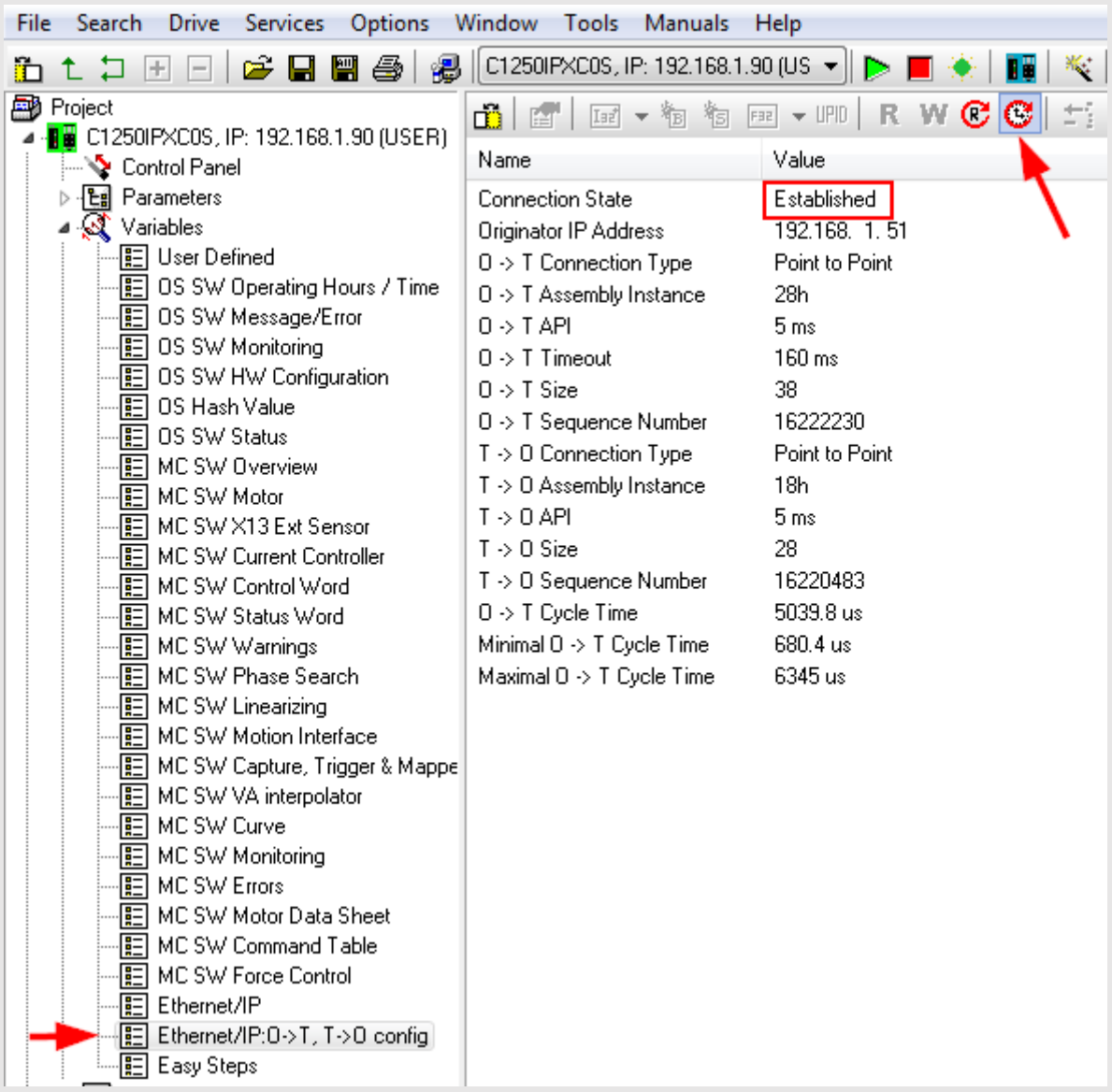


## 5.5 Check with LinMot-Talk if the Fieldbus is running

LinMot-Talk shows the status of the fieldbus. Open the variables *Ethernet/IP* for actual IP address, Net Mask or *Ethernet/IP:O->T, T->O config* to check the connection.

Check fieldbus status:




## 5.6 Next Steps

Now you can start implementing your application using the function blocks and documentation from the download link in chapter 5.1 Overview.

# 6 LinMot Profile: B&R Automation Studio

## 6.1 Overview

LinMot drives with *Powerlink* interface (e.g., C1250-PL-XC-1S) can be integrated in a B&R environment as point-to-point axis using the LinMot function block library.

**Download:**
Demo projects can be downloaded from:
http://download.linmot.com/plc_lib/libraries/BR/ (named *BR_AS4x_LinMot_Powerlink_noUpgrades*...)

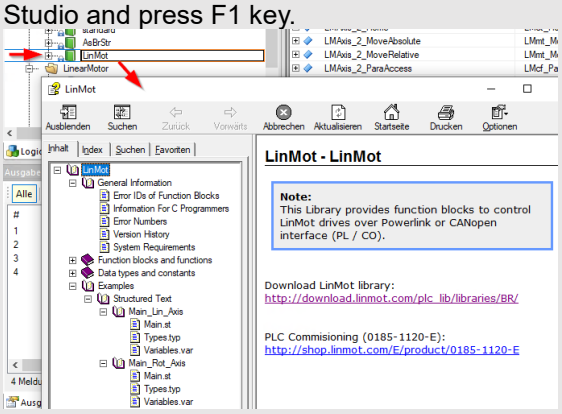
The library itself can be download from:
http://download.linmot.com/plc_lib/libraries/BR/ (named *LM_BR_Library_inclSources_*...)

> Choose the library …inclSources… to compile it on your system for your specific AS and AR versions.

For a detailed description of the library function blocks please check the Beckhoff library documentation which is part of the download in **chapter 3.1 (page 21).**

Both the Beckhoff and B&R library contain the same function blocks and are therefore identical in documentation.

The library itself contains a HTML Help that can be opened by selecting the library in Automation Studio and press F1 key.



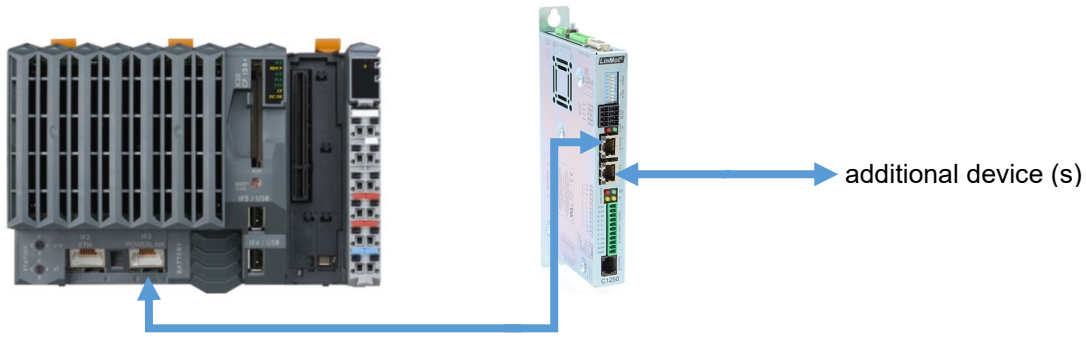




Image Source: http://www.br-automation.com/

For further information on Ethernet POWERLINK please visit: http://www.ethernet-powerlink.org

## 6.2 Configuration and Connection of the LinMot Drive

The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 6.2.1 Motor configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


See Appendix I: Basic Position Control Loop Tuning

### 6.2.2 XDD File

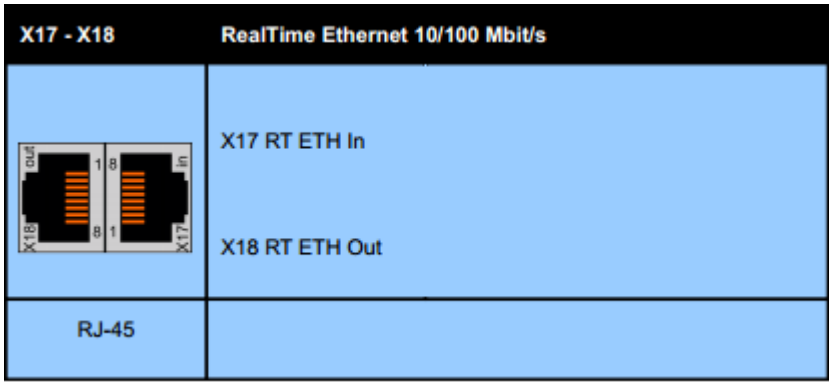
Install the XDD file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software.
It is located by default:

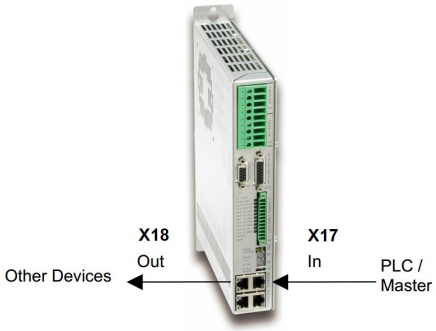
- Powerlink: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink\XDD
- Powerlink: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink_Nx\XDD (-MI drives)

There are separate XDD files available for either AS3 or AS4.

### 6.2.3 Powerlink Connection

The drive is connected to the Powerlink network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-PL-UC drive. On all other LinMot drives supporting Powerlink the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.



### 6.2.4 Node ID

The Node ID of the LinMot drive is set using the hardware address selectors.

| S1 - S2 | Address Selectors | | |
|---|---|---|---|
| E1100<br>E1200 V1 | E1200 V2<br>E1400<br>C1x00 | | |
|  |  | S1 (5 .. 8) | Bus ID High (0 ... F). Bit 5 is LSB, bit 8 MSB. |
| | | S2 (1 .. 4) | Bus ID High (0 ... F). Bit 1 is LSB, bit 4 MSB. |

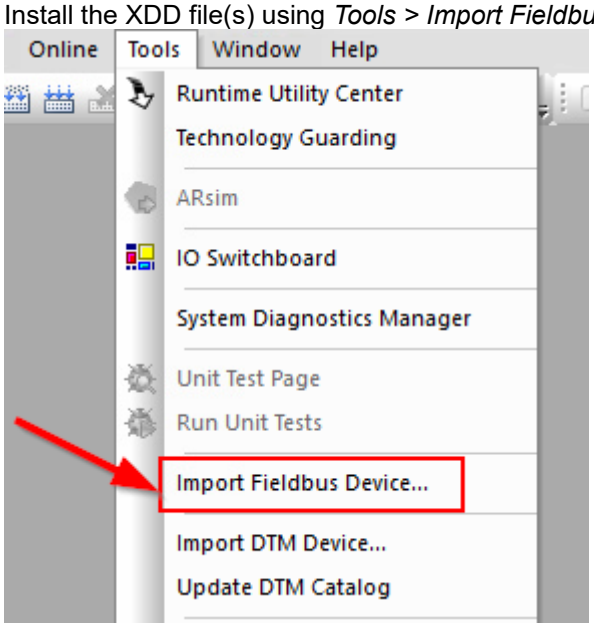
E.g.: Node ID 18 dec = 12 hex = 0001 0010 bin

=> S1 = 1 hex = 0001 bin (Dip Switch 5 = ON), S2 = 2 hex = 0010 bin (Dip Switch 2 = ON)

## 6.3 Hardware Configuration

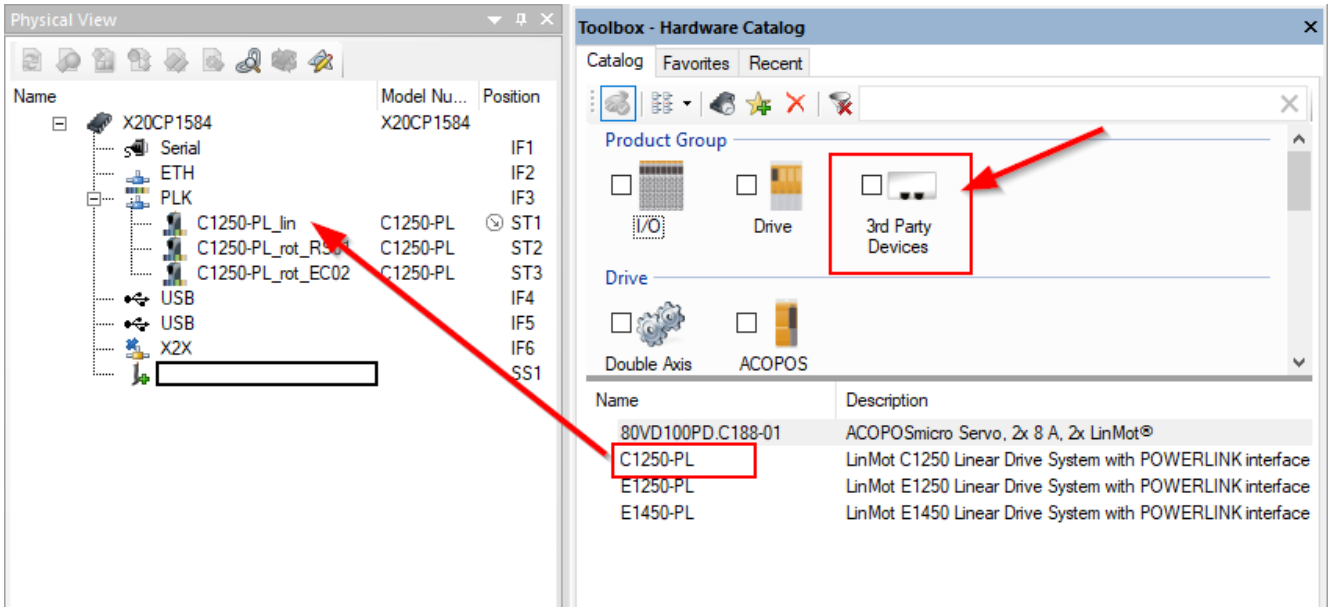
### 6.3.1 Add the LinMot drive to the Powerlink network

Install the XDD file(s) using *Tools* > *Import Fieldbus Device…* in Automation Studio.




The required XDD-file can be found in the following folder (default):
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\POWERLINK\XDD

In the Hardware Catalog search for the LinMot drive (e.g., C1250...) and drag&drop it onto the Powerlink interface (PLK).
Alternatively select the Product Group *3rd Party Devices* > all installed LinMot drives are shown.

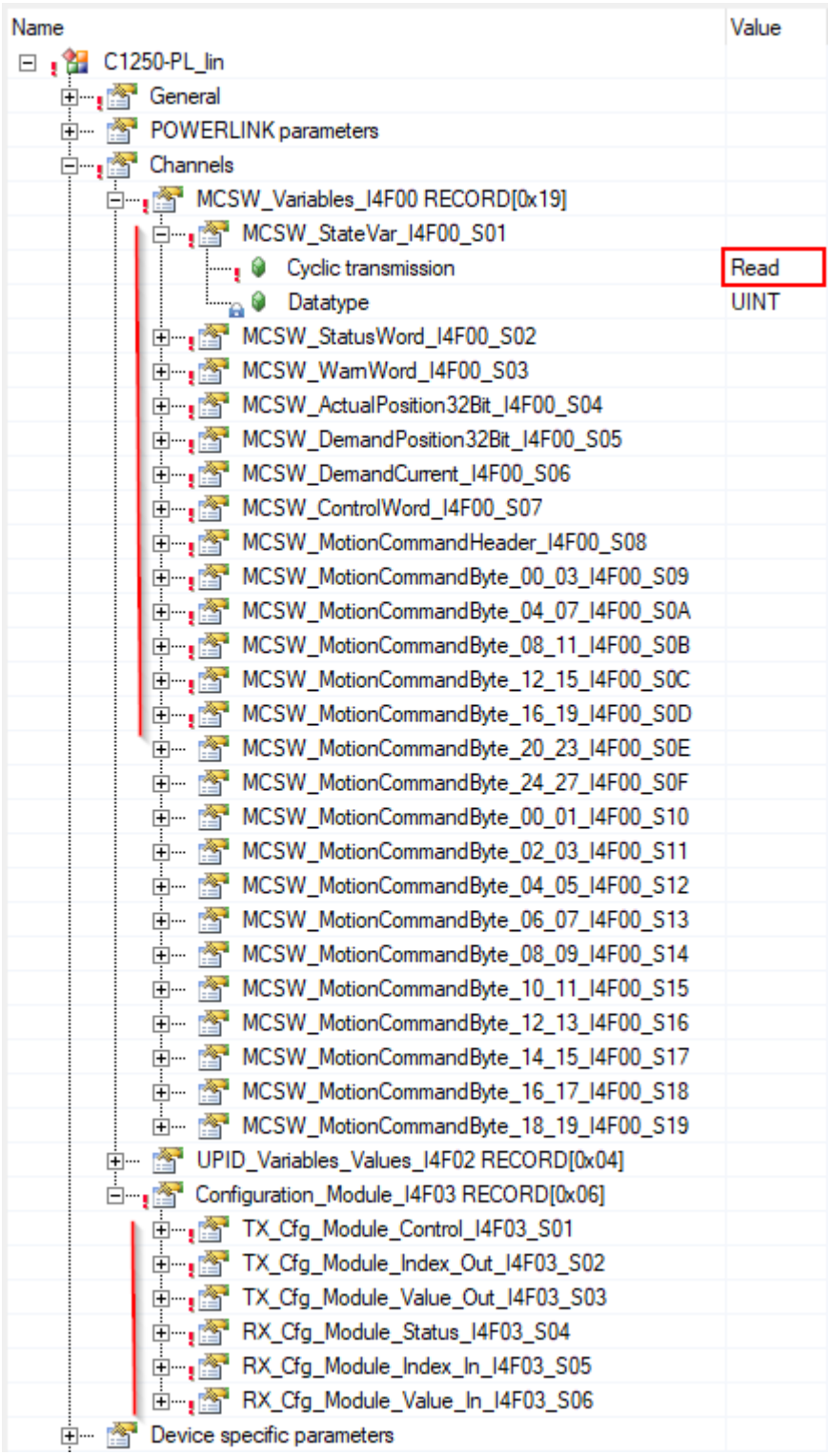



Set the Node ID according to your requirements.

### 6.3.2 Configuration

In the configuration of the LinMot slave module (right-click on device > *Configuration*) the following channels must be selected:

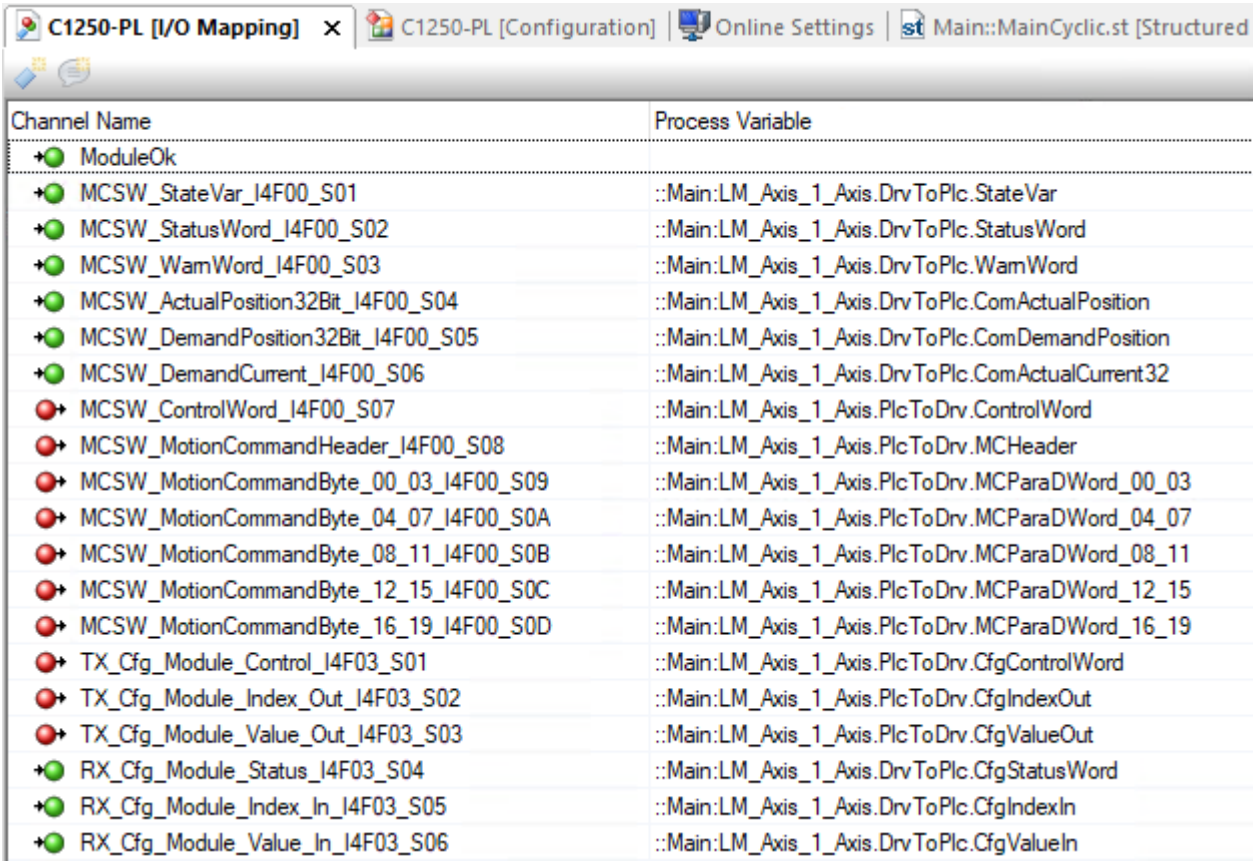
Set the **Cyclic transmission** of all channels marked with a red exclamation mark to **Read** or to **Write** respectively.



### 6.3.3 I/O Mapping

The inputs and outputs (right-click open I/O mapping) of the modules are mapped to the axis structure (datatype stsLM_Axis).

In the following example a variable *LM_Axis_1_Axis* of datatype *tstLM_Axis* is defined in the *Main* program.




| Channel Name | Process Variable |
|---|---|
| ModuleOk | |
| MCSW_StateVar_I4F00_S01 | ::Main:LM_Axis_1_Axis.DrvToPlc.StateVar |
| MCSW_StatusWord_I4F00_S02 | ::Main:LM_Axis_1_Axis.DrvToPlc.StatusWord |
| MCSW_WarnWord_I4F00_S03 | ::Main:LM_Axis_1_Axis.DrvToPlc.WarnWord |
| MCSW_ActualPosition32Bit_I4F00_S04 | ::Main:LM_Axis_1_Axis.DrvToPlc.ComActualPosition |
| MCSW_DemandPosition32Bit_I4F00_S05 | ::Main:LM_Axis_1_Axis.DrvToPlc.ComDemandPosition |
| MCSW_DemandCurrent_I4F00_S06 | ::Main:LM_Axis_1_Axis.DrvToPlc.ComActualCurrent32 |
| MCSW_ControlWord_I4F00_S07 | ::Main:LM_Axis_1_Axis.PlcToDrv.ControlWord |
| MCSW_MotionCommandHeader_I4F00_S08 | ::Main:LM_Axis_1_Axis.PlcToDrv.MCHeader |
| MCSW_MotionCommandByte_00_03_I4F00_S09 | ::Main:LM_Axis_1_Axis.PlcToDrv.MCParaDWord_00_03 |
| MCSW_MotionCommandByte_04_07_I4F00_S0A | ::Main:LM_Axis_1_Axis.PlcToDrv.MCParaDWord_04_07 |
| MCSW_MotionCommandByte_08_11_I4F00_S0B | ::Main:LM_Axis_1_Axis.PlcToDrv.MCParaDWord_08_11 |
| MCSW_MotionCommandByte_12_15_I4F00_S0C | ::Main:LM_Axis_1_Axis.PlcToDrv.MCParaDWord_12_15 |
| MCSW_MotionCommandByte_16_19_I4F00_S0D | ::Main:LM_Axis_1_Axis.PlcToDrv.MCParaDWord_16_19 |
| TX_Cfg_Module_Control_I4F03_S01 | ::Main:LM_Axis_1_Axis.PlcToDrv.CfgControlWord |
| TX_Cfg_Module_Index_Out_I4F03_S02 | ::Main:LM_Axis_1_Axis.PlcToDrv.CfgIndexOut |
| TX_Cfg_Module_Value_Out_I4F03_S03 | ::Main:LM_Axis_1_Axis.PlcToDrv.CfgValueOut |
| RX_Cfg_Module_Status_I4F03_S04 | ::Main:LM_Axis_1_Axis.DrvToPlc.CfgStatusWord |
| RX_Cfg_Module_Index_In_I4F03_S05 | ::Main:LM_Axis_1_Axis.DrvToPlc.CfgIndexIn |
| RX_Cfg_Module_Value_In_I4F03_S06 | ::Main:LM_Axis_1_Axis.DrvToPlc.CfgValueIn |

## 6.4 Parameter Access

For detailed information about accessing drive parameters please check chapter **16.12 Parameter Access (page 233)**.

## 6.5 Add Additional Parameters or Variables to the Process Data

To add additional parameters or variables for read and write access to the process data please see chapter **16.13 Add Additional Parameters or Variables to the Process Data (page 236)**.

## 6.6 Next Steps

Now you can start implementing your application using the function blocks and documentation from the download link(s) in chapter **6.1 Overview (page 53)**.

# 7 LinMot Profile: CODESYS

## 7.1 Overview

A LinMot drive with *EtherCAT (-EC)* interface (e.g., C1250-EC-XC-1S) can be integrated and setup in a CODESYS environment as I/O device using the LinMot function block library

**Download:**
A package including the library and example projects is available from:
http://download.linmot.com/plc_lib/libraries/CODESYS/3x/ (named *CoDeSys_3.x_LinMot_Library_...*)

For a detailed description of the library function blocks please check the Beckhoff library documentation which can be downloaded in **chapter 3.1 (page 21)**

Both the Beckhoff and CODESYS library contain the same function blocks and are therefore identical in documentation.

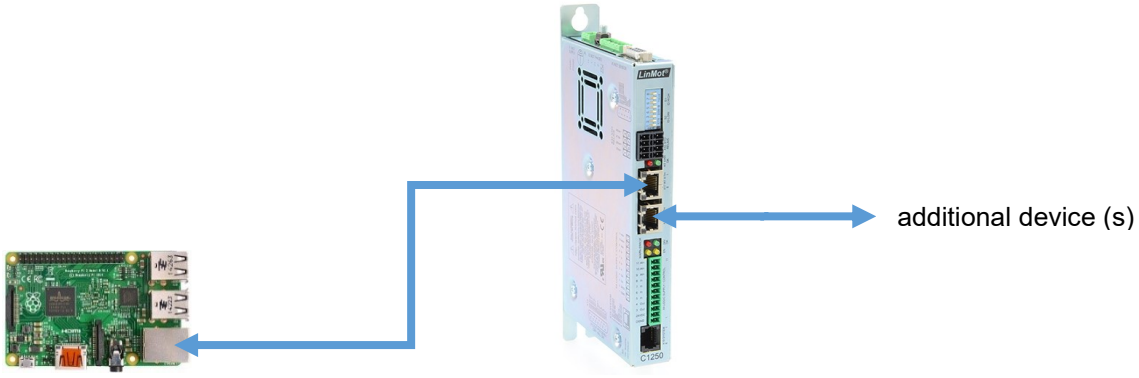



Image Source: http://www.raspberrypi.org/

EtherCAT is the real-time Ethernet network originally developed by Beckhoff. The LinMot acts as Slave in this network and is implemented with the standard ASIC ET1100 from Beckhoff.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

# 8 LinMot Profile: Mitsubishi GX Works

## 8.1 Overview

This chapter shows how a LinMot drive with CC-Link IE Field Basic Interface (LinMot Profile) can be connected to a Mitsubishi PLC using GX Works3.

**Download:**
The libraries for GX Works3 including documentation can be downloaded from:
http://download.linmot.com/plc_lib/libraries/Mitsubishi/

**Note:**
More information can be found in the library documentation (part of the above-mentioned download) and in the user manual CC-Link IE Field Basic Interface (see chapter Documentation / User Manuals).

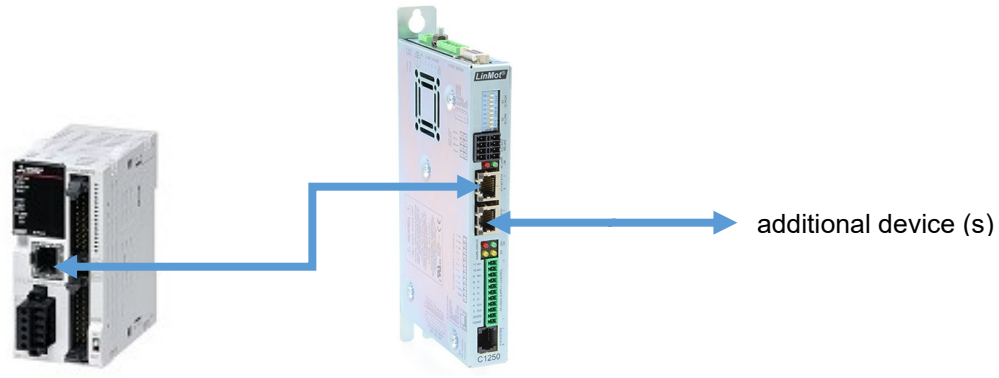



Image Source: https://www.mitsubishielectric.com

CC-Link IE Field Basic  is an Industrial Ethernet Standard. The LinMot CC-Link IE Field Basic drives act as CC-Link IE Field Basic slaves in this network.

For further information on CC-Link IE Field Basic please visit:
https://www.cc-link.org/en/index.html

## 8.2 Configuration and Connection of the LinMot Drive

The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 8.2.1 Motor configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


See Appendix I: Basic Position Control Loop Tuning

### 8.2.2 CSP+ files

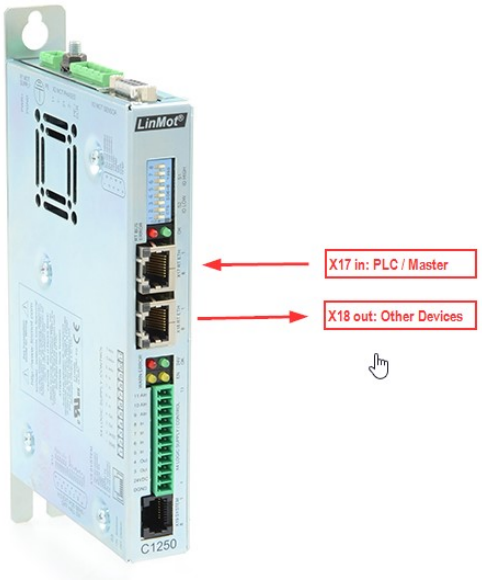
Install the CSP+ file(s) that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\CCLinkIEFB\CSPP

### 8.2.3 CC Link Connection

The drive is connected to the CC Link network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an C1250-CC drive.

| X17 – X18 | ETHERCAT Connector | | |
|---|---|---|---|
| | Pin | Wire color code | Assignment 100BASE-TX |
|  | 1 | WHT/ORG | Rx+ |
| | 2 | ORG | Rx- |
| | 3 | WHT/GRN | Tx+ |
| | 4 | BLU | - |
| | 5 | WHT/BLU | - |
| | 6 | GRN | Tx- |
| | 7 | WHT/BRN | - |
| | 8 | BRN | - |
| | case | - | - |
| RJ-45 | Use standard patch cables (twisted pair, S/UTP, AWG26) for wiring. This type of cable is usually referred to as a “Cat5e-Cable”. | | |

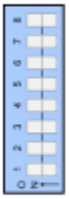


### 8.2.4 IP Address

The default IP address is 192.168.3.**xxx**, where the last byte xxx is defined via the two address selectors *S1 & S2*. *S1* sets the high and *S2* the low digit. E.g., S1 = 5, S2 = A -> 5A (hex) = 90 (dec) -> IP = 192.168.3.**90**

For further information about the IP address settings please check the interface manual > 0185-1171

Set IP address (last byte, **xxx** from above) by S1 & S2

| | NodeID Selectors | |
|---|---|---|
| C12x0 | | |
|  | S1 (5..8) | Station-ID High Nibble (0 ... F). Bit 5 is the LSB, bit 8 the MSB. |
| | S2 (1..4) | Station-ID Low Nibble (0 ... F). Bit 1 is the LSB, bit 4 the MSB.<br><br>**Setting the Station-ID high & low to 0xFF initiates resetting the drive to manufacturer settings!** |

## 8.3 PLC Setup CC Link

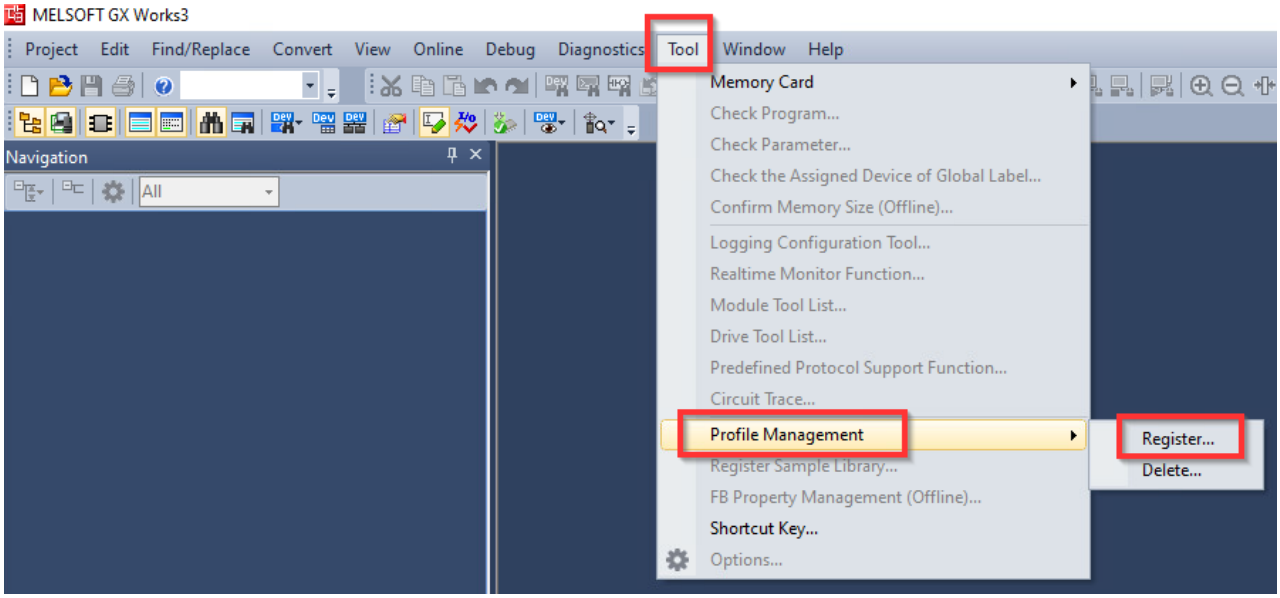
The following procedure can be used for all Mitsubishi PLCs compatible with GX Works 3.

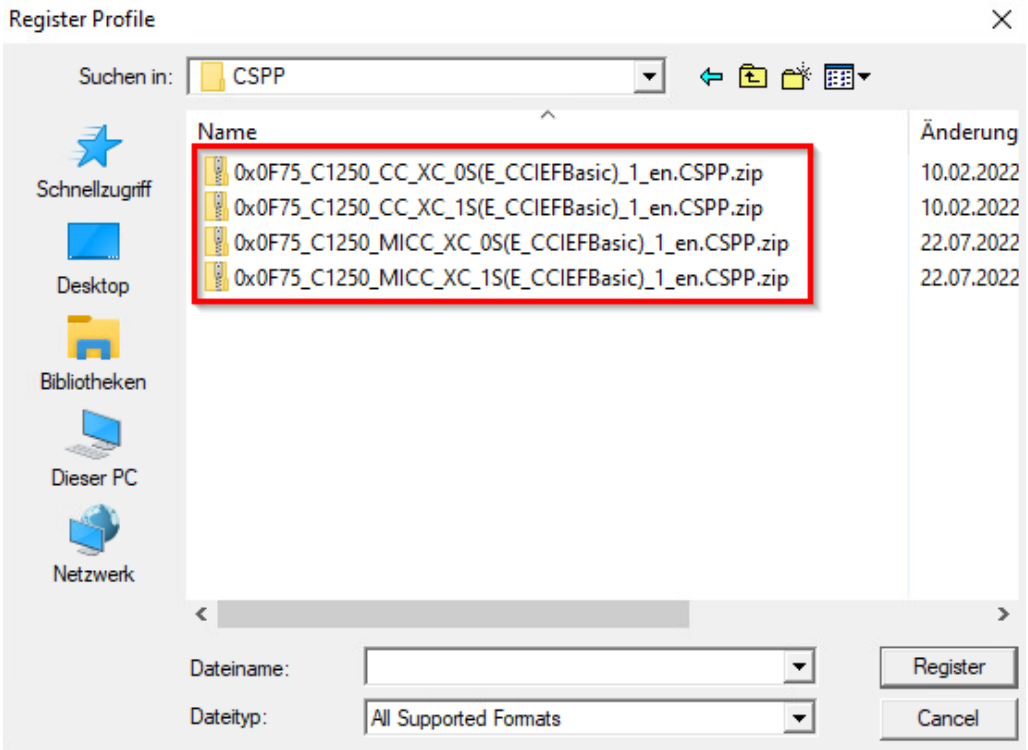
First, install the CSP+ file for the drive.
(GX Works 3 → Tool → Profile Management → Register

The required CSP+ file can be found by default in the following folder:

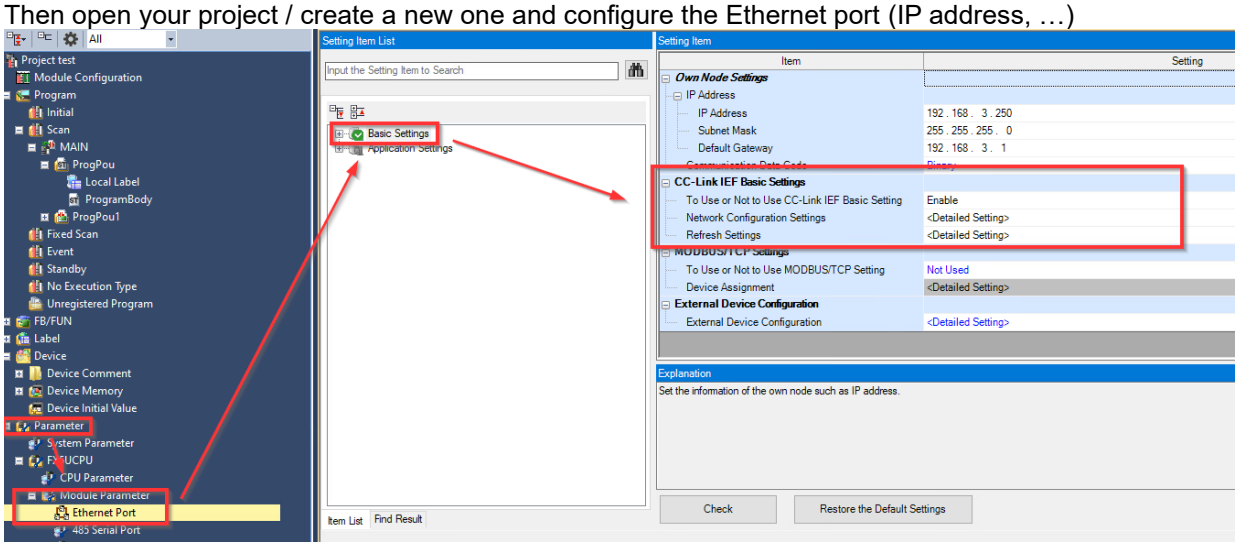
*C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\CCLinkIEFB\CSPP*

1. First the corresponding CSPP file of the LinMot Drive must be installed. No project must be open when doing so!

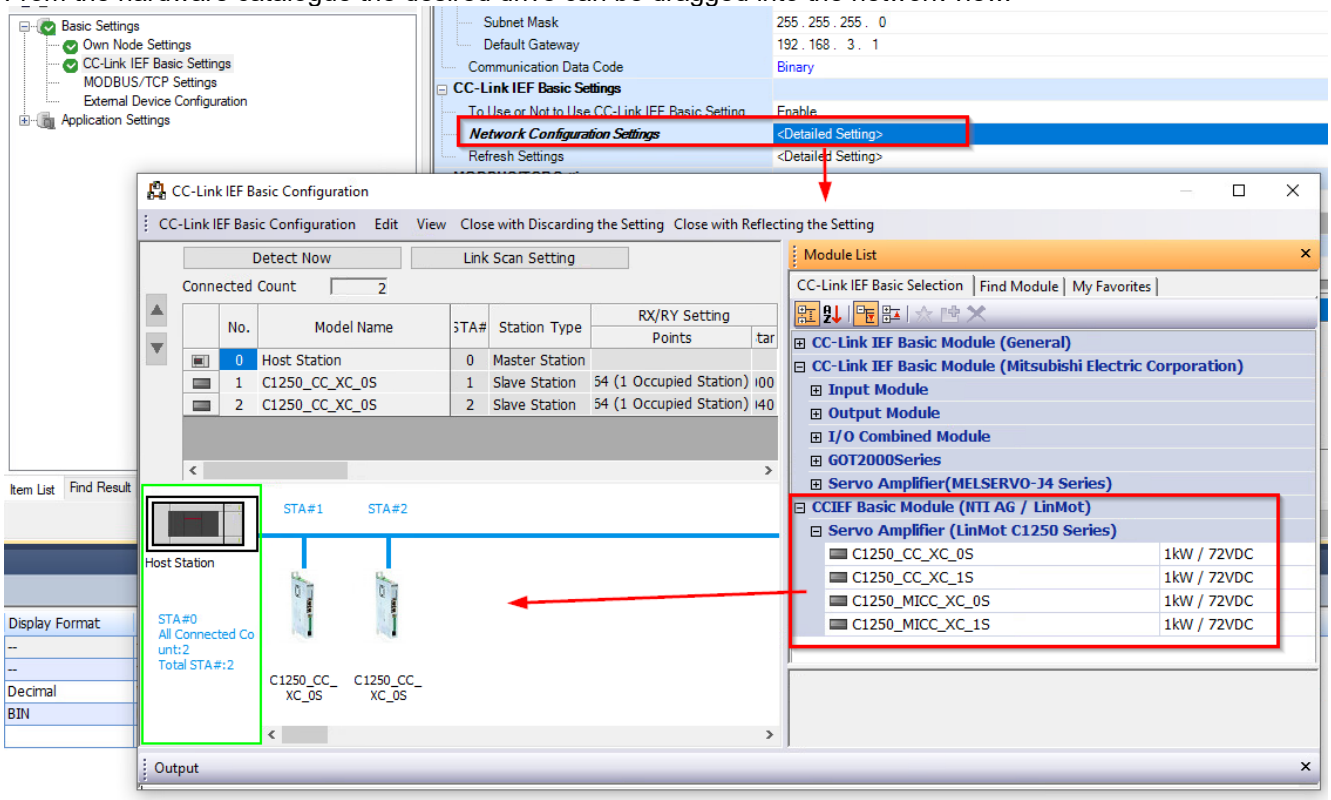


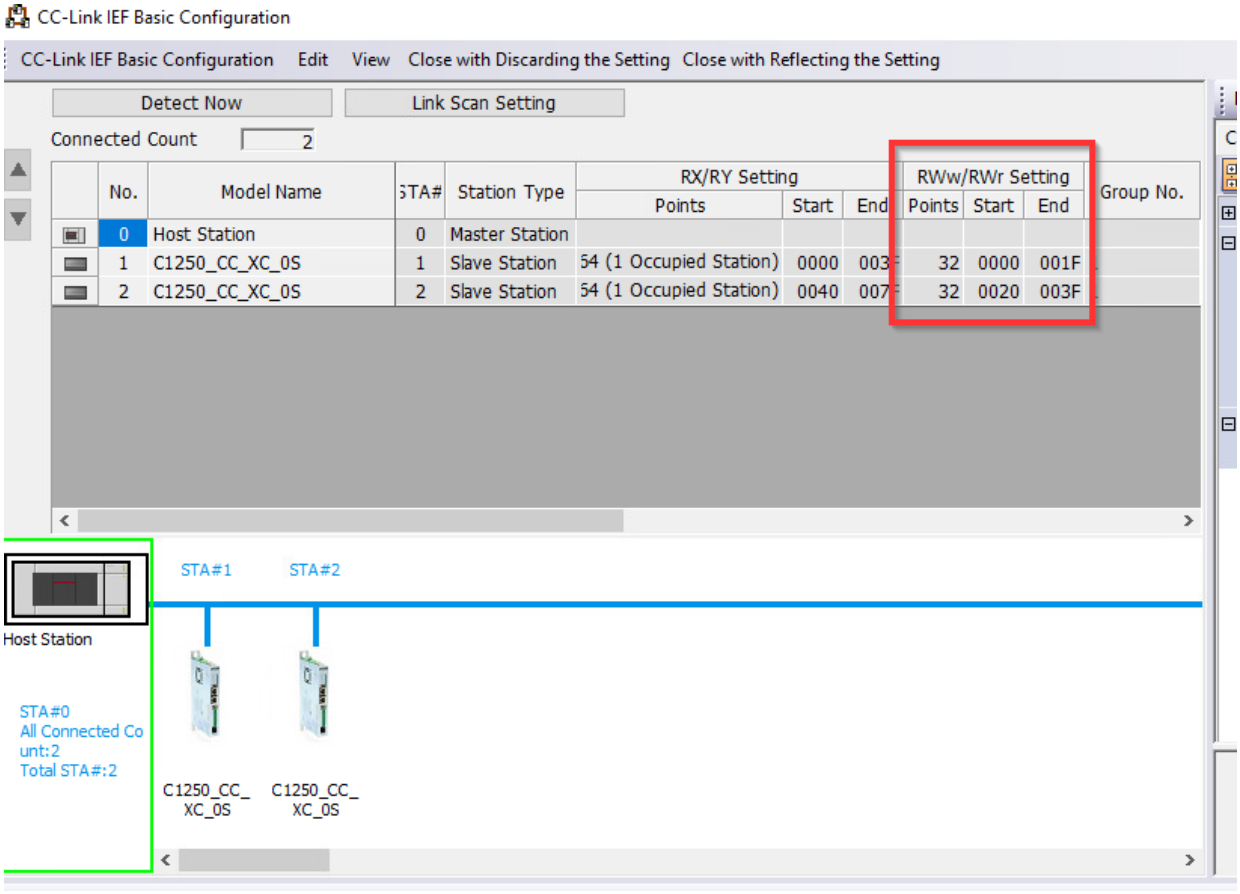
2. Then open your project / create a new one and configure the Ethernet port (IP address, …)




3. From the hardware catalogue the desired drive can be dragged into the network view.



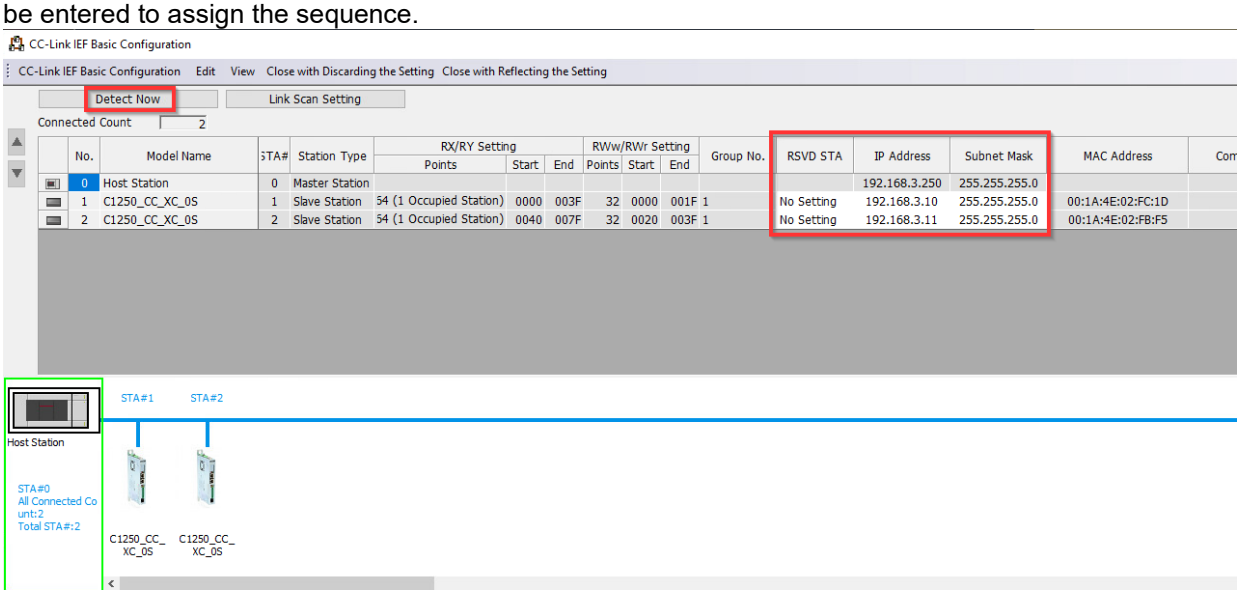
4. Afterwards the I/O data areas can be read.




**Note:**
Input registers W are counted hexadecimal, output registers R decimal!

Here: Read Drive 1: W0
Read Drive 2: W20
Write Drive 1: R0
Write Drive 2: R32 (1F: 31, 20: 32)

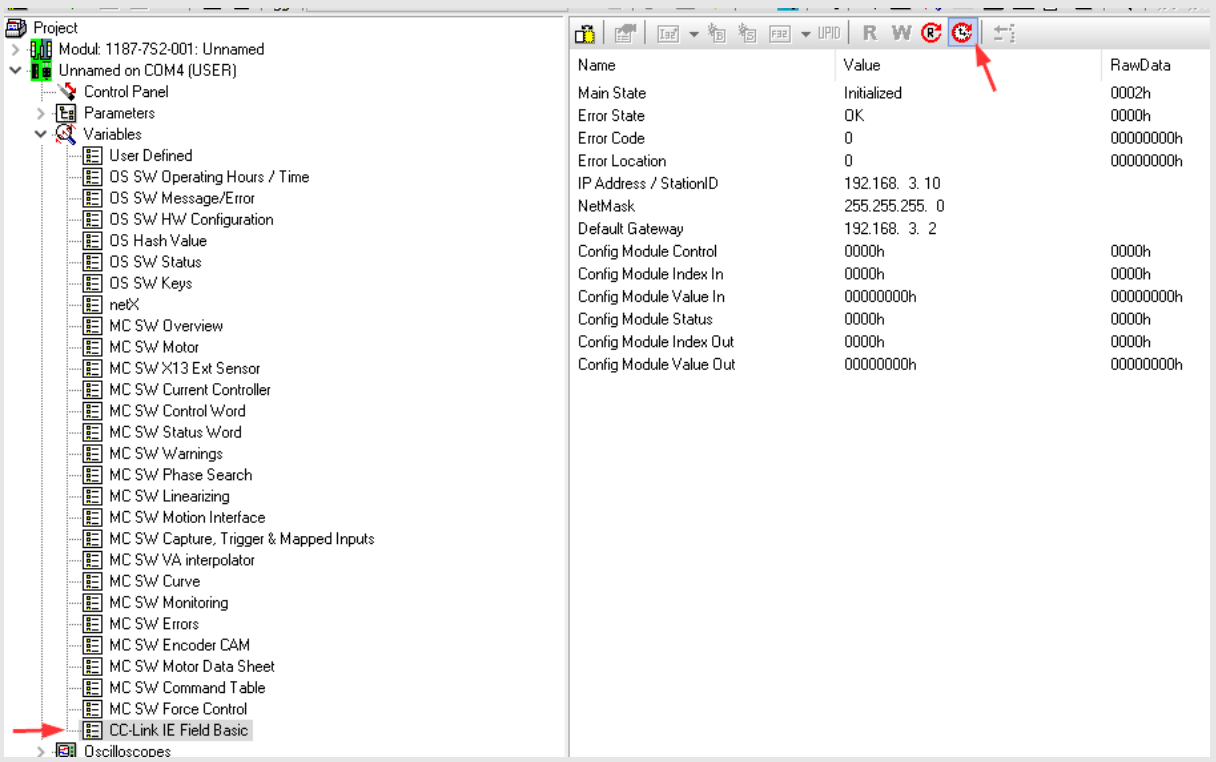
5. On the control side, the participants can then be recognized in GX3 Works via "Detect Now". The IP's can be entered to assign the sequence.



## 8.4 Check with LinMot-Talk if the Fieldbus is running

LinMot-Talk shows the status of the fieldbus. Open the variables CC-Link IF Field Basic and check if the IP Address is correct.

Check fieldbus status:




## 8.5 Next Steps

Now you can start implementing your application using the function blocks and documentation from the download link in chapter **8.1 Overview.**

# 9 Drive Profile: Rockwell Automation Motion (CIP Sync)

## 9.1 Overview

LinMot drives with *LinMot EtherNet/IP CIP Sync* interface (e.g., C1250-CM-XC-1S or C1250-MI-XC-1S) can be integrated and setup in a Rockwell Automation environment as motion axis.

**Download:**
A demo project including EDS files and a detailed documentation can be found here:
http://download.linmot.com/plc_lib/examples/Rockwell_CM/ (CIPSync_Demo...)

**Note:**
More information and step by step guides can be found in the project documentation (part of the above-mentioned download) and in the user manual Ethernet/IP (0185-1165, see chapter 1.4 Documentation / User Manuals)

**EDS Files:**
EDS files for **-CM** and **-MI** drives located by default :
\\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherNetIP_NX\EDS


Image Source: http://www.rockwellautomation.com/

EtherNet/IP is an industrial Ethernet network that implements the Common Industrial Protocol (CIP).

For further information on EtherNet/IP please visit:
http://www.odva.org

**YouTube video series** (LinMot Integration – Rockwell Automation CIP Sync):
https://youtube.com/playlist?list=PLMSLCScLnf95PmjyZv8sl6zuHKhci8uln

**Attention:** The following minimum requirements must be met!

Rockwell Automation controller revision **must be V30 or later**

LinMot drive firmware **must be 6.8 Build 20190315 or later**

# 10 Drive Profile: Siemens TIA, PROFIdrive

## 10.1 Overview

This chapter shows how a LinMot drive with *PROFIdrive* interface (e.g., a C1250-PD-XC-1S) can be integrated and setup in SIEMENS TIA Portal using SIEMENS Telegram 105. For this example, TIA 15 and a S7-1517F-3 PN/DP is used.

**Download:**
An example project can be downloaded from:
http://download.linmot.com/plc_lib/examples/Siemens_PD/

Components used:

- LinMot C1250-PD-XC-1S-000 (article number 0150-2619) with firmware 6.10 Build 20210521
- CPU 1517F-3 PN/DP (article number 6ES7 517-3FP00-0AB0) with firmware V 2.6.1
- TIA Portal V15.1 Update 4


Image Source: https://mall.industry.siemens.com

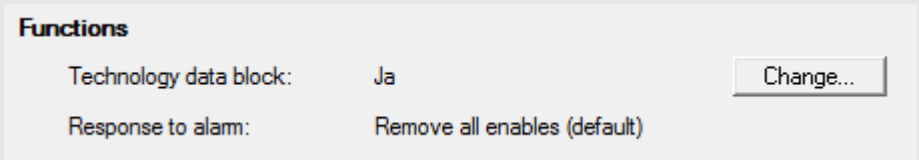
PROFINET is an open real-time Ethernet network
The LinMot drive acts as slave in this network and is implemented with the TPS1 chip from Renesas.

For further information on PROFINET and the PROFIdrive fieldbus protocols please visit:
http://www.profibus.com/

**YouTube video series** (LinMot Integration – Siemens PROFIdrive):
https://youtube.com/playlist?list=PLMSLCScLnf97_39VIYddSsIm57rr-N0su

**Attention:**
SIEMENS Telegram 105 requires firmware 2.0 and higher installed in the S7-1500 CPU. Some older S7-1500 hardware revisions do not support firmware 2.0. In that case use Standard Telegram 5.

**Note:**
From firmware 6.6 Build 20170522 the LinMot drive supports the SIEMENS Technology data block:



## 10.2 Configuration and Connection of the LinMot Drive

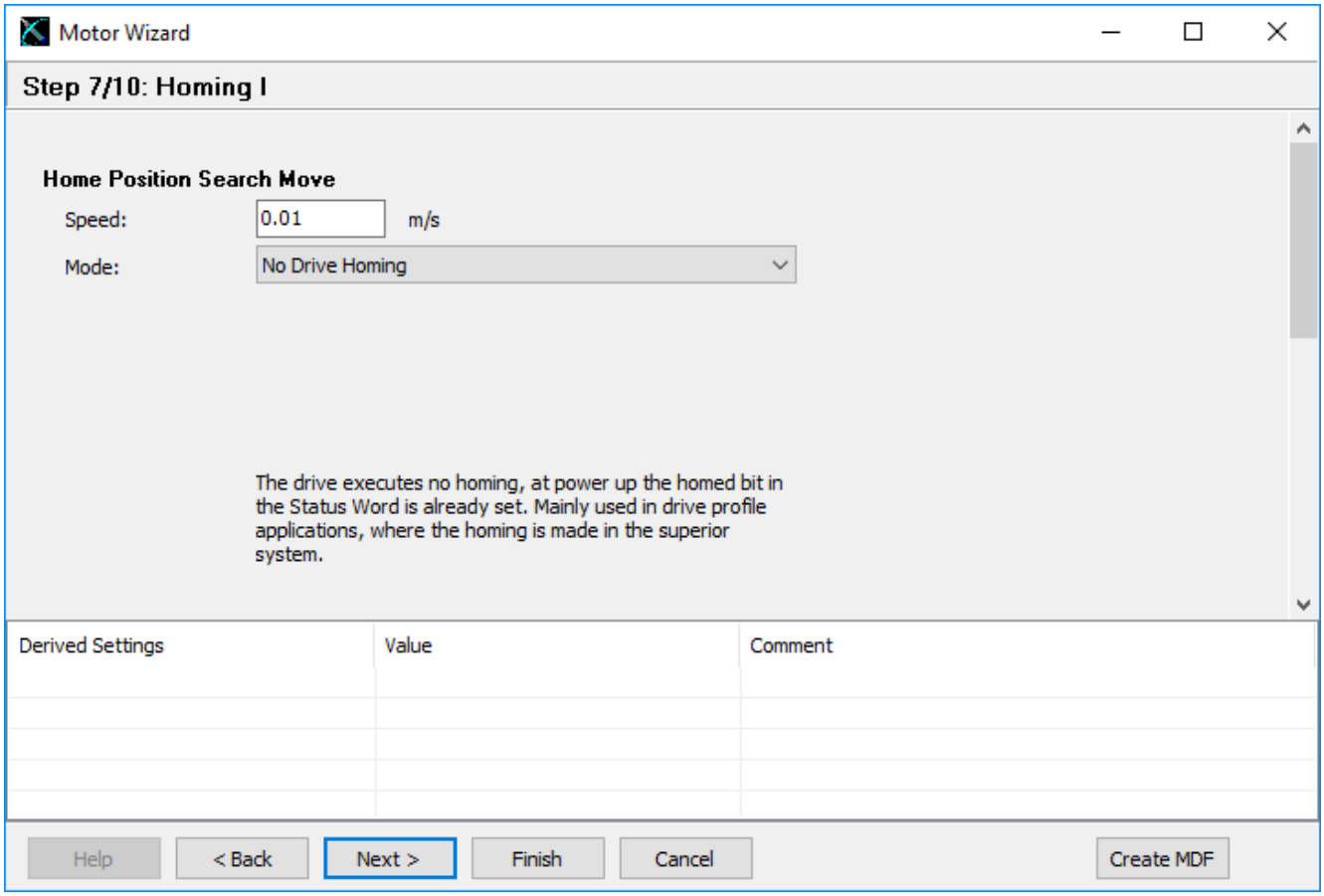
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 10.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select “No Drive Homing” in the Motor Wizard as with SIEMENS Telegram 105 the homing of the axis is done in the PLC.




See Appendix I: Basic Position Control Loop Tuning

### 10.2.2 GSDML File

Install the GSDML file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- PROFIdrive: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\ProfiNet\GSDML_PD
- PROFIdrive: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\ProfiNet_Nx\GSDML_PD (-MI drives)

### 10.2.3 PROFINET Connection

The drive is connected to the PROFINET network using the X17 (Port1, IN) & X18 (Port 2, OUT) connectors.

| X17 – X18 | RealTime Ethernet 10/100 Mbit/s | |
|---|---|---|
|  | X17 RT ETH In | SIEMENS: Port 1, X1.X1 |
|  | X18 RT ETH Out | SIEMENS: Port 2, X1.X2 |
| RJ-45 | | |

## 10.3 PLC Setup

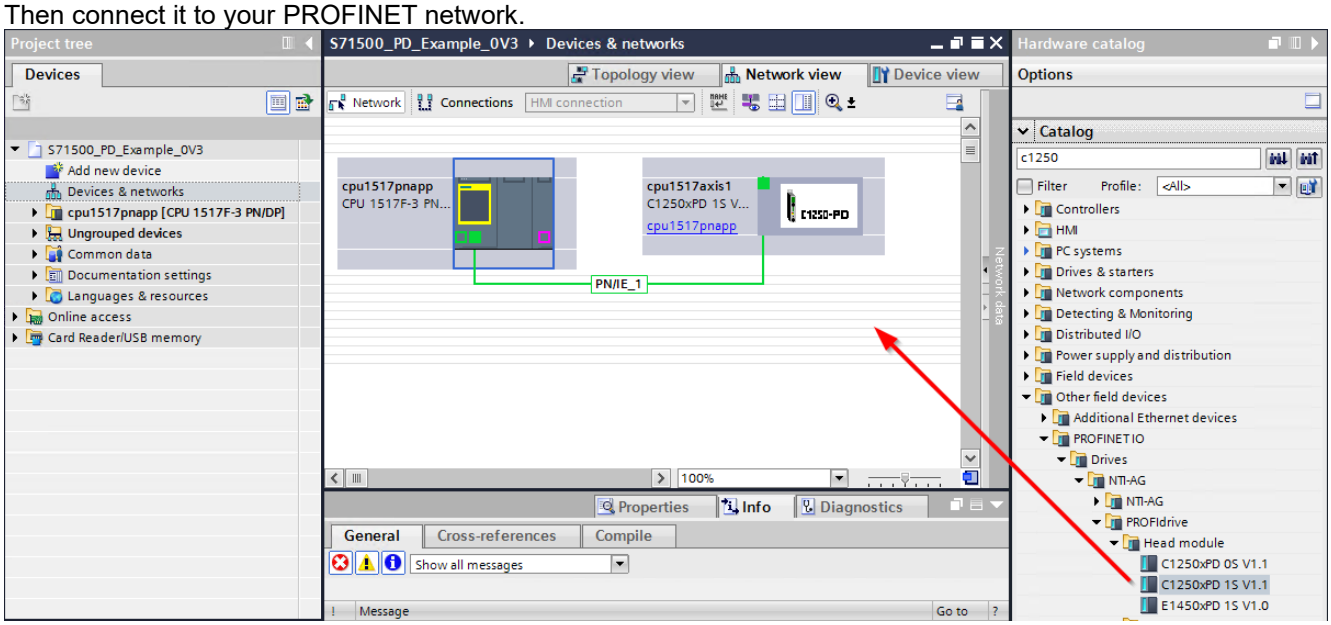
### 10.3.1 Install GSDML File(s)

The GSDML files for the drives are located by default in the LinMot-Talk installation path:
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\ProfiNet\GSDML_PD
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\ProfiNet_Nx\GSDML_PD

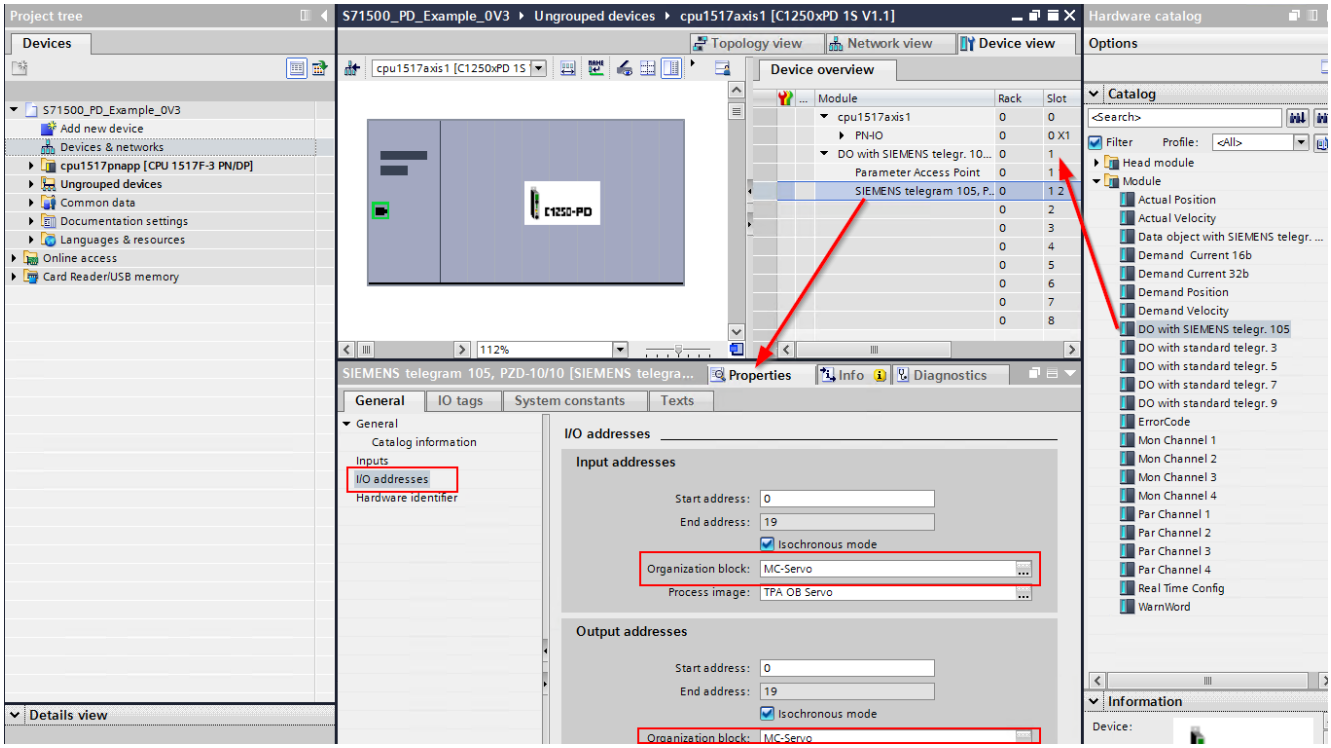
In TIA Portal select in the menu bar “Options > Manage general station description files (GSD)” and navigate to the directory you have put the GSDML file(s) in.

### 10.3.2 Insert the LinMot Drive as Device (-PD Drives)

Select the LinMot C1250-PD-XC-1S drive from the catalogue and drag&drop it to the network view.
Then connect it to your PROFINET network.



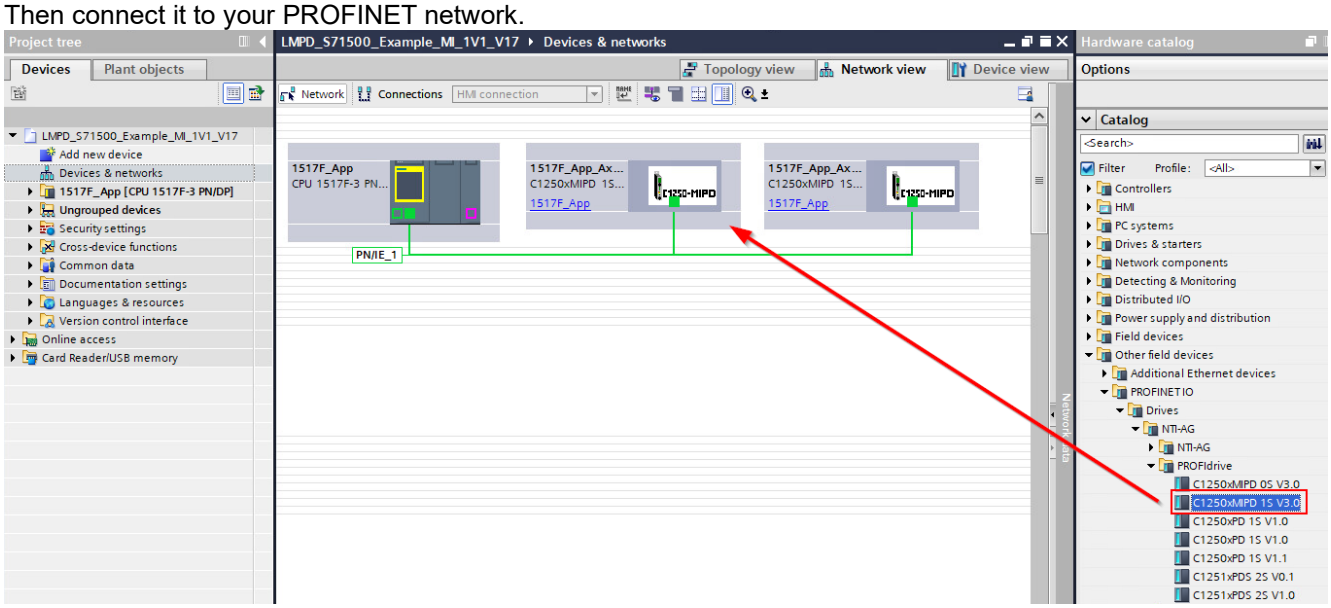

Add the SIEMENS Telegram 105 module in the device view to slot 1 and select the MC-Servo OB under I/O addresses:



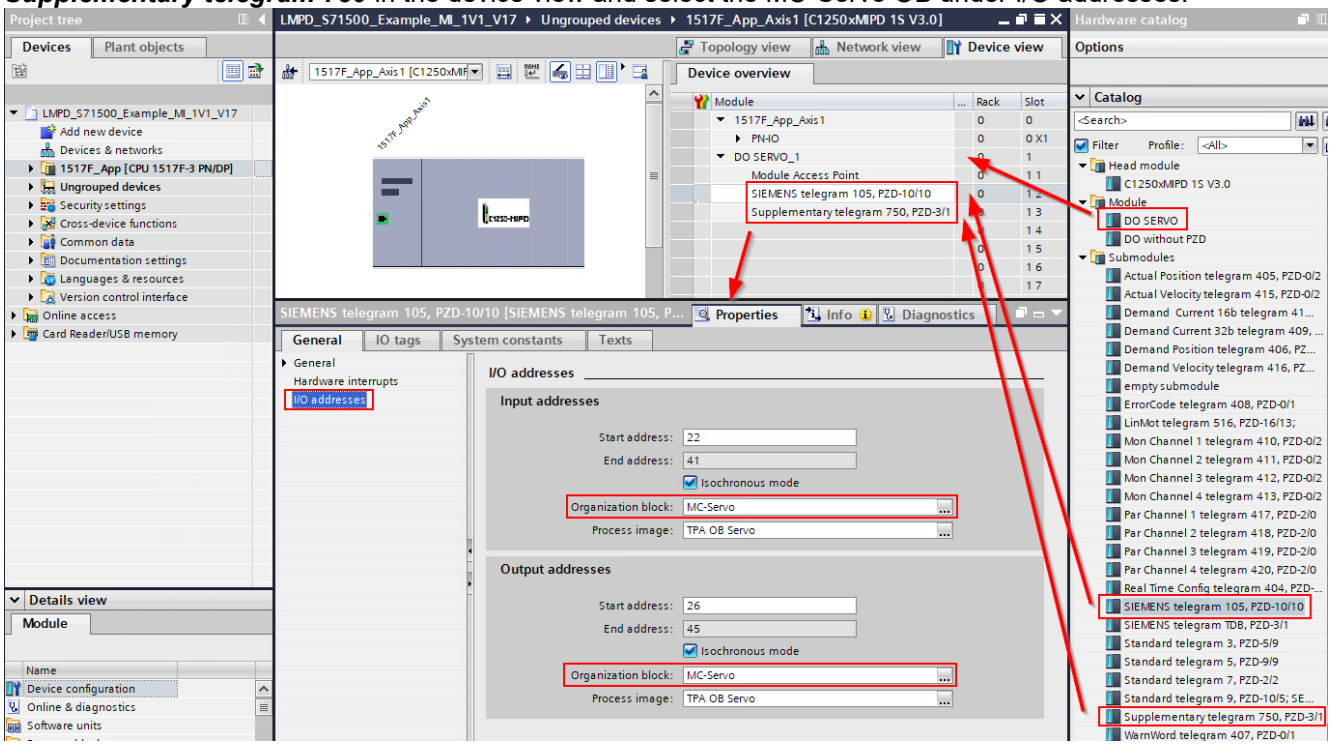
### 10.3.3 Insert the LinMot Drive as Device (-MI Drives)

Select the LinMot C1250-MI-XC-1S (C1250xMIPD 1S V3.0) drive from the catalogue and drag&drop it to the network view.

Then connect it to your PROFINET network.

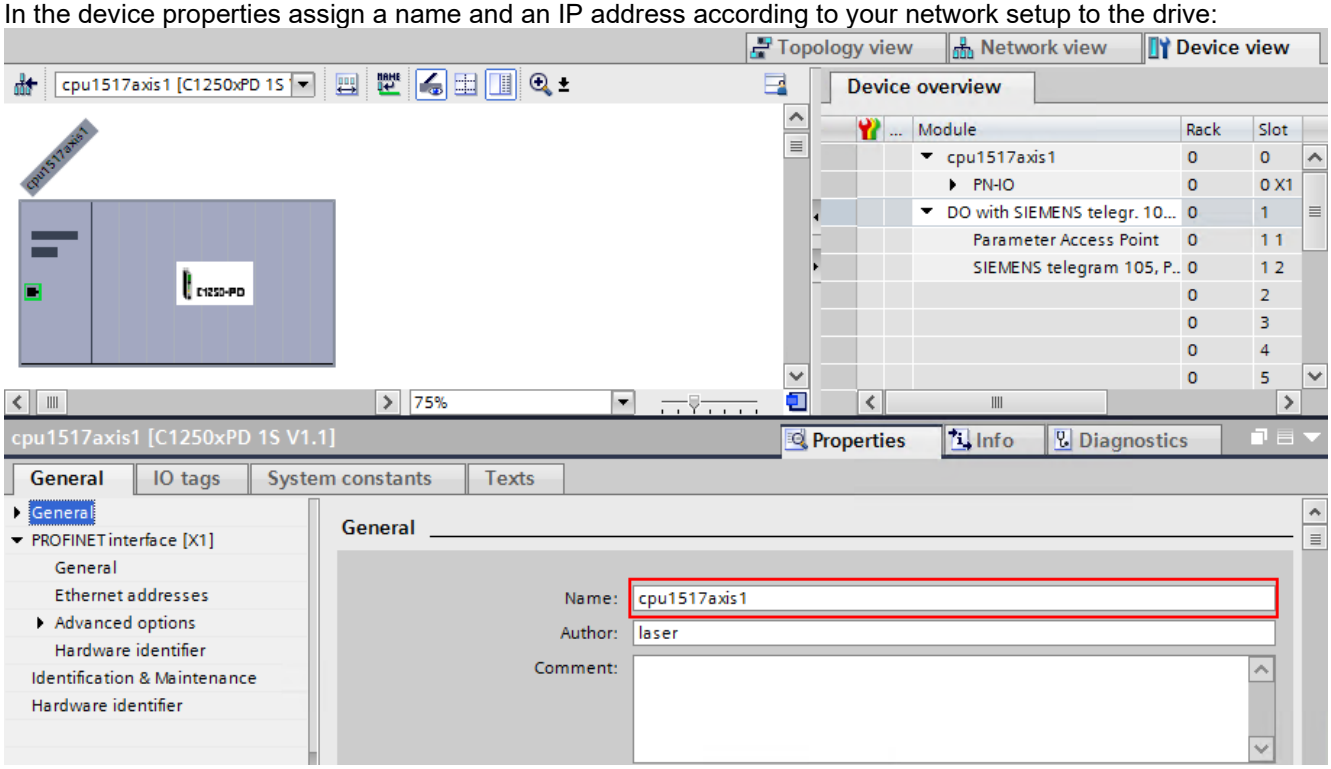


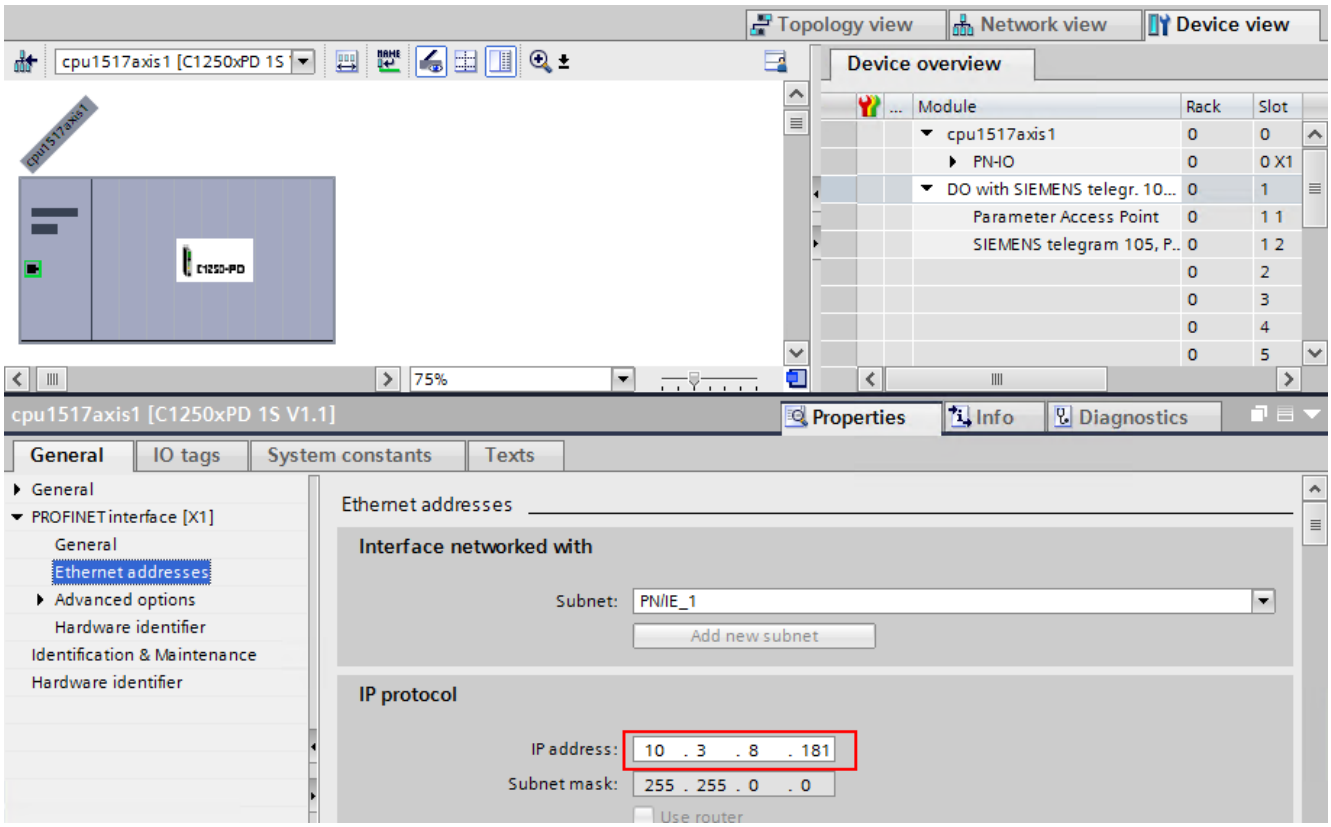

Add the ***DO Servo Module*** and afterwards the submodule ***SIEMENS telegram 105*** and optionally ***Supplementary telegram 750*** in the device view and select the MC-Servo OB under I/O addresses:



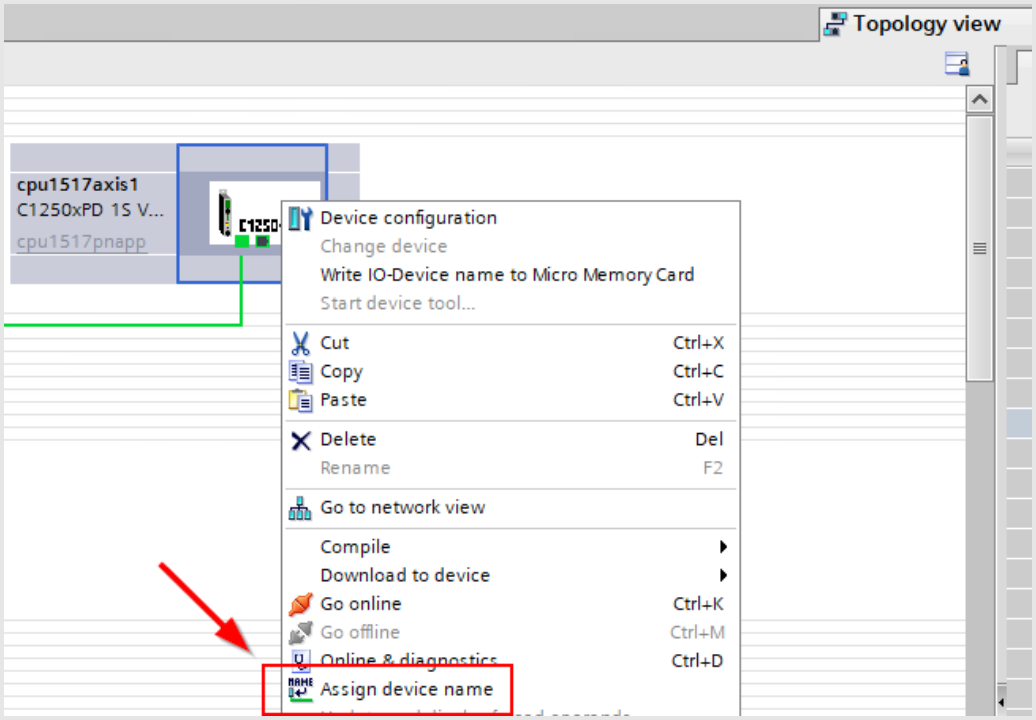
### 10.3.4 Set PROFINET Name and IP Address

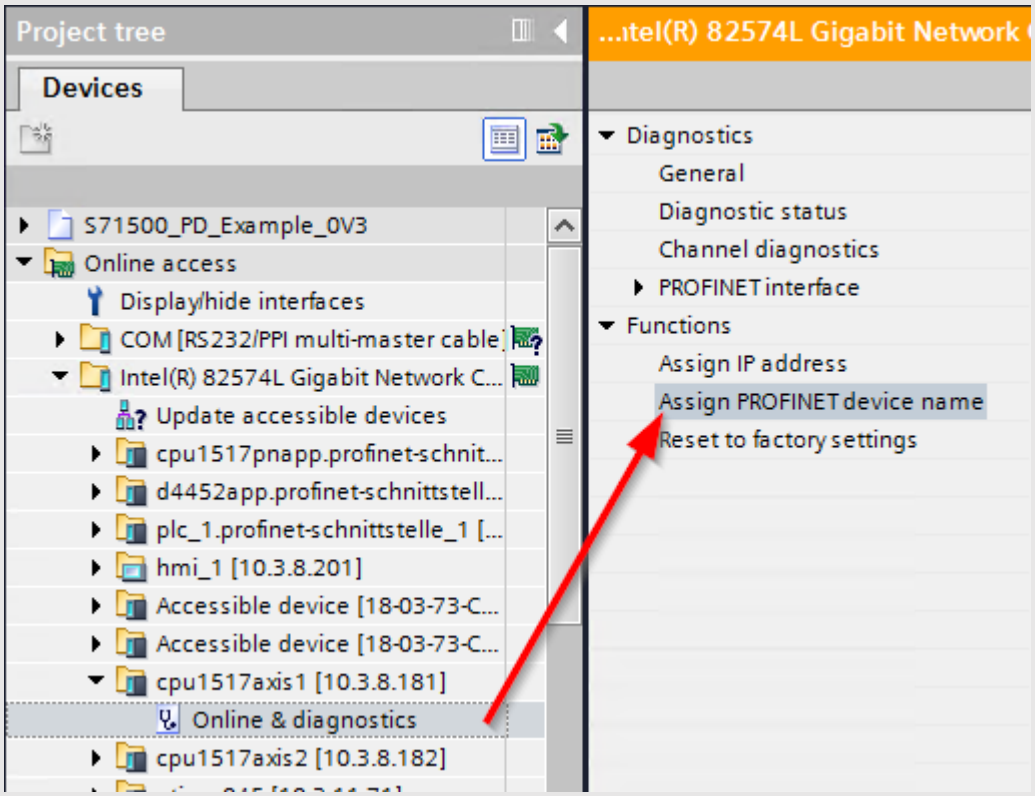
In the device properties assign a name and an IP address according to your network setup to the drive:



**Note:**
If the PROFINET device name is not assigned automatically you can either right-click the device in the topology view and select “Assign device name” or in the project tree under “Online access”.

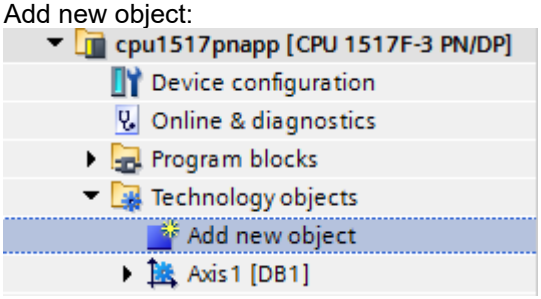


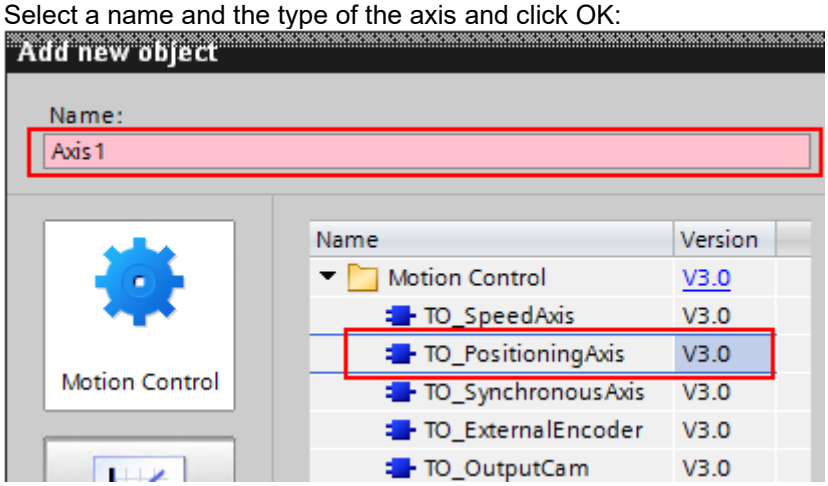

Normally the PROFINET name and IP address are assigned automatically to the devices if the topology is defined.

### 10.3.5 Insert and Setup a new Technology object (Linear Axis)

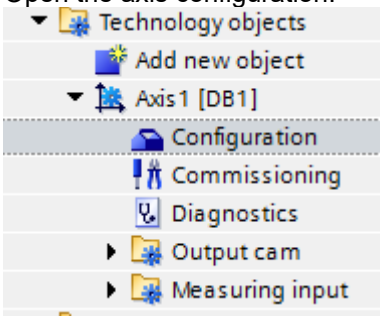
1. Add new object:



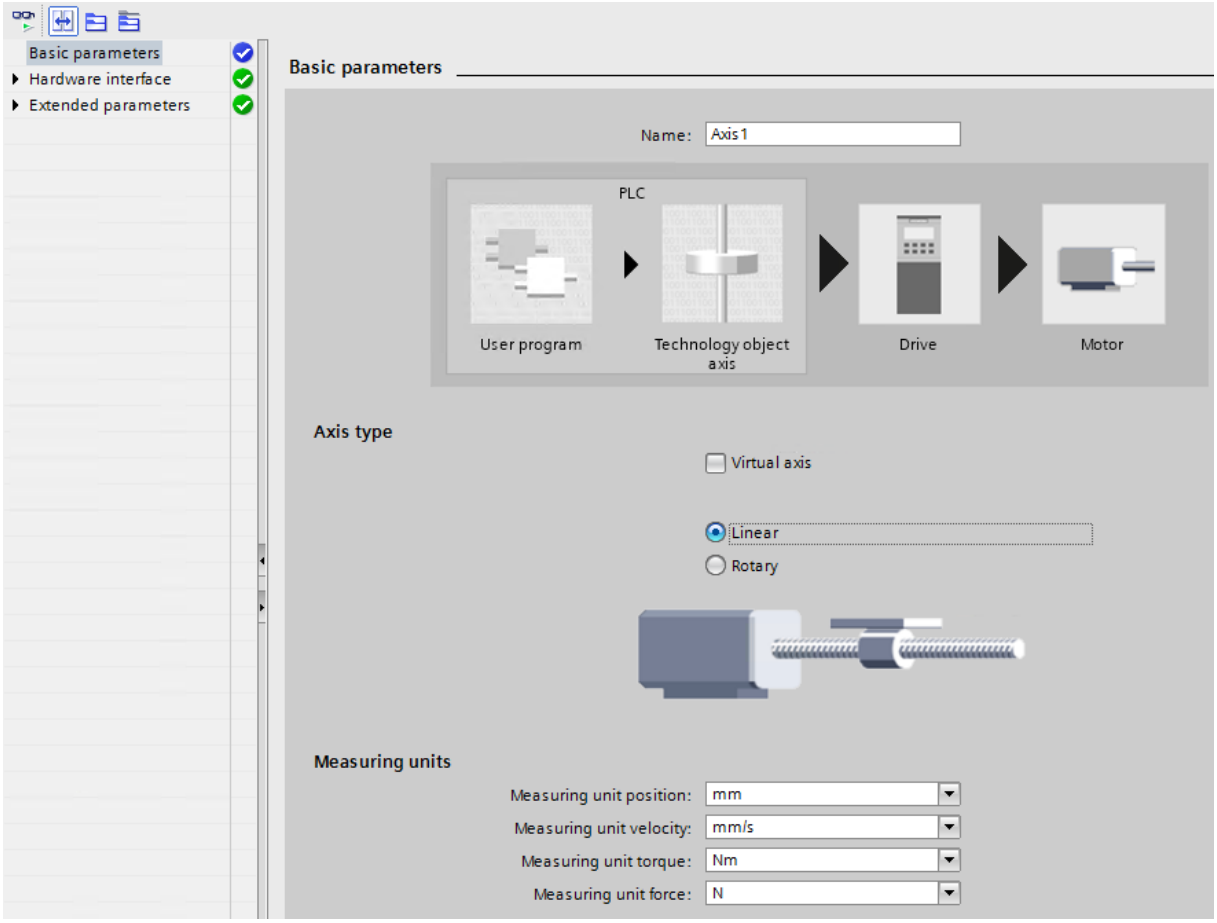

2. Select a name and the type of the axis and click OK:



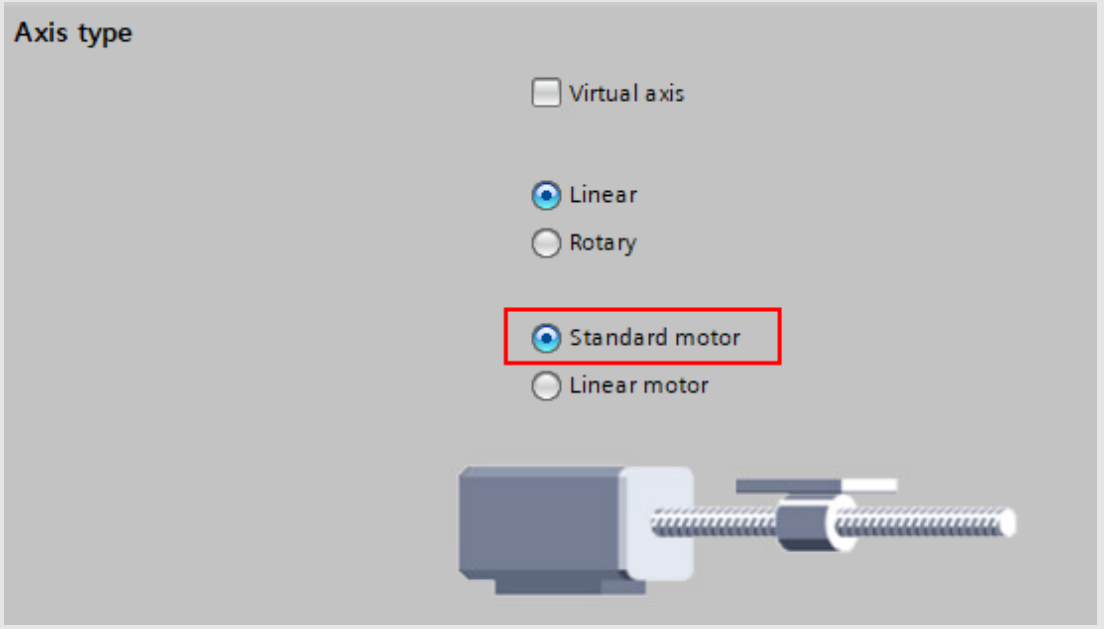

3. Open the axis configuration:



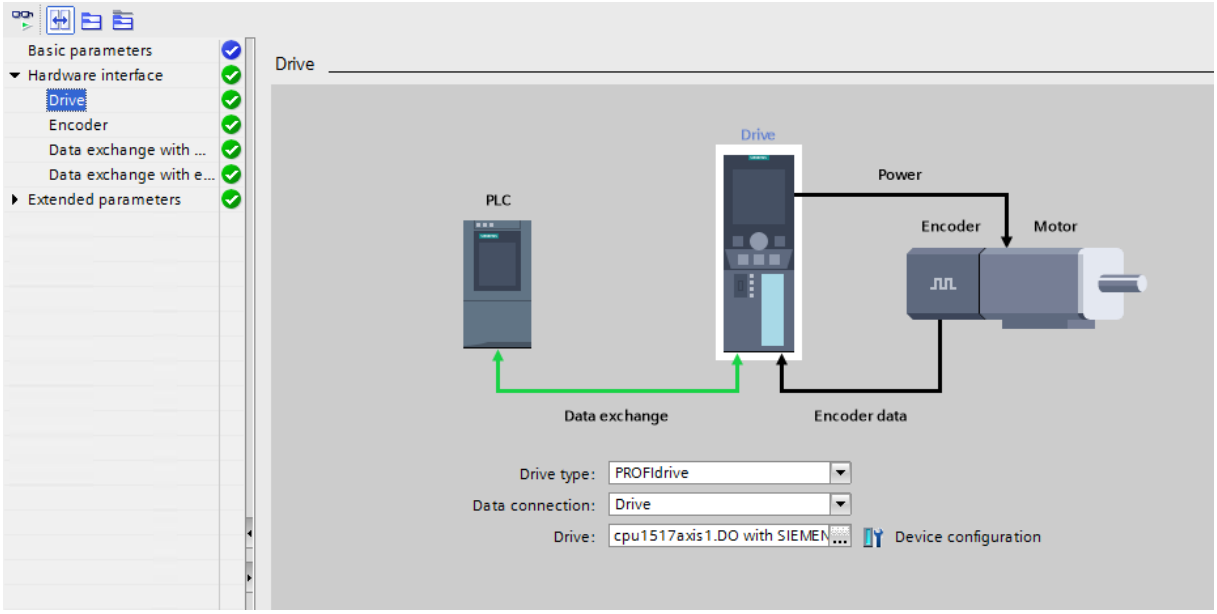
4. Basic Parameters:




**Note:**
From TO version V6.0 there is a new option to select “Standard motor” or “Linear motor”.
Choose “Standard motor”

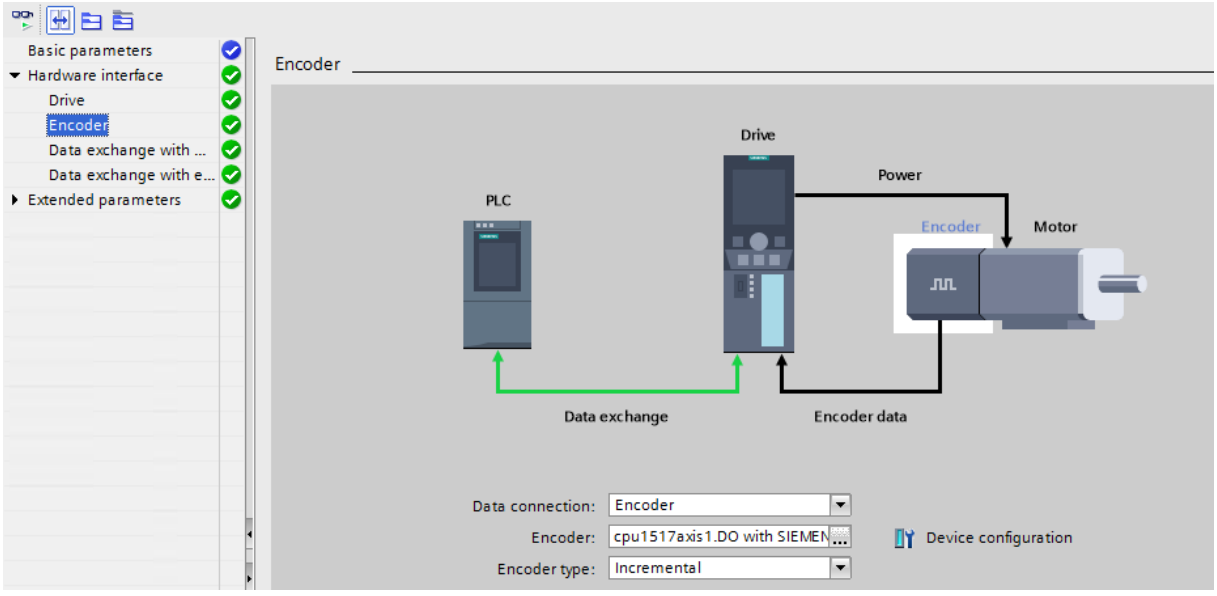


5. Hardware Interface > Drive:



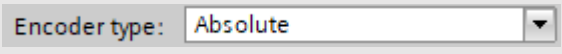

**Note:**
Make sure the correct drive is selected.

6. Hardware Interface > Encoder:



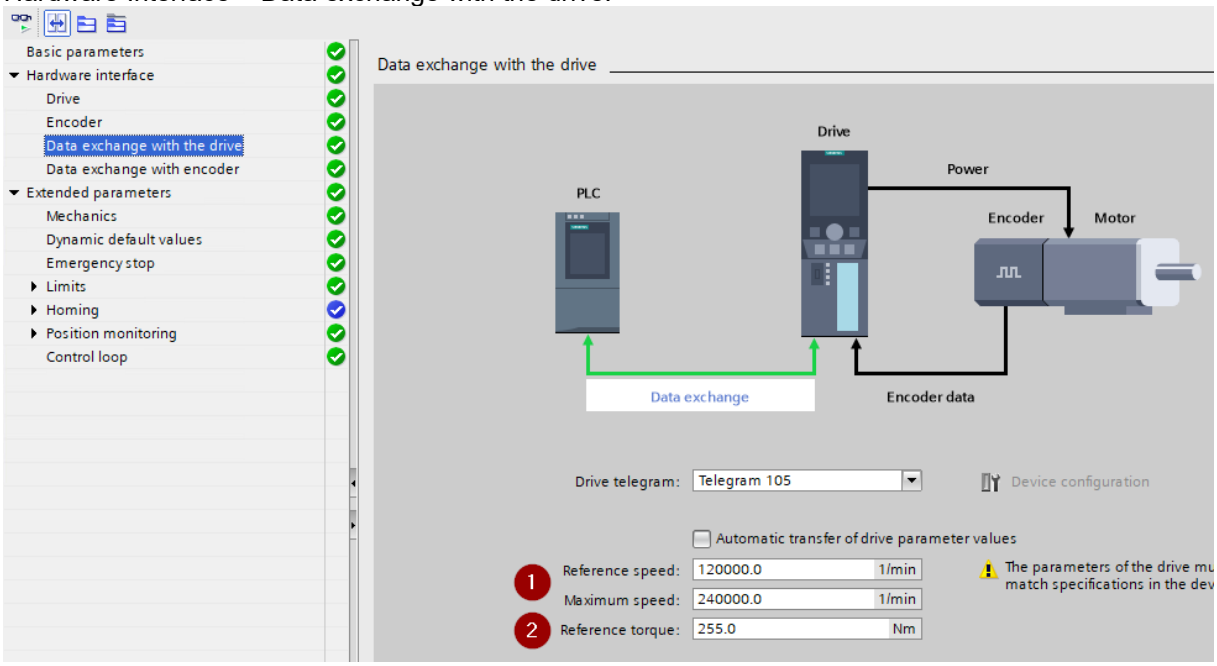

Set Encoder type to *"Incremental"*

**Note:**
If an absolute encoder is connected to the LinMot drive, select "Absolute" as Encoder Type.

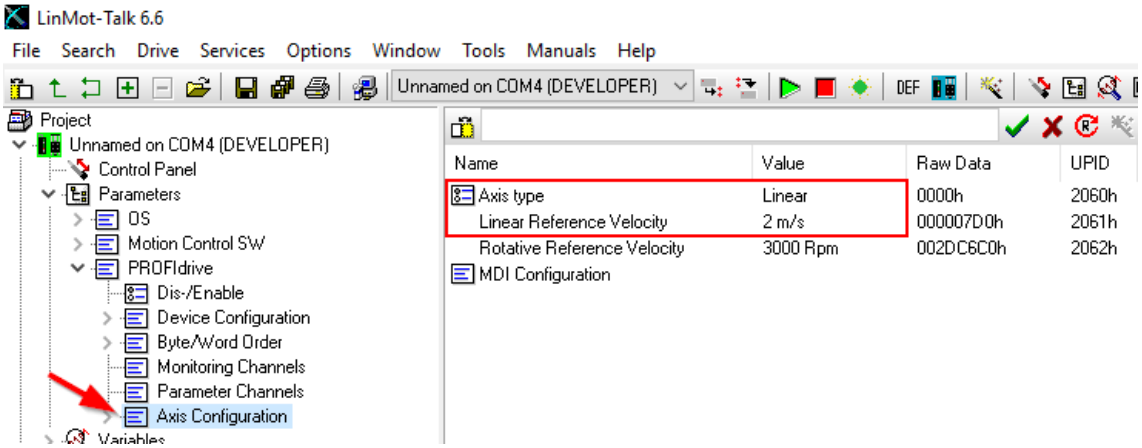



**Please also check the note in chapter 10.4.3 Application based example to** home a LinMot Linear Axis**!**

7. Hardware Interface > Data exchange with the drive:

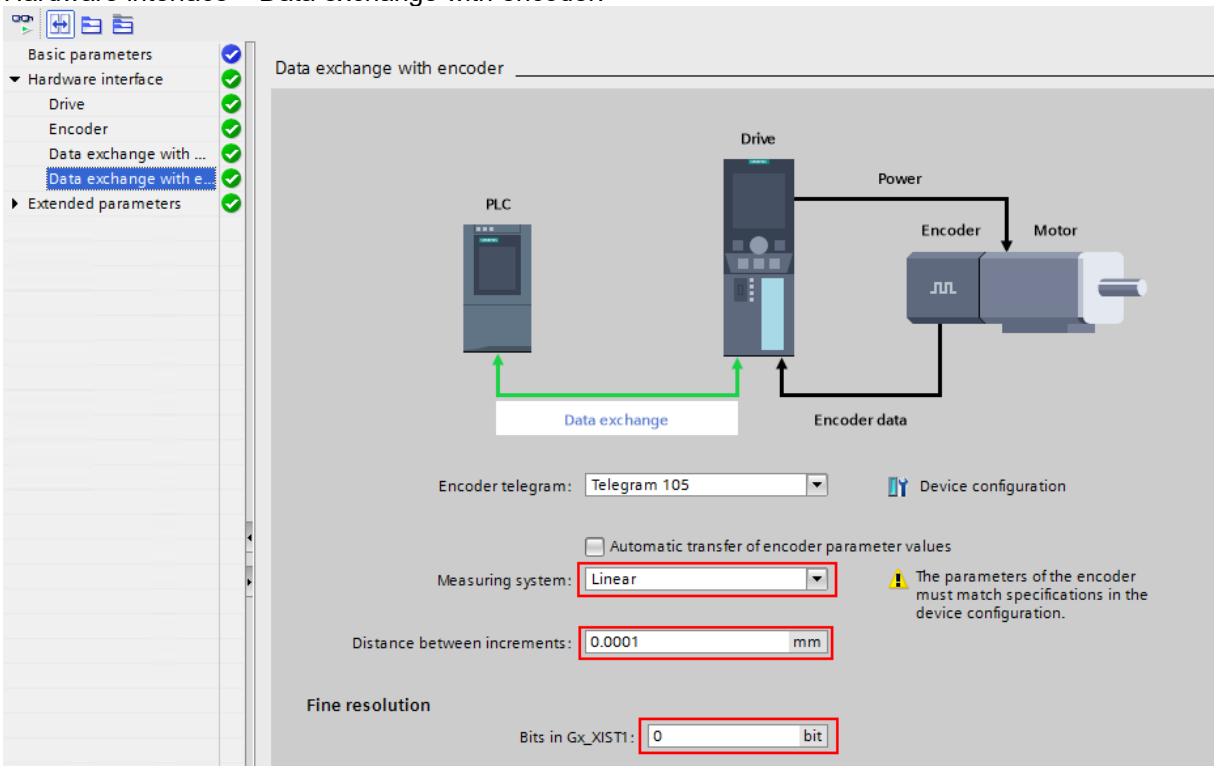



**1:** Set the reference speed according to the setting within the drive.
E.g., drive: 2 m/s => 2000 mm/s * 60 = 120’000 1/min




**2:** Set to the maximum force of the motor.
This value can be found in the data sheet of the connected motor (*Max. Force @72VDC*).
The motor in this example has 255 N peak force.

8. Hardware interface > Data exchange with encoder:

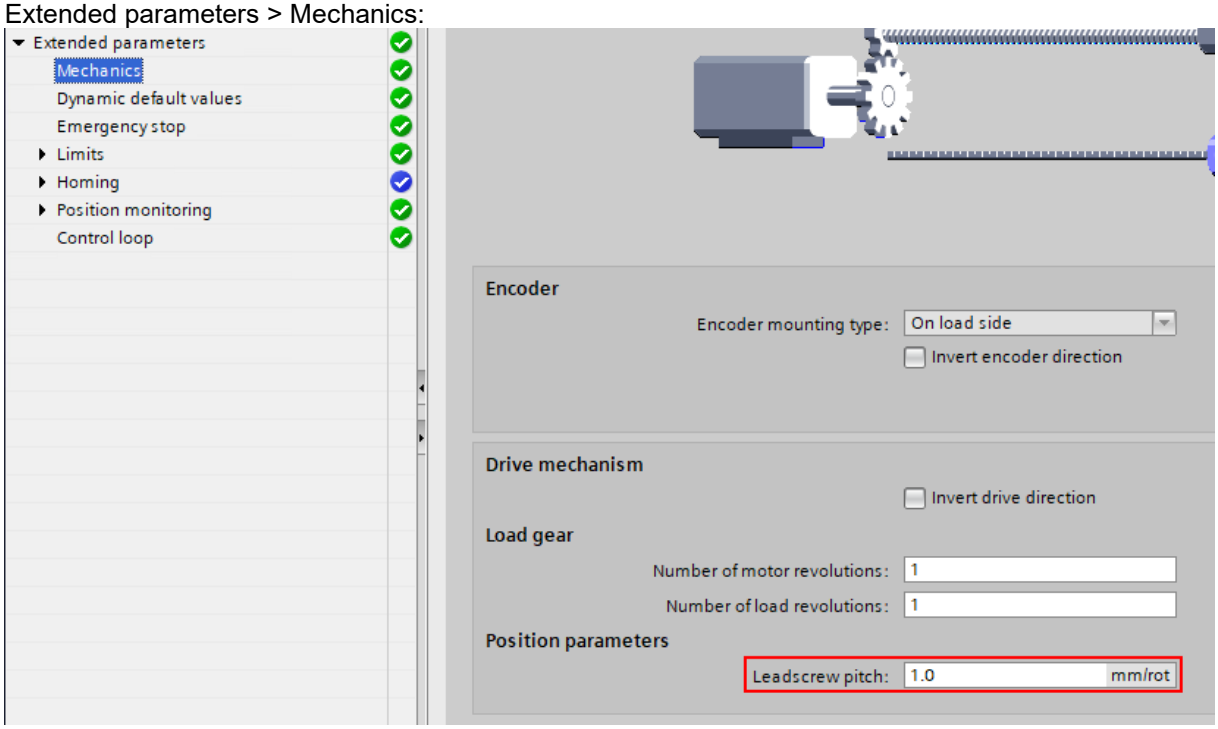



The internal resolution of the LinMot drive is 0.1um.

**Note:**
If an absolute encoder is connected to the LinMot drive and setup in *Hardware Interface* > *Encoder,* then set the fine resolution for both *Gx_XIST1* and *Gx_XIST2* to **0 bit**.


**Please also check the note in chapter 10.4.3 Application based example to** home a LinMot Linear Axis**!**

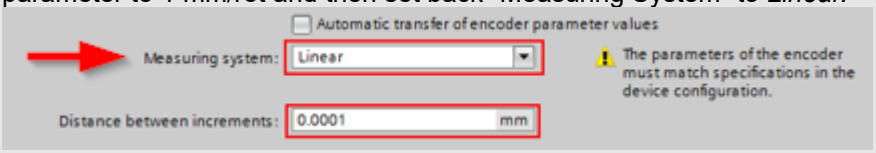
9. Extended parameters > Mechanics:



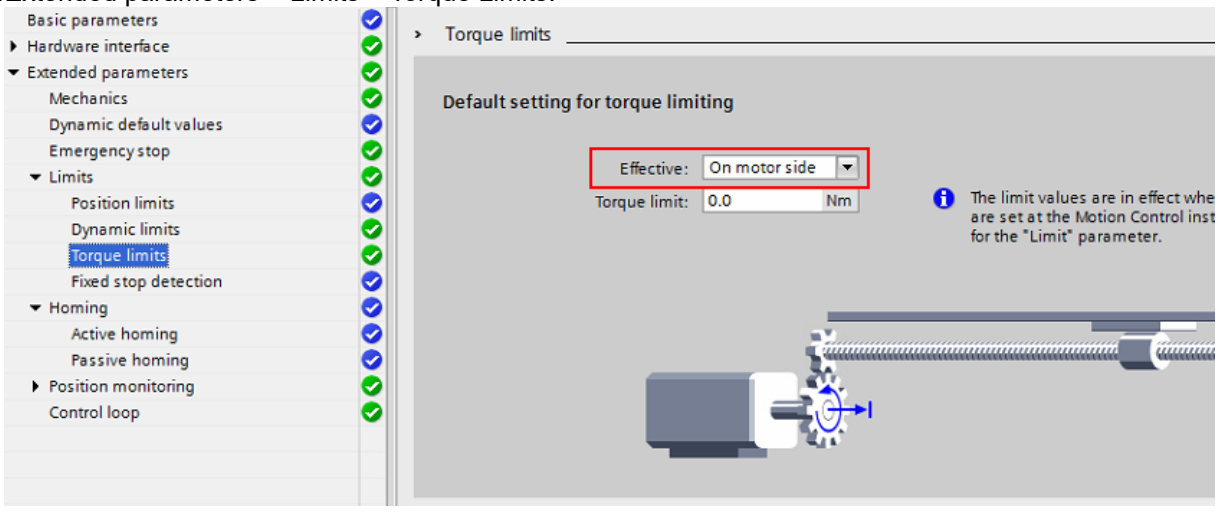

Make sure the Leadscrew pitch is set to 1 mm/rot.

**Attention TIA16:**
In TIA16 this parameter is not shown any more if the “Measuring system” is set to *Linear*. This seems to be a bug in TIA16 as the original value (10mm/rot) is taken anyway.

**Workaround**: Set in **Step 8. above** the “Measuring System” to *Rotary*, change the parameter to 1 mm/rot and then set back “Measuring System” to *Linear.*




10. Extended parameters > Limits > Torque Limits:

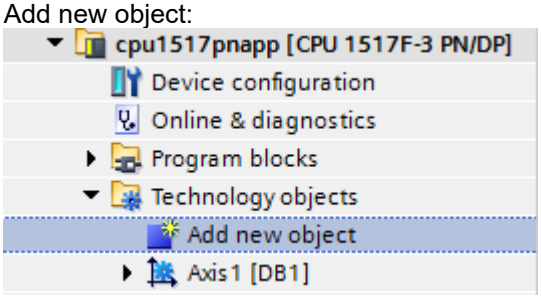



Set Effective to *On motor side.*

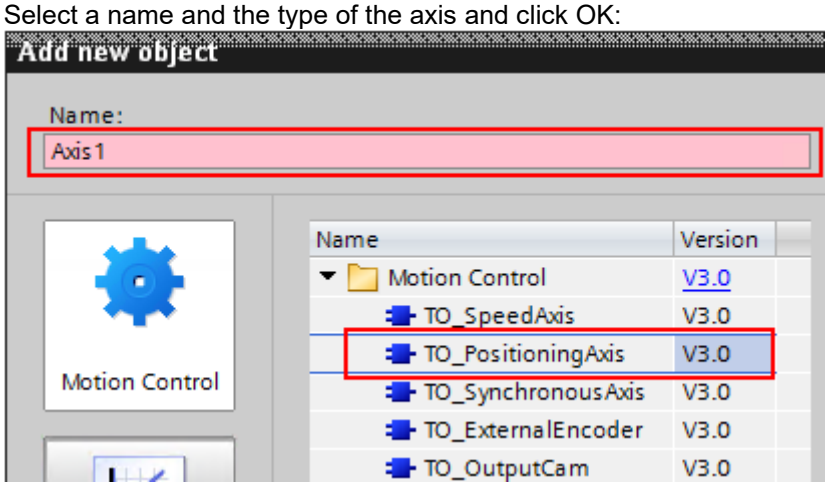
### 10.3.6 Insert and Setup a new Technology object (Rotary Axis)

In this example we setup a RS01-52x40 motor (rotary part of a LinMot PR01-52x40…)

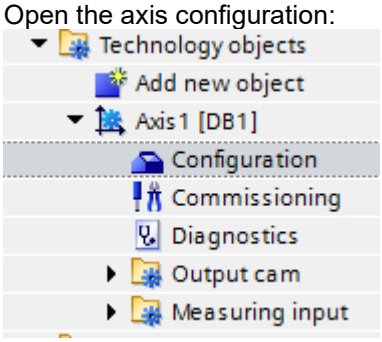
1. Add new object:



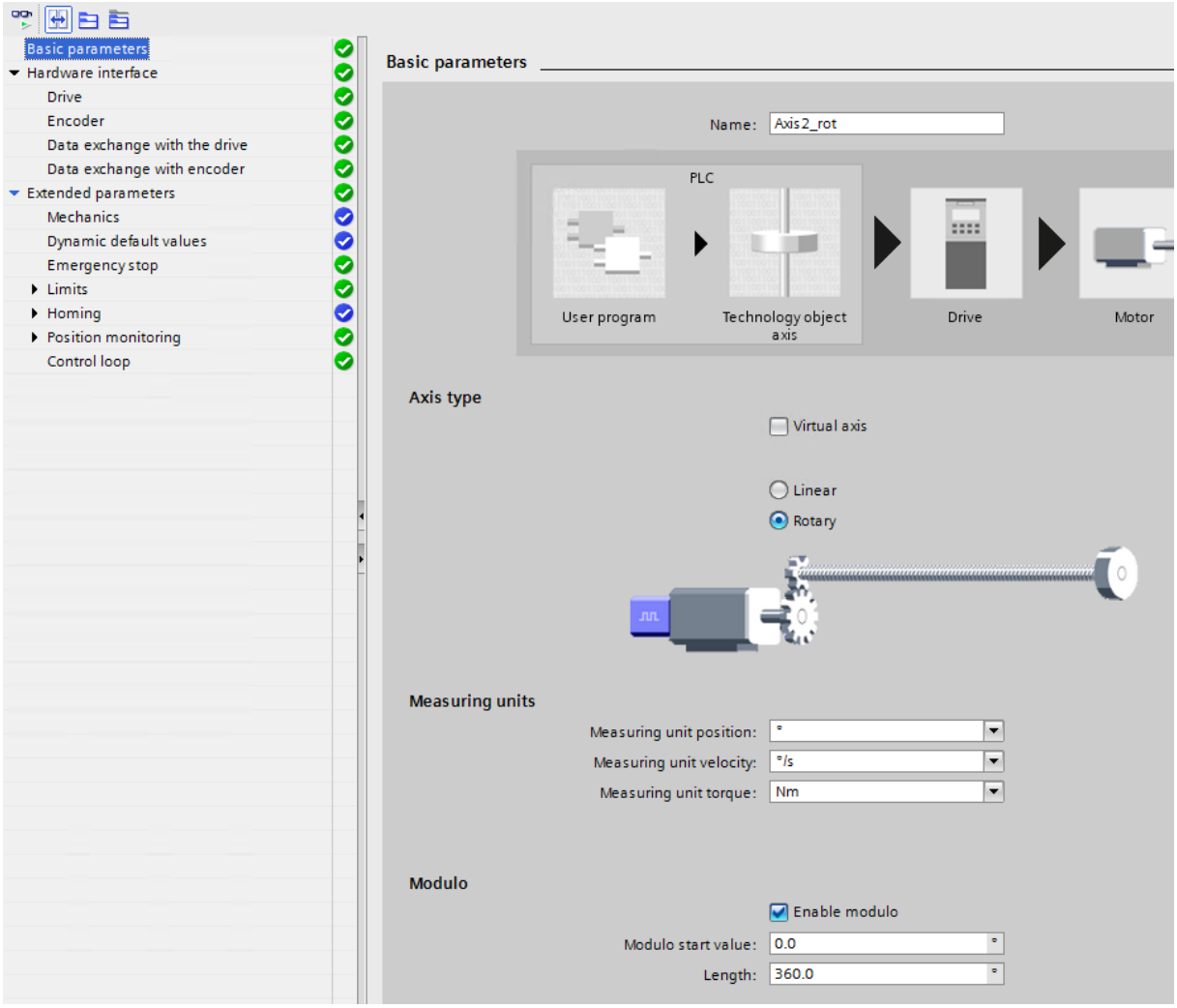

2. Select a name and the type of the axis and click OK:



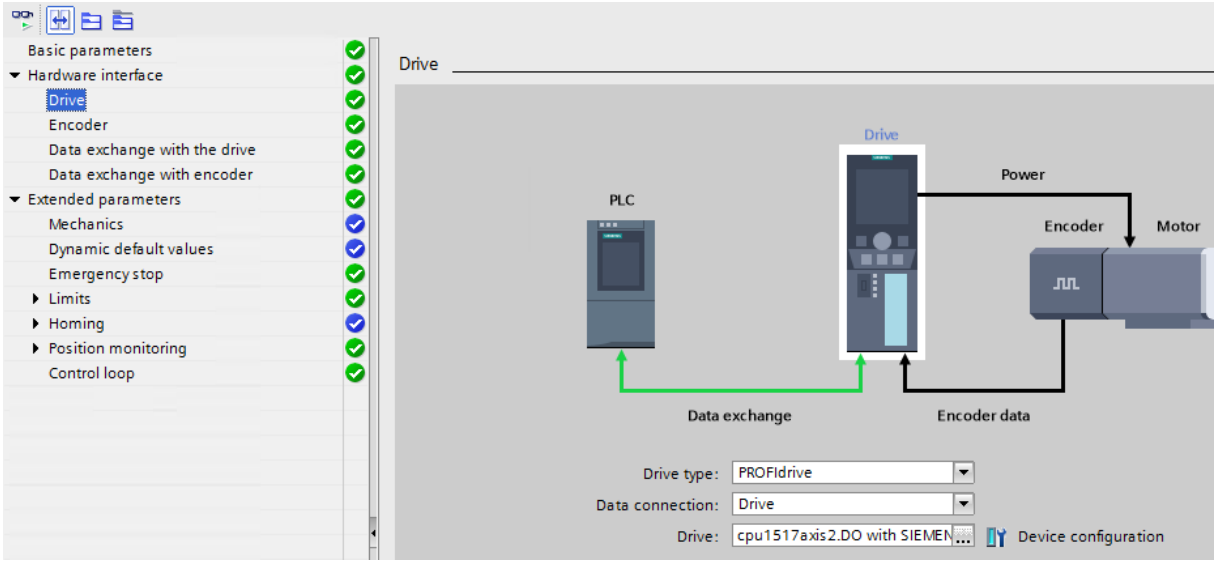

3. Open the axis configuration:



4. Basic Parameters:

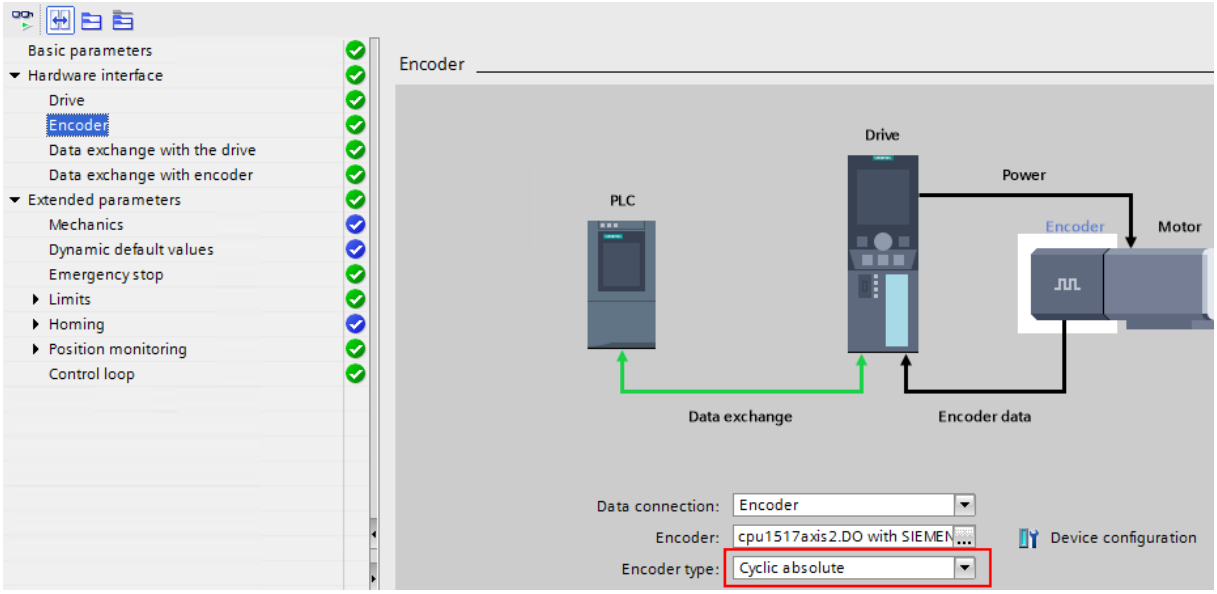



5. Hardware Interface > Drive:




**Note:**
Make sure the correct drive is selected.

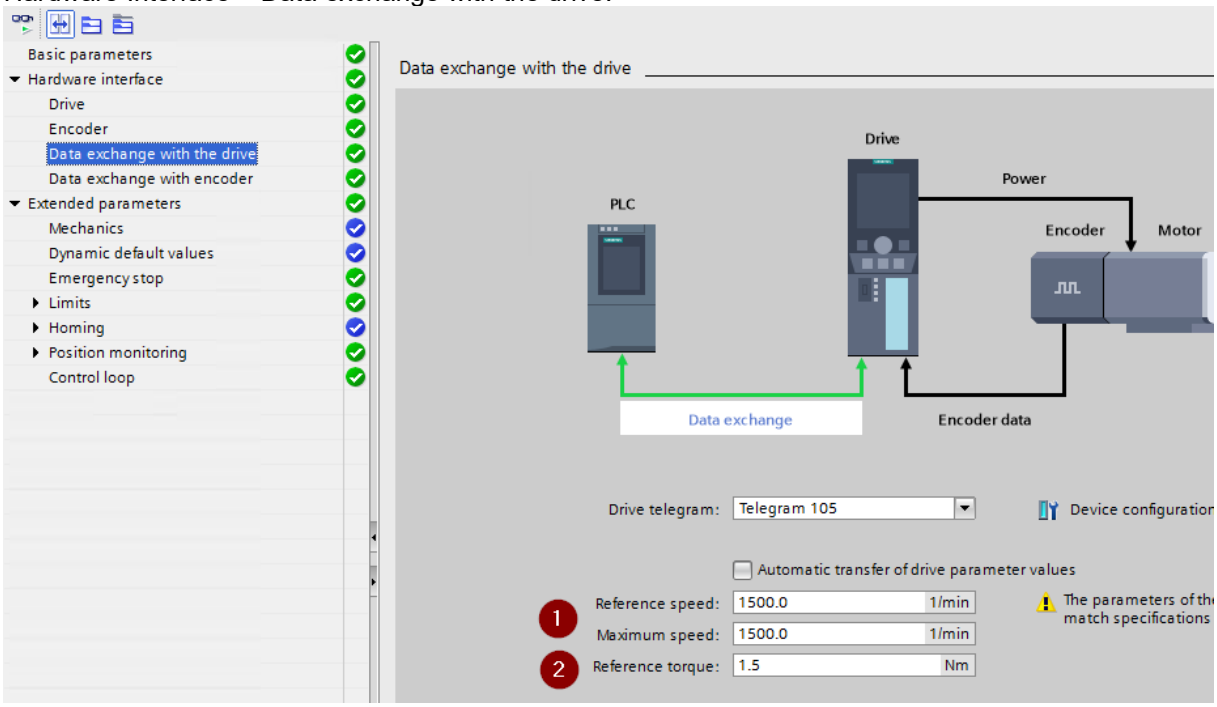
6. Hardware Interface > Encoder:



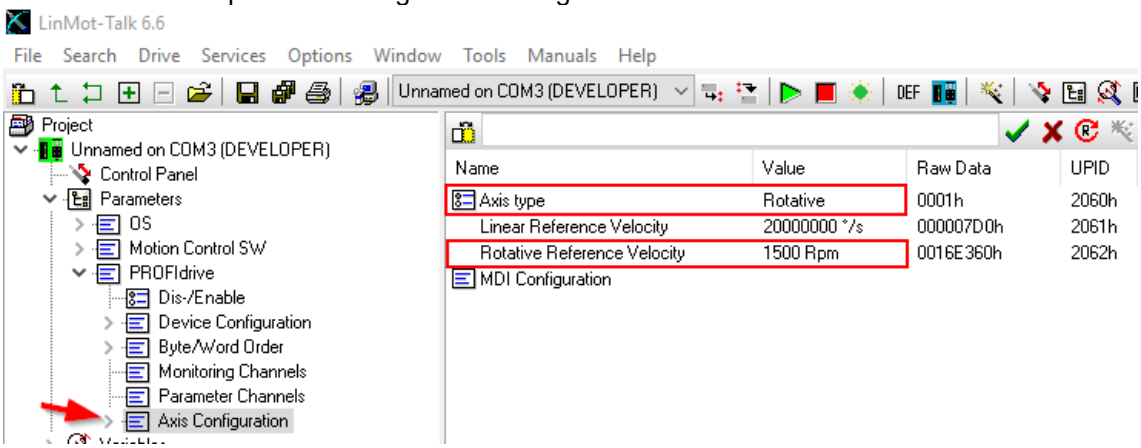

The used RS01-52x40 motor (rotary part of a LinMot PR01-52x40…) has a single turn absolute encoder feedback.

Therefor select “Cyclic absolute” as encoder type.

7. Hardware Interface > Data exchange with the drive:

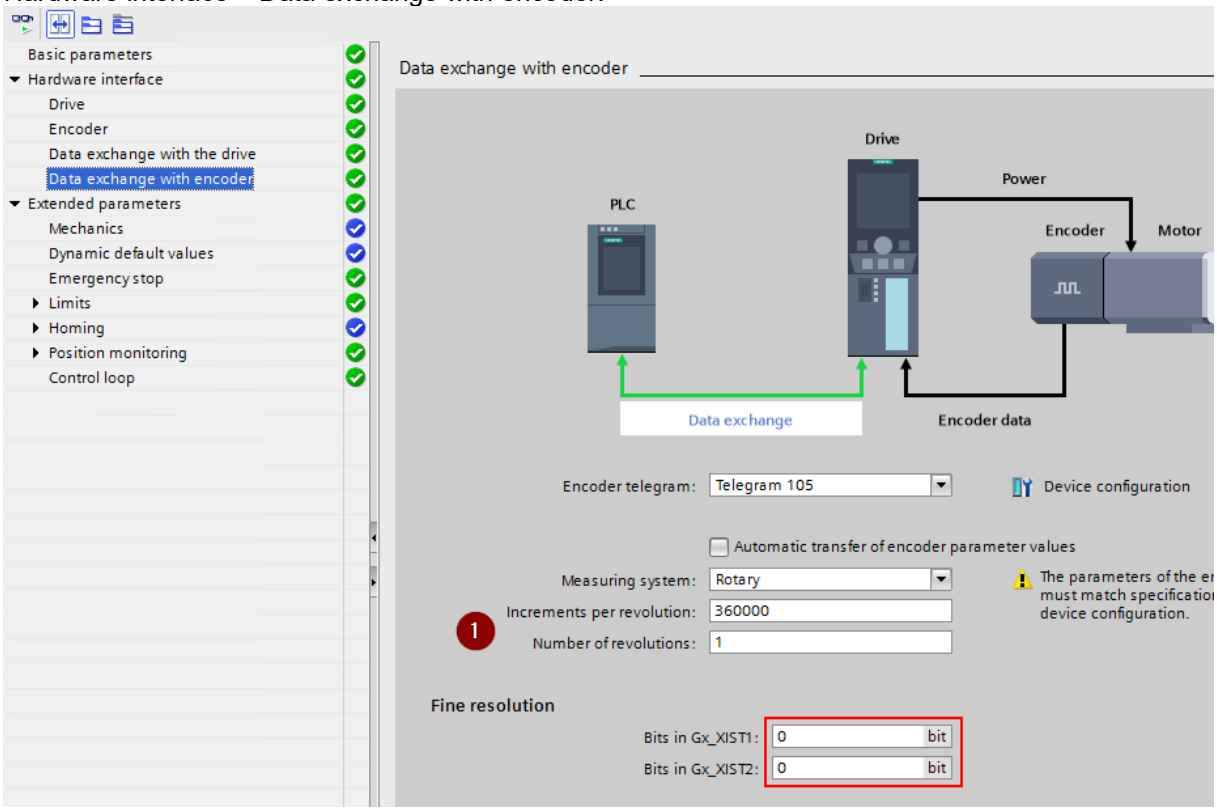



**1:** Set the reference speed according to the settings within the drive:




**2:** Set to the maximum torque of the motor according to the datasheet. The motor in this example has 1.5 Nm peak torque.

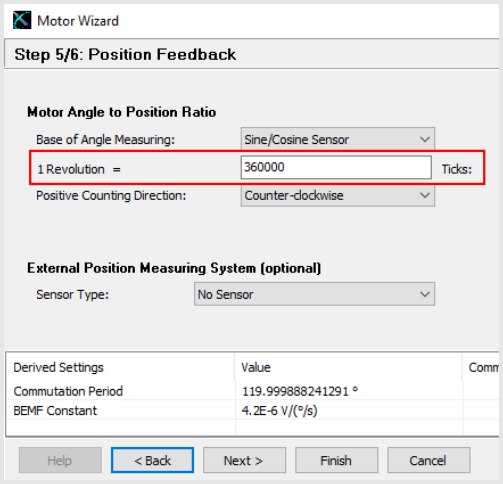
8. Hardware interface > Data exchange with encoder:



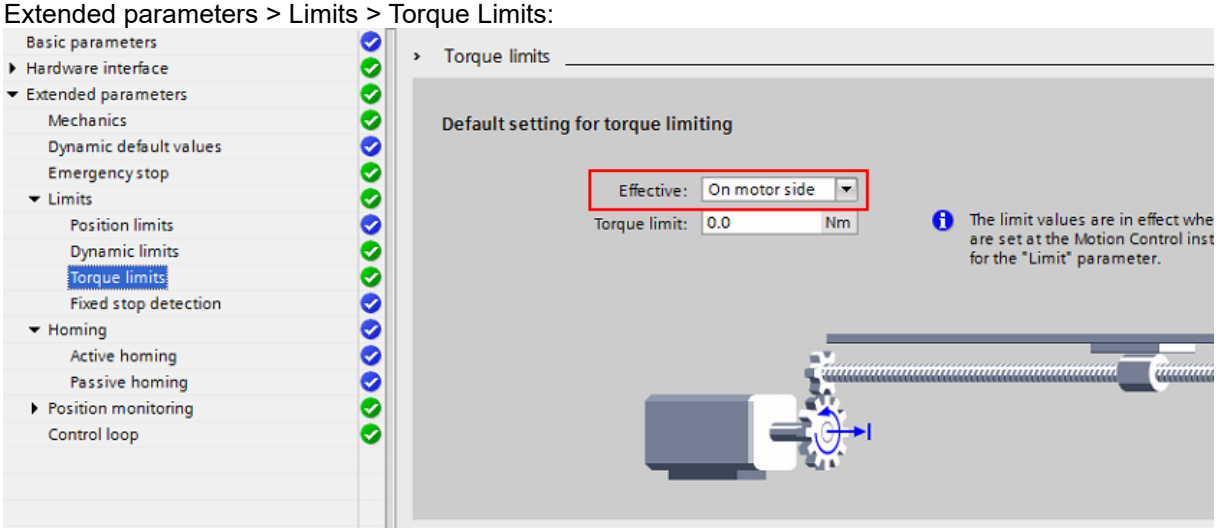

**1:** The used RS01-52x40 motor (rotary part of a LinMot PR01-52x40…) has a single turn absolute encoder feedback.
Set the increments according to the settings within the LinMot-Talk Motor Wizard (default 36mm / 360'000 ticks). By default, this motor has 360'000 ticks per revolution.

**Note:**
Following an overview of LinMot rotary motors showing encoder type and increments per revolution.
- RS01: cyclic absolute (single turn), 360'000 increments per revolution
- EC02: cyclic absolute (single turn), 524'288 increments per revolution

LinMot supports a wide range of rotary motor. Just check the motor wizard for how many ticks per revolution the motor used has.



9. Extended parameters > Limits > Torque Limits:

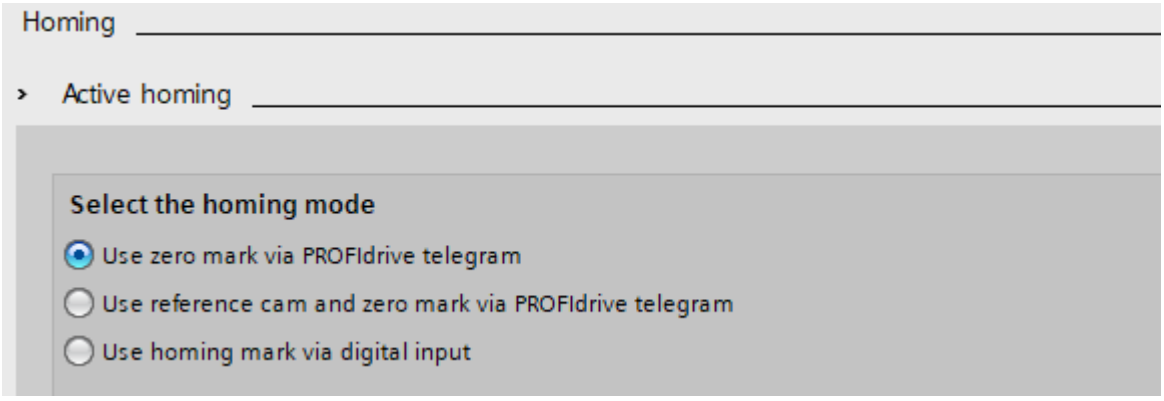



Set Effective to “*On motor side”.*

## 10.4 Home / Set Reference

### 10.4.1 Active Homing for both linear and rotary motors with incremental encoders

From firmware 6.8 Build 20190315 and later it is possible to use a zero mark via PROFIdrive telegram to execute an active homing using the MC_Home function block of SIEMENS.

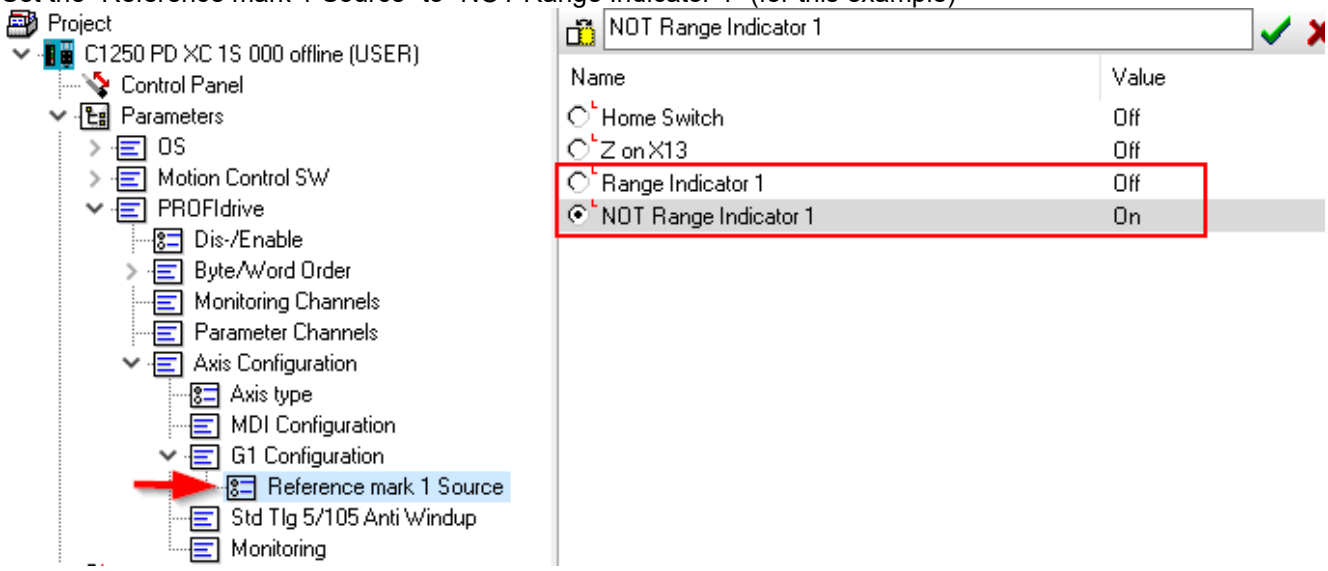



On the following pages there are two examples showing the setup in the LinMot drive using the LinMot-Talk software.

**Example 1:** Generating the zero mark using the Range Indicator 1 functionality of the LinMot drive. The current of the motor should be monitored to detect the hard stop. As soon as the motor current exceeds a set limit the zero mark is generated.

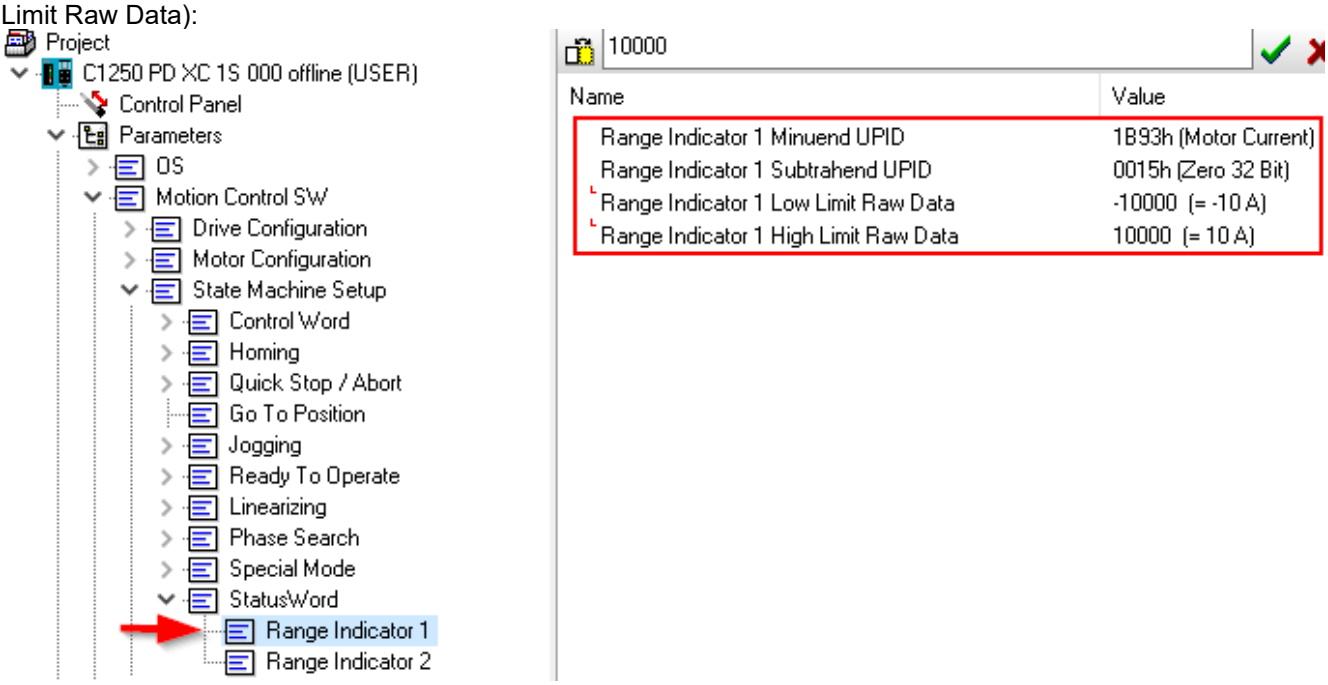
From firmware **6.9 Build xxxxxxx**x the Range Indicator 1 can be used to generate the zero mark via PROFIdrive telegram.

Set the “Reference mark 1 Source” to “NOT Range Indicator 1” (for this example)




As example the motor current or difference position can be used to generate the zero mark.

The *Range Indicator 1* can be setup here (Choose UPID 1B93h as Minuend UPID and set the Low and High Limit Raw Data):

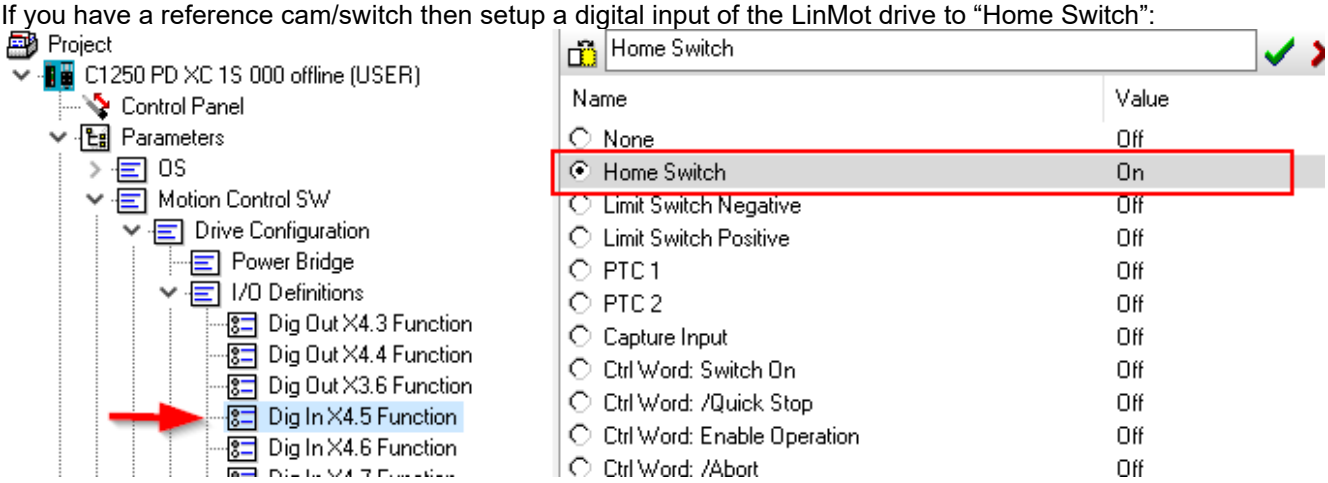



In this case the Range Indicator 1 Bit is set when the Demand Current is between -10A and 10A.
Therefor the zero mark will be generated when the motor current is >10A or <-10A.

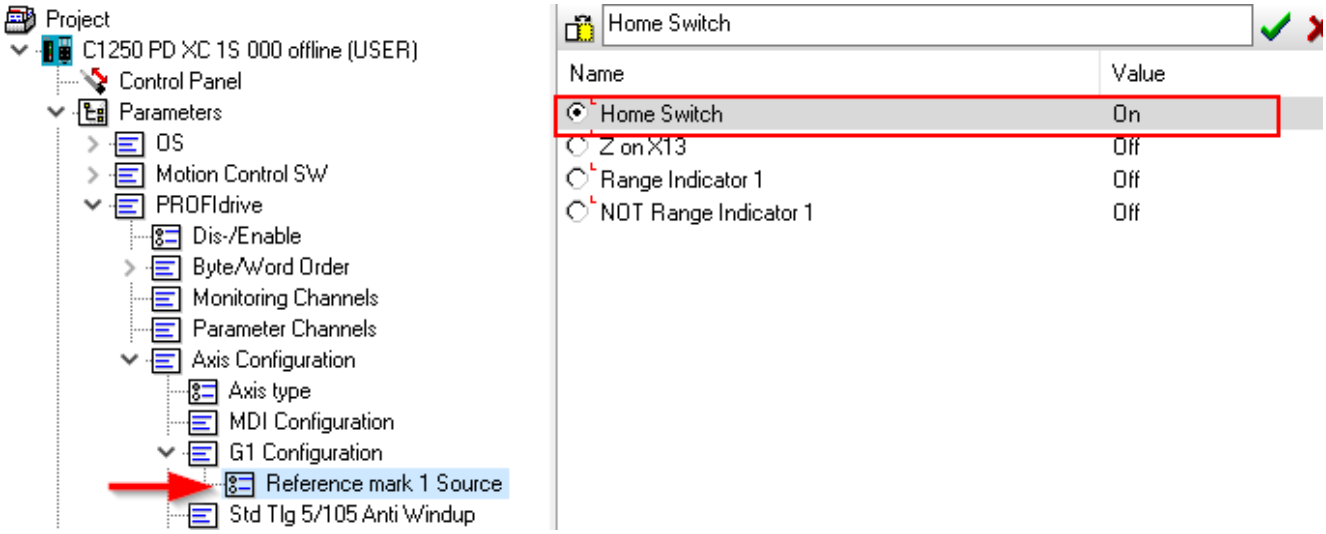
The **active homing** itself can then be executed using the **MC_HOME** function block with **mode 3 or 5**.

**Example 2:** Generating the zero mark using a Home Switch (external 24V DC signal connected to the drive)

If you have a reference cam/switch then setup a digital input of the LinMot drive to “Home Switch”:




Set “Reference mark 1 Source” to “Home Switch”:




The **active homing** itself can then be executed using the **MC_HOME** function block with **mode 3 or 5**.

### 10.4.2 Set Reference for both linear and rotary motors with absolute encoders

As the rotary motor used in this example has a single turn absolute encoder it is only required to set the reference flag of the axis. This is also valid for linear axes with absolute encoders too (please check the note below).

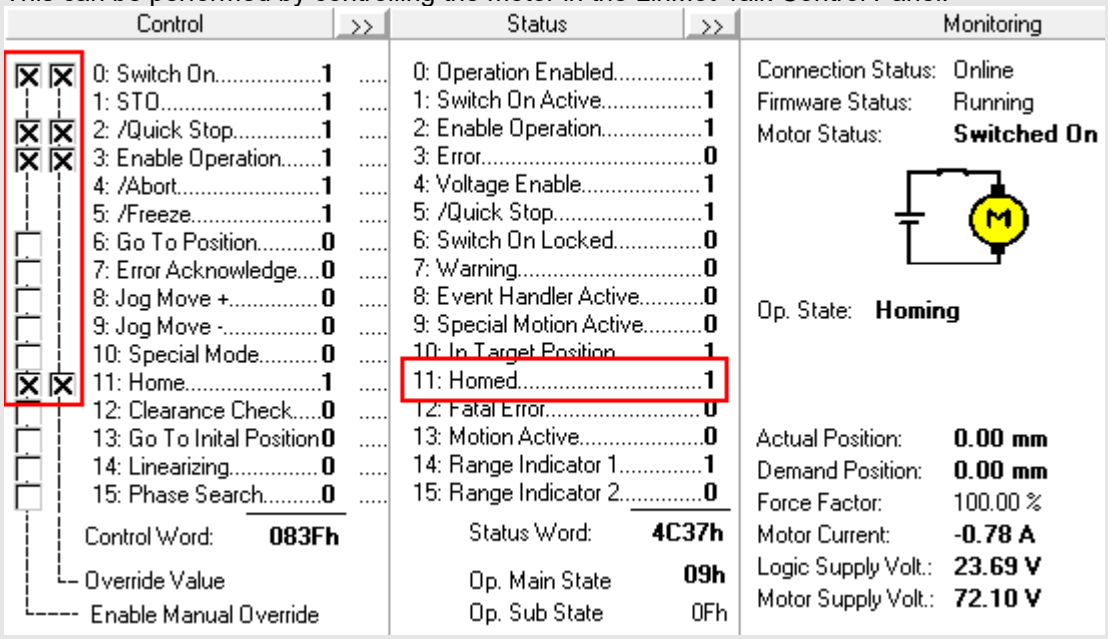
This can be done using the function block **MC_HOME with mode 6** (Absolut encoder adjustment (relative)) and Position = 0.0.

**Note:**
Please check the SIEMENS Information System for more information about the Motion Control function blocks.

**Note for linear absolute encoders:**

**1.** If a linear motor with absolute encoder is used, then first a homing in the drive must be completed once to set the absolute encoder offset in the drive. (e.g., use homing mode “Mechanical Stop Negative Search” > UPID 13C4h [set back to “No Drive Homing” when completed!]).

This can be performed by controlling the motor in the LinMot-Talk Control Panel:




**2.** After that the absolute encoder adjustment in the PLC must be performed **ONCE** using mode 6 or 7 of the function block MC_Home (check SIEMENS Information System):
Mode 6: Absolute encoder adjustment (relative)
Mode 7: Absolute encoder adjustment (absolute)

This will cause the PLC to store the calculated absolute value offset retentively in the CPU.
*(<TO>.StatusSensor[n].AbsEncoderOffset)*

***ATTENTION**: If the Actual Position in the Control Panel of LinMot-Talk is showing a negative value when using the MC_Home it is recommended to use mode 7 (absolute) and **set the input Position to the value of the Actual Position in the Control Panel.** Then the correct offset (negative) is stored in the PLC.*

### 10.4.3 Application based example to home a LinMot Linear Axis

If for any reason the active homing procedure as explained in chapter 10.4.1 can't be used it is also possible to do the homing application based.

Using the torque limiting functionality (MC_TORQUELIMITING) of SIEMENS Telegram 105 it is possible to make a referencing to a mechanical stop with a desired force limit.

If the hard stop should be detected when the motor reaches the set force limit then check for "InLimitation" in state 5 of the following example, or if the hard stop should be detected by the position lag (*"Axis1".Clamping.FollowingErrorDeviation*) then use "InClamping".

The next page shows how an application-based homing can be performed.

Following a very basic example homing routine that is also part of the example project:

```
CASE "GVL".iHomeState OF
    0:  //Idle
        IF "GVL".bHomeExecute THEN
            "GVL".iHomeState := 1;
        END_IF;

    1:  //Check motor powered
        IF "MC_POWER_Axis1".Status THEN
            "GVL".iHomeState := 2;
        ELSE
            "MC_POWER_Axis1".Enable := TRUE;
        END_IF;

    2:  //Enable Torque Limiting and start moving direction hard stop
        IF NOT "MC_TORQUELIMITING_Axis1".Busy THEN
            "MC_TORQUELIMITING_Axis1".Mode := 1; //Detect hard stop
            "MC_TORQUELIMITING_Axis1".Limit := 100; //Motor force limit in Newton [N]
            "MC_TORQUELIMITING_Axis1".Enable := TRUE; //Enable Torque Limiting
            "Axis1".Clamping.FollowingErrorDeviation := 0.5; //Set clamping detection window [mm]
            "Axis1".Clamping.PositionTolerance := 0.3; //Set clamping position tolerance [mm]
        END_IF;

        "MC_MOVEVELOCITY_Axis1".Velocity := -10; //Velocity to find the hard stop. Values <10mm/s!
        "MC_MOVEVELOCITY_Axis1".Execute := TRUE;

        IF "MC_MOVEVELOCITY_Axis1".Busy THEN
            "GVL".iHomeState := 3;
        END_IF;

    3: // Check axis in limitation & clamping (wait MoveVelocity aborted) and set axis reference
        IF "MC_TORQUELIMITING_Axis1".InLimitation AND "MC_MOVEVELOCITY_Axis1".CommandAborted THEN
            // Set reference position
            "MC_HOME_Axis1".Mode := 0; //Set absolute reference
            "MC_HOME_Axis1".Position := -2.0; //Reference position
            "MC_HOME_Axis1".Execute := TRUE;
        END_IF;

        IF "MC_HOME_Axis1".Done THEN
            "MC_HOME_Axis1".Execute := FALSE;
            "GVL".iHomeState := 5;
        END_IF;

    5:  //Move away from hard stop
        "MC_MOVEABSOLUTE_Axis1".Position := 0.0; //Move to 0 position
        "MC_MOVEABSOLUTE_Axis1".Execute := TRUE;
        "MC_MOVEABSOLUTE_Axis1".Velocity := 1; //Slow velocity
        IF "MC_MOVEABSOLUTE_Axis1".Done THEN
            "MC_MOVEABSOLUTE_Axis1".Execute := FALSE;
            "GVL".iHomeState := 10;
        END_IF;

    10: //Clean up
        "MC_MOVEABSOLUTE_Axis1".Execute := FALSE;
        "MC_MOVEABSOLUTE_Axis1".Velocity := -1;
        "MC_HALT_Axis1".Execute := FALSE;
        "MC_HOME_Axis1".Execute := FALSE;
        "MC_MOVEVELOCITY_Axis1".Execute := FALSE;
        "MC_TORQUELIMITING_Axis1".Enable := FALSE;

        "GVL".bHomeExecute := FALSE;
        "GVL".iHomeState := 0;

    ELSE
        ;
END_CASE;
```

## 10.5 Parameter Access (RAM)

Read/Write access to the RAM value of LinMot drive parameters/variables can be done using the SIEMENS function block “SINA-PARA” (FB286) from the library “DriveLib_S71500” that is installed with SINAMICS Startdrive.

The RAM and ROM values of the drive parameters can be accessed by their parameter number (UPID) added with an offset of 0x2000 (UPID+0x2000).

More information about accessing parameters can be found on the SIEMENS support page:
https://support.industry.siemens.com (Entry ID: 99412590)

**Example:**

Read the value of the parameter Maximal Current (UPID 16#13A6):

**Name:** Maximal Current
**UPID:** 16#13A6 = 5030
**Type:** SInt32 = DINT
**Scale:** 0.001 A

> .ParaNo = 1 (read 1 parameter)
> .sxParameter[1].siParaNo = 16#2000 + 16#13A6 = 16#33A6 = 13222

| Name | ... | Display format | Monitor value | Modify value | | Comment |
|---|---|---|---|---|---|---|
| "SINA_PARA_Axis1".Start | | Bool | TRUE | TRUE | ☑ ⚠ | |
| "SINA_PARA_Axis1".ReadWrite | | Bool | FALSE | FALSE | ☑ ⚠ | |
| "SINA_PARA_Axis1".ParaNo | | DEC+/- | 1 | 1 | ☑ ⚠ | |
| "SINA_PARA_Axis1".AxisNo | | Hex | 16#00 | | ☐ | |
| "SINA_PARA_Axis1".hardwareId | | DEC | 267 | | ☐ | |
| "SINA_PARA_Axis1".Done | | Bool | TRUE | | ☐ | |
| "SINA_PARA_Axis1".Busy | | Bool | FALSE | | ☐ | |
| "SINA_PARA_Axis1".Error | | Bool | FALSE | | ☐ | |
| "SINA_PARA_Axis1".ErrorId | | Hex | 16#0000_0000 | | ☐ | |
| "SINA_PARA_Axis1".sxParameter[1].sdValue | | DEC+/- | 7000 | 0 | ☑ ⚠ | |
| "SINA_PARA_Axis1".sxParameter[1].siIndex | | DEC+/- | 0 | | ☐ | |
| "SINA_PARA_Axis1".sxParameter[1].siParaNo | | DEC+/- | 13222 | 13222 | ☑ ⚠ | // 16#2000 + UPID = 16#33A6 = 13222 |
| "SINA_PARA_Axis1".sxParameter[1].srValue | | Floating-point nu... | 0.0 | | ☐ | |
| "SINA_PARA_Axis1".sxParameter[1].swErrorNo | | Hex | 16#0000 | | ☐ | |
| "SINA_PARA_Axis1".sxParameter[1].syFormat | | Hex | 16#07 | | ☐ | |

Set .Start TRUE.
As soon as Done becomes TRUE the read data is shown in .sxParameter[1].sdValue

**Supported Services:**

| Request ID | Description |
|---|---|
| 0x01 | Request Parameter, reads the RAM value of the parameter |
| 0x02 | Change parameter, changes the RAM value of the parameter |
| 0x41 | Read ROM value of parameter, only valid with value attribute |
| 0x42 | Write ROM value of parameter, only valid with the value attribute |

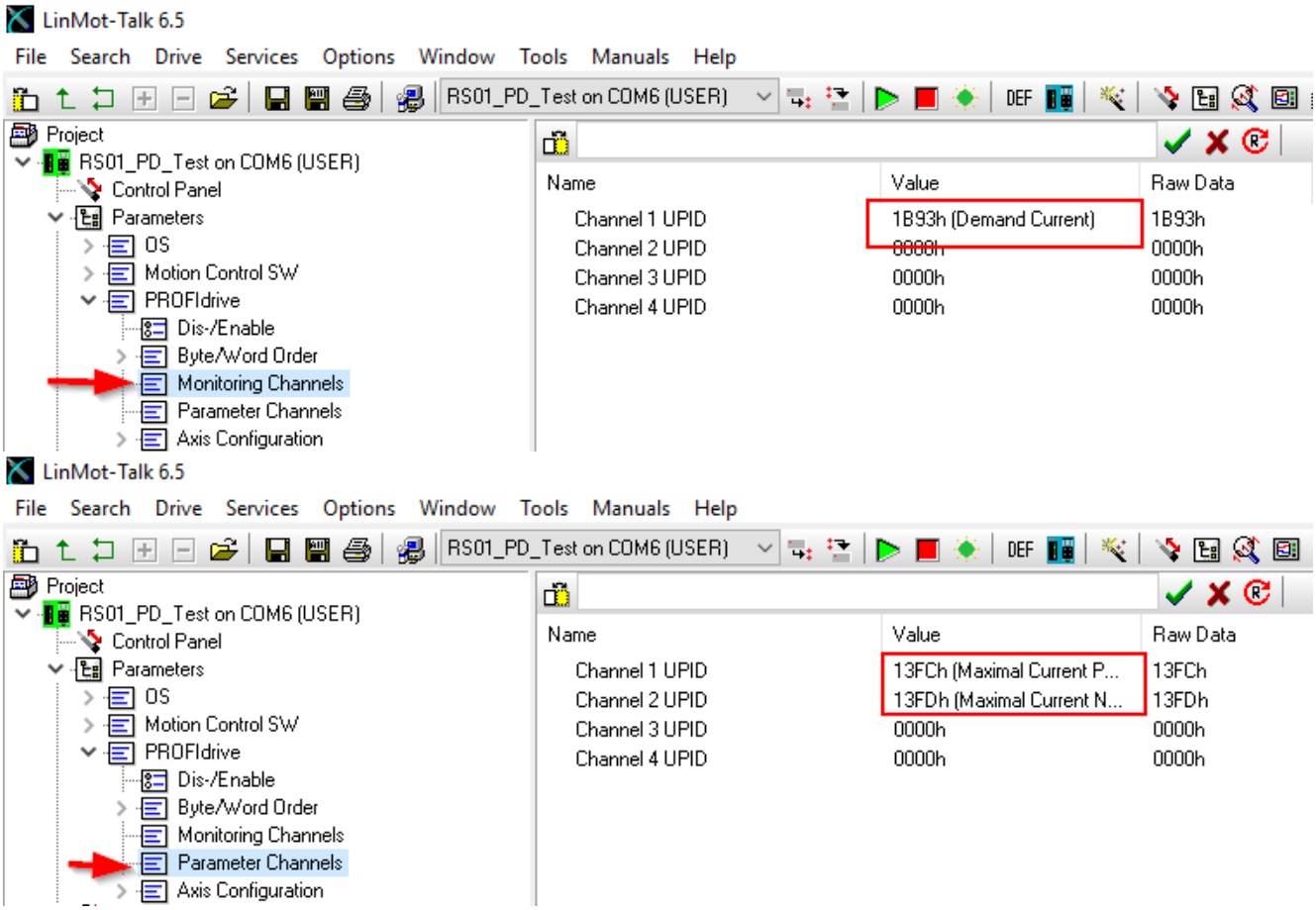
## 10.6 Add Additional Parameters / Variables to the Real-Time Channel

It is possible to add additional parameters or variables of the drive to the real-time channel to allow read and write access (RAM). There are 4 channels from the Drive to the PLC (32Bit each, Mon Channel 1-4) and 4 channels from the PLC to the Drive (32Bit each, Par Channel 1-4) available.

### 10.6.1 Drive Configuration

In LinMot-Talk the UPID (parameter/variable address) is entered in the particular channel.

As example in Monitoring Channel 1 the UPID *1B93h (Demand Current)* and in Parameter Channel 1 & 2 the UPID *13FCh (Maximal Current Positive)* and *13FDh (Maximal Current Negative)* are entered.

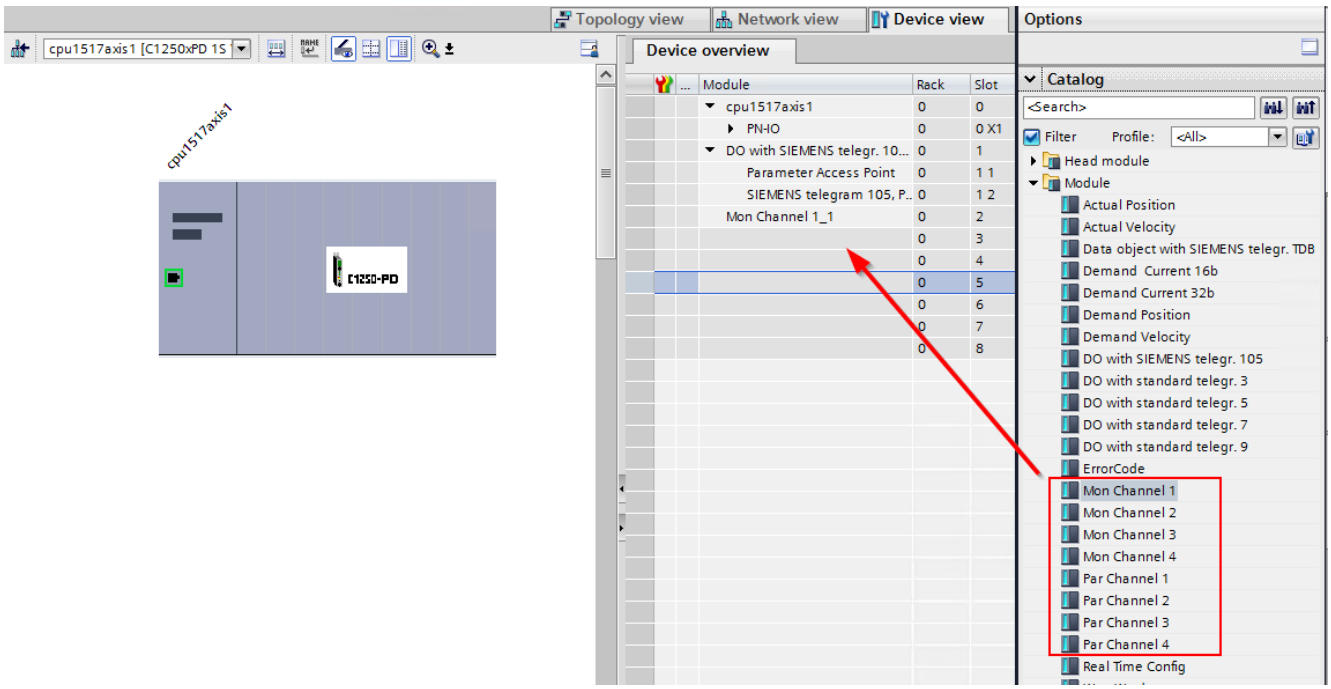



**Attention:**
The above used parameters are thought to be an example only and should not be used if SIEMENS telegram 105 is used.

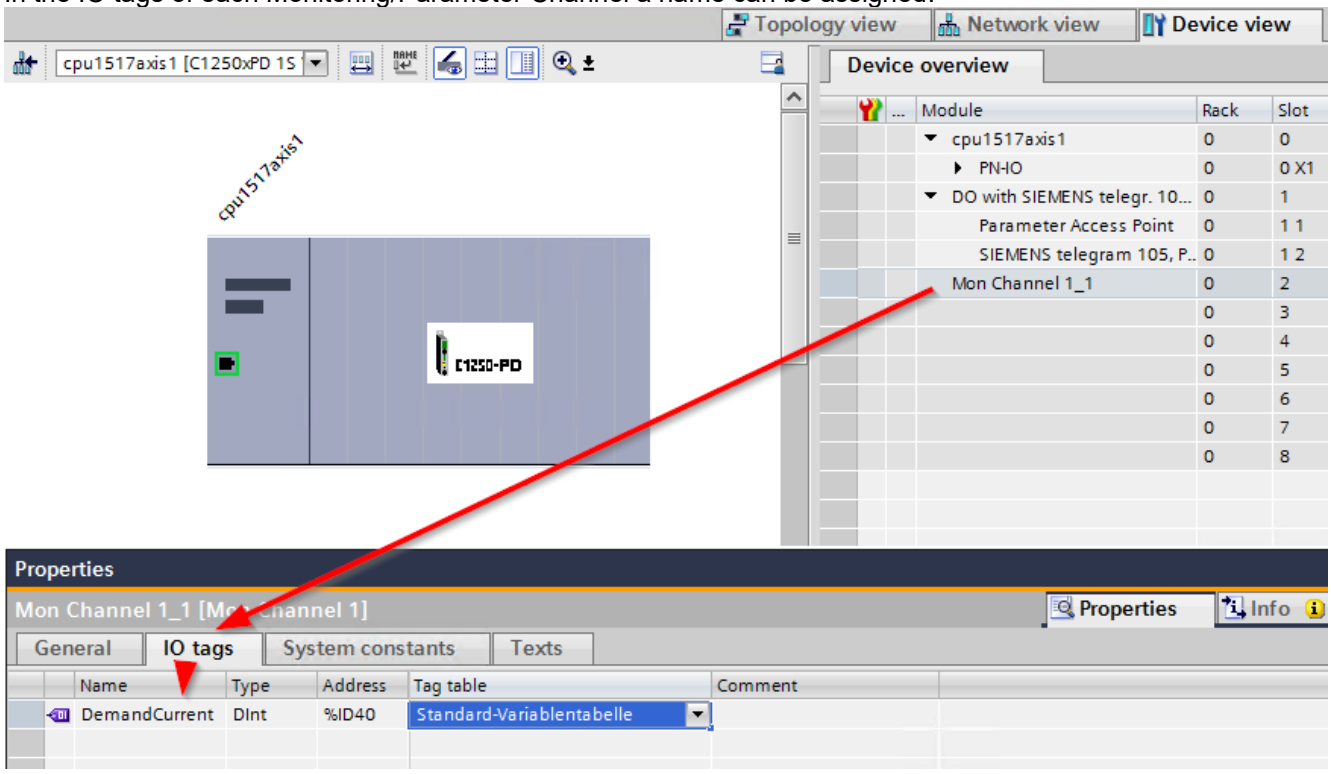
If the parameters in Parameter Channel 1 & 2 have a value of 0 the motor cannot move as they limit the maximal allowed positive and negative motor current in the drive.

### 10.6.2 PLC Configuration

In the HW Config drag&drop the desired modules from the catalogue to the slots of the drive:




In the IO tags of each Monitoring/Parameter Channel a name can be assigned:




**Note:**
On SIEMENS PLCs, the double words are transmitted in big endian organization. Therefore, it is required to swap the high and the low word.
This can be done by using the function ROR().
E.g. *DemandCurrent:=**ROR**(DemandCurrent,16);*

## 10.7 Telegram 750 / TDB

It is possible to add telegram 750 for additional torque/force functionality using a data block.

### 10.7.1 LinMot MI Drives

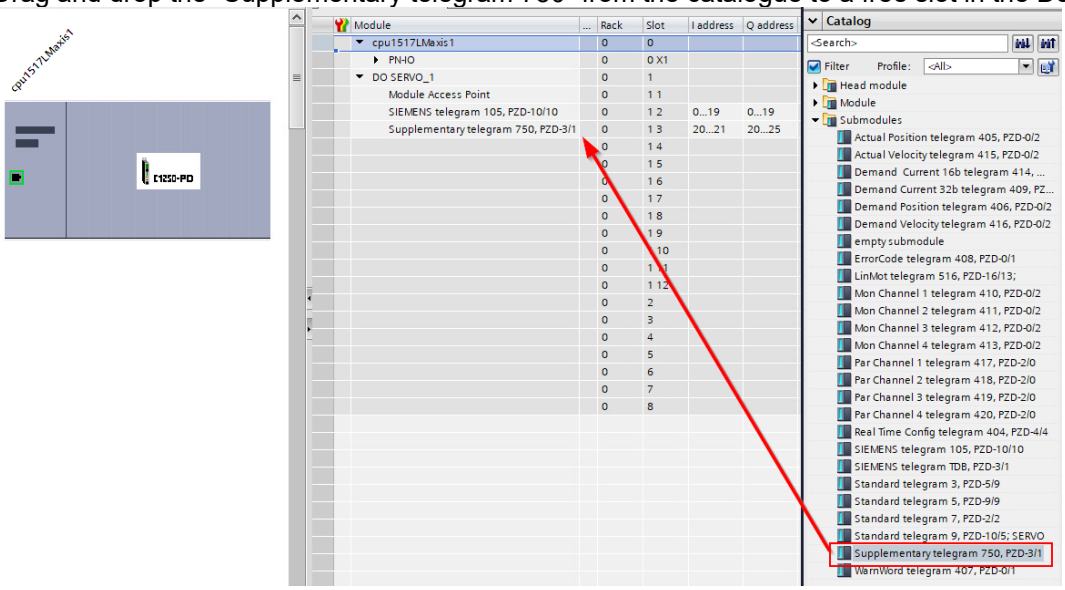
Using LinMot MI drives the telegram 750 can be / is directly connected in the TO configuration.

Telegram 750 is a supplementary telegram for torque/force control containing the following data:

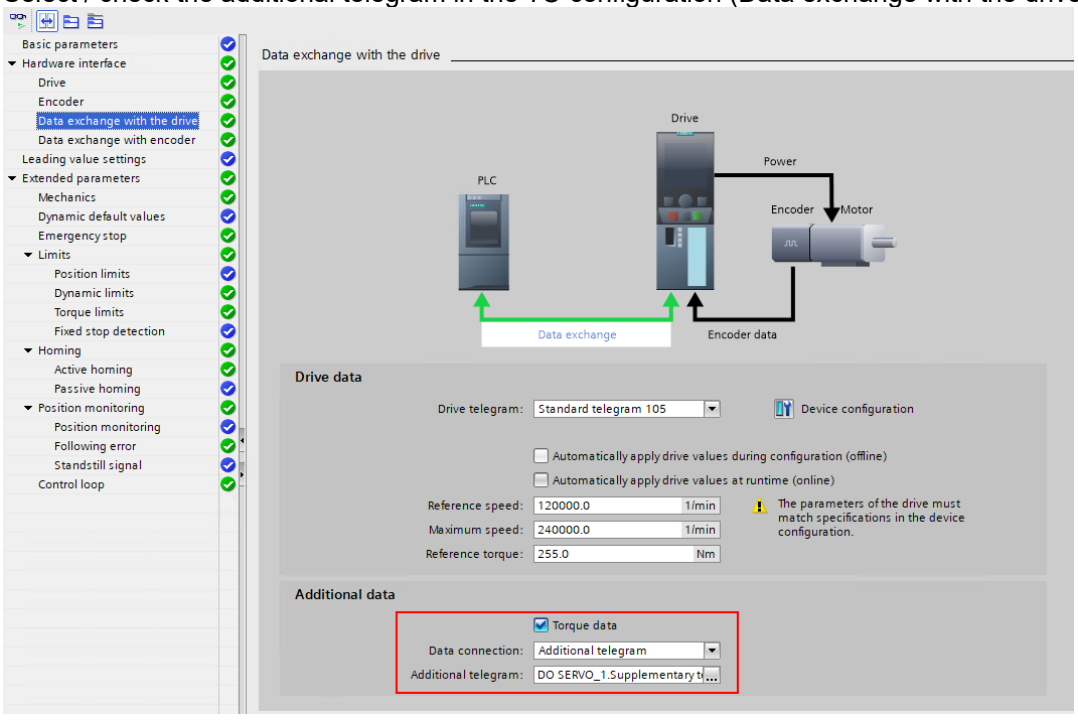
- The PLC sends the additive torque/force (M_ADD) as well as the positive and negative torque/force limit (M_LIMIT_POS, M_LIMIT_NEG) to the drive.
- The drive sends the actual torque/force (M_ACT) to the PLC.

| PZD01 | PZD02 | PZD03 |
|---|---|---|
| M_ADD | M_LIMIT_POS | M_LIMIT_NEG |
| M_ACT | --- | |

1. Drag and drop the “Supplementary telegram 750” from the catalogue to a free slot in the Device view:




2. Select / check the additional telegram in the TO configuration (Data exchange with the drive):



### 10.7.2 LinMot PD Drives

Using LinMot PD drives the telegram 750 can be used by following these instructions.

Telegram 750 is a supplementary telegram for torque/force control containing the following data:

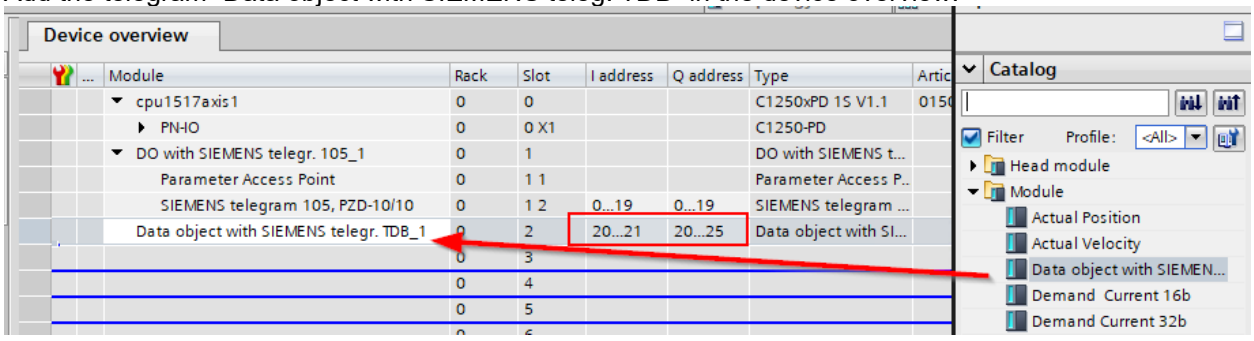
- The PLC sends the additive torque/force (M_ADD) as well as the positive and negative torque/force limit (M_LIMIT_POS, M_LIMIT_NEG) to the drive.
- The drive sends the actual torque/force (M_ACT) to the PLC.

| | |
|---|---|
| Axis1_TorqueData | PD_TEL750 |
| Input | PD_TEL750_IN |
| ActualTorque | Word |
| Output | PD_TEL750_OUT |
| AdditiveTorque | Word |
| UpperTorqueLimit | Word |
| LowerTorqueLimit | Word |

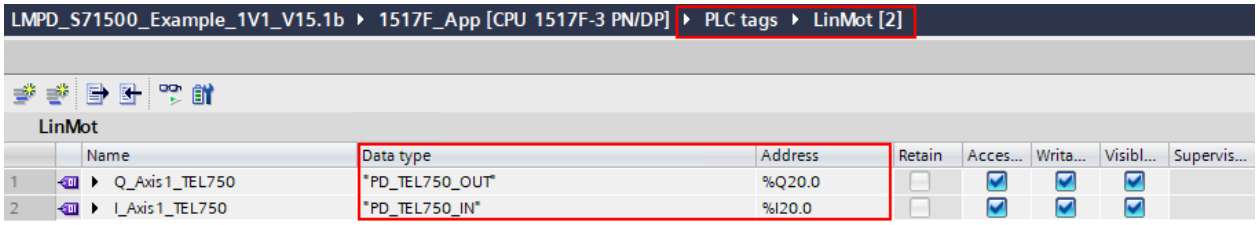
| PZD01 | PZD02 | PZD03 |
|---|---|---|
| M_ADD | M_LIMIT_POS | M_LIMIT_NEG |
| M_ACT | --- | |

**Note:**
The data types **PD_TEL750**, **PD_TEL750_IN** and **PD_TEL750_OUT** are automatically added to the project when you set them in the Data type column (they don't show up in the dropdown list).
Do not create your own data types.

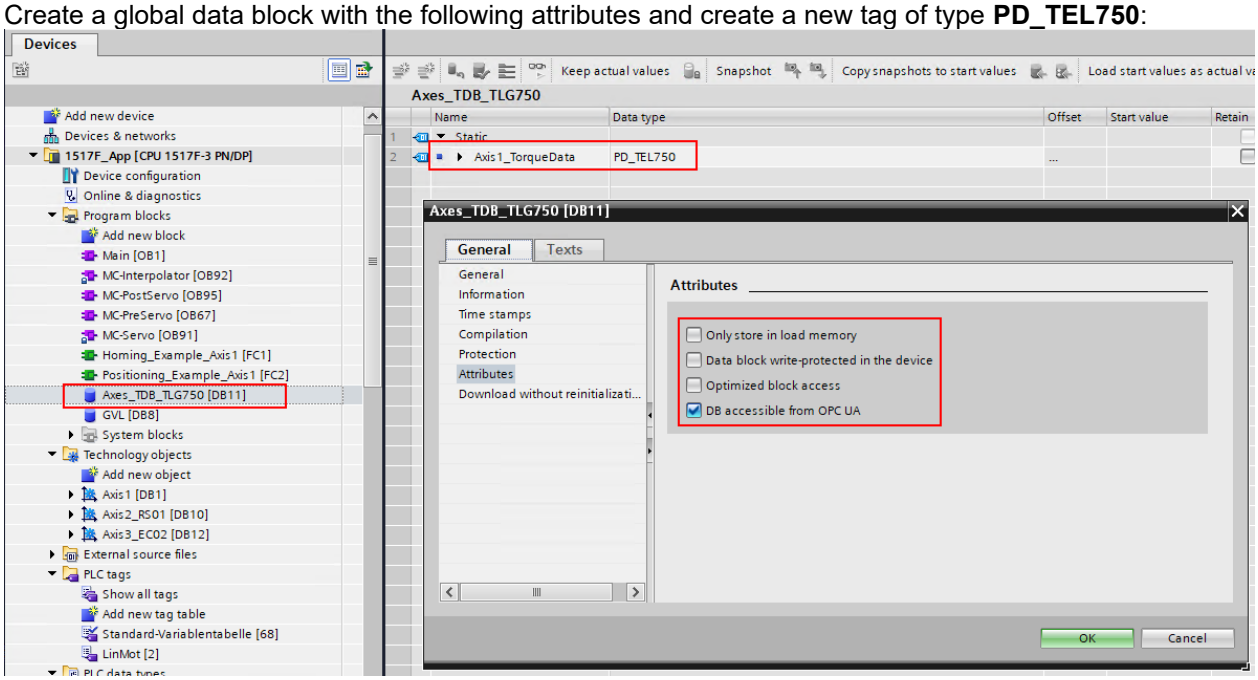
1. Add the telegram "Data object with SIEMENS teleg. TDB" in the device overview:



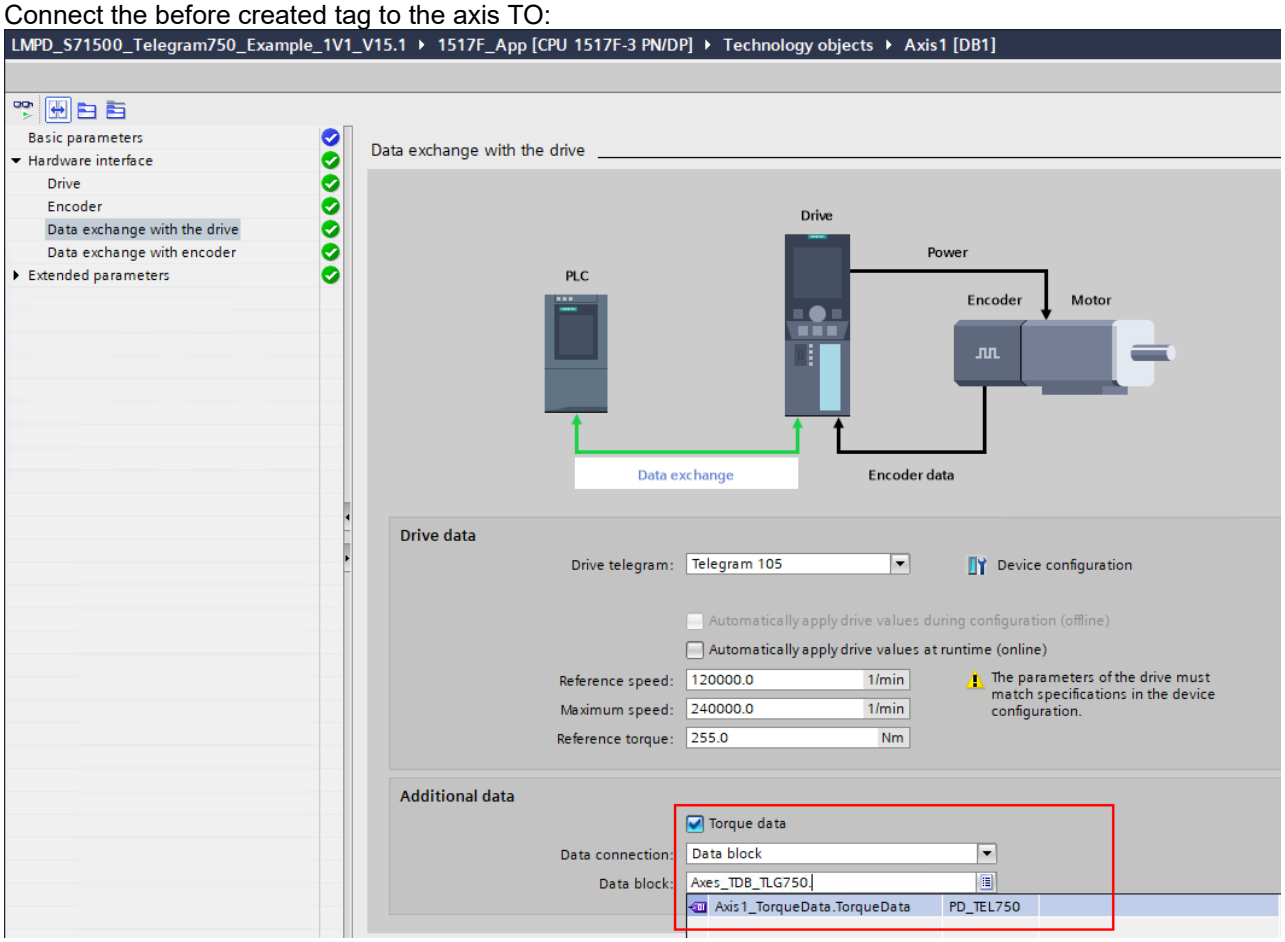

2. Add two tags in the PLC tags of type **PD_TEL750_IN** and **PD_TEL750_OUT** and assign the addresses from above:



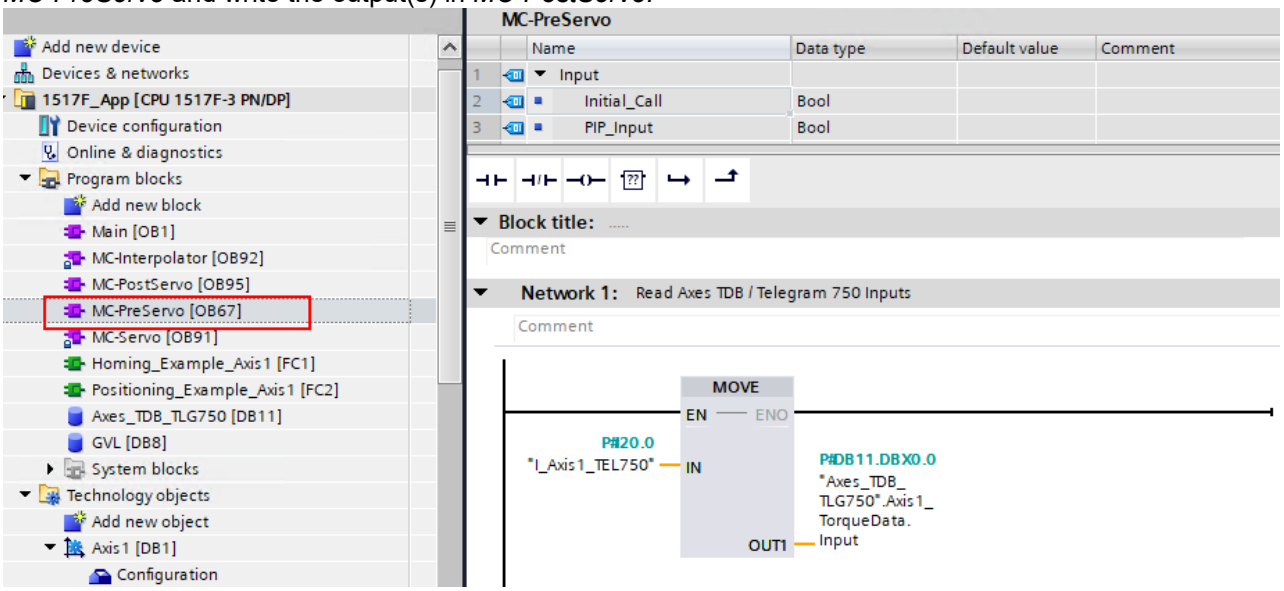
3. Create a global data block with the following attributes and create a new tag of type **PD_TEL750**:

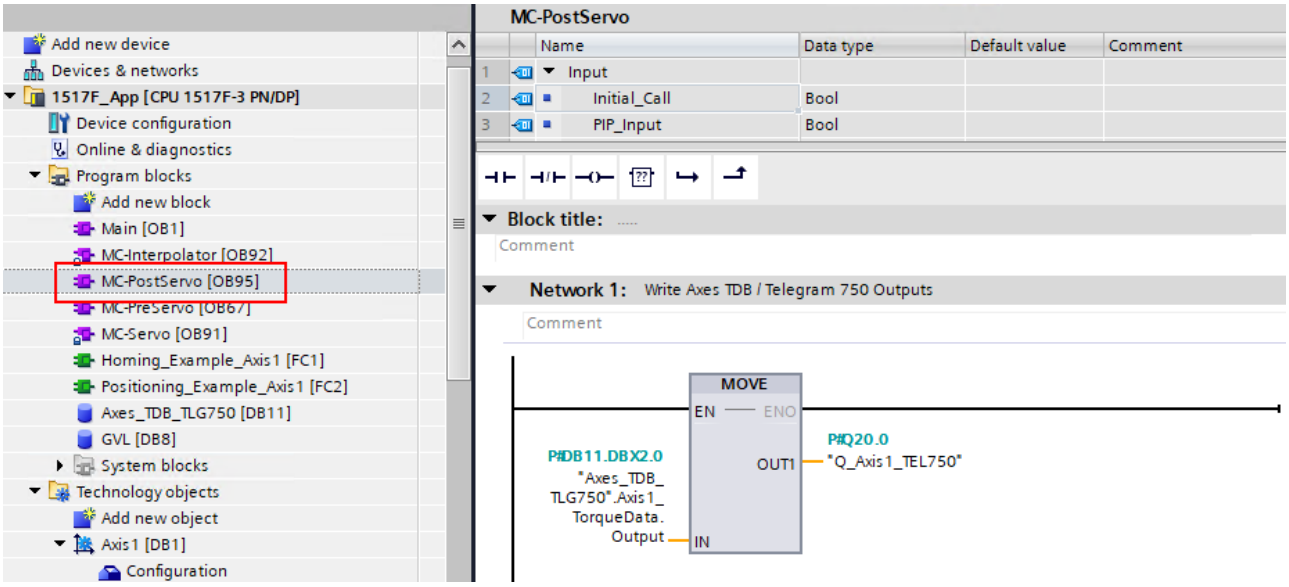



4. Connect the before created tag to the axis TO:



5. Add the organization blocks *MC-PreServo* and *MC-PostServo* to your program blocks. Read the input in *MC-PreServo* and write the output(s) in *MC-PostServo:*



## 10.8 Using the Closed Loop Force/Torque Control Technology Function

This chapter shows a way to switch from using Telegram 5/105 with a TO Axis to the closed loop force/torque control run mode inside the LinMot drive and vice versa.

This chapter does not cover the general setup of force/torque sensor and the according control loop parameters.

**Attention:**
To use closed loop force/torque control the technology function **TF-Force Control** (0150-2503) is required. Check the manual for more information: https://shop.linmot.com/E/product/0150-2503

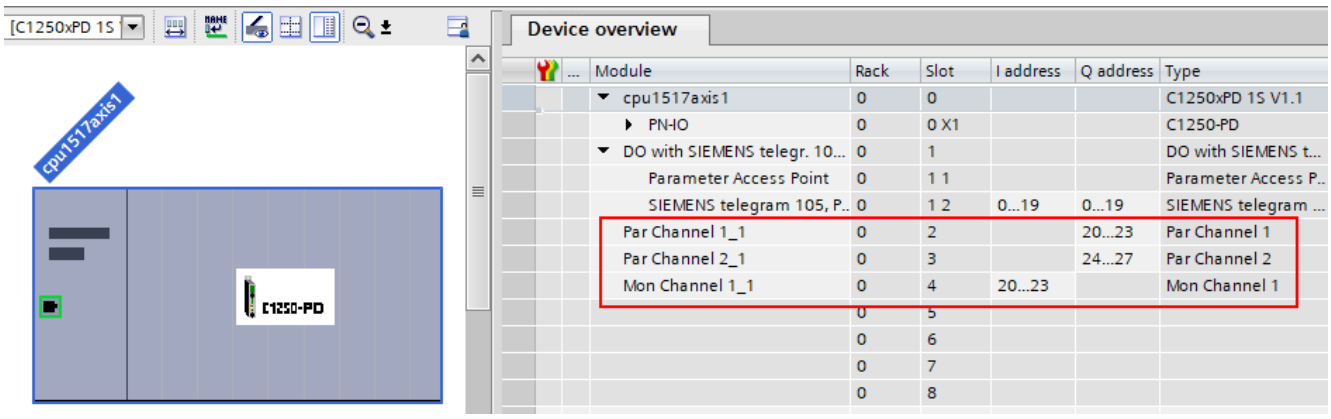
**Attention:**
Drive firmware **6.10 Build 20210521 or later** is required.

**Note:**
This example was done using the rotary part of a LinMot PR02 Linear Rotary Module with torque sensor. Nevertheless, the same principle is applicable for linear motors with force sensors.

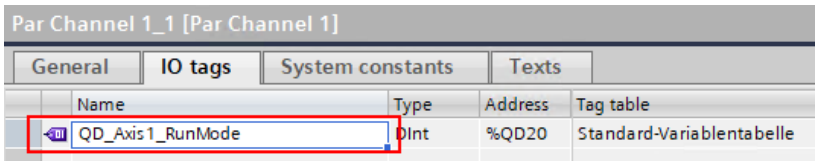
See Appendix II: Basic Closed Loop Torque/Force Control Loop Tuning

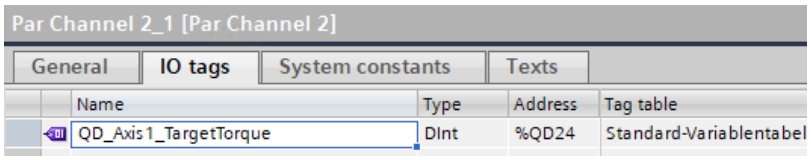
### 10.8.1 Add 1 Monitoring and 2 Parameter Channels / Drive Setup

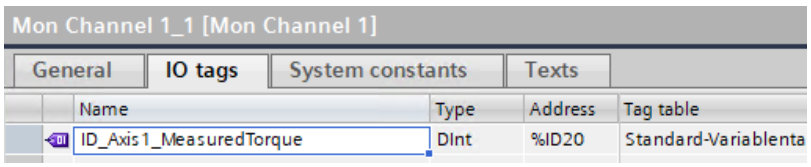
1. In the TIA Portal in the Device overview add the following modules from the catalogue:



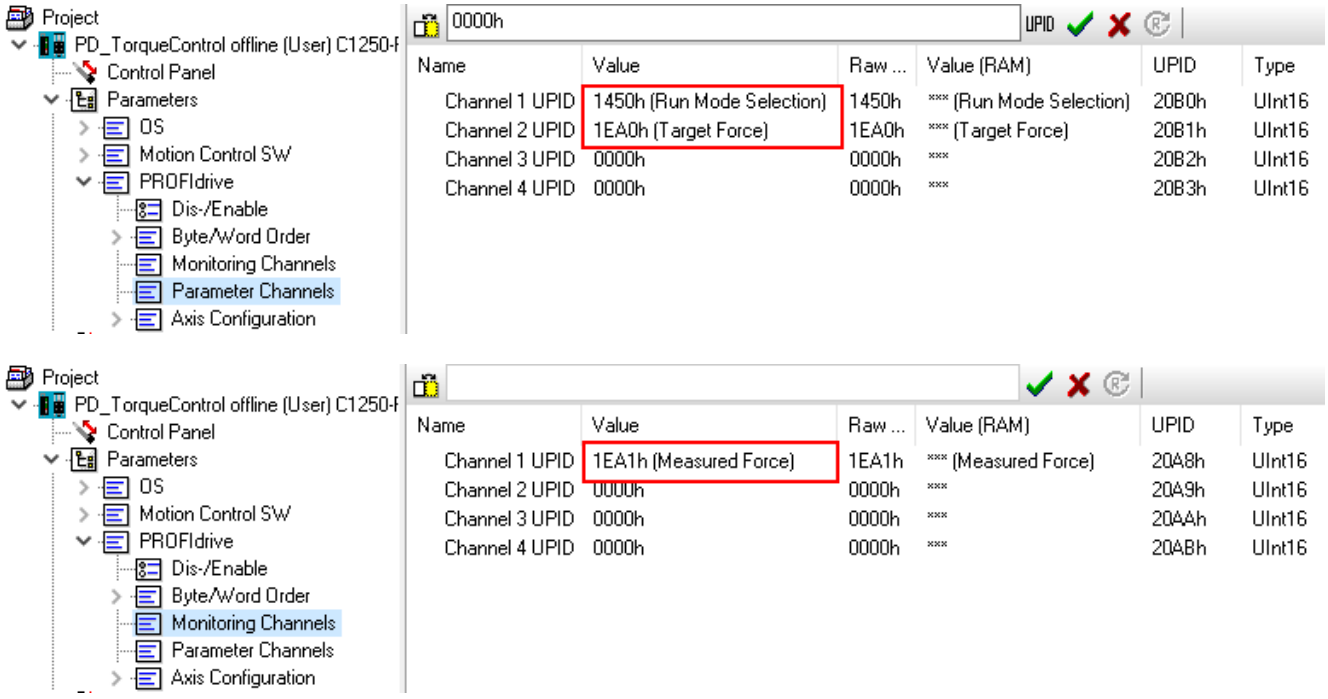

2. Assign a tag name in their properties to use them in your program:



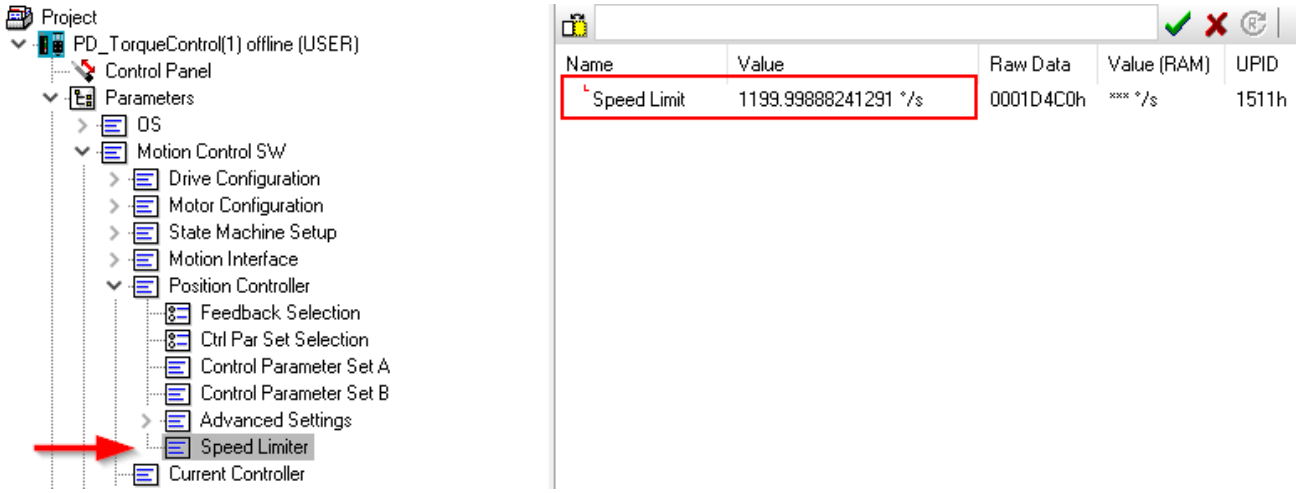



3. Setup the UPIDs for the channels withing the drive using the LinMot-Talk software:




4. Enable a Speed Limit to avoid too fast moves in closed loop force/torque control:



### 10.8.2 Scaling the Parameter and Monitoring Channels

The *Axis1_TargetTorque* and *Axis1_MeasuredTorque* tags can be scaled and rotated to Nm. This is done at the beginning (monitoring) and the end (parameter) of the program block (here in OB1).

**Network 1:** Read Axis 1 Monitoring Channels

Comment

```
1  // Torque scaling is 0.000572957795113082 Nm for the used rotary motor (PR02 with 360000 ticks per motor revolution)
2  "Axis1_MeasuredTorque" := INT_TO_REAL((ROR(IN := "ID_Axis1_MeasuredTorque", N:=16))) * 0.000572957795113082;
3
```

**Network 7:** Write Axis 1 Parameter Channels

Comment

```
1  // Torque scaling is 0.000572957795113082 Nm for the used rotary motor (PR02 with 360000 ticks per motor revolution)
2  "QD_Axis1_TargetTorque" := ROR(IN := REAL_TO_DINT("Axis1_TargetTorque" / 0.000572957795113082), N := 16);
3
4  // "Axis1_RunModeSelection" allows switching the drive run mode
5  // 1    = Motion Command Interface [normal operation with telegram 5/105
6  // 14   = Closed Loop Force Control [only visble in LinMot-Talk when logged in as "Service"]
7  "QD_Axis1_RunMode" := ROR(IN := "Axis1_RunModeSelection", N := 16);
```

### 10.8.3 Example Sequence

The following state machine is an example how the mode can be changed **from *controlled by the TO (technology object)*** to ***closed loop force/torque control inside the drive*** and back.
In this example the axis is controlling a rotary motor.

```
//************************************************************************
// LinMot Drive Based Torque Control Demo
// - Switch from Telegram 5/105 DSC to closed loop force control and back
//************************************************************************
//

CASE "GVL".iTCtrlState OF
    0:  // Idle
        // *********************************************
        IF "GVL".bTCtrlExecute THEN
            "Axis1_TargetTorque" := 0;
            "Axis1_RunModeSelection" := 1; // Positioning Mode (Drive Run Mode 1, Telegram 5 / 105)
            "GVL".iTCtrlState := "GVL".iTCtrlState + 1;
        END_IF;

    1:  // Move Velocity to disable position-controlled mode and position monitoring
        // *********************************************
        "MC_MOVEVELOCITY_Axis1".Execute := TRUE;
        "MC_MOVEVELOCITY_Axis1".Velocity := 120;
        "MC_MOVEVELOCITY_Axis1".PositionControlled := FALSE; //Important: Disable position
        controlled. Otherwise, there will be jumps when switching back

        IF "MC_MOVEVELOCITY_Axis1".InVelocity AND "Axis1_MeasuredTorque" > 0.15 THEN // Check actual
         torque to switch
            "MC_MOVEVELOCITY_Axis1".Execute := FALSE;

            // Change to drive based Torque Control and set target torque
            // *********************************************
            "Axis1_TargetTorque" := 0.2; // Target Torque 0.2 Nm
            "Axis1_RunModeSelection" := 14; // Torque Control Mode (Drive Run Mode 14)

            "GVL".iTCtrlState := "GVL".iTCtrlState + 1;
        END_IF;

    2:  // Check target torque reached
        // *********************************************
        IF "Axis1_MeasuredTorque" >= "Axis1_TargetTorque" - 0.02
           // AND ABS(IN := "Axis1_RS02".ActualSpeed) < 2
        THEN
            // Halt axis
            // *********************************************
            "MC_HALT_Axis1".Execute := TRUE;
            "GVL".iTCtrlState := "GVL".iTCtrlState + 1;
        END_IF;

    3:  // Check axis halted
        // *********************************************
        IF "MC_HALT_Axis1".Done THEN
            "MC_HALT_Axis1".Execute := FALSE;

            // Reset axis to position control mode (Telegram 5/105)
            // *********************************************
            "Axis1_RunModeSelection" := 1; // Positioning Mode (Drive Run Mode 1, Telegram 5 / 105)

            // Decrement Actual Position a little
            // *********************************************
            "MC_MOVEABSOLUTE_Axis1".Position := "Axis1_RS02".ActualPosition - 0.1; // Increment
            Actual Position
            "MC_MOVEABSOLUTE_Axis1".Velocity := 3600; //
            "MC_MOVEABSOLUTE_Axis1".Acceleration := 72000; //
            "MC_MOVEABSOLUTE_Axis1".Deceleration := 72000; //
            "MC_MOVEABSOLUTE_Axis1".Direction := 3; // Shortest way!!!
            "MC_MOVEABSOLUTE_Axis1".Execute := TRUE;

            "GVL".iTCtrlState := "GVL".iTCtrlState + 1;
        END_IF;
```

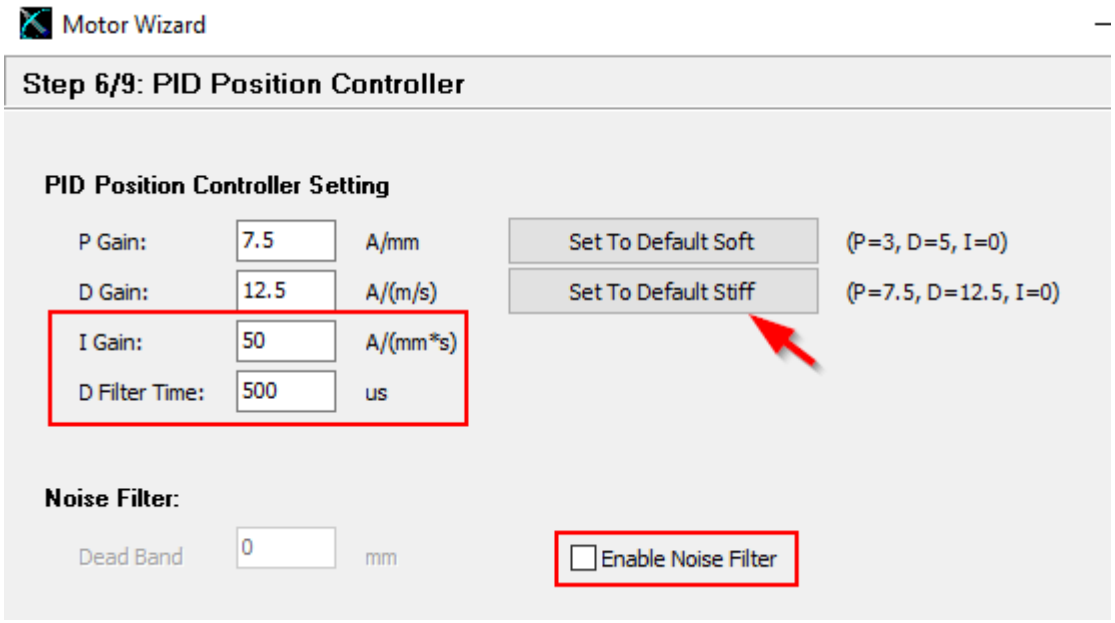
```
    4:  // Wait move absolute done
        // ********************************************
        IF "MC_MOVEABSOLUTE_Axis1".Done THEN
            // Clean up
            // ********************************************
            "MC_MOVEABSOLUTE_Axis1".Execute := FALSE;
            "MC_MOVEVELOCITY_Axis1".Execute := FALSE;
            "MC_HALT_Axis1".Execute := FALSE;

            "Axis1_TargetTorque" := 0;
            "Axis1_RunModeSelection" := 1; // Positioning Mode (Drive Run Mode 1, Telegram 5 / 105)

            "GVL".bTCtrlExecute := FALSE;

            "GVL".iTCtrlState := 0; // Cycle completed
        END_IF;


    100:// Reset
        // ********************************************
        "MC_MOVEABSOLUTE_Axis1".Execute := FALSE;
        "MC_MOVEVELOCITY_Axis1".Execute := FALSE;
        "MC_HALT_Axis1".Execute := FALSE;

        "Axis1_TargetTorque" := 0;
        "Axis1_RunModeSelection" := 1; // Positioning Mode (Drive Run Mode 1, Telegram 5 / 105)

        "GVL".bTCtrlExecute := FALSE;

        "GVL".iTCtrlState := 0;
    ELSE
        ;
END_CASE;
```

## 10.9 Tuning the Control Loops

### 10.9.1 Drive Control Loop

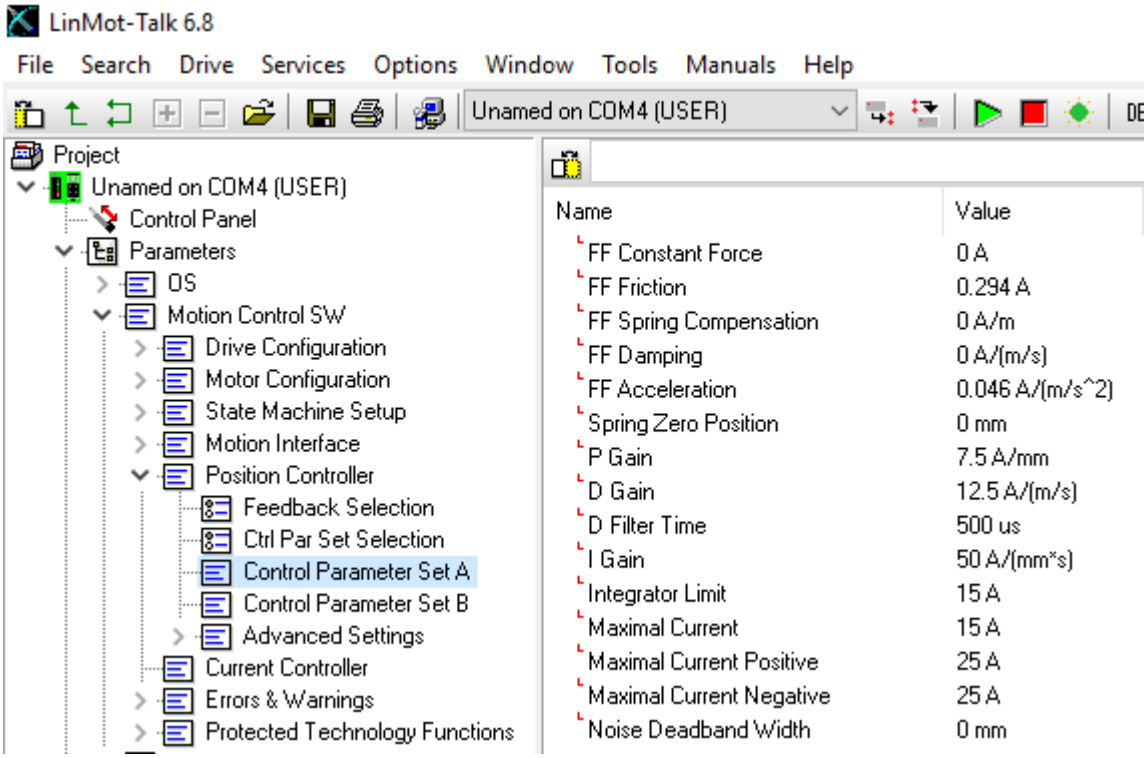
To start tuning use the **default stiff** settings in the Motor Wizard within LinMot-Talk (motor depending).

In case the motor becomes too loud and vibrates reduce the P and D Gains.




- Make sure the “Enable Noise Filter” is unchecked.
- Set the D Filter Time to 250-500us.
- Set the I Gain to 20-50 A/(mm*s).

After finishing the motor wizard and restarting the drive, the control loop parameters can be tuned in the *Control Parameter Set A* (mainly P, D and I Gains):



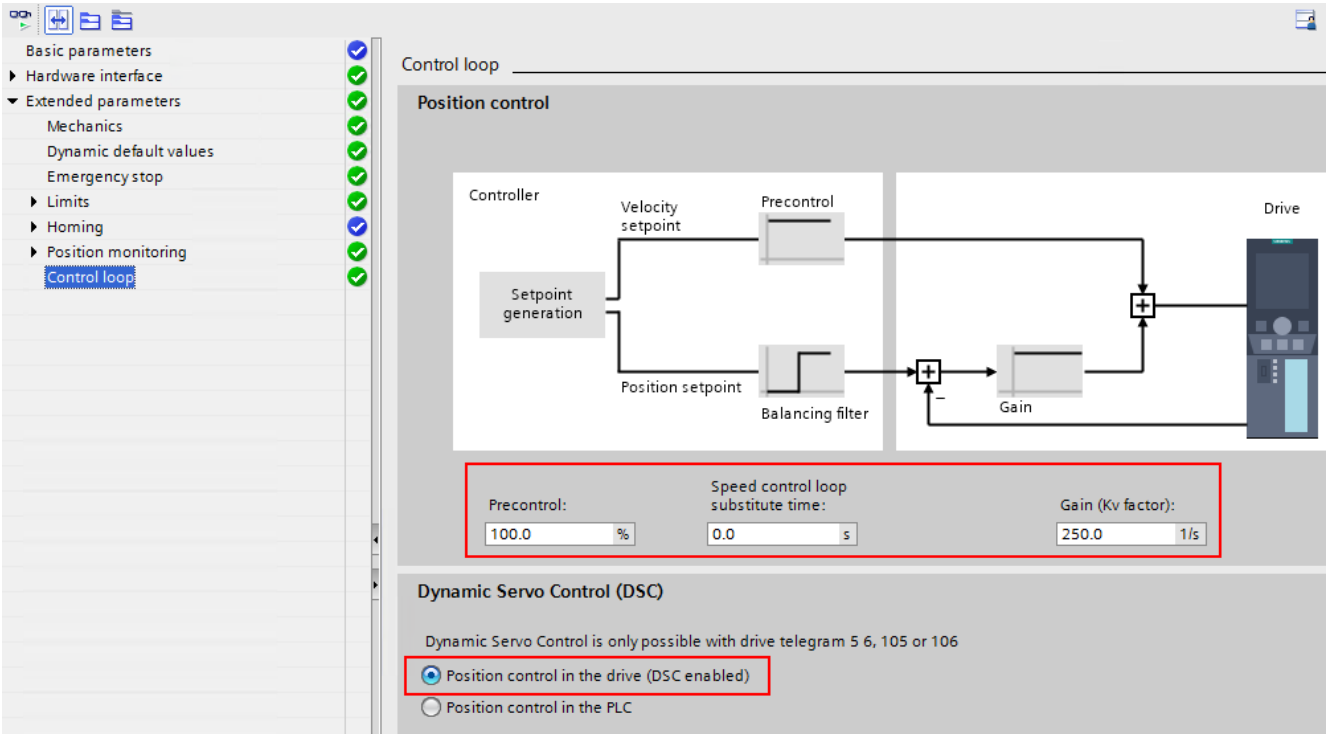

**Up to firmware 6.6 Build 20170410 the I Gain** in the Control Parameters Set A of the LinMot drive must be set to 0 (zero).

**From firmware 6.6 Build 20170522 the I-Gain must be** set to an appropriate value to exactly reach the end position.

Also check chapter Appendix I: Basic Position Control Loop Tuning

### 10.9.2 PLC Control Loop

Within the technology object configuration set the Gain (Kv factor) to 100.
From this value increase until you reach the required stiffness and accuracy.



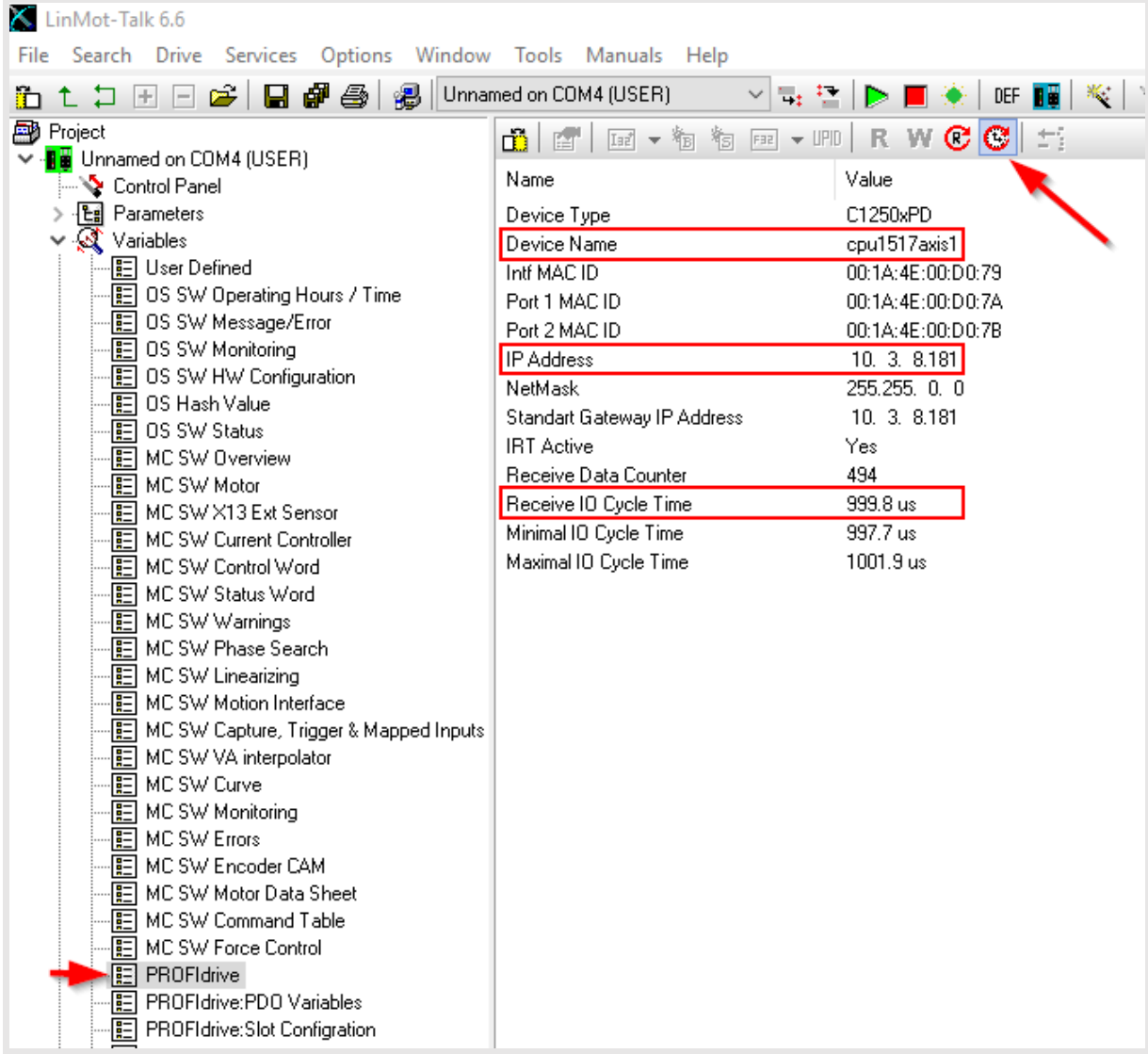

Make sure DSC is enabled.

Now you are ready to enable the motor and start some movements to optimize the control parameters in both the drive and the PLC.

## 10.10 Check with LinMot-Talk if the Fieldbus is running

LinMot-Talk shows the status of the fieldbus. Open the variables *PROFIdrive* and check if the Node Address / IP Address is correct.

Check fieldbus status:



# 11 Drive Profile: Beckhoff TwinCAT 2/3, CoE DS402

## 11.1 Overview

This chapter shows how a LinMot drive with *DS402 (CoE) / CiA402* interface (e.g., C1250-DS-XC-1S) can be integrated and setup in a Beckhoff TwinCAT environment.
For this example, TwinCAT 3 is used. The steps are generally similar for TwinCAT 2.

**Attention:**
Drive firmware **6.10 Build 20210521 or later** is required to use all the functionality shown in this chapter.

**Download:**
Example projects can be downloaded from:
http://download.linmot.com/plc_lib/examples/Beckhoff_CoE/ (named *LM_CoE_Demo_TCx_...*)

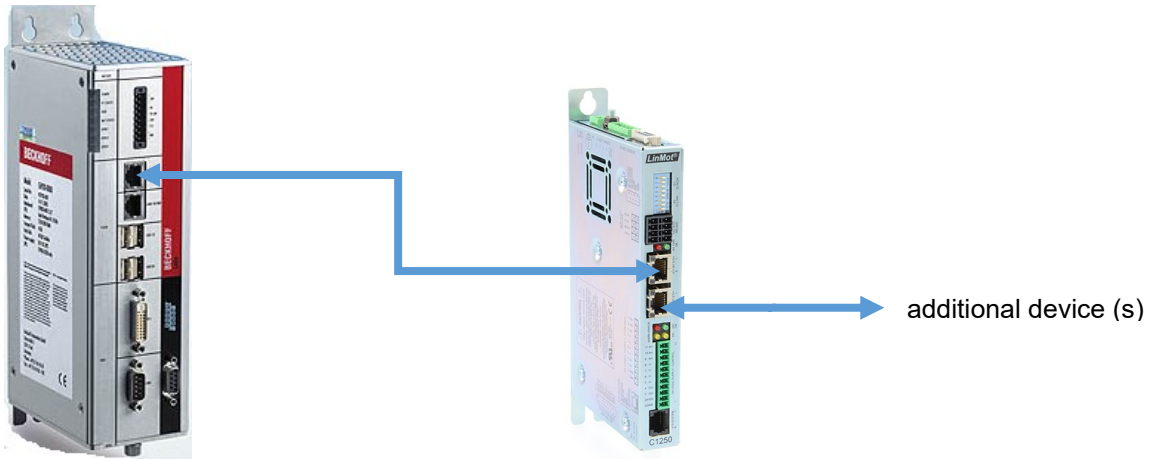



Image Source: http://www.beckhoff.com/

EtherCAT is the real-time Ethernet network originally developed by Beckhoff. The LinMot acts as Slave in this network and is implemented with the standard ASIC ET1100 from Beckhoff.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

**Note:**
In this chapter the following Modes of Operation of the DS402 / CiA402 profile are used:
6 = homing mode
8 = cyclic synchronous position (csp)
10= cyclic synchronous torque (cst)

The modes are switched by setting object 6060h (Mode of operation) to the according value.
The drive shows the active mode in object 6061h (Mode of operation display).

## 11.2 Configuration and Connection of the LinMot Drive

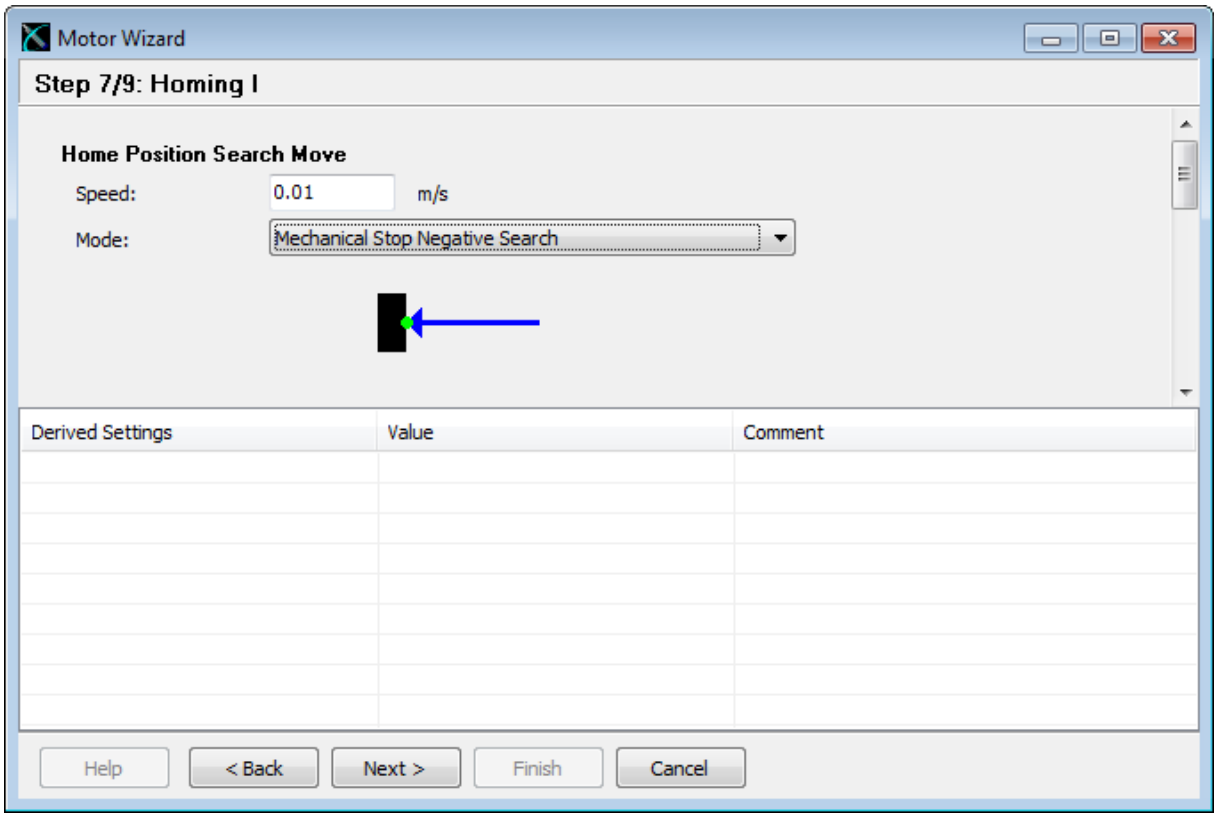
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 11.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select a homing mode in the Motor Wizard when the drive-based homing should be used as shown in chapter 11.5.2 Homing using the drive-controlled homing (Mode of Operation 6) (e.g., *Mechanical Stop Negative Search*):




Otherwise select “No Drive Homing” as Mode.

See Appendix I: Basic Position Control Loop Tuning

For ***Drive Based Homing*** please refer to chapter ***11.5.2***

### 11.2.2 XML File

Install the XML file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML\
- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML\ (-MI drives)

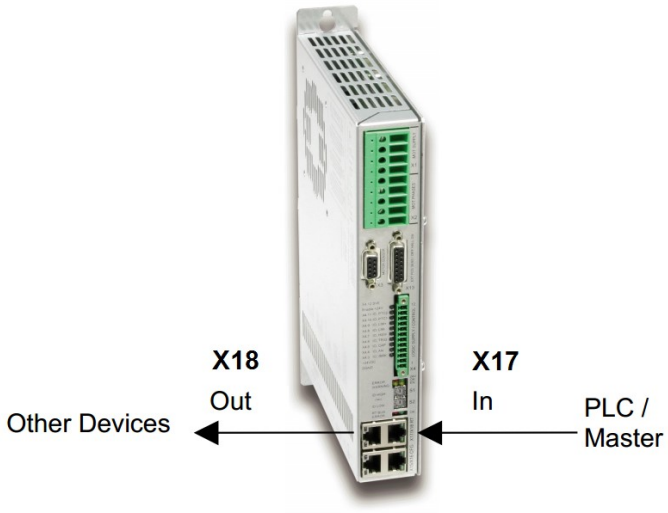
**XML file names:**

| | |
|---|---|
| NTIL_CiA402_Servos_Vx_xrx.xml | LinMot DS drives SG6-7 |
| NTIL_CiA402_SG5_Servos_Vx_xrx.xml | LinMot DS drives SG5 |
| NTIL_CiA402_Servos_MI_Vx_xrx.xml | LinMot MI drives SG6 |

### 11.2.3 EtherCAT Connection

The drive is connected to the EtherCAT network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-DS-UC drive. On all other LinMot drives supporting EtherCAT CoE the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.

| X17 - X18 | RealTime Ethernet 10/100 Mbit/s |
|---|---|
|  | X17 RT ETH In (Beckhoff: EtherCAT IN, from PLC)<br>X18 RT ETH Out (Beckhoff: EtherCAT OUT, to next device) |
| RJ-45 | |



## 11.3 PLC Setup

### 11.3.1 EtherCAT Device Description File XML

The EtherCAT device description file (XML) is located by default in the LinMot-Talk installation path:
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML

Copy the required file to your local TwinCAT installation path:

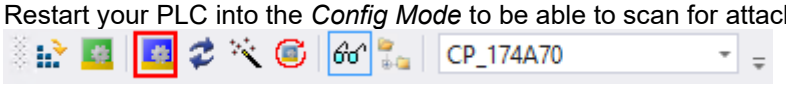
- C:\TwinCAT\Io\EtherCAT\ (for TwinCAT 2)
- C:\TwinCAT\3.1\Config\Io\EtherCAT\ (for TwinCAT 3)

**Note:** You may have to restart the developing environment to have the new XML file to be recognized.

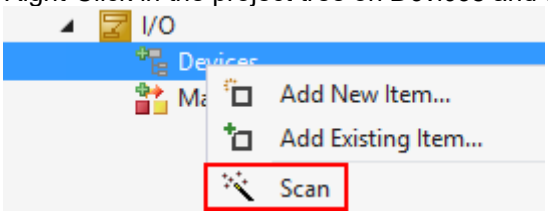
### 11.3.2 Scanning the EtherCAT for new Devices

Follow these steps to add the LinMot drive to your Beckhoff PLC.

1. Restart your PLC into the *Config Mode* to be able to scan for attached devices.

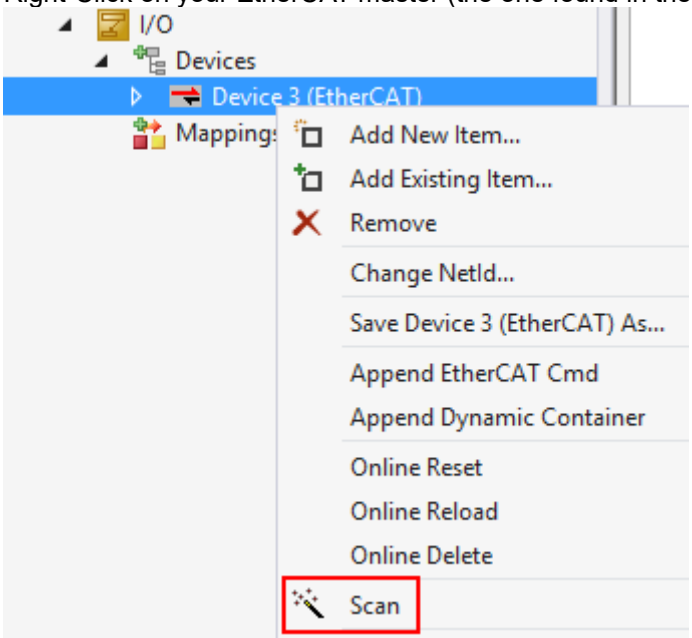



2. Right-Click in the project tree on *Devices* and select *Scan*

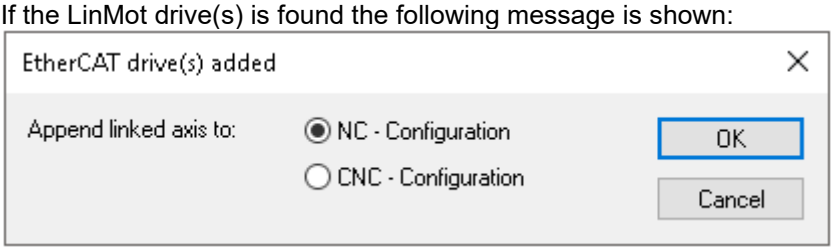



Select and insert the found EtherCAT master (e.g., *Device 3 (EtherCAT)*).
You can now directly scan for boxes. If you select yes, then skip the next step.

3. Right-Click on your EtherCAT master (the one found in the previous step) and select *Scan*

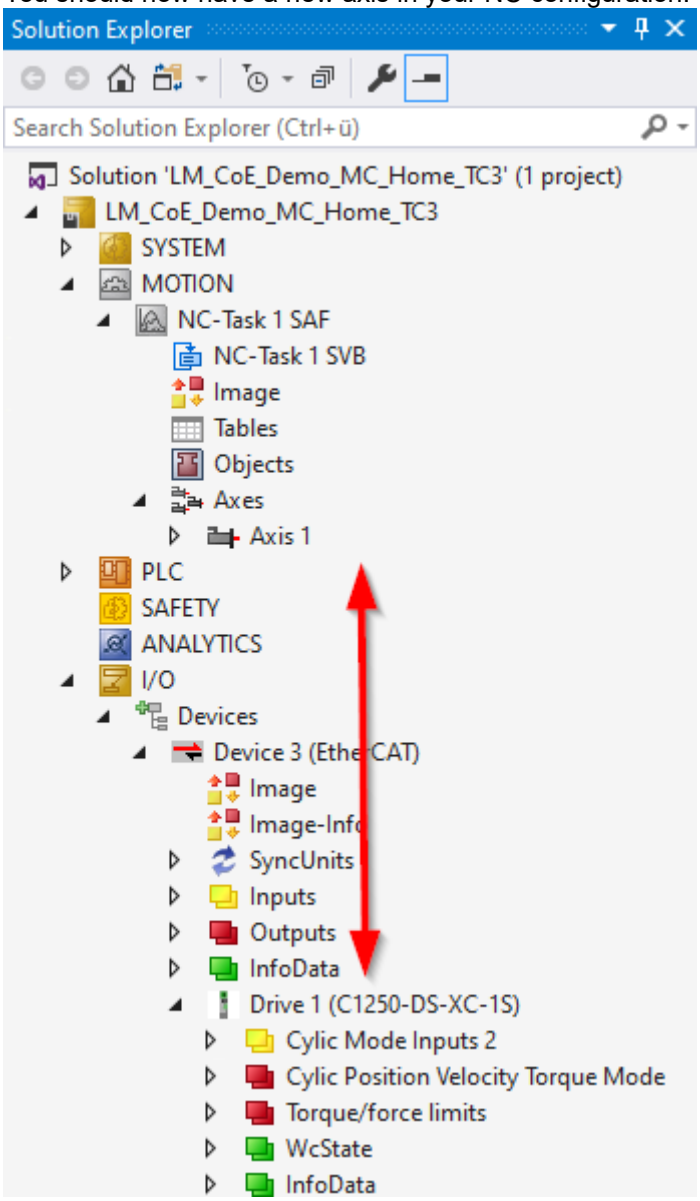


4. If the LinMot drive(s) is found the following message is shown:




Select OK

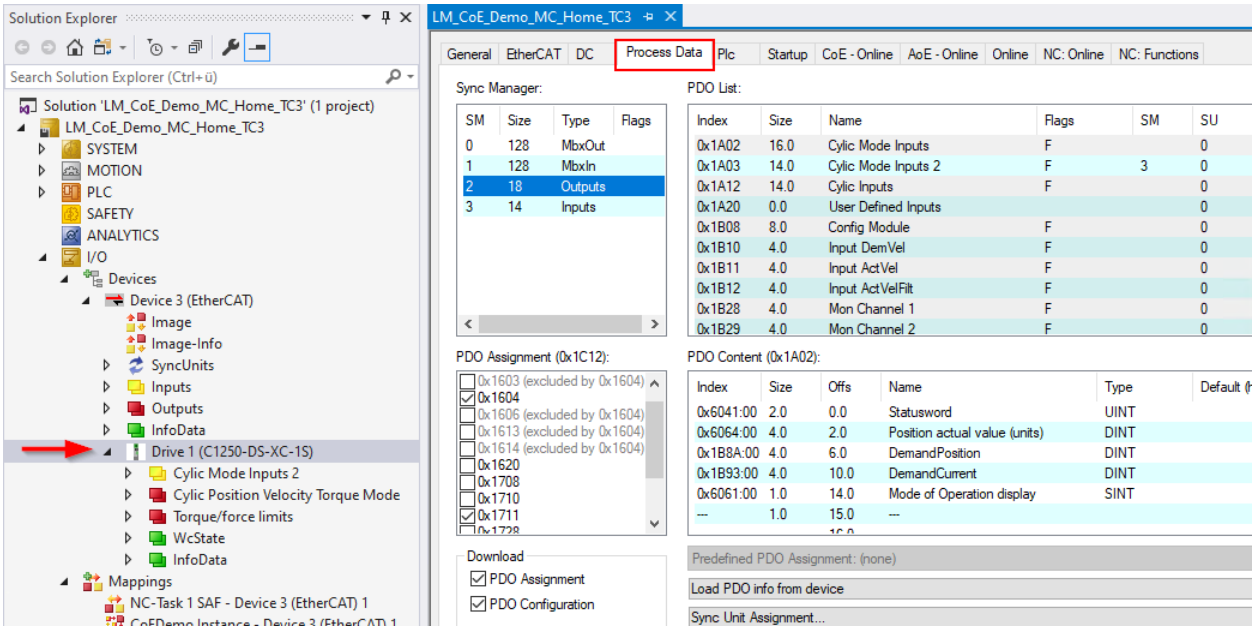
5. You should now have a new axis in your NC configuration:



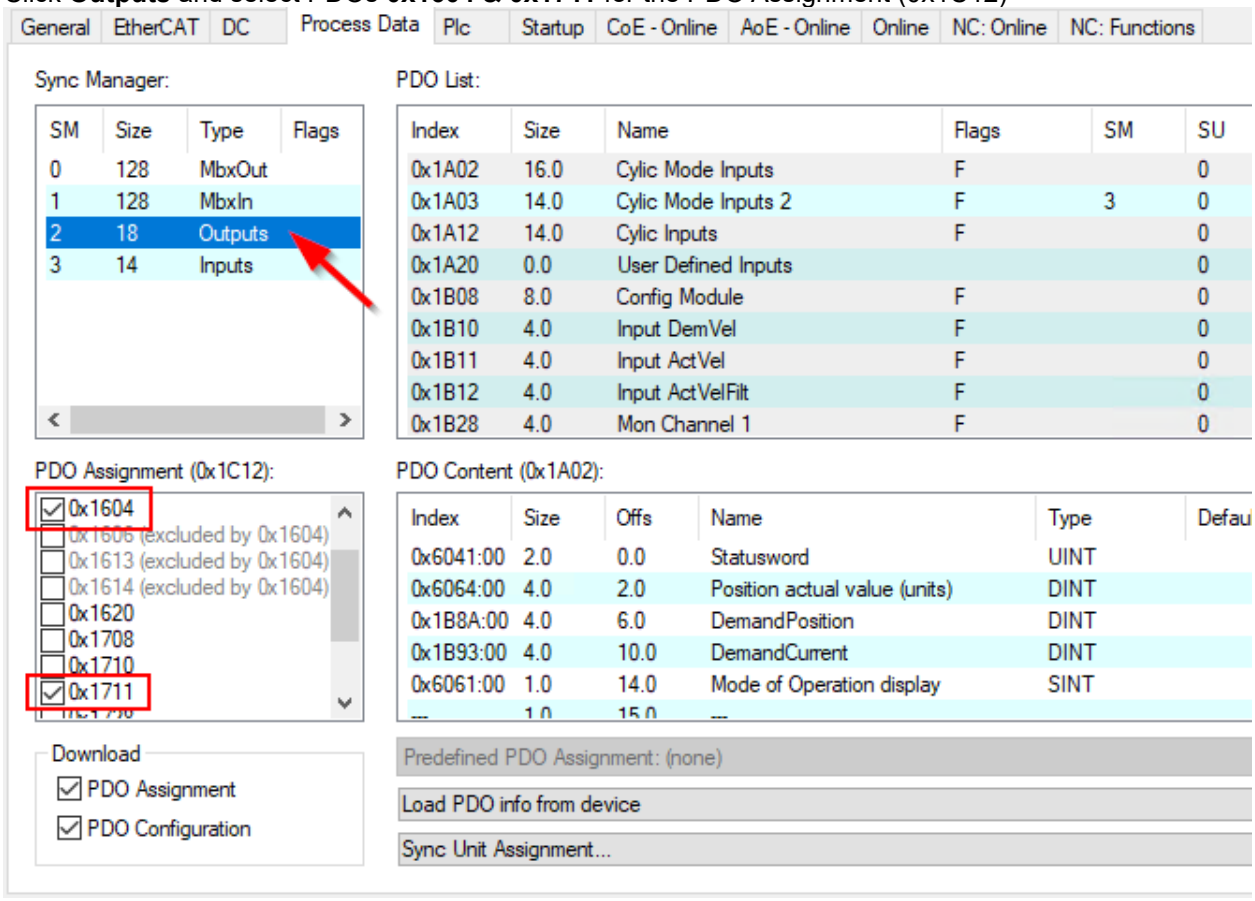

In the next chapter the configuration of the process data of the LinMot drive is shown

### 11.3.3 Setting up the Process Data of the LinMot Drive

1. Move to the tab *Process Data*

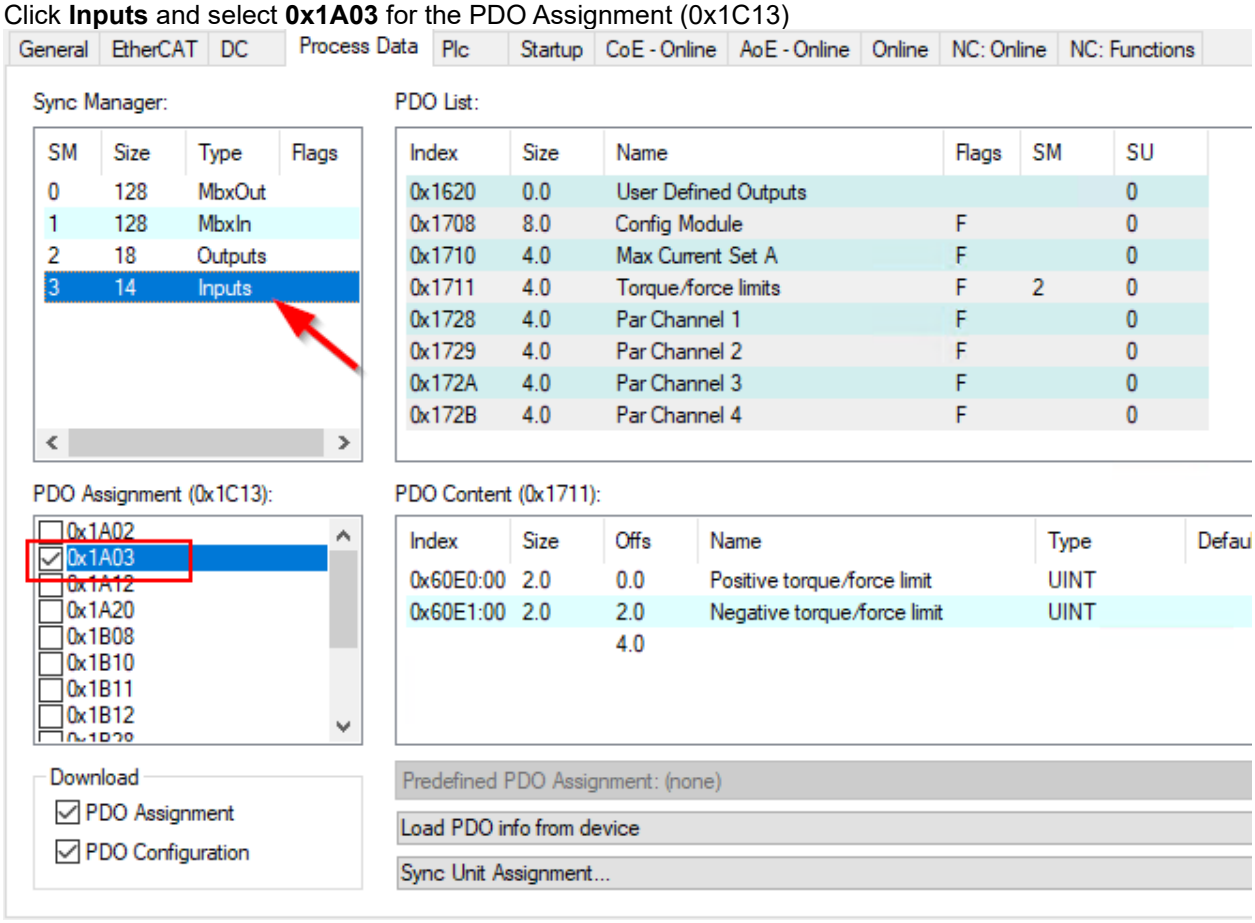



2. Click **Outputs** and select PDOs **0x1604 & 0x1711** for the PDO Assignment (0x1C12)




0x1604 = Cyclic Position Velocity Torque Mode
0x1711 = Torque/Force Limits (optional)

3. Click **Inputs** and select **0x1A03** for the PDO Assignment (0x1C13)




The mapping of drive and NC axis should look as follows:

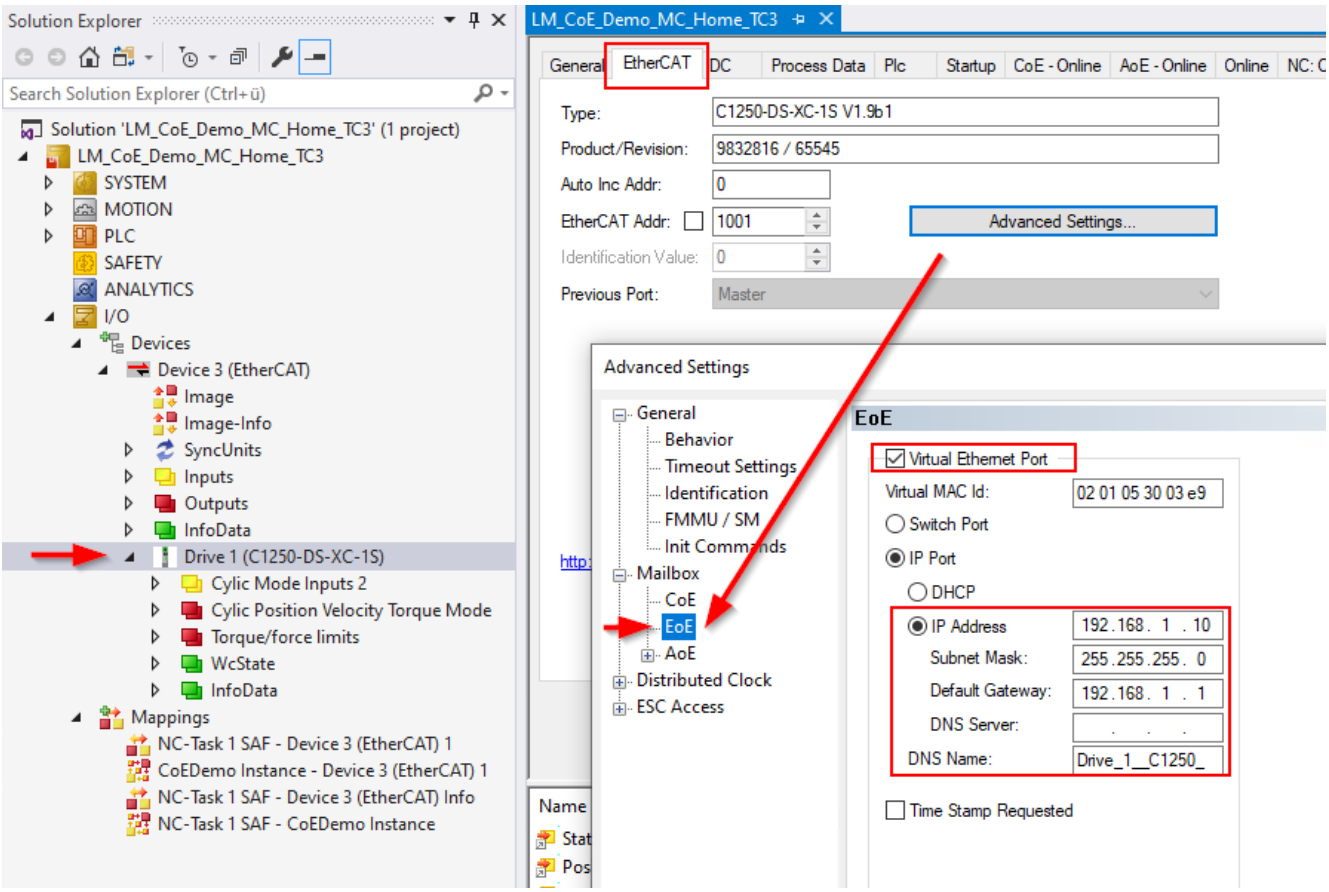
| Name | Type | Size | In/Out | Linked to |
|---|---|---|---|---|
| Statusword | UINT | 2.0 | Input | nState1, nState2 |
| Position actual value (units) | DINT | 4.0 | Input | nDataIn1 . In . Inputs . Enc . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Position demand value | DINT | 4.0 | Input | |
| Torque actual value | INT | 2.0 | Input | nDataIn3[0] . nDataIn3 . In . Inputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Mode of Operation display | SINT | 1.0 | Input | nState5 . In . Inputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| WcState | BIT | 0.1 | Input | nState4, nState4 |
| InputToggle | BIT | 0.1 | Input | nState4, nState4 |
| State | UINT | 2.0 | Input | |
| AdsAddr | AMSADDR | 8.0 | Input | |
| AoeNetId | AMSNETID | 6.0 | Input | |
| Chn0 | USINT | 1.0 | Input | |
| DcOutputShift | DINT | 4.0 | Input | nDcOutputTime . In . Inputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| DcInputShift | DINT | 4.0 | Input | nDcInputTime . In . Inputs . Enc . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Controlword | UINT | 2.0 | Output | nCtrl1, nCtrl2 |
| Target position | DINT | 4.0 | Output | nDataOut1 . Out . Outputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Target velocity | DINT | 4.0 | Output | nDataOut2 . Out . Outputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Target torque | INT | 2.0 | Output | nDataOut4[0] . nDataOut4 . Out . Outputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Mode of Operation | SINT | 1.0 | Output | nCtrl5 . Out . Outputs . Drive . Axis 1 . Axis 1 . Axes . NC-Task 1 SAF |
| Positive torque/force limit | UINT | 2.0 | Output | GVL.iAxisTorqueLimitPos[1] . GVL.iAxisTorqueLimitPos . PlcTask Outputs . CoEDemo Instance . CoEDemo |
| Negative torque/force limit | UINT | 2.0 | Output | GVL.iAxisTorqueLimitNeg[1] . GVL.iAxisTorqueLimitNeg . PlcTask Outputs . CoEDemo Instance . CoEDemo |

**Important:**
At this point the positive and negative force/torque limits must be mapped/linked as shown above. Otherwise, the motor current will be set to 0 when activating the project and it won't be possible to move.

### 11.3.4 Check EoE settings

On supported drives (C1250-..., C1450-… & E1450-...) please check the **EoE** (Ethernet over EtherCAT) settings.

LinMot drives do NOT support the DHCP mode yet. A fix IP must be assigned (select *IP Address*) or the EoE feature must be completely disabled (disable Virtual Ethernet Port).

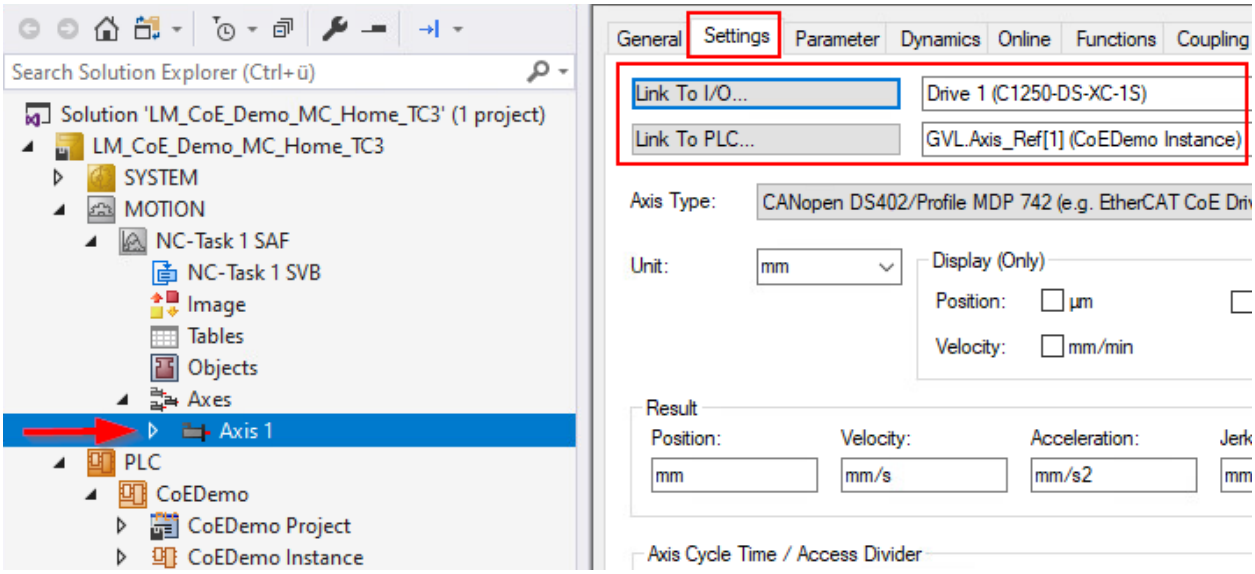



**Note:**
EoE can be used to login into the drive with LinMot-Talk directly over EtherCAT.
E.g., to change drive settings or for monitoring and tracing.

### 11.3.5 Setting up the NC Axis

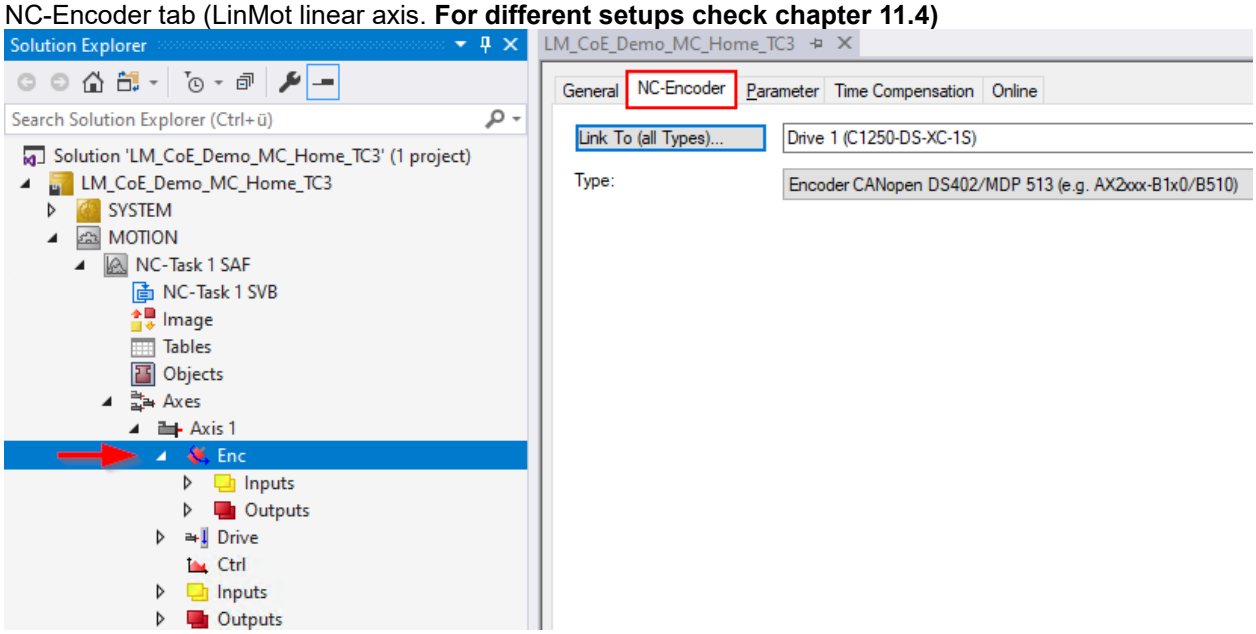
The following steps show how to basically setup the NC axis (red framed = mandatory).

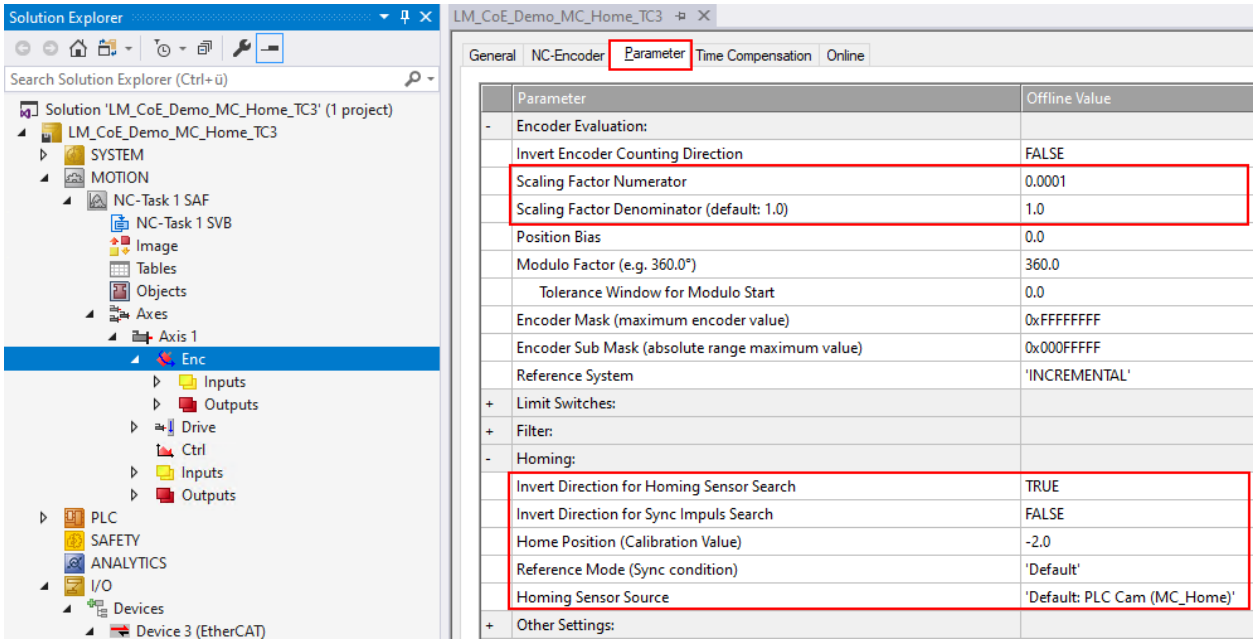
1. **Axis Drive and AXIS_REF:** Make sure the correct drive and AXIS_REF are selected under “Settings” > “Link To I/O…” and “Link To PLC..”



2. **Axis Encoder:** Setup the axis encoder as shown. Make sure the correct drive is selected “Link To (all….).

NC-Encoder tab (LinMot linear axis. **For different setups check chapter 11.4)**

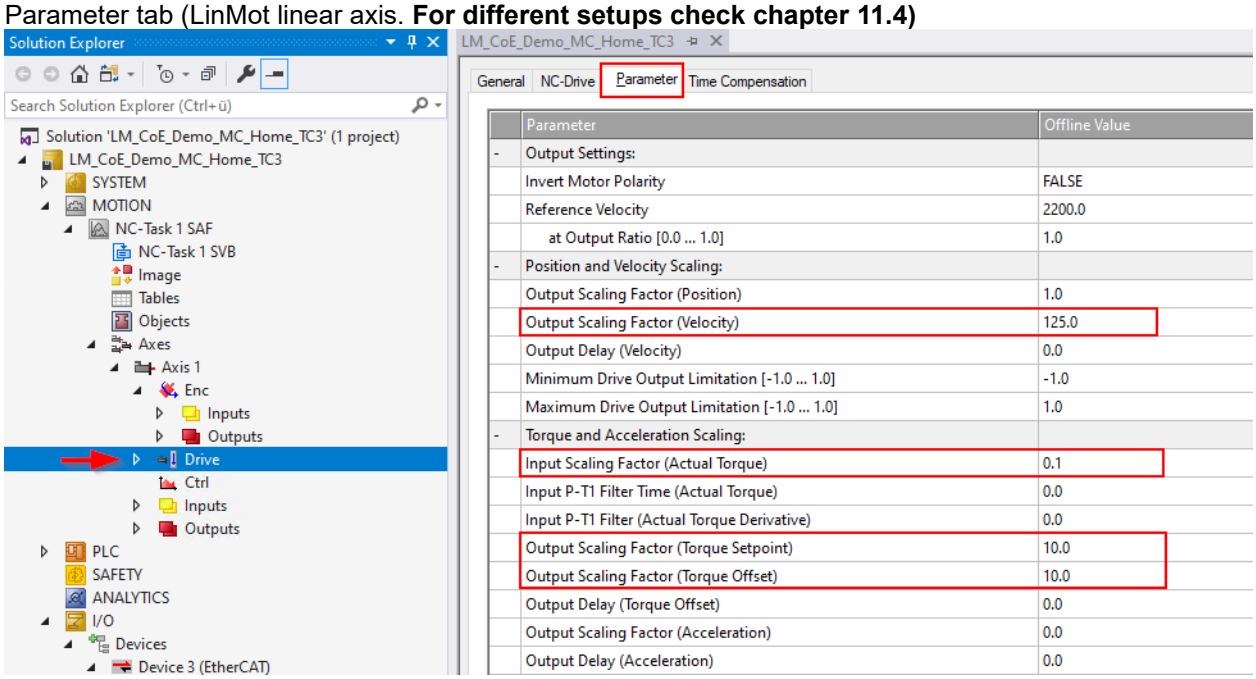



Depending on what Homing sequence you prefer you can setup some of the Homing parameters here.

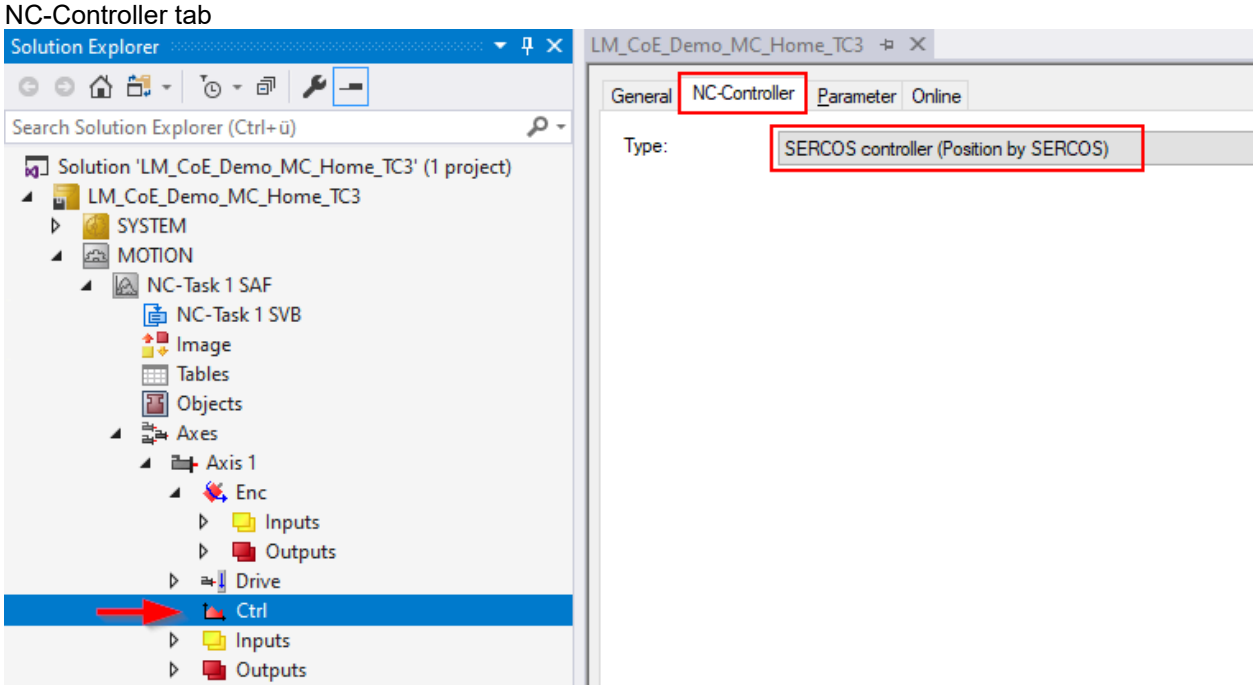
3. **Axis Drive:** Setup the drive as shown (red framed = mandatory)

   Parameter tab (LinMot linear axis. **For different setups check chapter 11.4)**

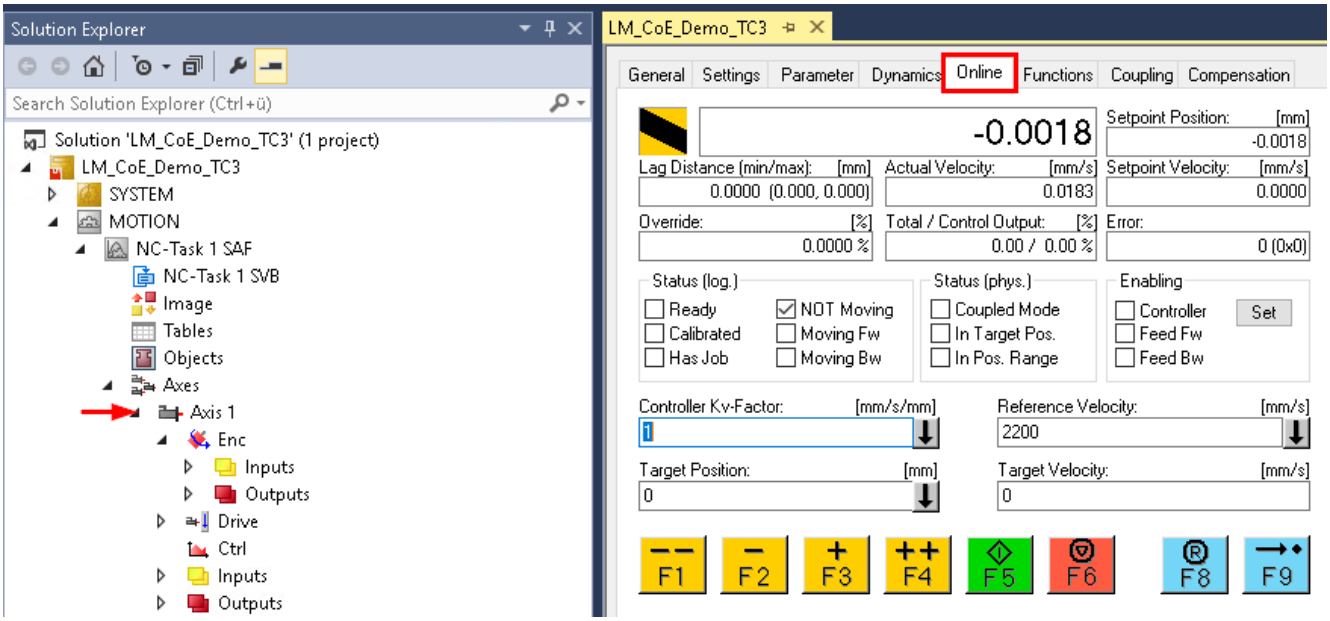



4. **Axis Ctrl:** Set Type as SERCOS controller

   NC-Controller tab



You are now ready to activate the configuration of the PLC and to enable and move the axis in the Online tab of the NC axis



## 11.4 Encoder Parameter Examples for Linear and Rotary Motors (Axis > Enc)

### 11.4.1 Linear Motors PS01 / PS10

The internal position resolution of the LinMot drive is 0.1um for linear motors.

Therefor the following parameters must be set accordingly:

- *Scaling Factor Numerator* must be set to **0.0001** (=1mm / 10'000 Increments)
- *Scaling Factor Denominator* is left to **1.0**

General | NC-Encoder | Parameter | Sercos | Time Compensation | Online

| | Parameter | Offline Value |
|---|---|---|
| - | Encoder Evaluation: | |
| | Invert Encoder Counting Direction | FALSE |
| | Scaling Factor Numerator | 0.0001 |
| | Scaling Factor Denominator (default: 1.0) | 1.0 |
| | Position Bias | 0.0 |
| | Modulo Factor (e.g. 360.0°) | 360.0 |
| | Tolerance Window for Modulo Start | 0.0 |
| | Encoder Mask (maximum encoder value) | 0xFFFFFFFF |
| | Encoder Sub Mask (absolute range maximum value) | 0x000FFFFF |
| | Reference System | 'INCREMENTAL' |

### 11.4.2 Rotary Motors RS01 & RS02

The LinMot RS01 motors (rotary part of LinMot PR01 & PR02 linear rotary motors) have a single turn absolute encoder and by default 360'000 Ticks (Increments) per motor revolution.

Therefor the following parameters must be set accordingly:

- *Scaling Factor Numerator* must be set to **0.001** (=360° / 360'000 Increments)
- *Scaling Factor Denominator* is left to **1.0**

Encoder tab

General | NC-Encoder | Parameter | Sercos | Time Compensation | Online

| | Parameter | Offline Value |
|---|---|---|
| - | Encoder Evaluation: | |
| | Invert Encoder Counting Direction | FALSE |
| | Scaling Factor Numerator | 0.001 |
| | Scaling Factor Denominator (default: 1.0) | 1.0 |
| | Position Bias | 0.0 |
| | Modulo Factor (e.g. 360.0°) | 360.0 |
| | Tolerance Window for Modulo Start | 0.0 |
| | Encoder Mask (maximum encoder value) | 0xFFFFFFFF |
| | Encoder Sub Mask (absolute range maximum value) | 0x00057E40 |
| | Reference System | 'ABSOLUTE SINGLETURN RANGE (with single overflow)' |

Drive tab:

General | NC-Drive | Parameter | Time Compensation

| | Parameter | Offline Value |
|---|---|---|
| - | Output Settings: | |
| | Invert Motor Polarity | FALSE |
| | Reference Velocity | 7200.0 |
| | at Output Ratio [0.0 ... 1.0] | 1.0 |
| - | Position and Velocity Scaling: | |
| | Output Scaling Factor (Position) | 1.0 |
| | Output Scaling Factor (Velocity) | 42.91538516558186 |
| | Output Delay (Velocity) | 0.0 |
| | Minimum Drive Output Limitation [-1.0 ... 1.0] | -1.0 |
| | Maximum Drive Output Limitation [-1.0 ... 1.0] | 1.0 |
| - | Torque and Acceleration Scaling: | |
| | Input Scaling Factor (Actual Torque) | 0.1 |
| | Input P-T1 Filter Time (Actual Torque) | 0.0 |
| | Input P-T1 Filter (Actual Torque Derivative) | 0.0 |
| | Output Scaling Factor (Torque Setpoint) | 10.0 |
| | Output Scaling Factor (Torque Offset) | 10.0 |
| | Output Delay (Torque Offset) | 0.0 |

The *Output Scaling Factor (Velocity)* depends on the selected *Encoder Sub Mask*.

**Scaling** = **125** * **360000** (16#00057E40) / **1048575** (16#000FFFFF) = **42.915385165581861097203347400043**

### 11.4.3 Rotary Motors EC02

The LinMot EC02 motors have a single turn absolute encoder and by default 524'288 (2^19) Ticks (Increments) per motor revolution.

Therefor the following parameters must be set accordingly:

- *Scaling Factor Numerator* must be set to **0.006866455078125** (=360° / 524'288 Increments * 10)
- *Scaling Factor Denominator* is set to **10.0** (=1.0 * 10)

Factor 10 is required to have the full resolution in the *Scaling Factor Numerator*

General | NC-Encoder | Parameter | Sercos | Time Compensation | Online

| | Parameter | Offline Value |
|---|---|---|
| - | Encoder Evaluation: | |
| | Invert Encoder Counting Direction | FALSE |
| | Scaling Factor Numerator | 0.006866455078125 |
| | Scaling Factor Denominator (default: 1.0) | 10.0 |
| | Position Bias | 0.0 |
| | Modulo Factor (e.g. 360.0°) | 360.0 |
| | Tolerance Window for Modulo Start | 0.0 |
| | Encoder Mask (maximum encoder value) | 0xFFFFFFFF |
| | Encoder Sub Mask (absolute range maximum value) | 0x00080000 |
| | Reference System | 'ABSOLUTE SINGLETURN RANGE (with single overflow)' |

Drive tab:

General | NC-Drive | Parameter | Time Compensation

| | Parameter | Offline Value |
|---|---|---|
| - | Output Settings: | |
| | Invert Motor Polarity | FALSE |
| | Reference Velocity | 7200.0 |
| | at Output Ratio [0.0 ... 1.0] | 1.0 |
| - | Position and Velocity Scaling: | |
| | Output Scaling Factor (Position) | 1.0 |
| | Output Scaling Factor (Velocity) | 62.500059604701619 |
| | Output Delay (Velocity) | 0.0 |
| | Minimum Drive Output Limitation [-1.0 ... 1.0] | -1.0 |
| | Maximum Drive Output Limitation [-1.0 ... 1.0] | 1.0 |
| - | Torque and Acceleration Scaling: | |
| | Input Scaling Factor (Actual Torque) | 0.1 |
| | Input P-T1 Filter Time (Actual Torque) | 0.0 |
| | Input P-T1 Filter (Actual Torque Derivative) | 0.0 |
| | Output Scaling Factor (Torque Setpoint) | 10.0 |
| | Output Scaling Factor (Torque Offset) | 10.0 |
| | Output Delay (Torque Offset) | 0.0 |
| | Output Scaling Factor (Acceleration) | 0.0 |
| | Output Delay (Acceleration) | 0.0 |

The *Output Scaling Factor (Velocity)* depends on the selected *Encoder Sub Mask*.

**Scaling** = **125** * **524288** (16#00080000) / **1048575** (16#000FFFFF) = **62.500059604701618863695968337983**

### 11.4.4 Other motors

For other motors, the principle is the same to calculate the Numerator, Denominator, and other encoder parameters as well as the Output Scaling Factor (Velocity)

## 11.5 Homing

There are several ways to home an axis.

Following one option by using the MC_Home function block (11.5.1) and a second option by using a LinMot function block (LM_DSHoming) that controls the drive-based homing (Mode of Operation 6) (11.5.2).

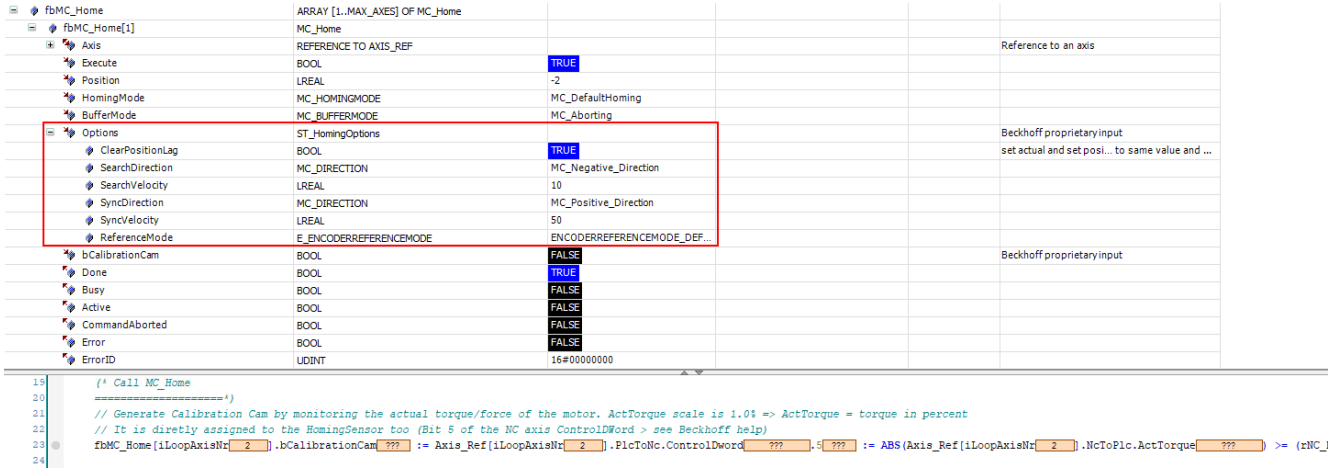
### 11.5.1 Homing using the MC_Home function block and simulate a reference CAM

The MC_Home function block does not support a block search yet. Nevertheless, it is possible to simulate a calibration cam by a certain force/torque value of the motor.

**Attention:**
To have the ActualTorque feedback in percent (500% = max. torque/force of the motor) at least firmware **6.10 Build 20210521 or later** is required.

The MC_Home function block *Options* in this example are set to start the search direction negative with 10 unit/s, clear position lag and move away from the calibration cam in positive direction with 50 unit/s.

The calibration cam is set TRUE when the actual motor torque/force is bigger than the set limit in variable rNC_HomeTorqueLimit (500% = max. motor torque/force).



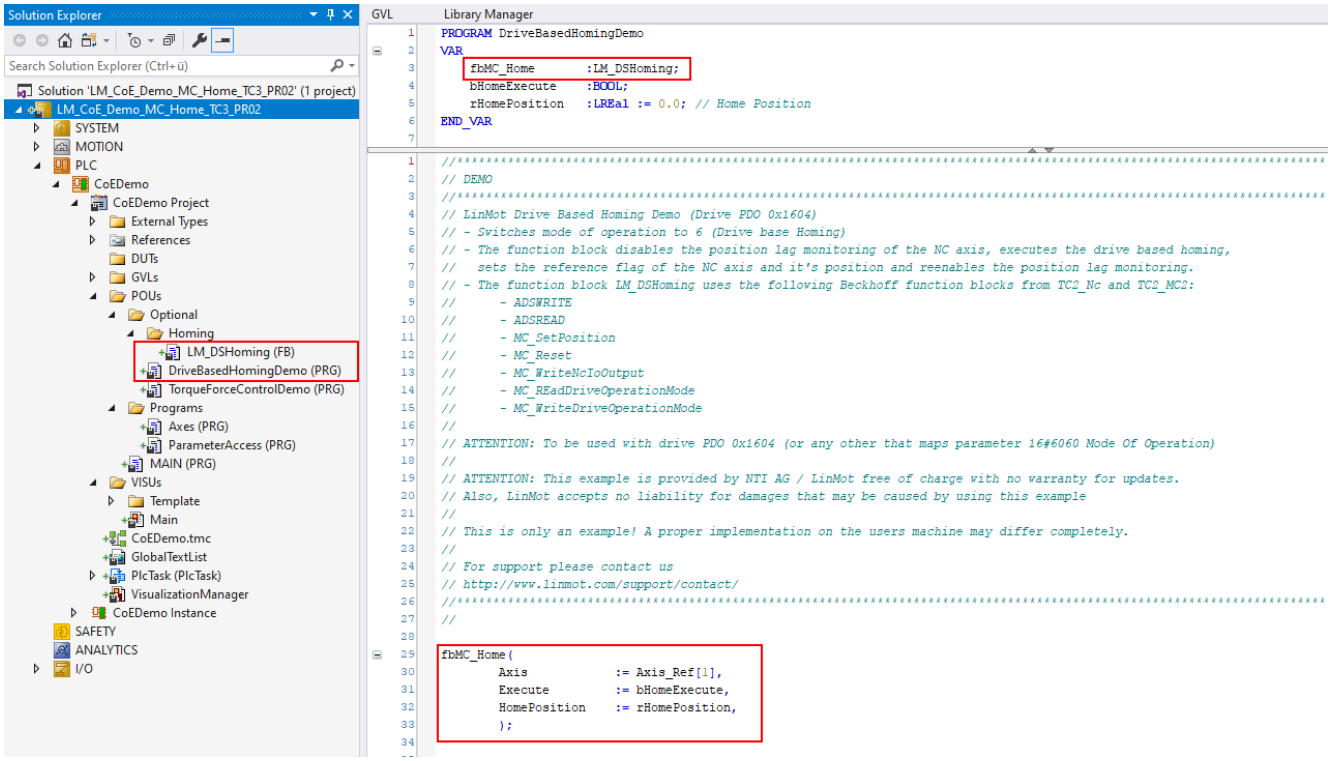

The actual torque is visible in the AXIS_REF structure (***AXIS_REF.NcToPlc.ActTorque***).

**Note:**
For more details check the example project (link in chapter 11.1 Overview)

### 11.5.2 Homing using the drive-controlled homing (Mode of Operation 6)

To home the axis, the drive-controlled homing (Mode of Operation 6) can be used.

The example project (see chapter 11.1) contains a function block (LM_DSHoming) to execute the drive-controlled homing.



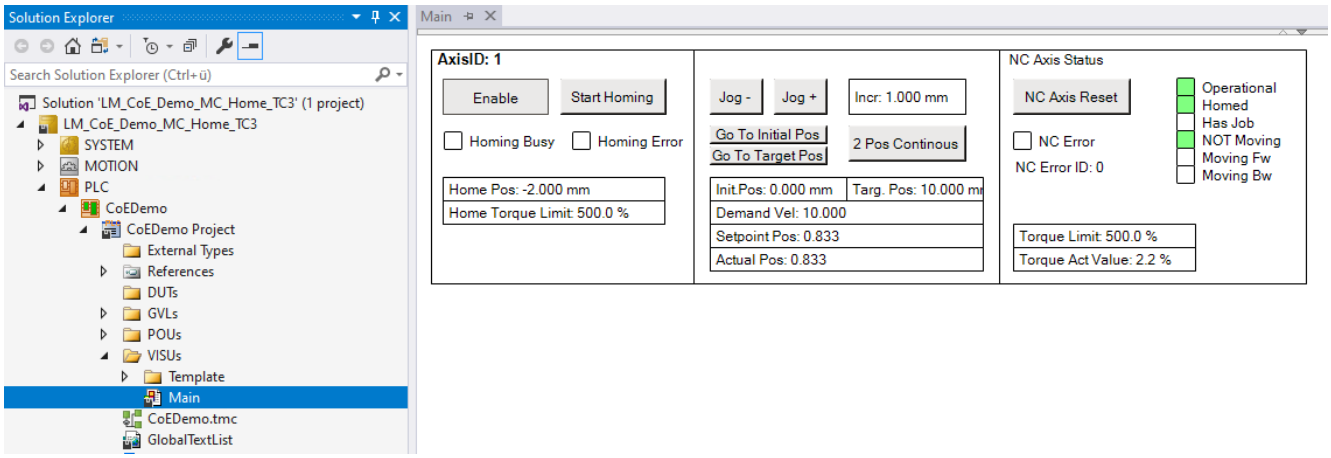

Inputs of the function block:

| | |
|---|---|
| Execute (*BOOL*) | Start drive-controlled homing |
| HomePosition (*LREAL*) | Position the NC axis actual position is set to after homing is completed |
| Axis (*AXIS_REF*) | Axis reference of the NC axis |

**Note:**
The LM_DSHoming function block uses several Beckhoff function blocks. The following Beckhoff libraries are required:
- **TC2_Standard** (R_TRIG, TON)
- **TC2_System** (ADSREAD, ADSWRITE)
- **TC2_MC2** (MC_Reset, MC_SetPosition, MC_WriteNcIoOutput, MC_ReadDriveOperationMode, MC_WriteDriveOperationMode)

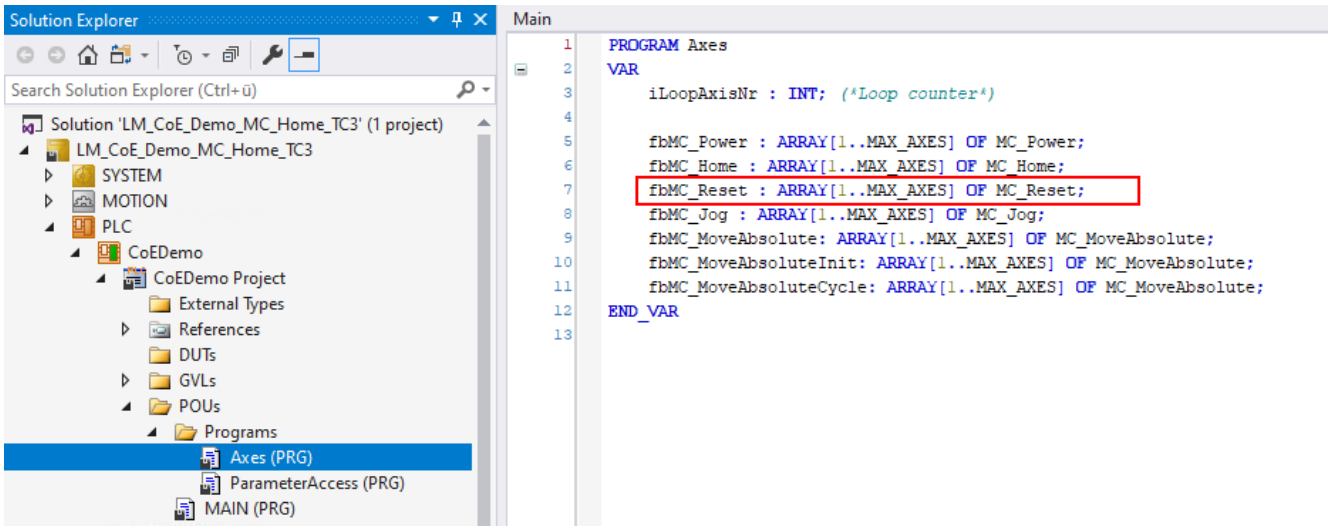
## 11.6 Visualization

The example project contains a visualization to control the NC axis (including the mentioned homing method in the chapter 11.5.1):




## 11.7 Drive Reset

To acknowledge an error of a CoE drive the function block *MC_Reset* must be used on TwinCAT systems. This function block is part of the Beckhoff library *TC2_MC2 (TwinCAT 3) / MC (TwinCAT 2)*.



## 11.8 Asynchronous Parameter Access

Parameters of the LinMot drive can be accessed using the Beckhoff function block *FB_EcCoESdoRead* and *FB_EcCoESdoWrite* which are part of the *Tc2_EtherCAT (TwinCAT 3) / EtherCAT (TwinCAT 2)* library.

More information about this function blocks can be found in the Beckhoff Information System: http://infosys.beckhoff.com (search for *FB_EcCoESdoRead & FB_EcCoESdoWrite).*

Example calls of this function blocks can be found in the example project.

```
PROGRAM ParameterAccess
VAR
    iLoopAxisNr :INT; (*Loop counter*)

    fbGetAxisAmsAddr : ARRAY[1..MAX_AXES] OF FB_GetAxisAmsAddr; (* Get the NetID and the Port of the drive linked to the NC axis by AxisID *)

    CoERead     :ARRAY[1..MAX_AXES] OF FB_EcCoESdoRead;
    CoEWrite    :ARRAY[1..MAX_AXES] OF FB_EcCoESdoWrite;

    UPID        :ARRAY[1..MAX_AXES] OF WORD;
    SubIndex    :ARRAY[1..MAX_AXES] OF BYTE;
    DataIN      :ARRAY[1..MAX_AXES] OF DINT;
    DataOUT     :ARRAY[1..MAX_AXES] OF DINT;
END_VAR
```

```
    (* Read parameter
    ============================================================*)
    CoERead[iLoopAxisNr](
        sNetId:=g_aDrive_AmsAddr[iLoopAxisNr].netId,
        nSlaveAddr:=g_aDrive_AmsAddr[iLoopAxisNr].port,
        nIndex:= 16#2000 + UPID[iLoopAxisNr],
        nSubIndex:= SubIndex[iLoopAxisNr], (*e.g. 16#01 = Access RAM Value*)
        pDstBuf:= ADR(DataIN[iLoopAxisNr]),
        cbBufLen:= SIZEOF(DataIN[iLoopAxisNr]),
        bExecute:= ,
        tTimeout:= ,
        bBusy=> ,
        bError=> ,
        nErrId=> );

    (* Write parameter
    ============================================================*)
    CoEWrite[iLoopAxisNr](
        sNetId:=g_aDrive_AmsAddr[iLoopAxisNr].netId,
        nSlaveAddr:=g_aDrive_AmsAddr[iLoopAxisNr].port,
        nIndex:= 16#2000 + UPID[iLoopAxisNr],
        nSubIndex:= SubIndex[iLoopAxisNr], (*e.g. 16#01 = Access RAM Value*)
        pSrcBuf:= ADR(DataOUT[iLoopAxisNr]),
        cbBufLen:= SIZEOF(DataOUT[iLoopAxisNr]),
        bExecute:= ,
        tTimeout:= ,
        bBusy=> ,
        bError=> ,
        nErrId=> );
```

**sNetID, nSlaveAddr:** AmsNetID & port of the drive.
These can be read using the function block *FB_GetAxisAmsAddr* which is part of the Beckhoff library *Tc2_NC (TwinCAT 3) / TcNC (TwinCAT 2)*.

**nDriveNo:** NC axis ID

**nIndex:** Parameter address to be accessed = **16#2000** + UPID
(E.g., Maximal Current (UPID 16#13A6) ▶ **nIndex** = 16#2000 + 16#13A6 = 16#33A6)

**nSubIndex:** The following commands can be performed
(See user manual 0185-1103 for more information > search on http://shop.linmot.com)

| SDO Service | Index | SubIndex | Description |
|---|---|---|---|
| Read | 2000h+UPID | 0x01 | Parameter UPID read RAM value |
| Write | 2000h+UPID | 0x01 | Parameter UPID write RAM value |
| Read | 2000h+UPID | 0x02 | Parameter UPID read ROM value |
| Write | 2000h+UPID | 0x02 | Parameter UPID write ROM value |
| Read | 2000h+UPID | 0x03 | Parameter UPID read minimal value |
| Read | 2000h+UPID | 0x04 | Parameter UPID read maximal value |
| Read | 2000h+UPID | 0x05 | Parameter UPID read default value |
| Write | 2000h+UPID | 0x06 | Parameter UPID write RAM and ROM value |
| Write | 2000h | 0x07 | Parameter Default OS SW instance |
| Write | 2000h | 0x08 | Parameter Default MC SW instance |
| Write | 2000h | 0x09 | Parameter Default Intf SW instance |
| Write | 2000h | 0x0A | Parameter Default Appl SW instance |
| Write | 2000h | 0x0B | Reset device |

**pDstBuf, pSrcBuf:** Address to the variable containing the value to be read / written (must be of size 4 Bytes!)

**cbBufLen:** Size of the variable of the value to be read / written (must be of size 4 Bytes!)

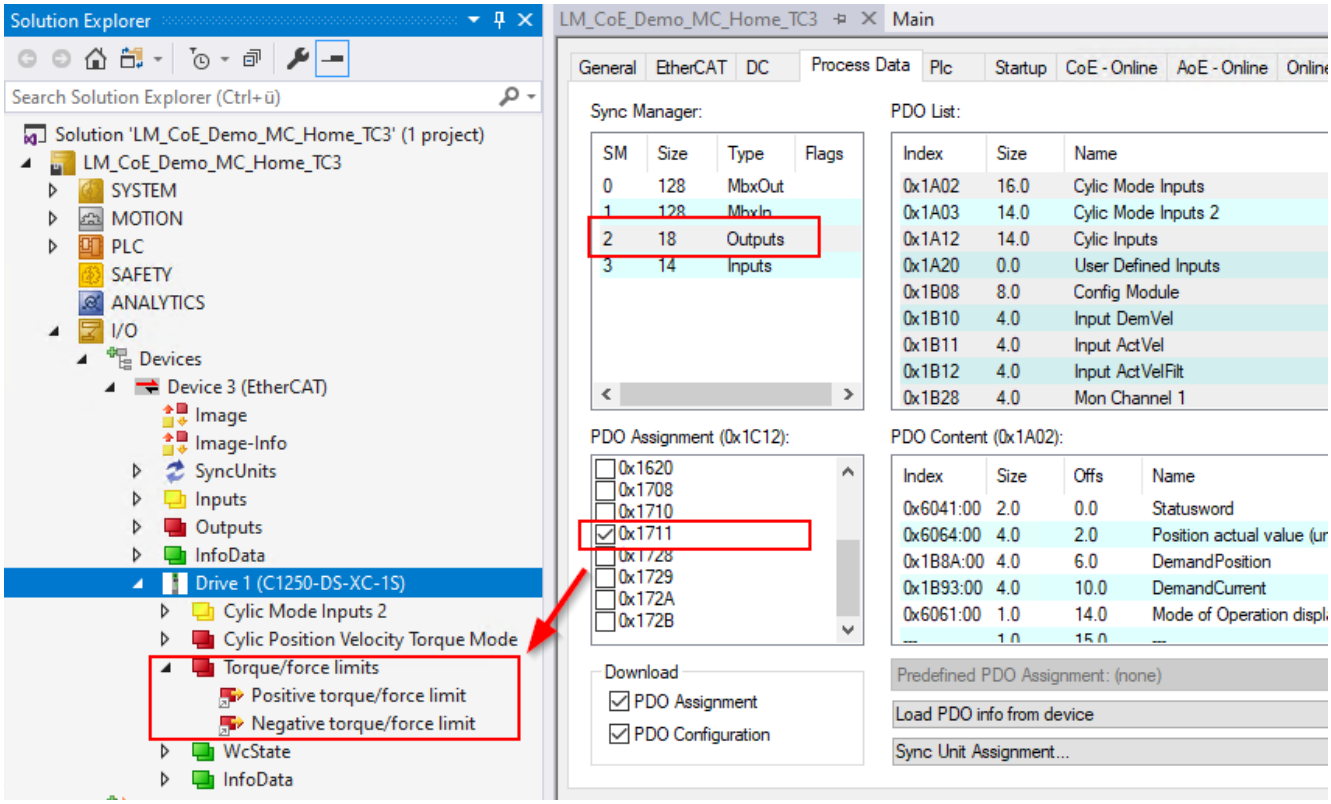
**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

**Note:**
Use *FB_EcCoESdoRead & FB_EcCoESdoWrite* from Beckhoff to access drive parameters.

For every access (read or write) the length of the variable connected to pDstBuf/pSrcBuf must be of **size 4 Bytes** (e.g., datatypes DWORD, DINT, UDINT).
If you access a parameter/variable of a different size (e.g., INT, SINT, …) then make a typecast afterwards.

## 11.9 Add Positive and Negative torque/force limit (0x1711) to the Process Data

It is possible to add additional Process Data to PDO for cyclical write access. The additional Output *Torque/force Limits* (0x60E0 & 0x60E1) control the *Maximal Current Positive (UPID 13FCh)* and the *Maximal Current Negative (UPID 13FDh)* of the drive cyclically by adding the PDO 0x1711 in the PDO Assignment.




**Note:**
The maximum Torque/force Limit value is **5000 (500.0%)** and corresponds to the **maximal Current** of the connected motor. The value **1000 (100.0%)** corresponds approximately to the **nominal Current** of the motor.

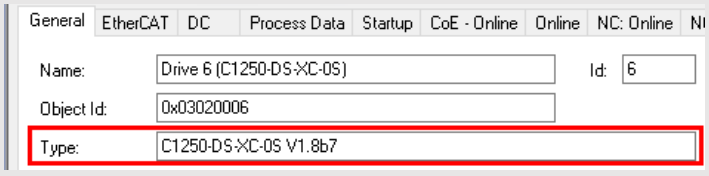
**Example 1:**

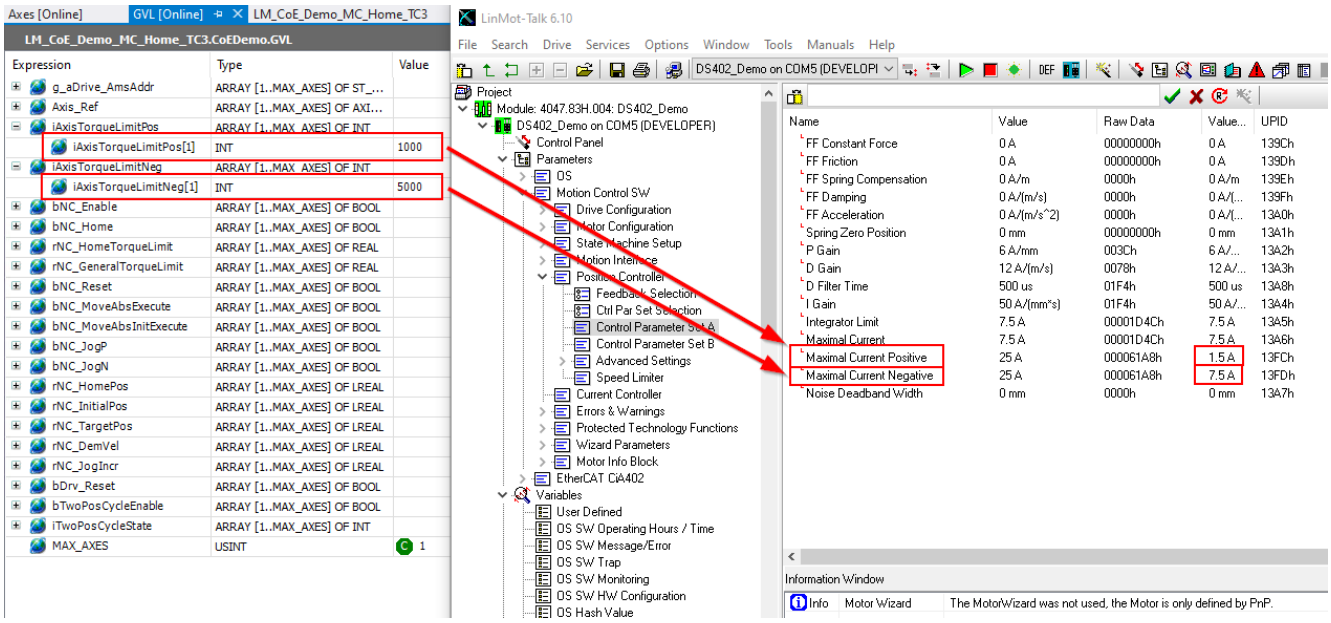
| | |
|---|---|
| Torque/Force Limit Positive: | **1000 (100.0%)** |
| Motor Maximal Current: | 7.5 A |
| = Max. Current Positive: | 1.5 A |

**Example 2:**

| | |
|---|---|
| Torque/Force Limit Negative: | **5000 (500.0%)** |
| Motor Maximal Current: | 7.5 A |
| = Max. Current Negative: | 7.5 A |

**Note:**
Torque/force Limit value is supported from xml version V1.8b7 with Firmware Build 6.8 Build 20190315 and newer.



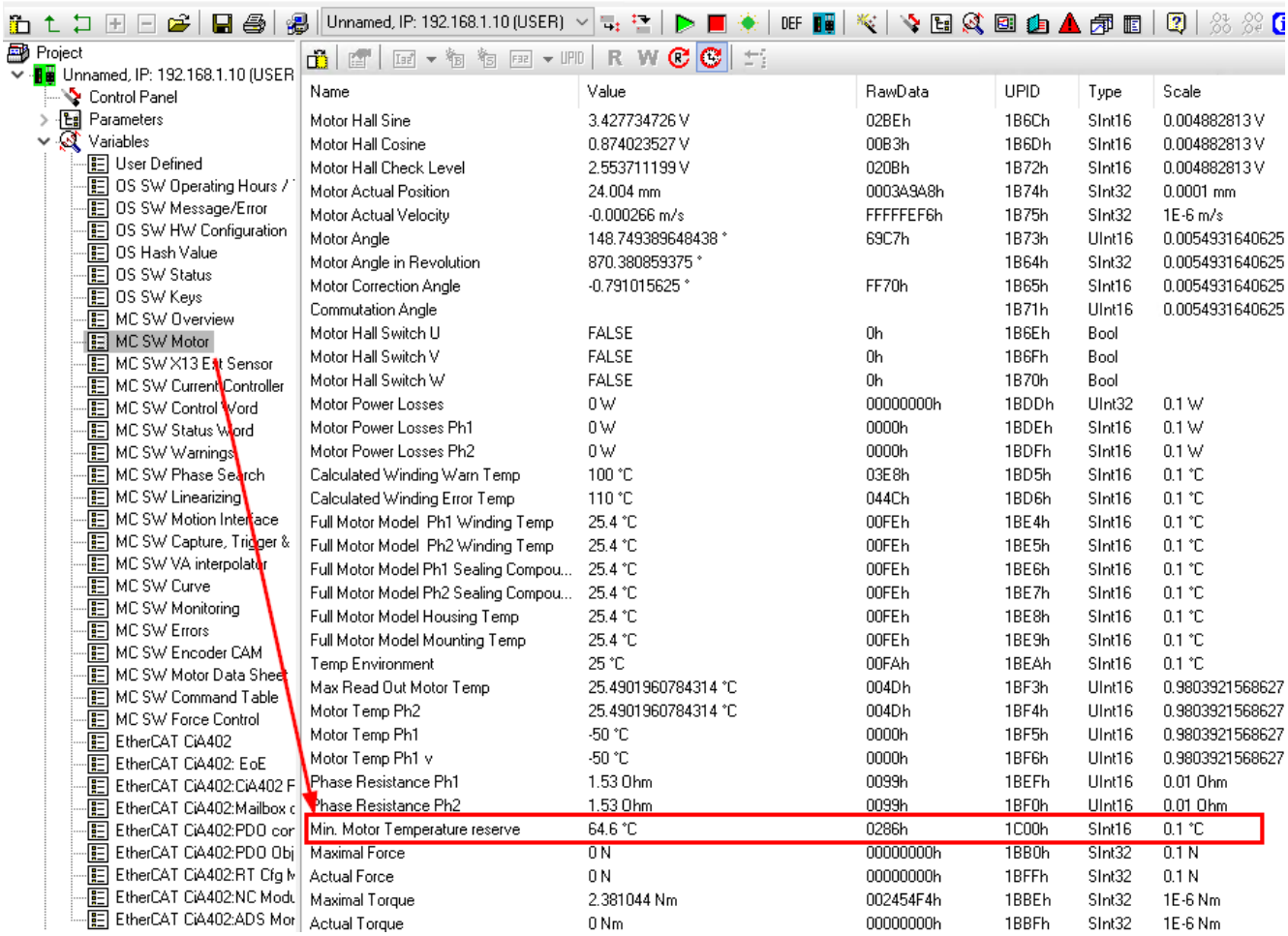



**Attention:**
Make sure, that the torque/force limit is not 0. Otherwise, the motor cannot move with a current of 0A.

## 11.10 Add Additional Parameters or Variables to the Process Data

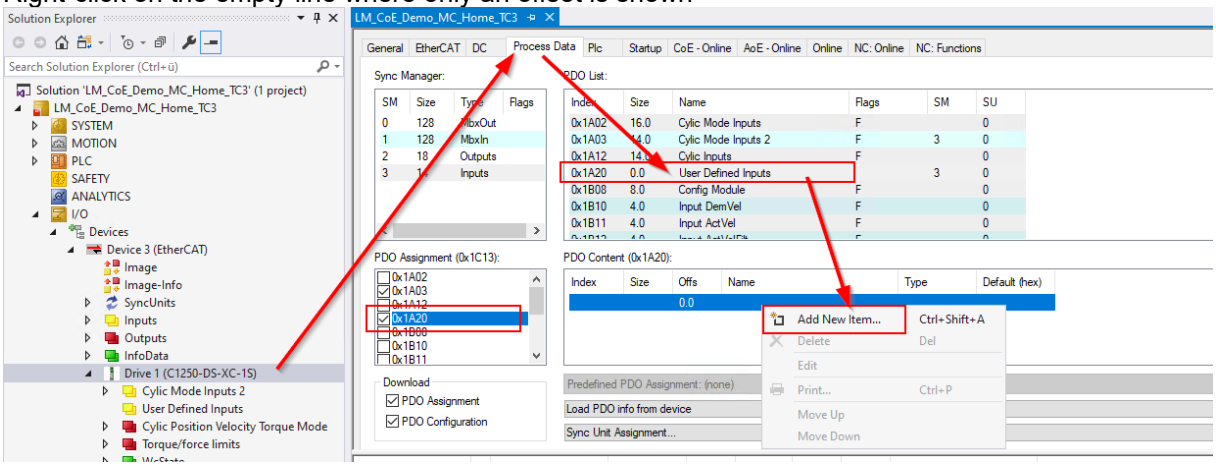
It is possible to add additional parameters or variables of the drive to PDO for cyclical write or read access.

**Example 1:** Read the *Min. Motor Temperature reserve (UPID 1C00h)* of the drive cyclically by adding it into the User Defined Inputs (0x1A20):

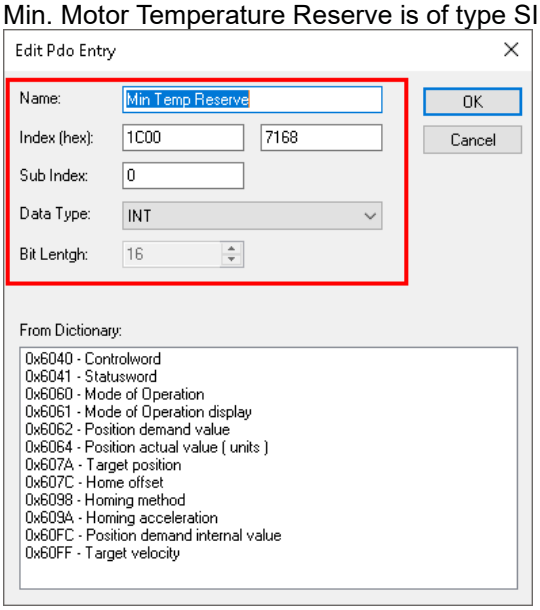



**Name:** Min. Motor Temperature reserve
**UPID:** 1C00h
**Type:** Sint16 = INT
**Scale:** 0.1 °C

1. In the Process Data select *0x1A20 User Defined Inputs* from the PDO List.
2. Right-click on the empty line where only an offset is shown

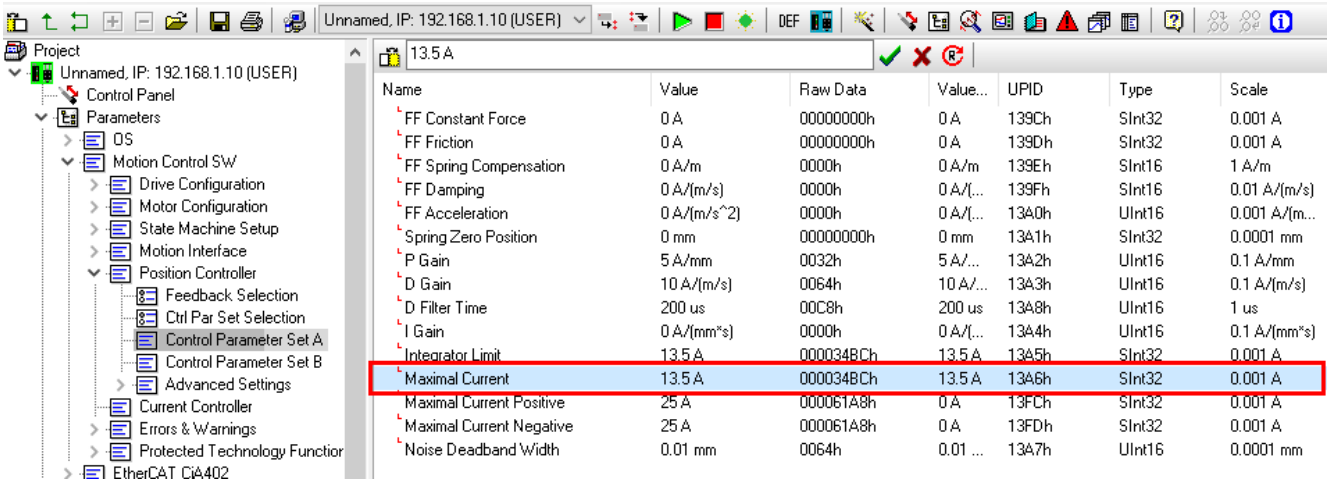


3. In the *Edit PDO Entry* window enter the name, the UPID as *Index* and set the *Data Type* to INT (as the Min. Motor Temperature Reserve is of type SINT16)




4. Click OK to finish

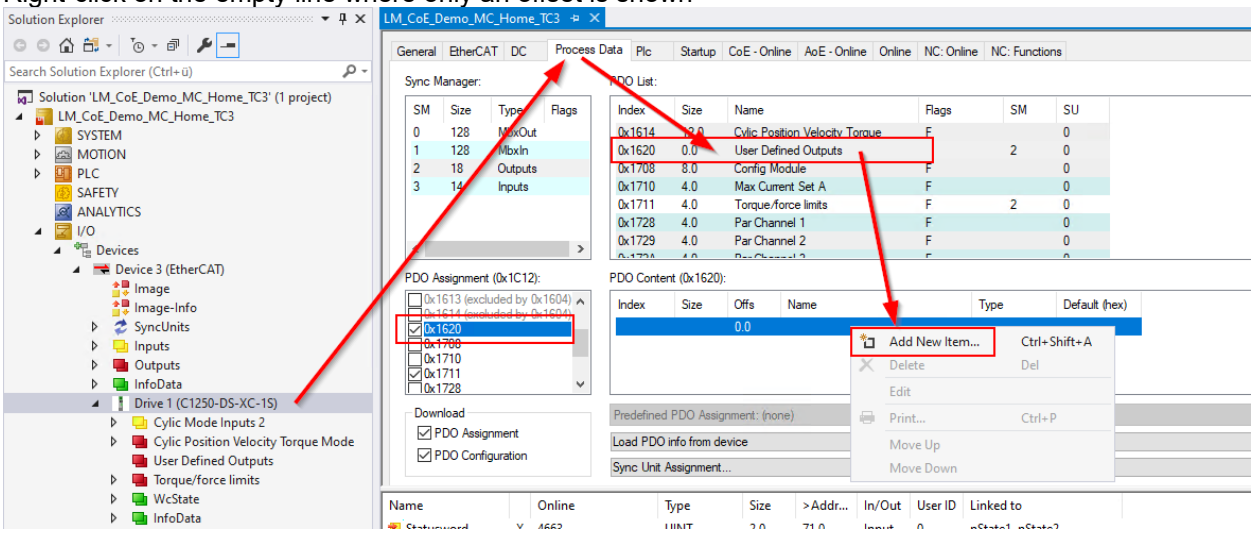
**Example 2:** Write the *Maximal Current (UPID 13A6h)* of the drive cyclically by adding it into the User Defined Outputs (0x1620):



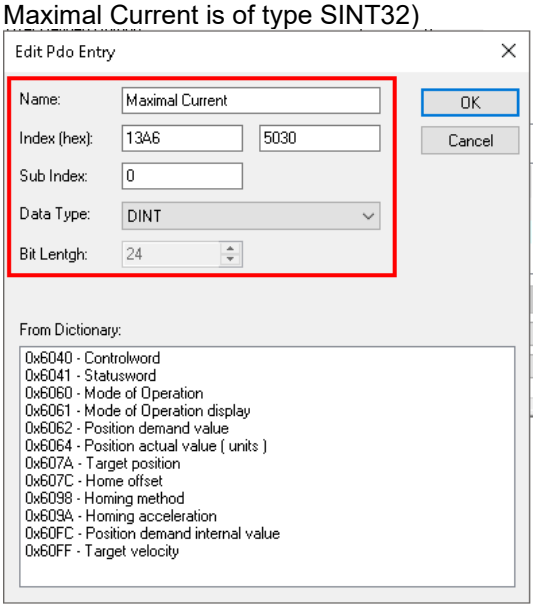

**Name:** Maximal Current
**UPID:** 13A6h
**Type:** SInt32 = DINT
**Scale:** 0.001 A

1. In the Process Data select 0x1620 *User Defined Outputs* from the PDO List.

2. Right-click on the empty line where only an offset is shown

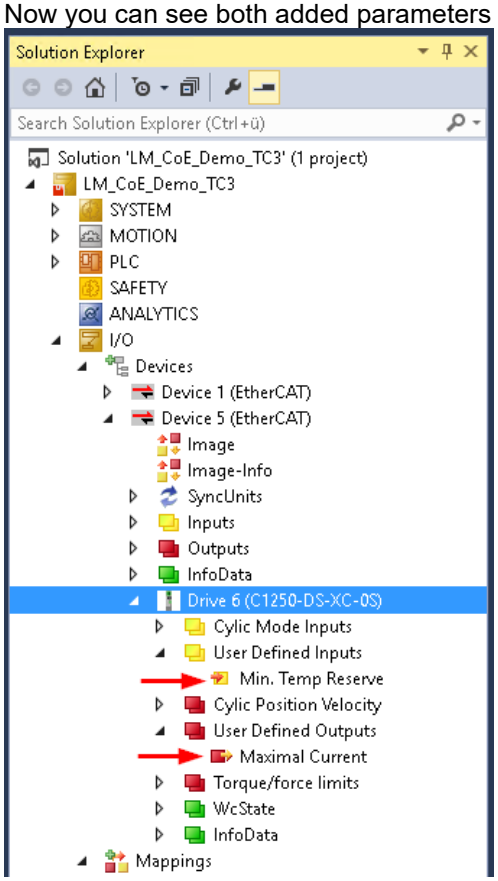



3. In the *Edit PDO Entry* window enter the name, the UPID as *Index* and set the *Data Type* to DINT (as the Maximal Current is of type SINT32)




4. Click OK to finish

Now you can see both added parameters in the solution tree and link them:




Variable Definition in the GVLs

```
iMotorSensorTemp    AT %I*  :ARRAY[1..MAX_AXES] OF INT;
uiMaximalCurrent    AT %Q* :ARRAY[1..MAX_AXES] OF DINT := [MAX_AXES(1000)]; (*1A*)
uiForceLimitPos     AT %Q* :ARRAY[1..MAX_AXES] OF UINT := [MAX_AXES(5000)]; (*500$*)
uiForceLimitNeg     AT %Q* :ARRAY[1..MAX_AXES] OF UINT := [MAX_AXES(5000)]; (*500$*)
```

## 11.11 Using the Closed Loop Force/Torque Control Technology Function

This chapter shows an application example of how the technology function *closed loop force/torque control* inside the LinMot drive can be used.

This chapter does not cover the general setup of force/torque sensor and the according control loop parameters. Please check the according documentation.

**Attention:**
To use closed loop force/torque control the technology function **TF-Force Control** (0150-2503) is required. Check the manual for more information: https://shop.linmot.com/E/product/0150-2503

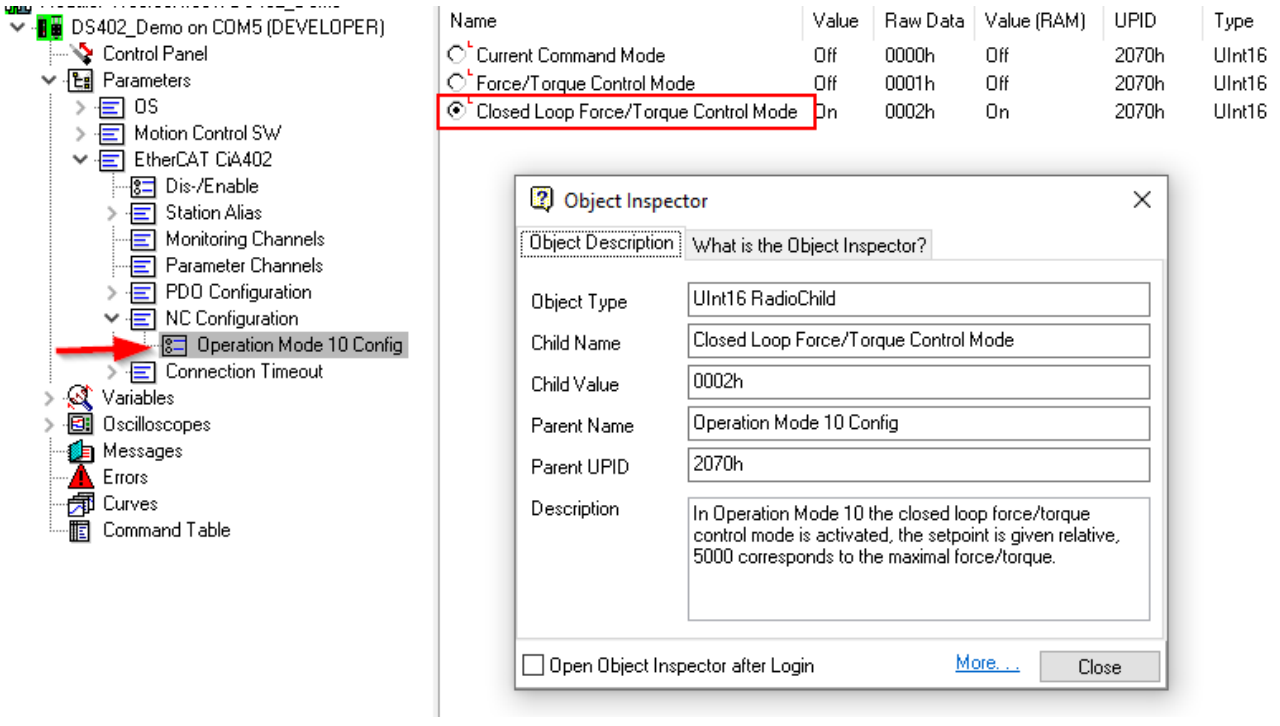
**Attention:**
Drive firmware **6.10 Build 20210521 or later** is required.

**Note:**
This example was done using the rotary part of a LinMot PR02 Linear Rotary Module with torque sensor. Nevertheless, the same principle is applicable for linear motors with force sensors.

See Appendix II: Basic Closed Loop Torque/Force Control Loop Tuning

### 11.11.1 Drive Setup

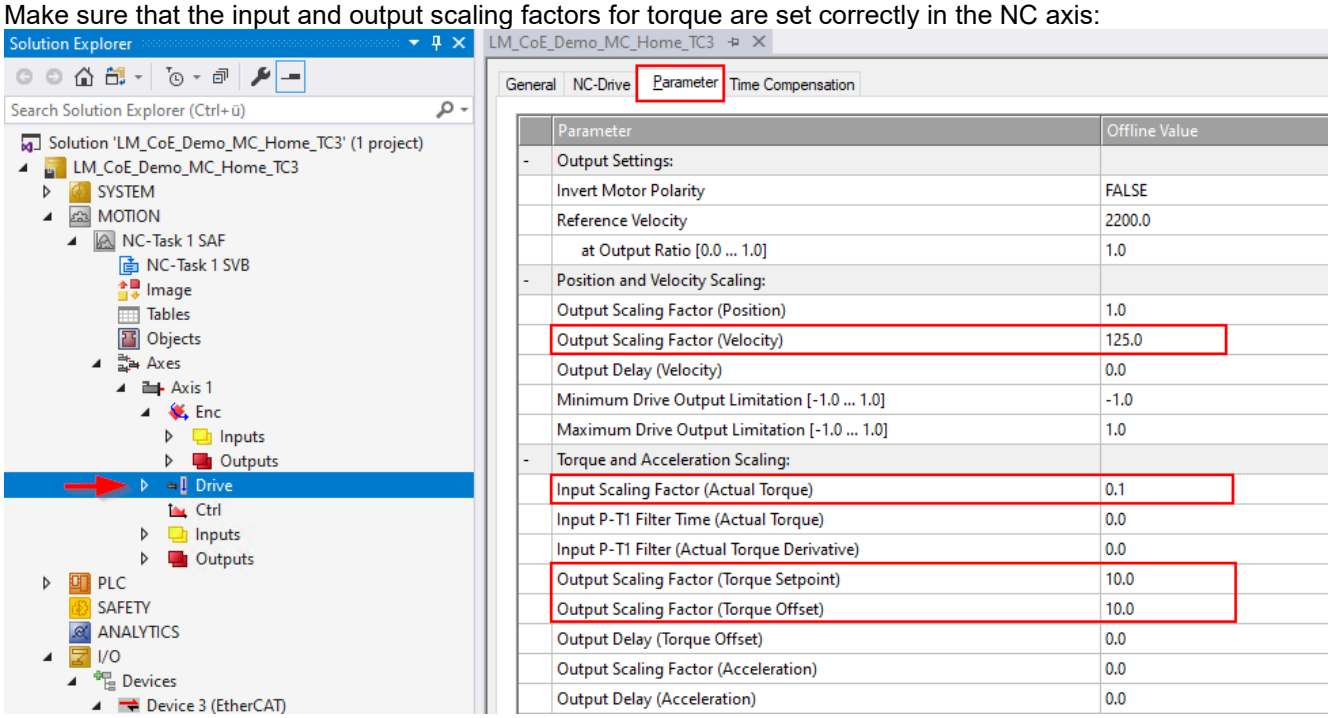
To enable the closed loop/force control in *Mode of Operation 10 (CST)* UPID 2070h must be set to “Closed Loop Force/Torque Control Mode”:




**Note:**
If an open loop force/torque control is enough for the application (less accurate and influenced by mechanical properties) one can also us “Force/Torque Control Mode” which sets the requested current.

### 11.11.2 PLC Setup

Make sure that the input and output scaling factors for torque are set correctly in the NC axis:




| | Parameter | Offline Value |
|---|---|---|
| - | Output Settings: | |
| | Invert Motor Polarity | FALSE |
| | Reference Velocity | 2200.0 |
| | at Output Ratio [0.0 ... 1.0] | 1.0 |
| - | Position and Velocity Scaling: | |
| | Output Scaling Factor (Position) | 1.0 |
| | Output Scaling Factor (Velocity) | 125.0 |
| | Output Delay (Velocity) | 0.0 |
| | Minimum Drive Output Limitation [-1.0 ... 1.0] | -1.0 |
| | Maximum Drive Output Limitation [-1.0 ... 1.0] | 1.0 |
| - | Torque and Acceleration Scaling: | |
| | Input Scaling Factor (Actual Torque) | 0.1 |
| | Input P-T1 Filter Time (Actual Torque) | 0.0 |
| | Input P-T1 Filter (Actual Torque Derivative) | 0.0 |
| | Output Scaling Factor (Torque Setpoint) | 10.0 |
| | Output Scaling Factor (Torque Offset) | 10.0 |
| | Output Delay (Torque Offset) | 0.0 |
| | Output Scaling Factor (Acceleration) | 0.0 |
| | Output Delay (Acceleration) | 0.0 |

**Note:**
The torque unit is in 0.1 percent [%].
Decimal 5000 reflects 500% = motor maximal torque/force.
Torque Actual value (object 0x6077) and Target torque (object 0x6071) are used to read and write the torque/force of the motor in percent.

Therefore, the input scaling factor must be 0.1 and the output scaling factor must be 10.

**Attention:**
This example requires the drive PDOs setup as shown in chapter 11.3.3.

### 11.11.3 Example Sequence

The example project from chapter 11.1 contains an example sequence ***TorqueForceControlDemo*** following that shows how a closed loop force control process could look like.

In the example the axis is controlling a rotary motor.

**Attention:**
This example should not directly be used as it is shown!
Please adapt it to your machine and error handling concept.

Tested using:
- TwinCAT target version **V3.1 Build 4024.32**
- LinMot C1250-DS-XC-1S drive with **Firmware 6.11 Build 20220628**

```
PROGRAM TorqueForceControlDemo
VAR
    fbMC_TorqueControl      :MC_TorqueControl;
    fbMC_MoveAbsolute       :MC_MoveAbsolute;
    fbMC_MoveVelocity       :MC_MoveVelocity;
    fbMC_Halt               :MC_Halt;
    fbMC_Reset              :MC_Reset;

    rTargetTorque           :REAL := 100.0; // 100.0 % target torque/force (+/-500.0 % = maximal torque/force of the motor)
    rLimitTorque            :REAL := 70.0; // 70.0 % torque/force limit to switch to torque/force control
    rSearchSpeed            :REAL := 10.0; // 10 Units
    rSearchAccDec           :REAL := 100.0; // 100 Units
    rFastSpeed              :REAL := 4000.0; // 4000 Units
    rFastAccDec             :REAL := 5000.0; // 5000 Units
    rStopDec                :REAL := 50000.0; // 50000 Units
    rDistance               :REAL := 0.05; // 0.05 Units, Distance to move away from target after process completed
    rSmoothJerk             :REAL := 1000000.0; //
    rStopJerk               :REAL := 10000.0; //

    State                   :INT; // State machine
    Execute                 :BOOL; // Start process

    Error                   :BOOL; // Error flag
    ErrorId                 :UDINT; // Error code
    ErrorState              :INT; // State Error happened

    Axis                    :Axis_Ref;

    ExecuteOffTrigger       :F_TRIG;
END_VAR
```

```
// Read inputs
// ***********************************************
Axis := Axis_Ref[1]; // Assign axis (1 in this example)

//*************************************************************************************************************************************************
// DEMO
//*************************************************************************************************************************************************
// LinMot Drive Based Torque/Force Control Demo
// - Switches mode of operation to 10 (CST, TorqueControl) by using MC_TorqueControl (tc2_mc2 3.3.48.0)
// - Inside the drive set parameter UPID 2070h to the desired mode:
//      Force/Torque Mode (1): without force/torque cell
//      Closed Loop Force/Torque Control Mode (2): closed loop force control inside the drive with force/torque cell feedback
//
// ATTENTION: Make sure to enable the Speed Limiter in the drive (UPID 1511h) before starting with tests
//
// ATTENTION: This example is provided by NTI AG / LinMot free of charge with no warranty for updates.
// Also, LinMot accepts no liability for damages that may be caused by using this example
//
// This is only an example! A proper implementation on the users machine may differ completely.
```

# 12 Drive Profile: Beckhoff TwinCAT 2/3, SoE (Outdated)

**Important Note:**

For new applications with Beckhoff PLCs LinMot recommends using LinMot drives with **CoE (DS402/CiA402)** interface.

See chapter **11 Drive Profile: Beckhoff TwinCAT 2/3, CoE DS402**

## 12.1 Overview

This chapter shows how a LinMot drive with *Servo Drive Profile over EtherCAT (SoE)* interface (e.g., C1250-SE-XC-1S) can be integrated and setup in a Beckhoff TwinCAT environment.
For this example, TwinCAT 3 is used. The steps are generally similar for TwinCAT 2.

**Download:**
Example projects can be downloaded from:
http://download.linmot.com/plc_lib/examples/Beckhoff_SoE/ (named *LM_SoE_Demo_TCx_...*)

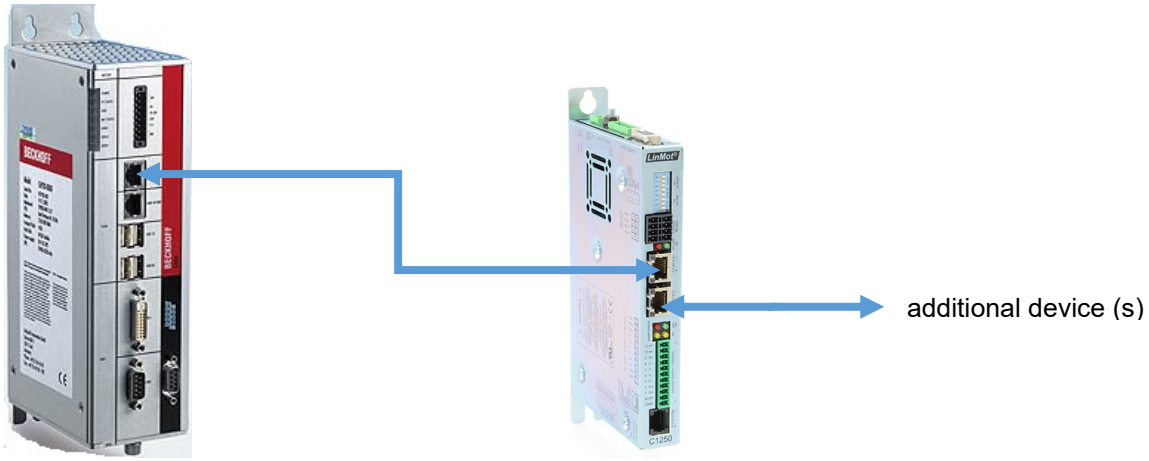



Image Source: http://www.beckhoff.com/

EtherCAT is the real-time Ethernet network originally developed by Beckhoff. The LinMot acts as Slave in this network and is implemented with the standard ASIC ET1100 from Beckhoff.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

## 12.2 Configuration and Connection of the LinMot Drive

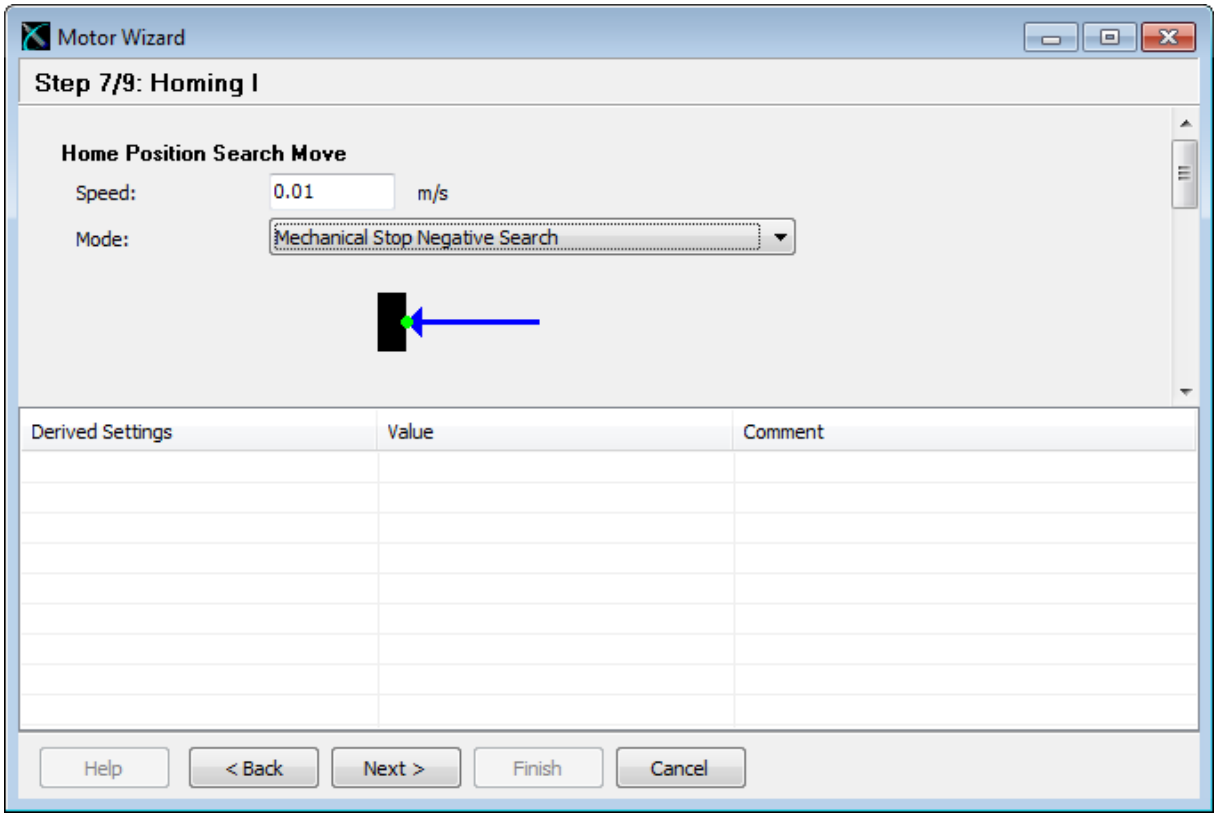
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 12.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select a homing mode in the Motor Wizard as with the SoE interface a drive-based homing is possible (e.g. *Mechanical Stop Negative Search*):




See Appendix I: Basic Position Control Loop Tuning

### 12.2.2 XML File

Install the XMLfile that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- EtherCAT SoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT_SoE\XML

### 12.2.3 EtherCAT Connection

The drive is connected to the EtherCAT network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-SE-UC drive. On all other LinMot drives supporting EtherCAT SoE the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.

| X17 - X18 | RealTime Ethernet 10/100 Mbit/s | |
|---|---|---|
|  | X17 RT ETH In | (Beckhoff: EtherCAT IN, from PLC) |
| | X18 RT ETH Out | (Beckhoff: EtherCAT OUT, to next device) |
| RJ-45 | | |

## 12.3 PLC Setup

### 12.3.1 EtherCAT Device Description File XML

The EtherCAT device description file (XML) is located by default in the LinMot-Talk installation path:
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT_SoE\XML

Copy this file to your local TwinCAT installation path:

- C:\TwinCAT\Io\EtherCAT\ (for TwinCAT 2)
- C:\TwinCAT\3.1\Config\Io\EtherCAT (for TwinCAT 3)

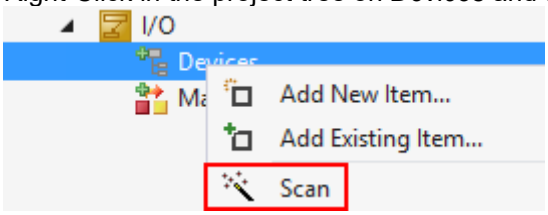
**Note:** You may have to restart the developing environment to have the new XML file to be recognized.

### 12.3.2 Scanning the EtherCAT for new Devices

Follow these steps to add the LinMot drive to your Beckhoff PLC.

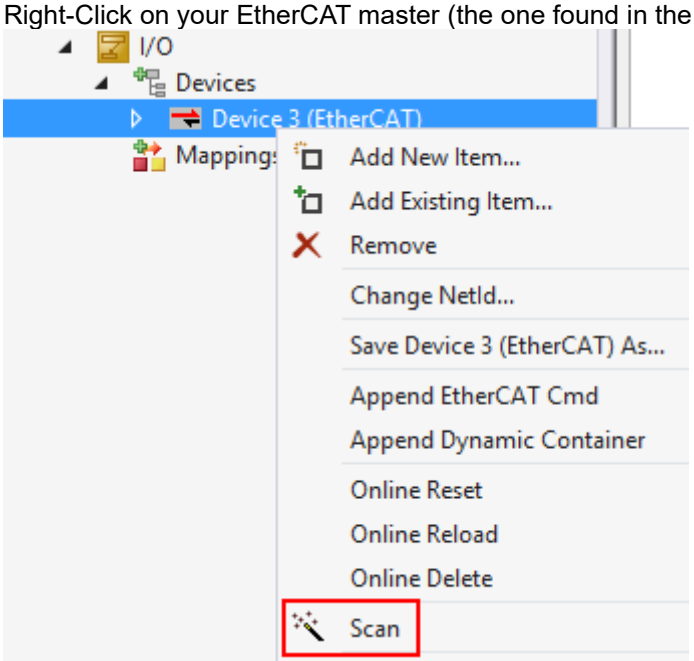
1. Restart your PLC into the *Config Mode* to be able to scan for attached devices.


2. Right-Click in the project tree on *Devices* and select *Scan*

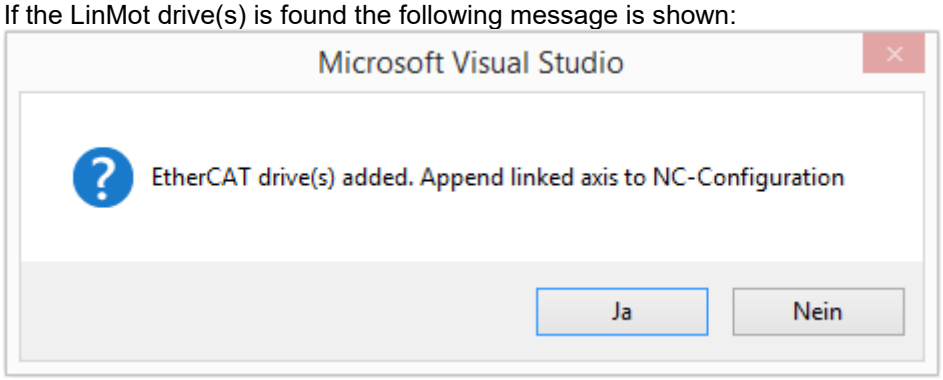



Select and insert the found EtherCAT master (e.g. *Device 3 (EtherCAT)*).
You can now directly scan for boxes. If you select yes, then skip the next step.

3. Right-Click on your EtherCAT master (the one found in the previous step) and select *Scan*

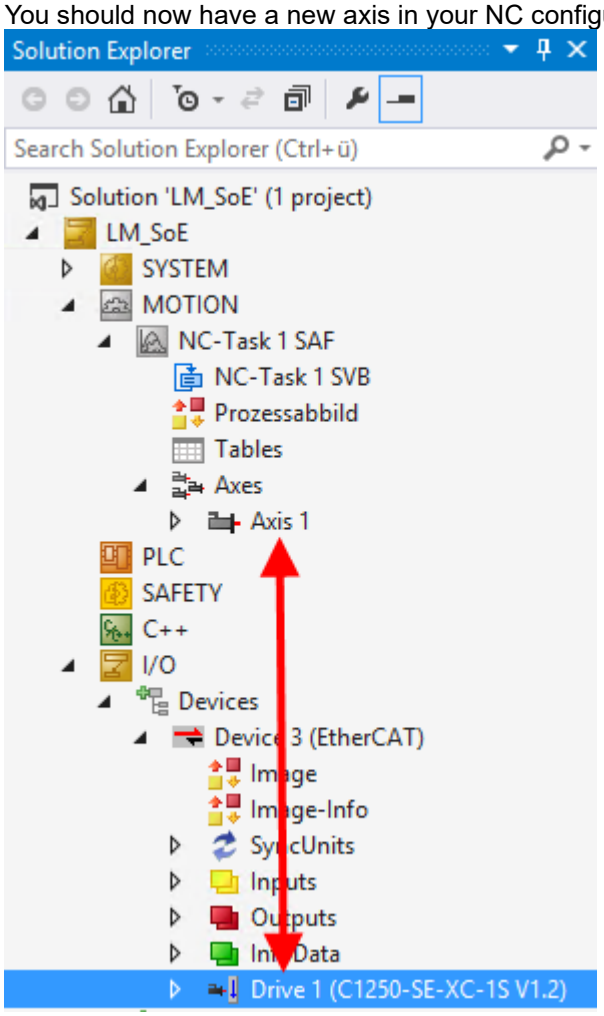


4. If the LinMot drive(s) is found the following message is shown:




Select Yes (Ja)

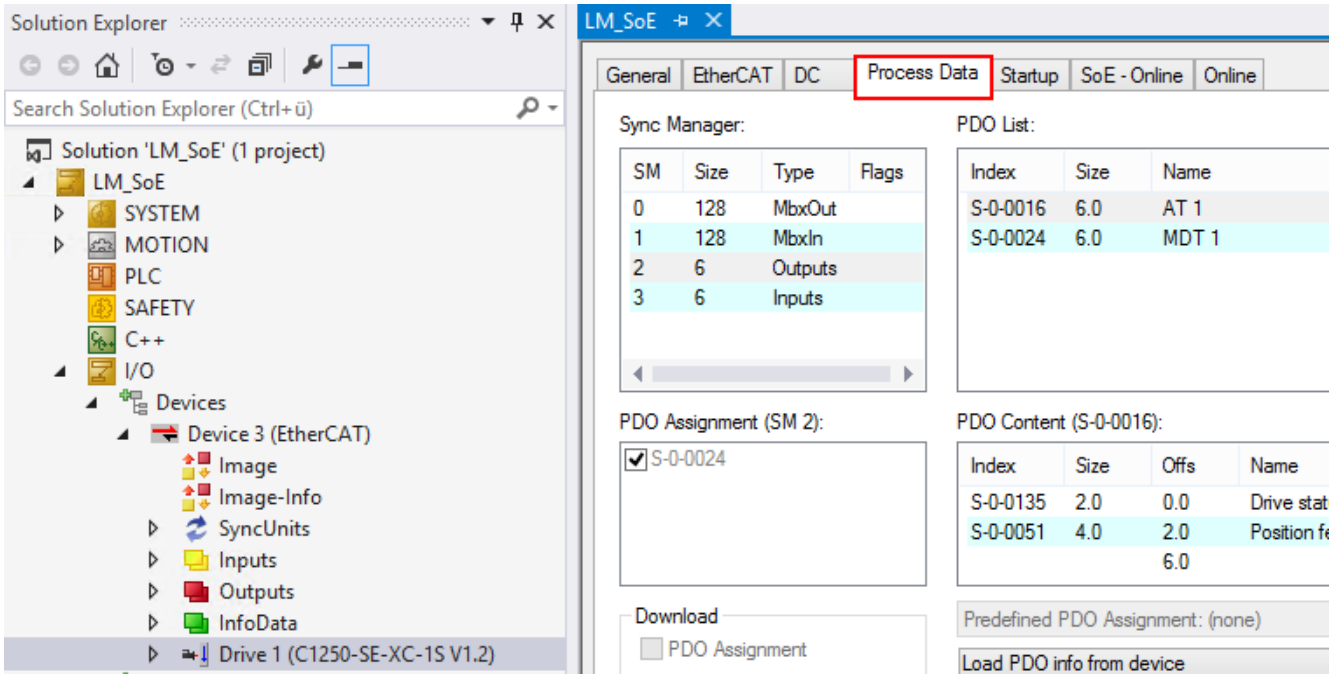
5. You should now have a new axis in your NC configuration:



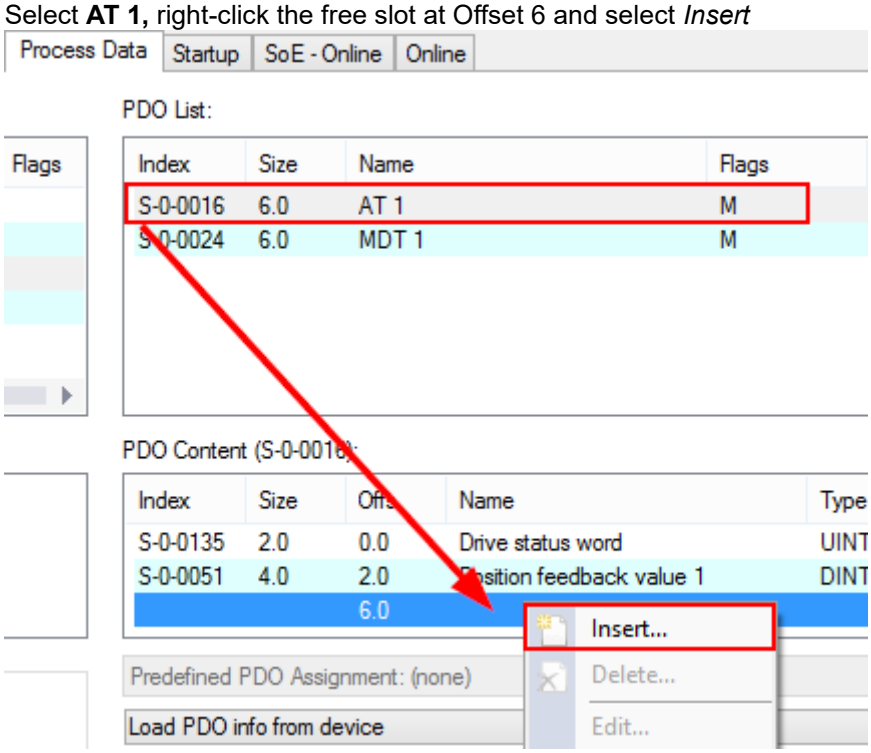

In the next chapter the configuration of the process data of the LinMot drive is shown

### 12.3.3 Setting up the Process Data of the LinMot Drive

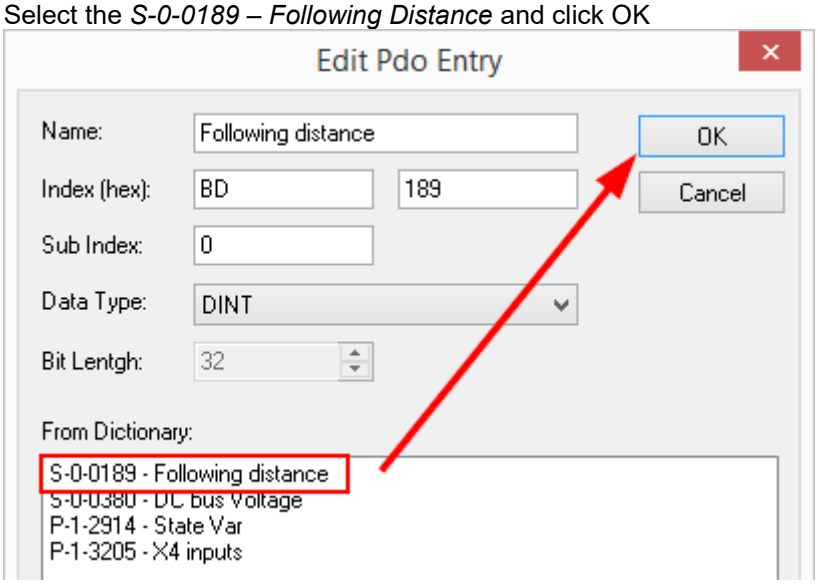
1. Move to the tab *Process Data*



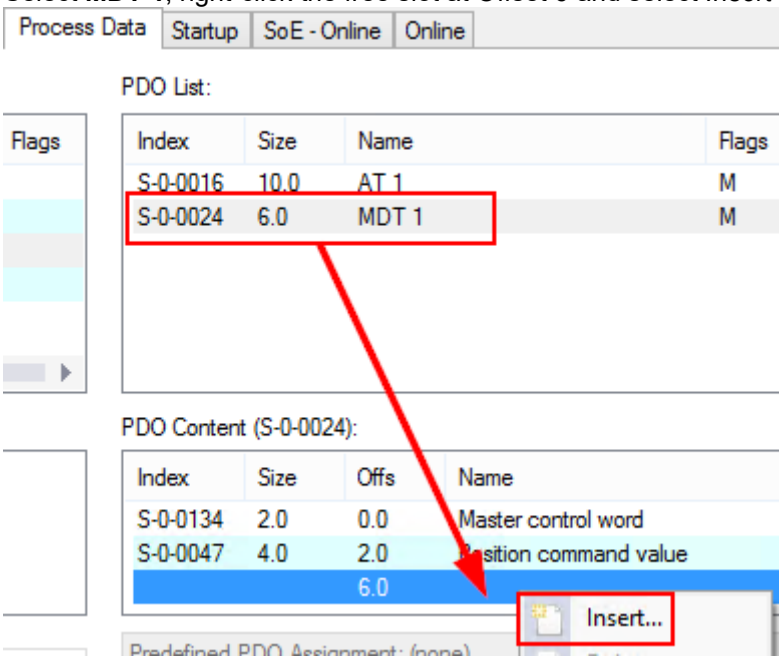

2. Select **AT 1,** right-click the free slot at Offset 6 and select *Insert*



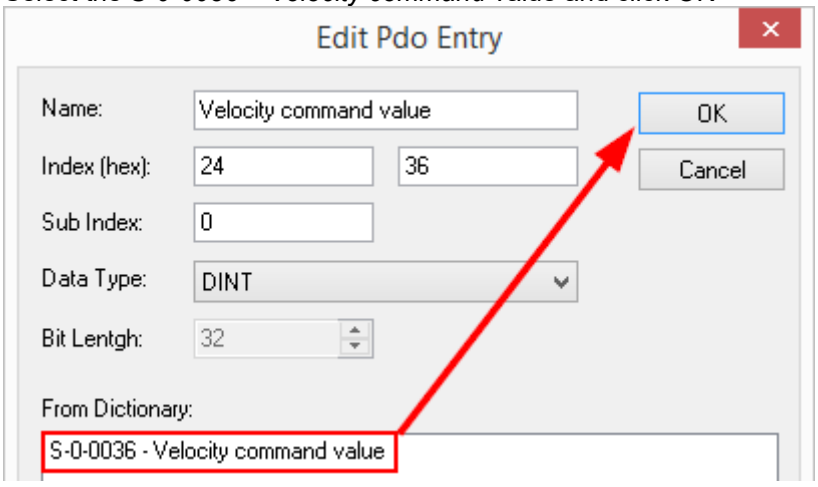
3. Select the *S-0-0189 – Following Distance* and click OK




4. Select **MDT 1**, right-click the free slot at Offset 6 and select *Insert*



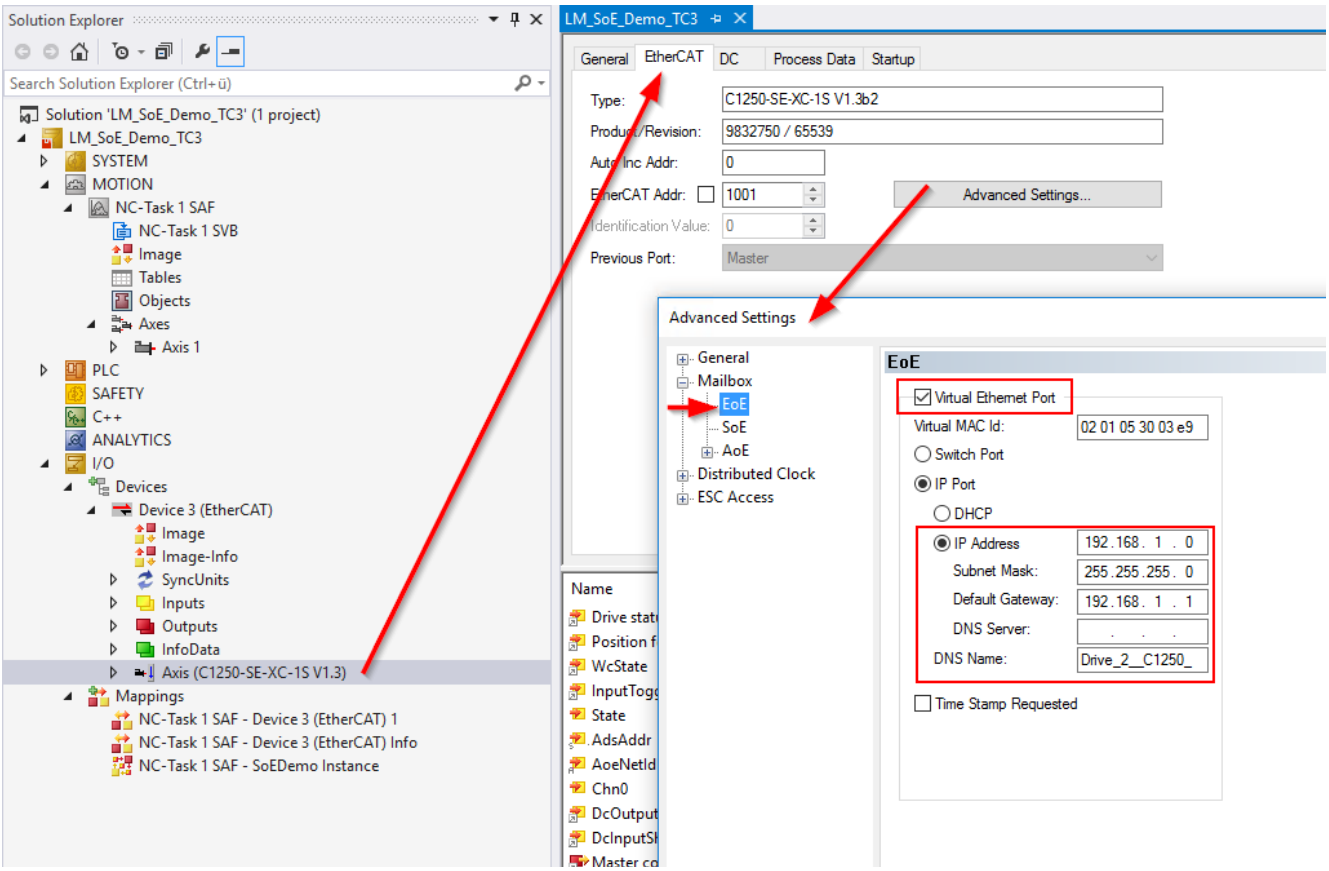

5. Select the *S-0-0036 – Velocity command* value and click OK



### 12.3.4 Check EoE settings

On supported drives (C1250-..., C1450-… & E1450-...) please check the EoE (Ethernet over EtherCAT) settings.

LinMot drives do NOT support the DHCP mode yet. A fix IP must be assigned (select *IP Address*) or the EoE feature must be completely disabled (disable Virtual Ethernet Port).



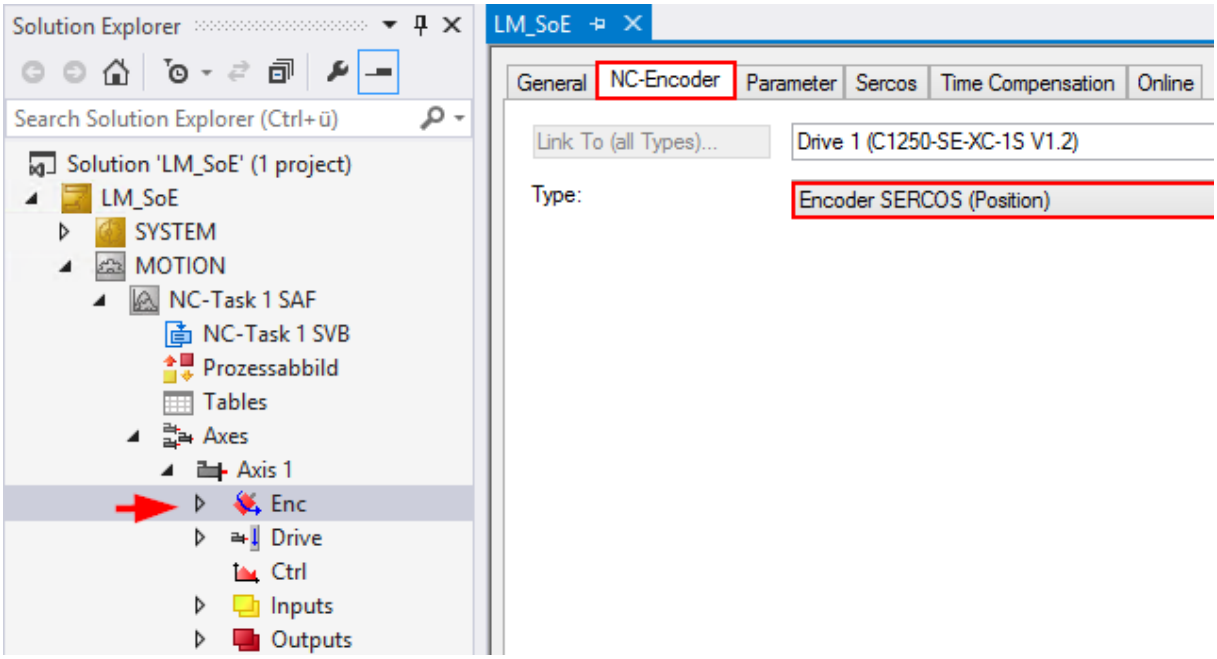

**Note:**
EoE can be used to login into the drive with LinMot-Talk directly over EtherCAT.
E.g. to change drive settings or for monitoring and tracing.

### 12.3.5 Setting up the NC Axis

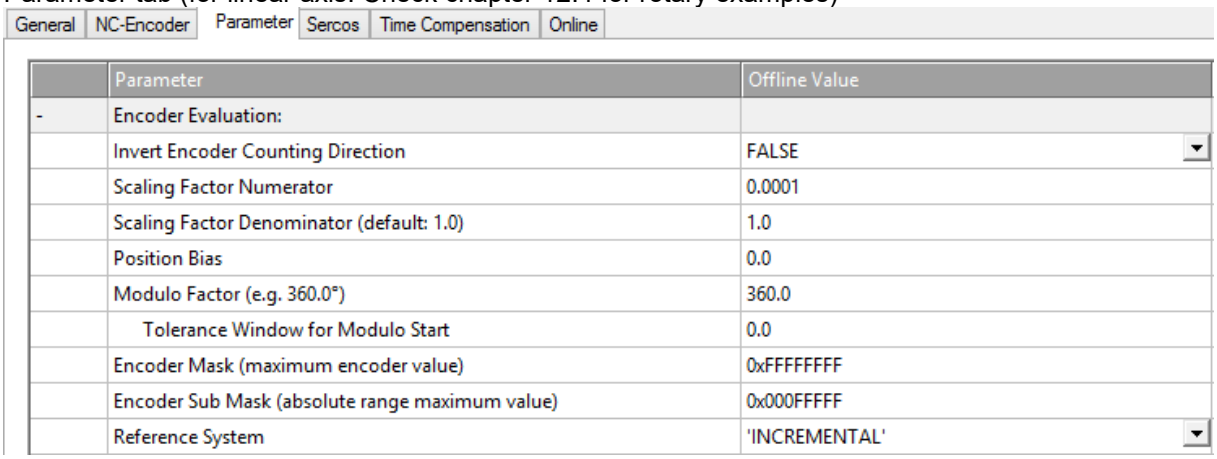
The following steps show how to basically setup the NC axis.

1. **Axis Encoder:** Setup the axis encoder as shown:

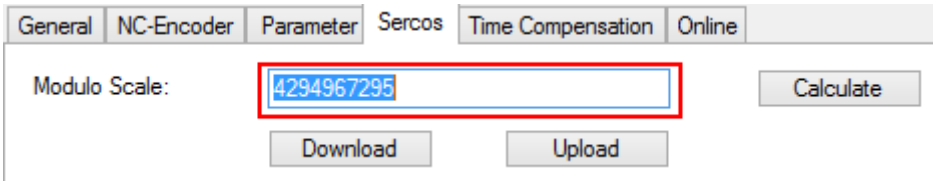
NC-Encoder tab




Parameter tab (for linear axis. Check chapter 12.4 for rotary examples)

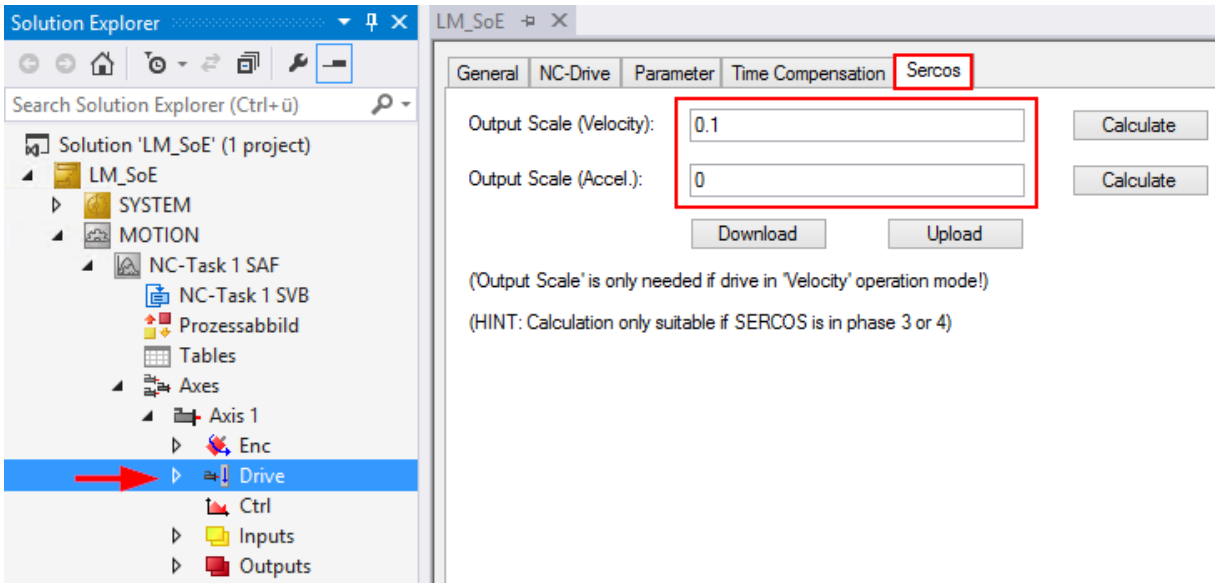



Sercos tab



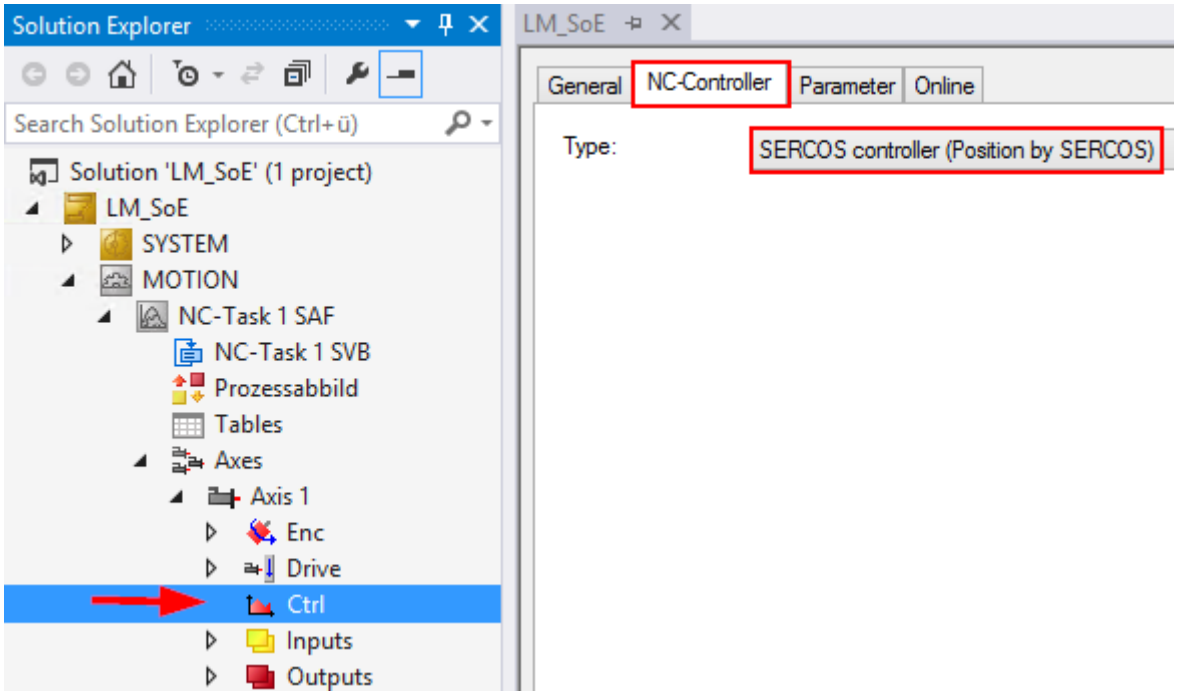
2. **Axis Drive:** Setup the drive as shown:

   Sercos tab




3. **Axis Controller:** Setup the controller as shown:

   NC-Controller tab




4. You're now ready to activate the configuration of the PLC and to enable and move the axis in the Online tab of the NC axis.

## 12.4 Encoder Parameter Examples for Linear and Rotary Motors

### 12.4.1 Linear Motors PS01 / PS10

The internal position resolution of the LinMot drive is 0.1um for linear motors.

Therefor the following parameters must be set accordingly:
- *Scaling Factor Numerator* must be set to **0.0001** (=1mm / 10'000 Increments)
- *Scaling Factor Denominator* is left to **1.0**

General | NC-Encoder | Parameter | Sercos | Time Compensation | Online

| | Parameter | Offline Value |
|---|---|---|
| - | Encoder Evaluation: | |
| | Invert Encoder Counting Direction | FALSE |
| | Scaling Factor Numerator | 0.0001 |
| | Scaling Factor Denominator (default: 1.0) | 1.0 |
| | Position Bias | 0.0 |
| | Modulo Factor (e.g. 360.0°) | 360.0 |
| | Tolerance Window for Modulo Start | 0.0 |
| | Encoder Mask (maximum encoder value) | 0xFFFFFFFF |
| | Encoder Sub Mask (absolute range maximum value) | 0x000FFFFF |
| | Reference System | 'INCREMENTAL' |

### 12.4.2 Rotary Motors RS01 & RS02

The LinMot RS01 motors (rotary part of LinMot PR01 & PR02 linear rotary motors) have a single turn absolute encoder and by default 360'000 Ticks (Increments) per motor revolution.

Therefor the following parameters must be set accordingly:
- *Scaling Factor Numerator* must be set to **0.001** (=360° / 360'000 Increments)
- *Scaling Factor Denominator* is left to **1.0**

General | NC-Encoder | Parameter | Sercos | Time Compensation | Online

| | Parameter | Offline Value |
|---|---|---|
| - | Encoder Evaluation: | |
| | Invert Encoder Counting Direction | FALSE |
| | Scaling Factor Numerator | 0.001 |
| | Scaling Factor Denominator (default: 1.0) | 1.0 |
| | Position Bias | 0.0 |
| | Modulo Factor (e.g. 360.0°) | 360.0 |
| | Tolerance Window for Modulo Start | 0.0 |
| | Encoder Mask (maximum encoder value) | 0xFFFFFFFF |
| | Encoder Sub Mask (absolute range maximum value) | 0x00057E40 |
| | Reference System | 'ABSOLUTE SINGLETURN RANGE (with single overflow)' |

### 12.4.3 Rotary Motors EC02

The LinMot EC02 motors have a single turn absolute encoder and by default 524’288 (2^19) Ticks (Increments) per motor revolution.

Therefor the following parameters must be set accordingly:

- *Scaling Factor Numerator* must be set to **0.006866455078125**
  (=360° / 524’288 Increments * 10)
- *Scaling Factor Denominator* is set to **10.0** (=1.0 * 10)

Factor 10 is required to have the full resolution in the *Scaling Factor Numerator*

General | NC-Encoder | Parameter | Sercos | Time Compensation | Online

| | Parameter | Offline Value |
|---|---|---|
| - | Encoder Evaluation: | |
| | Invert Encoder Counting Direction | FALSE |
| | Scaling Factor Numerator | 0.006866455078125 |
| | Scaling Factor Denominator (default: 1.0) | 10.0 |
| | Position Bias | 0.0 |
| | Modulo Factor (e.g. 360.0°) | 360.0 |
| | Tolerance Window for Modulo Start | 0.0 |
| | Encoder Mask (maximum encoder value) | 0xFFFFFFFF |
| | Encoder Sub Mask (absolute range maximum value) | 0x00080000 |
| | Reference System | 'ABSOLUTE SINGLETURN RANGE (with single overflow)' |

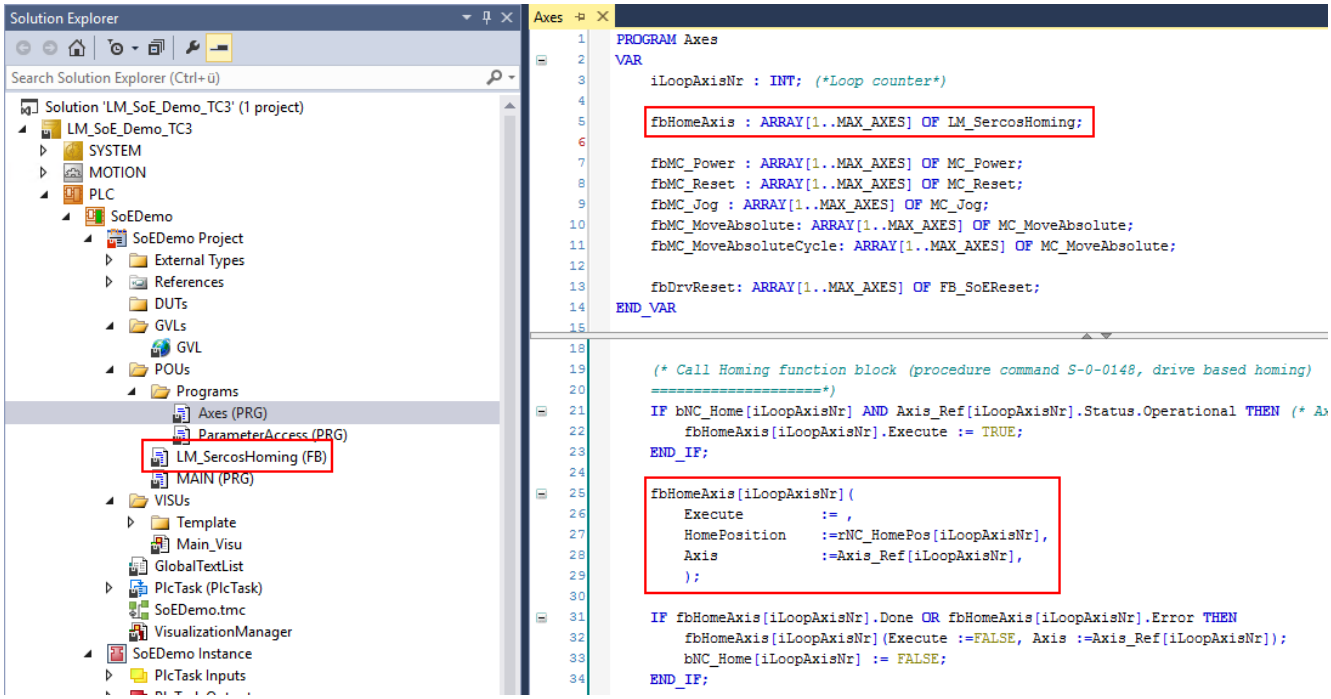
### 12.4.4 Other motors

For other motors the principle is the same to calculate the Numerator, Denominator and other encoder parameters.

## 12.5 Homing

To home the axis, the drive-controlled homing procedure command (IDN S-0-0148) can be used.

The example project (see chapter 12.1 Overview) contains a function block (LM_SercosHoming) to do the drive-controlled homing.



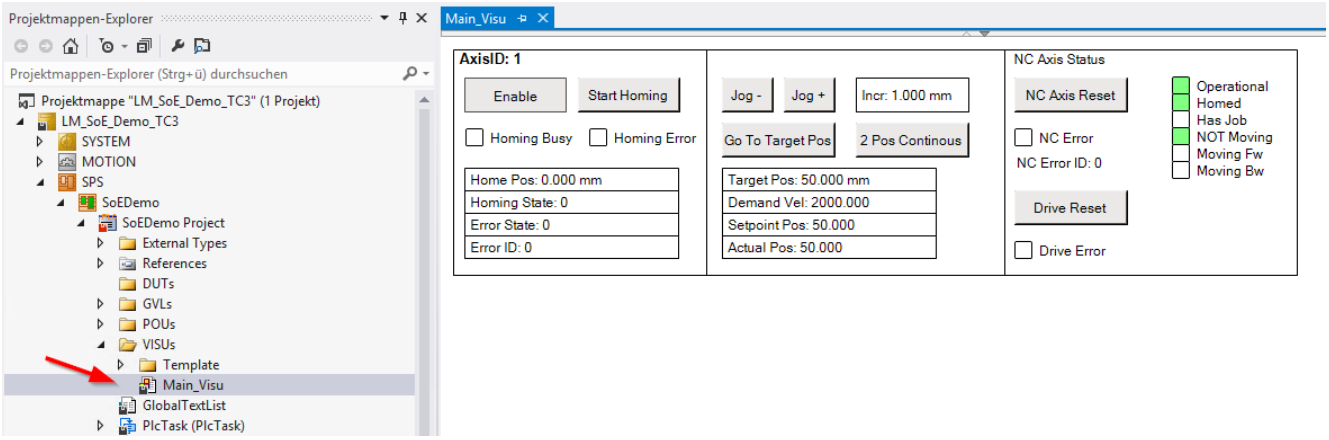

Inputs of the function block:

| | |
|---|---|
| HomePosition | Position the NC axis actual position is set to after homing is done |
| Axis | Axis reference of the NC axis |

The LM_SercosHoming is using the function block *FB_GetAxisAMsAddr* which is part of the Beckhoff library *Tc2_NC (TwinCAT 3) / TcNC (TwinCAT 2)*.

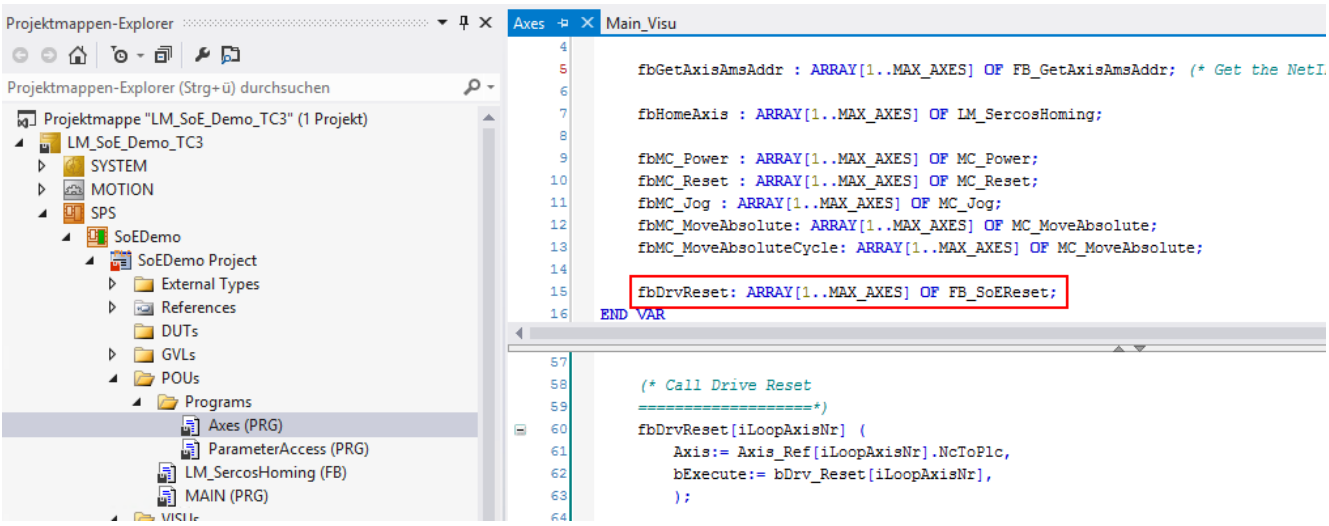
## 12.6 Visualization

The example project contains a visualization to control the NC axis (including the mentioned homing function block in the chapter before):




## 12.7 Drive Reset

To acknowledge an error of a SoE drive the function block *FB_SoEReset* must be used on TwinCAT systems. This function block is part of the Beckhoff library *Tc2_NcDrive (TwinCAT 3) / TcMC2Drive (TwinCAT 2)*.




**Note:**
Use FB_SoEReset from Beckhoff to reset drive errors.

## 12.8 Asynchronous Parameter Access

Parameters of the LinMot drive can be accessed using the Beckhoff function block *FB_EcSoeRead* and *FB_EcSoeWrite* which are part of the *Tc2_EtherCAT (TwinCAT 3) / EtherCAT (TwinCAT 2)* library.

More information about this function blocks can be found in the Beckhoff Information System: http://infosys.beckhoff.com (search for *FB_EcSoeRead & FB_EcSoeWrite*

Example calls of this function blocks can be found in the example project (see chapter 12.1)

MAIN | ParameterAccess | Library Manager | Axes | Main

```
PROGRAM ParameterAccess
VAR
    iLoopAxisNr :INT; (*Loop counter*)

    SoERead     :ARRAY[1..MAX_AXES] OF FB_EcSoERead;
    SoEWrite    :ARRAY[1..MAX_AXES] OF FB_EcSoEWrite;

    UPID        :ARRAY[1..MAX_AXES] OF WORD;

    DataIN      :ARRAY[1..MAX_AXES] OF DINT;
    DataOUT     :ARRAY[1..MAX_AXES] OF DINT;
END VAR
```

```
FOR iLoopAxisNr := 1 TO MAX_AXES DO (*Call the following function blocks per axis*)

SoERead[iLoopAxisNr](
    sNetId:=F_CreateAmsNetId(g_aAmsAddr[iLoopAxisNr].netId),
    nSlaveAddr:=g_aAmsAddr[iLoopAxisNr].port,
    nIdn:= 16#8000 + UPID[iLoopAxisNr],
    nElement:= 16#40, (*Value*)
    nDriveNo:= DWORD_TO_BYTE(Axis_Ref[iLoopAxisNr].NcToPlc.AxisId),
    bCommand:= ,
    pDstBuf:= ADR(DataIN[iLoopAxisNr]),
    cbBufLen:= SIZEOF(DataIN[iLoopAxisNr]),
    bExecute:= ,
    tTimeout:= ,
    bBusy=> ,
    bError=> ,
    nErrId=> );

SoEWrite[iLoopAxisNr](
    sNetId:=F_CreateAmsNetId(g_aAmsAddr[iLoopAxisNr].netId),
    nSlaveAddr:=g_aAmsAddr[iLoopAxisNr].port,
    nIdn:=16#8000 + UPID[iLoopAxisNr],
    nElement:=16#40, (*Value*)
    nDriveNo:= DWORD_TO_BYTE(Axis_Ref[iLoopAxisNr].NcToPlc.AxisId),
    bCommand:= ,
    pSrcBuf:= ADR(DataOUT[iLoopAxisNr]),
    cbBufLen:= SIZEOF(DataOUT[iLoopAxisNr]),
    bExecute:= ,
    tTimeout:= ,
    bBusy=> ,
    bError=> ,
    nErrId=> );
```

**sNetID, nSlaveAddr:** AmsNetID & port of the drive.
These can be read using the function block *FB_GetAxisAmsAddr* which is part of the Beckhoff library *Tc2_NC (TwinCAT 3) / TcNC (TwinCAT 2)*.

**nDriveNo:** NC axis ID

**nIdn:** Parameter address to be accessed = **16#8000** + UPID
(E.g., Maximal Current (UPID 16#13A6) ▶ **nIdn** = 16#8000 + 16#13A6 = 16#93A6)

**nElement:** Element of the parameter to be accessed. The following values are allowed

| Value | Description |
|---|---|
| 0x01 | Data Status |
| 0x02 | Name (read only) |
| 0x04 | Attribute |
| 0x08 | Unit |
| 0x10 | Minimum |
| 0x20 | Maximum |
| **0x40** | **Value** |
| 0x80 | Default |

Reading and writing the value (nElement := 16#40) accesses the **RAM** value of the UPID.
Writing to the default value (nElement := 16#80) accesses the **ROM** value of the UPID.

**pDstBuf, pSrcBuf:** Address to the variable containing the value to be read / written (must be of size 4 Bytes!)

cbBufLen: Size of the variable of the value to be read / written (must be of size 4 Bytes!)

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the life-time of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

**Note:**
Use *FB_EcSoeRead* and *FB_EcSoeWrite* from Beckhoff to access drive parameters.

**Advanced: Stop / Start MC_SW and reboot drive**
The parameter UPID 2080 allows:
1. stopping the MC_SW for flash access or to setup the drive from the PLC
2. starting the MS_SW after flash access and to load changed ROM values to RAM
3. reset the drive.

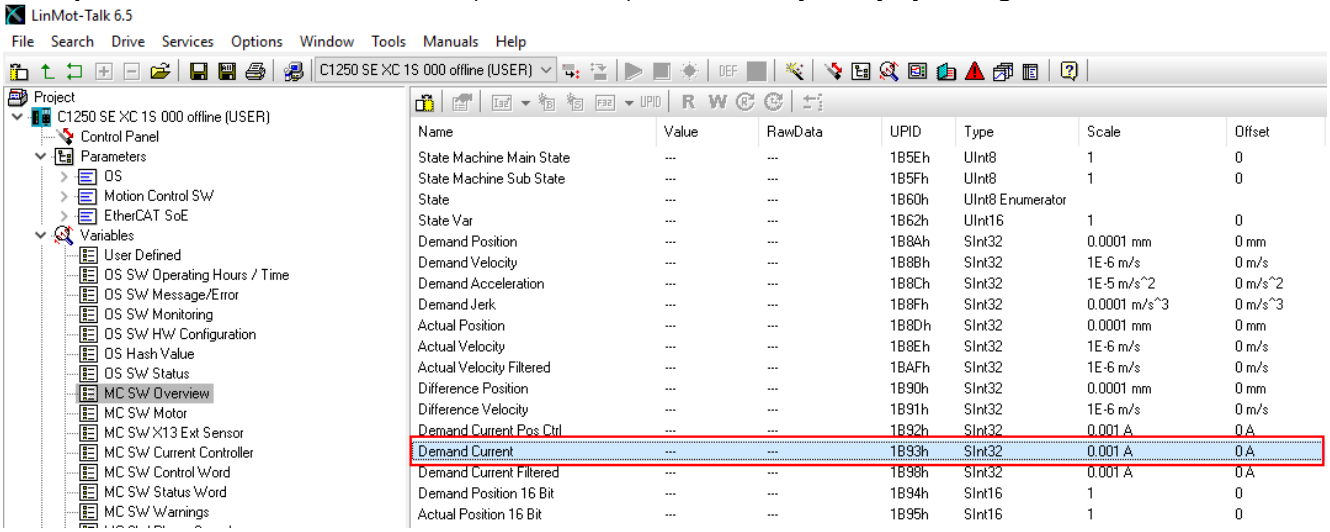
Write to RAM (nElement 0x40) and value 0 -> Stop MC_SW
Write to RAM (nElement 0x40) and value 1 -> Start MC_SW
Write to ROM (nElemen 0x80) and value 1 -> Reboot drive

For every access (read or write) the length of the variable connected to pDstBuf/pSrcBuf must be of **size 4 Bytes** (e.g., datatypes DWORD, DINT, UDINT).
If you access a parameter/variable of a different size (e.g., INT, SINT, …) then make a typecast afterwards.

## 12.9 Add Additional Parameters or Variables to the Process Data (MDT 1 / AT 1)

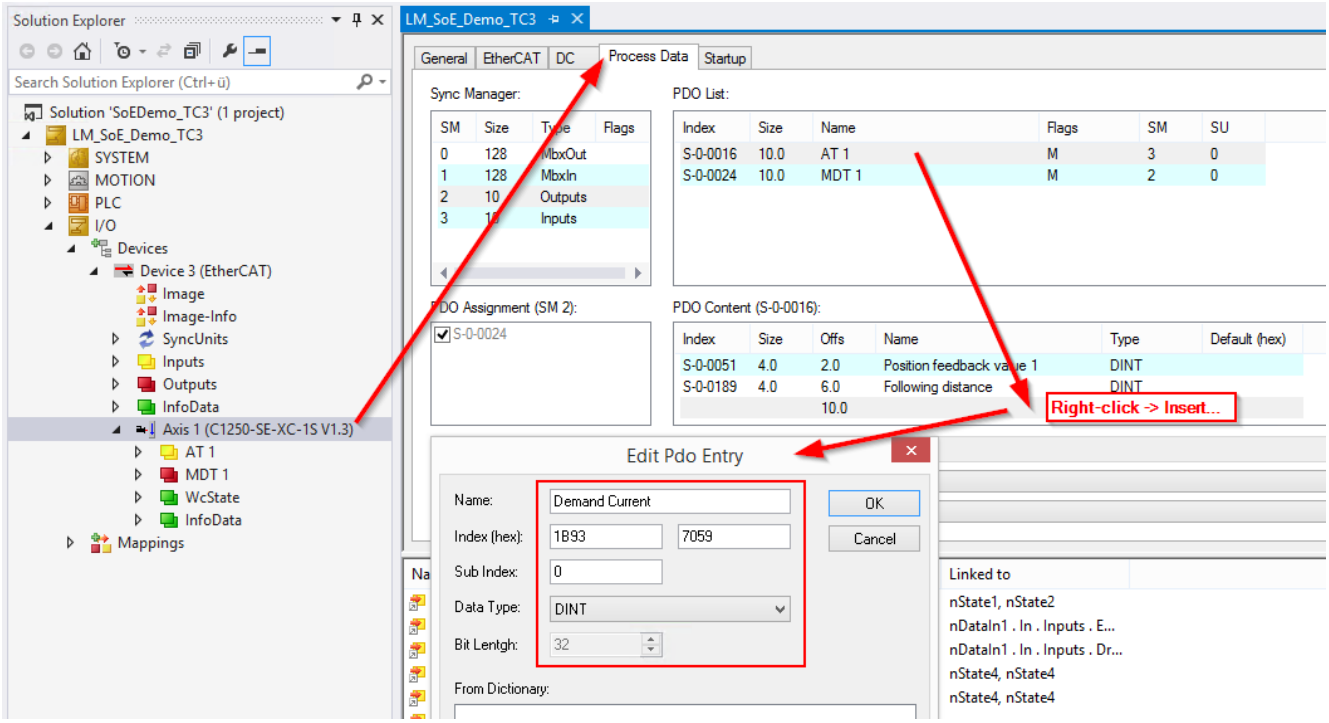
It is possible to add additional parameters or variables of the drive to PDO for cyclical write or read access.

**Example 1:** Read the *Demand Current (UPID 1B93h)* of the drive cyclically by adding it to AT 1:

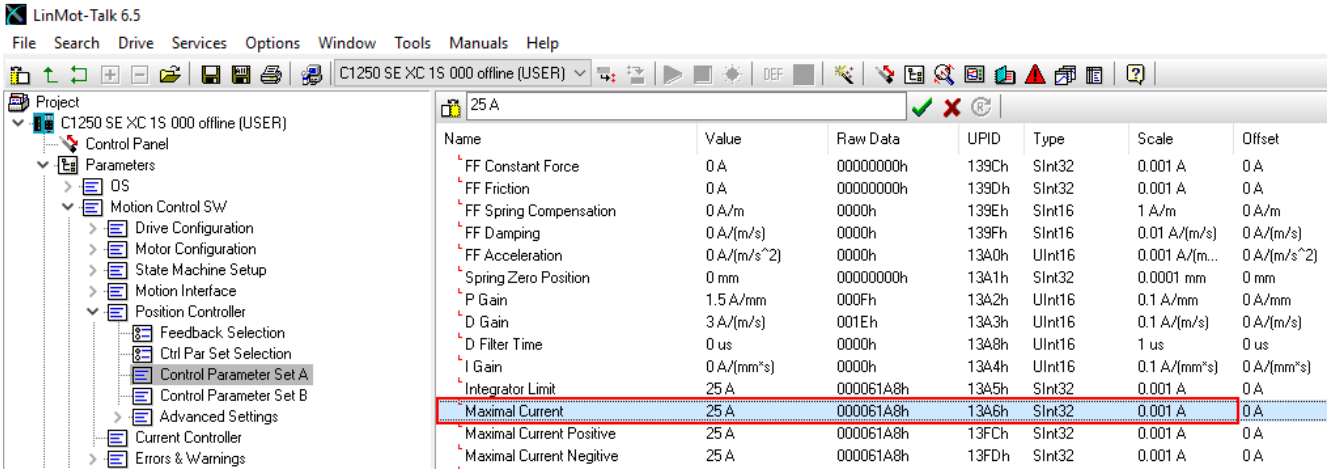



**Name:** Demand Current
**UPID:** 1B93h
**Type:** SInt32 = DINT
**Scale:** 0.001 A

1. In the Process Data select *AT 1* from the PDO List.
2. Right-click on the empty line where only an offset is shown
3. In the *Edit PDO Entry* window enter the name, the UPID as *Index* and set the *Data Type* to DINT (as the Demand Current is of type SINT32)
4. Click OK to finish

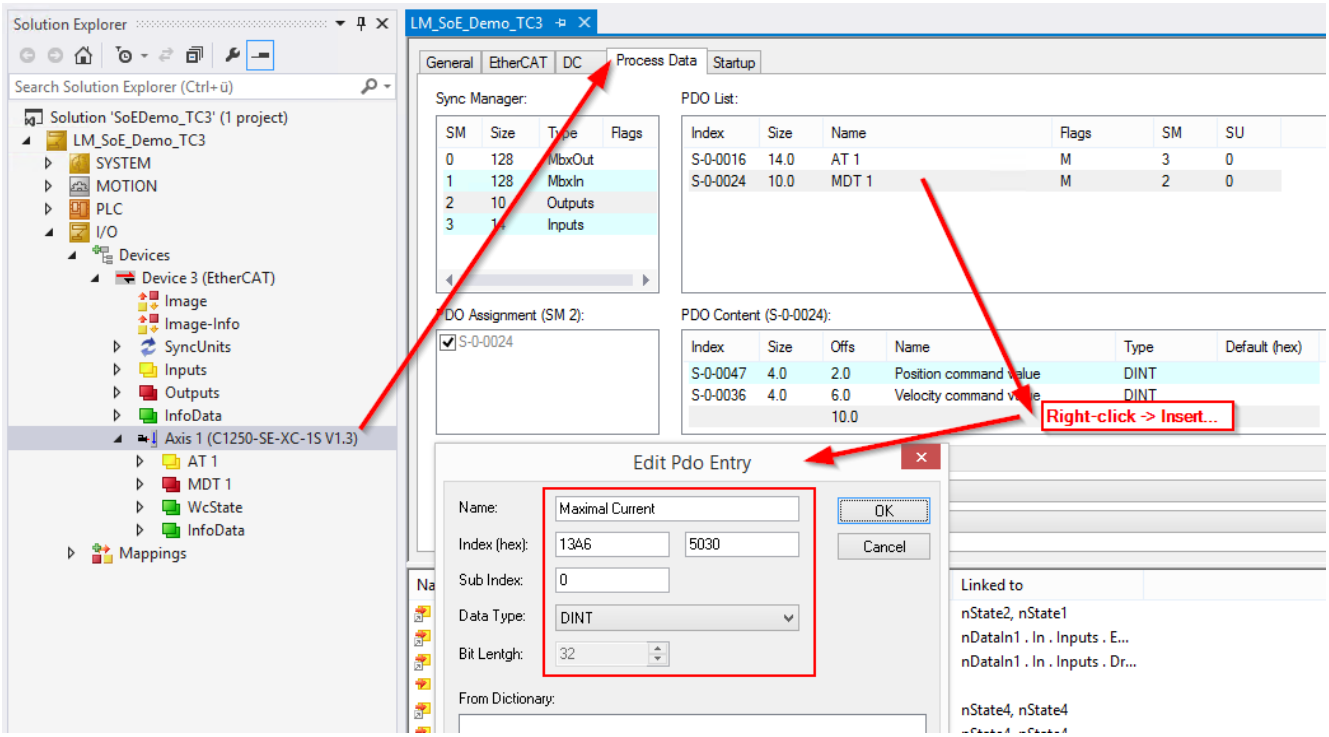


**Example 2:** Write the *Maximal Current (UPID 13A6h)* of the drive cyclically by adding it to MDT 1:

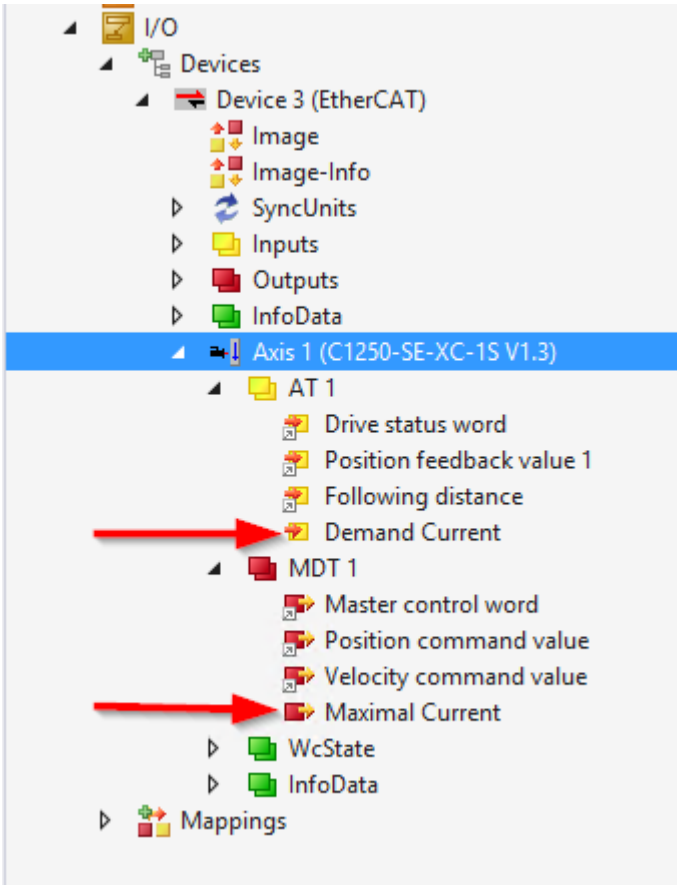



**Name:** Maximal Current
**UPID:** 13A6h
**Type:** SInt32 = DINT
**Scale:** 0.001 A

1. In the Process Data select *MDT 1* from the PDO List.
2. Right-click on the empty line where only an offset is shown
3. In the *Edit PDO Entry* window enter the name, the UPID as *Index* and set the *Data Type* to DINT (as the Maximal Current is of type SINT32)
4. Click OK to finish



Now you can see both added parameters in the solution tree and link them:



# 13 Drive Profile: Schneider Electric PacDrive 3, FSP_DRIVE

## 13.1 Overview

This chapter shows how a LinMot drive with SERCOS (FSP_DRIVE) interface (e.g., C1250-SC-XC-1S) can be integrated and setup in a Schneider Electric PacDrive environment.

**Download:**
An example project can be downloaded from (The example is based on the QuickMotionProgramming example from Schneider Electric):
http://download.linmot.com/plc_lib/examples/Schneider (named *LMC_LinMot_AxisModule_Example_...*)

Components used:

- LinMot C1250-SC-XC-0S-000 (article number 0150-1887) with firmware 6.8 Build 20191003
- LMC 400C with firmware V01.56.21.13
- SoMachine Motion Logic Builder V4.41 (V4.4 SP1)

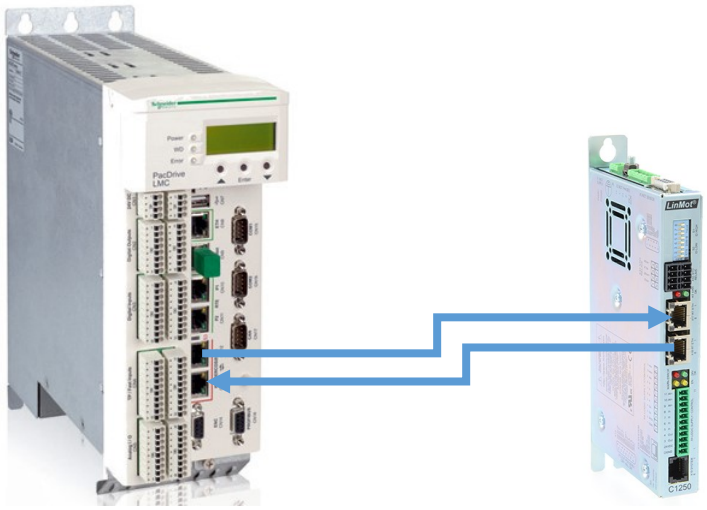


Image Source: http://www.schneider-electric.com/

Sercos is an open IEC standard universal bus for Ethernet-based real-time communication.

For further information on sercos please visit:
http://www.sercos.org/

## 13.2 Configuration and Connection of the LinMot Drive

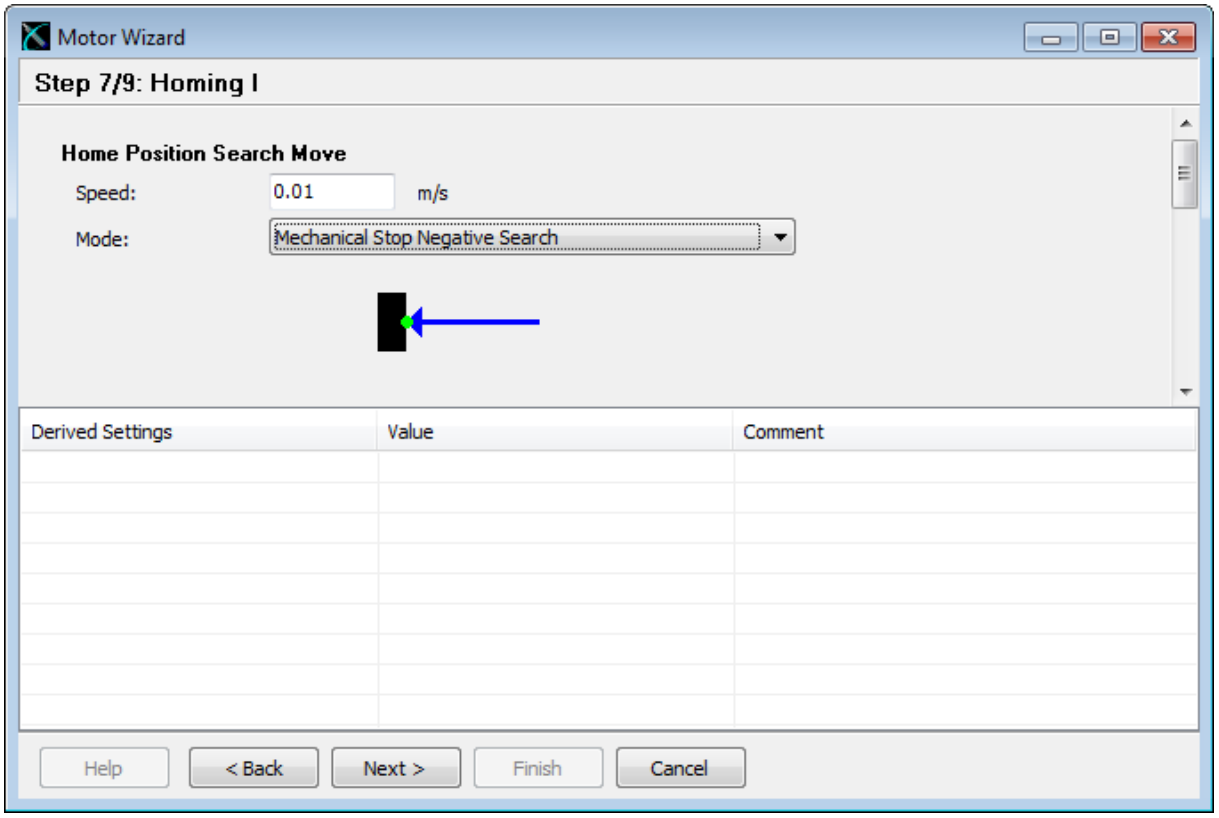
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 13.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select a homing mode in the Motor Wizard if you want to use the drive based homing as described in chapter 13.4.2 (e.g. *Mechanical Stop Negative Search*):




If the homing should be done as shown in chapter *13.4.1 Homing using the Schneider template (FB_Home)* then choose “No Drive Homing”.

See Appendix I: Basic Position Control Loop Tuning

### 13.2.2 Sercos Connection

The drive is connected to the sercos network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-SC-UC drive. On all other LinMot drives supporting sercos the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.

| X17 - X18 | RealTime Ethernet 10/100 Mbit/s | |
|---|---|---|
|  | X17 RT ETH In | SERCOSIII P1 |
| | X18 RT ETH Out | SERCOSIII P2 |
| RJ-45 | | |

### 13.2.3 Sercos Address

The sercos address is set via the two ID-switches S1 and S2, where S1 sets the high digit and S2 the low digit. The address can have a value between 1 (01h) and 255 (FFh).

Set the sercos address by S1 & S2

| S1 - S2 | Address Selectors | | |
|---|---|---|---|
| E1100<br>E1200 V1 | E1200 V2<br>E1400<br>C1x00 | | |
|  |  | S1 (5 .. 8) | Bus ID High (0 ... F). Bit 5 is LSB, bit 8 MSB. |
|  | | S2 (1 .. 4) | Bus ID High (0 ... F). Bit 1 is LSB, bit 4 MSB. |

**Note:**
The sercos address has no use with some identification modes like topology-based addressing. If both address selectors are set to 0, the topology address is used as the sercos address when topology-based addressing is configured in the PLC.

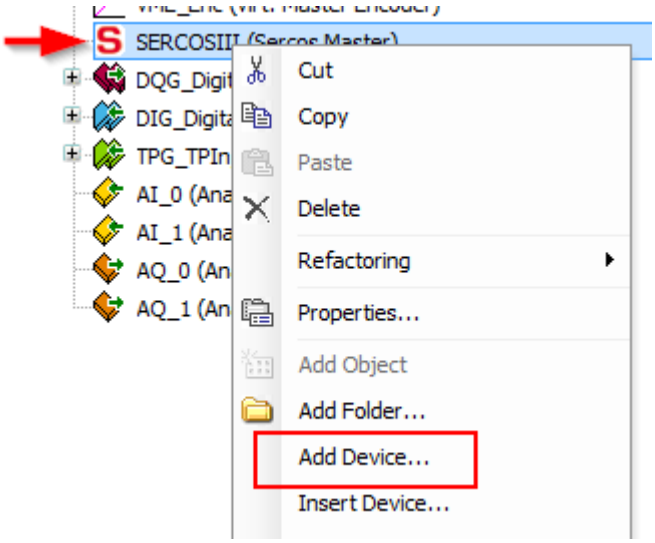
## 13.3 PLC Setup

### 13.3.1 General Device Properties

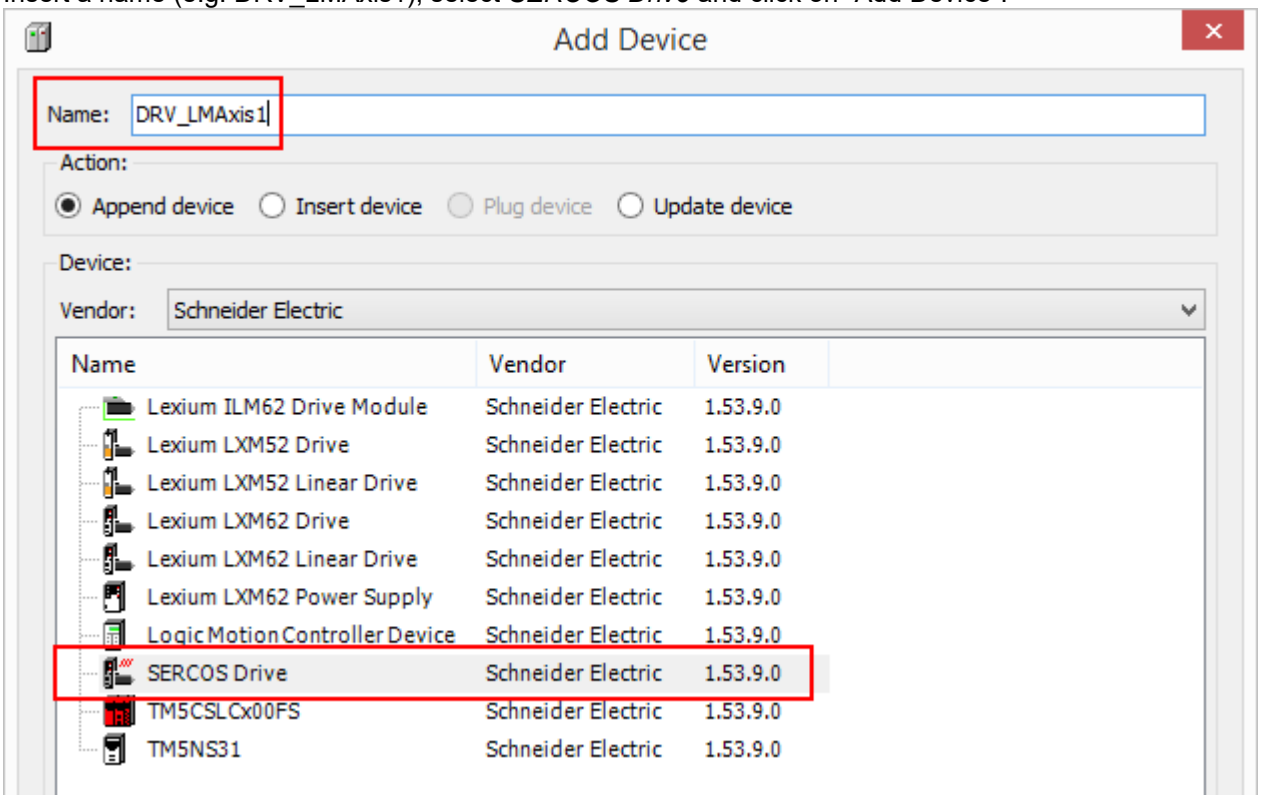
| Device Properties | Value | |
|---|---|---|
| sercos generation | Third generation | |
| sercos version | SC: sercos III V1.1.2; MI: sercos III V1.3.2 | |
| Hot-Plug support | only on MI-devices | |
| SERCON100 image version | V2.11 (SC only) | |
| Supported profiles and telegram types | E1x50-SC-xx-xx, C1x50-SC-xx-xx | FSP_DRIVE, FSP_IO |
| | C1x50-MI-xx-xx, C1251-MI-XC-2S | FSP_DRIVE |
| | B8050-ML-SC, MB8050-ML-SC | FSP_IO |
| Minimal sercos cycle time | 250 µs | |
| Vendor Device ID | '0150-1764' (E1250-SC-UC)<br>'0150-1785' (E1450-SC-QN-0S)<br>'0150-2357' (E1450-SC-QN-1S)<br>'0150-1881' (B8050-ML-SC)<br>'0150-2032' (MB8050-ML-SC)<br>'0150-1887' (C1250-SC-XC-0S-000, C1250-SC-XC-0S-C00)<br>'0150-2349' (C1250-SC-XC-1S-000, C1250-SC-XC-1S-C00)<br>'0150-30149' (C1250-MI-XC-0S-000, C1250-MI-XC-0S-C00)<br>'0150-30169' (C1250-MI-XC-1S-000, C1250-MI-XC-1S-C00)<br>'0150-30009' (C1251-MI-XC-2S-000, C1251-MI-XC-2S-C00) | |
| Vendor Code | 342 (0156h) | |

### 13.3.2 Add the LinMot Drive to the Sercos

Right-click on SERCOSIII (Sercos Master) and select *Add Device*:




Insert a name (e.g. DRV_LMAxis1), select *SERCOS Drive* and click on “Add Device”:




Double-click on the new added device to open the configuration.

### 13.3.3 Device Configuration

#### 13.3.3.1 Identification

All identification modes are support (SerialNumberController (0), SerialNumberMotor (1), TopologyAddress (2), ApplicationTyp (3) and SercosAddress (4)).

Following an example setup where the device is identified by *TopologyAddress (2)*
ConfiguredTopologyAddress: **1** (As it is the first device in the ring in this project)
VendorCode: **342**
VendorDeviceID: **'0150-2349'** (LinMot C1250-SC-XC-1S-000, for others check chapter 13.3.1)

| Parameter | Type | Value | Default Value |
|---|---|---|---|
| Identification | | | |
| Name | STRING(40) | '' | '' |
| WorkingMode | Enumeration of DINT | real / 1 | virtual / 0 |
| WorkingState | Enumeration of DINT | virtual / 0 | virtual / 0 |
| IdentifyDevice | Enumeration of BOOL | off / 0 | off / 0 |
| IdentificationMode | Enumeration of DINT | TopologyAddress / 2 | TopologyAddress / 2 |
| ConfiguredTopologyAddress | UINT(1..512) | 1 | 1 |
| TopologyAddress | UINT | | |
| ConfiguredApplicationType | STRING(40) | '' | '' |
| ApplicationType | STRING(40) | '' | '' |
| ConfiguredSercosAddress | UINT(1..512) | 100 | 100 |
| SercosAddress | UINT(1..299) | 1 | 1 |
| ConfiguredSerialNumber | STRING(80) | '' | '' |
| ConfiguredSerialNumberMotor | STRING(20) | '' | '' |
| ConfiguredVendorCode | WORD | 342 | 0 |
| VendorCode | WORD | | |
| ConfiguredVendorDeviceId | STRING(80) | '0150-2349' | 'SercDrv' |
| VendorDeviceId | STRING(80) | '' | '' |

#### 13.3.3.2 Motor/Mechanic using a LinMot linear motor

As unit millimeter [mm] is used.

The MaxRPM parameter limits the maximal possible velocity to around 1m/s when using 1.0 as *FeedConstant*.

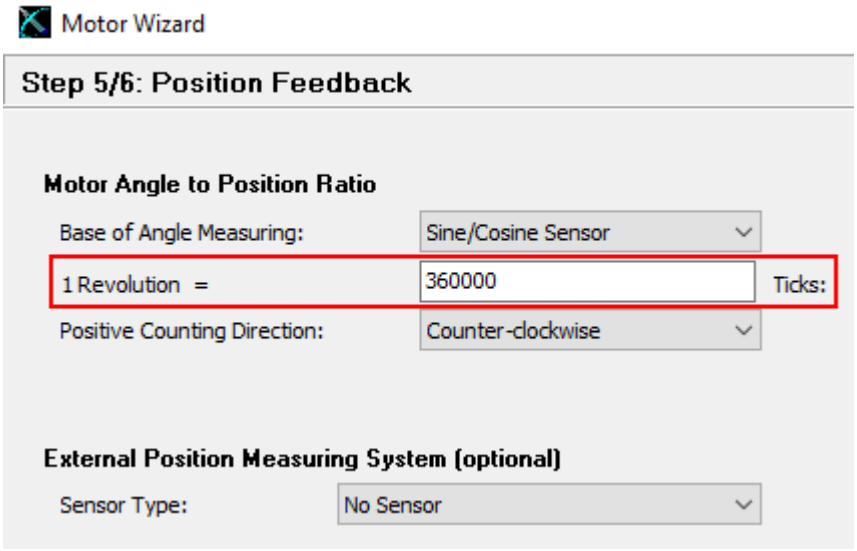
Therefore, the *FeedConstant* is set to **1'000** and the *FeedbackResolution* to **10'000'000**.

| Parameter | Value | Unit |
|---|---|---|
| GearIn | 1 | - |
| GearOut | 1 | - |
| Direction | right | - |
| FeedConstant | 1'000 | [Units/Revolution] |
| FeedbackResolution | 10'000'000 | [Increments] |
| MaxRPM | See below | [1/min] |
| ModuloValue | 0 | [Increments] |
| J total | Leave on default or set as required | [kg*cm^2] |
| Torque Constant | Leave on default or set as required | [0,001*Nm/A] |

**MaxRPM:**
Max. Speed (@72VDC, see motor datasheet): e.g., 5.2 [m/s] * 60 => 312 [m/min]
Input value for MaxRPM: 312 [m/min]

#### 13.3.3.3 Motor/Mechanic using a rotary motor

As unit degree [°] is used. As rotary motor, the *FeedbackResolution* depends on the settings done within the LinMot-Talk Motor Wizard. It must be set to the same value as Ticks per revolution in the Motor Wizard.



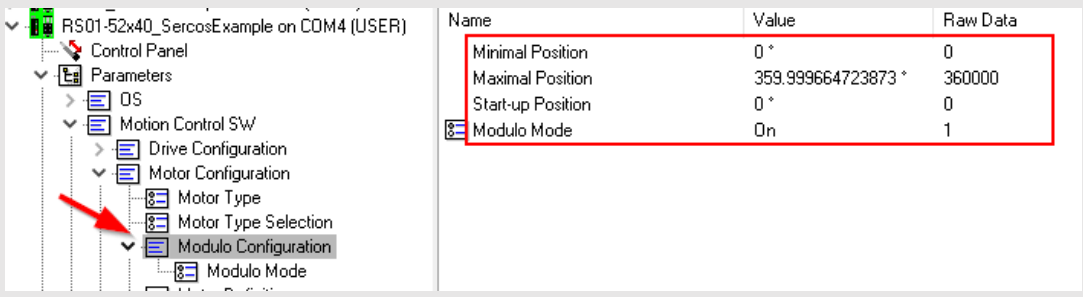

For a LinMot **RS01** (rotary part of the linear rotary motor PR01) the *FeedbackResolution* is **360'000**.
For a LinMot **EC02** the *FeedbackResolution* is **524'288**.

If the motor should be used as modulo axis (1 revolution) the *ModuloValue* must be set accordingly (same as *FeedbackResolution*).

Example for a LinMot **RS01** motor as modulo axis without gearbox:

| Parameter | Value | Unit |
|---|---|---|
| GearIn | 1 | - |
| GearOut | 1 | - |
| Direction | right | - |
| FeedConstant | 360.0 | [Units/Revolution] |
| FeedbackResolution | 360'000 | [Increments] |
| MaxRPM | Take from motor datasheet | [1/min] |
| ModuloValue | 0 | [Increments] |
| J total | Leave on default or set as required | [kg*cm^2] |
| Torque Constant | Leave on default or set as required | [0,001*Nm/A] |

**Attention:**
From drive firmware 6.9 it is possible to use the motor in modulo mode. The modulo mode must be enabled and setup in drive. Make sure that the *ModuloValue* in the PLC is set to the same value as the RawData of *Maximal Position* in the drive. E.g.: 360'000.



#### 13.3.3.4 Realtime channel

Following the recommended configuration of the real-time channel.

If the Realtime channel is configured with a list of IDNs the LinMot StatusWord, StateVar, the velocity feedback (S-0-0040), torque/force feedback (S-0-0084) and the effective current (S-0-0389) can also be mapped directly as part of the real-time process data.

In this example in the MDT also the **Bipolar torque limit** value (S-0-0092) is mapped. It allows to limit the maximum force/torque in percent (scale = 0.1%, > value = 0 to 1000).

| Parameter | Value |
|---|---|
| TelegramType | 7 |
| PrimaryOperationMode | 3 |
| ConfigurationListAT | 'S-0-0051.0.0;P-0-0100.0.0;P-0-0101.0.0;S-0-0040.0.0;S-0-0084.0.0;S-0-0389.0.0' |
| ConfigurationListATLength | 9 |
| PositionFeedbackValueOffset | 0 |
| ConfigurationListMDT | 'S-0-0047.0.0;S-0-0092.0.0' |
| ConfigurationListMDTLength | 3 |
| PositionCommandValueOffset | 0 |

| | | |
|---|---|---|
| Echtzeitkanal | | |
| TelegramType | UINT(0..7) | 7 |
| PrimaryOperationMode | UINT | 3 |
| ConfigurationListAT | STRING(254) | 'S-0-0051.0.0;P-0-0100.0.0;P-0-0101.0.0;S-0-0040.0.0;S-0-0084.0.0;S-0-0389.0.0' |
| ConfigurationListATLength | UINT(0..750) | 9 |
| PositionFeedbackValueOffset | UINT(0..748) | 0 |
| ConfigurationListMDT | STRING(254) | 'S-0-0047.0.0;S-0-0092.0.0' |
| ConfigurationListMDTLength | UINT(0..750) | 3 |
| PositionCommandValueOffset | UINT(0..748) | 0 |

**Attention:**
The order of the list's objects and its content must be exactly as stated in the table above to work properly.

Make sure the Bipolar torque limit value is not 0 or the motor cannot move.

**Note:**
Writing to the MDT from the PLC program (e.g., to write the Bipolar torque limit) could be done as follows:

```
39      // Bipolar Torque Limit (S-0-0092)
40      pBipolarLimit           : POINTER TO UINT;
41      rBipolarLimit           : REAL := 100.0; // in percent;
```

```
17  // ******************************************************************************
18  // Set Bipolar torque limit (S-0-0092) in MDT
19  // ATTENTION: Only valid with ConfigurationListMDT = 'S-0-0047.0.0;S-0-0092.0.0'
20  // - Bipolar torque limit has scaling 0.1% percent (100% = maximal force / torque of motor)
21  // - for other ConfigurationListMDT the address offset(s) must be adapted accordingly
22  // ******************************************************************************
23
24  pBipolarLimit := (DRV_LMAxis1.MDTAdr + 4); // Byte offset is 4
25  pBipolarLimit^ := REAL_TO_UINT(rBipolarLimit * 10);
26
```

## 13.4 Homing (Linear Motors)

There are several possible ways to home the axis. If the template(s) of Schneider is used the user may want to use the integrated FB_Home function block (see chapter 13.4.1). An alternative is using the drive-based homing as shown in chapter 13.4.2.

### 13.4.1 Homing using the Schneider template (FB_Home)

As homing mode *NegDirectionPosEdgeSensor* is used in this example project. As sensor the tracking deviation is evaluated. The axis moves in negative direction until it reaches the mechanical stop, and the tracking deviation increases. At a certain value this triggers *i_xSensor*.

#### 13.4.1.1 Setup

Set the homing mode:

```
55  AXM.FC_SetHomeMode( i_etModeVirtualDrive    := PDL.ET_HomeMode.SetPosAxisAndLogEncoderPosition,
56                      i_etModeRealDrive       := PDL.ET_HomeMode.NegDirectionPosEdgeSensor,
57                      iq_stAxisModuleItf      := stAxisInterface,
58                      q_etDiag                => etDiag,
59                      q_etDiagExt             => etDiagExtAXM,
60                      q_sMsg                  =>  );
```

Setup the homing mode Sensor:

```
72  (* Possible modes for homing mode Sensor:
73      PDL.ET_HomeMode.PosDirectionPosEdgeSensor
74      PDL.ET_HomeMode.NegDirectionPosEdgeSensor
75      PDL.ET_HomeMode.NegDirectionNegEdgeSensor
76      PDL.ET_HomeMode.PosDirectionNegEdgeSensor
77  *)
78  stAxisInterface.stHome.stSensor.i_lrHomePosition        := 0.0;       (* Position at the end of homing in units *)
79  stAxisInterface.stHome.stSensor.i_lrVel                 := 5.0;       (* Homing ceiling velocity in units/s *)
80  stAxisInterface.stHome.stSensor.i_lrAcc                 := 1000.0;    (* Homing acceleration in units/s^2 *)
81  stAxisInterface.stHome.stSensor.i_lrDec                 := 1000.0;    (* Homing deceleration in units/s^2 *)
82  stAxisInterface.stHome.stSensor.i_lrJerk                := 10000.0;   (* Homing jerk in units/s^3 *)
83  stAxisInterface.stHome.stSensor.i_lrOffset              := 10.0;      (* Offset from home signal in units *)
84  stAxisInterface.stHome.stSensor.i_lrMaxTravel           := 720.0;     (* Max. travel for referencing by normal input *)
85  stAxisInterface.stHome.stSensor.i_xRotativeSystem       := FALSE;     (* Rotative system true or false for referencing by norm
86  stAxisInterface.stHome.stSensor.i_xSensor               := FALSE;     (* Input for home signal. Can be a signal from a control
```

Make sure to keep the velocity *i_lrVel* low for best accuracy and set *i_lrOffset* to move away from the mechanical stop afterwards.

In a cyclic task trigger *i_xSensor*. E.g., as follows:

```
38  // ******************************************************************************
39  // Sensor for homing
40  //
41  // Can be used with homing modes:
42  //      PDL.ET_HomeMode.PosDirectionPosEdgeSensor
43  //      PDL.ET_HomeMode.NegDirectionPosEdgeSensor
44  //
45  // Recommended solution:
46  // - set i_xSensor if the tracking deviation reaches a certain level (2mm in this example)
47  // - keep the search velocity (stAxisInterface.stHome.stSensor.i_lrVel) low to get accurate results. Best below 5-10mm/s
48  //
49  // Alternative (probably less accurate):
50  // - If iTorque feedback (in 0.1% of the motors maximal force/torque) is 100% the sensor becomes TRUE
51  // - As limit by default 100% (1000) is used. The user is free to use other limits.
52  // ******************************************************************************
53
54  stAxisInterface.stHome.stSensor.i_xSensor := (ABS(DRV_LMAxis1.TrackingDeviation) >= 2); // Trig sensor on tracking deviation level (2mm in this example)
55
```

### 13.4.2 Drive-Based Homing (S-0-0148)

By executing IDN S-0-0148 as a procedure command, the drive-controlled homing is started.

**Attention:**
Executing this command while the drive is not in AxisState 3 (Wait for job) will result in an error.

Use the function block ***FB**_SercosWriteServiceData*! Using the ***FC**_SercosWriteServiceData* will interrupt the PLC task until the homing has been completed.

Example sequence that is used in the example project (see chapter 13.1 Overview).

```
VAR
// [Action: Homing_Active]
    xHome                   : BOOL;
    diHomeState             : DINT := 0;
    fbHoming                : FB_SercosWriteServiceData;
    diBuffer                : DINT := 0; // Value not relevant
    etDiag_homing           : GD.ET_Diag;
    etDiagExt_homing        : PDL.ET_DiagExt;

    diResult                : DINT;
    diBuffer2               : DINT;
END_VAR
```

…

```
100: // Disable Module if active
    IF stSubModuleInterface.q_xActive THEN
        stSubModuleInterface.i_xEnable := FALSE;
    END_IF
    diHomeState:=110;

110: // Check Module disabled and axis ready for enable or enabled(Axis State = 1 or 3)
    IF NOT stSubModuleInterface.q_xActive AND (DRV_LMAxis1.AxisState = 1 OR DRV_LMAxis1.AxisState = 3) THEN
        diHomeState:=120;
    END_IF

120: // Set Controller Enable and check for Axis State = 3
    DRV_LMAxis1.ControllerEnableSet := TRUE;
    IF DRV_LMAxis1.AxisState = 3 THEN
            diHomeState:=130;
    END_IF

130: // Start homing procedure
    fbHoming(
        i_stLogicalAddress:= DRV_LMAxis1.stLogicalAddress,
        i_xPSercosParameter:= FALSE,
        i_dwIDN:= 148, // S-0-0148 drive based homing command
        i_dwIDN_SI:= 0,
        i_dwIDN_SE:= 0,
        i_wType:= 7, // Execute command
        i_pdwData:= ADR(diBuffer),
        i_wMaxDataLen:= SIZEOF(diBuffer)
        );
    PDL.G_ifAsyncMgr.Start(i_ifAsync:= fbHoming, q_etDiag=> etDiag_homing , q_etDiagExt=> etDiagExt_homing); // Pass job to async manager

    IF etDiag_homing <> GD.ET_Diag.Ok THEN
        diHomeState:= 994; // Error Reaction see Online Help for possible diag codes
    END_IF

    diHomeState := 131;

131: // Check if homing function block has completed
    IF fbHoming.xDone THEN
        fbHoming.xDone := FALSE;
        diHomeState:=0;
        xHome := FALSE;
    ELSIF fbHoming.q_diResultWrite <> 0 THEN
        diHomeState:=995;
    END_IF
```

…

## 13.5 Parameter Access

Every parameter and variable in a LinMot system has its own UPID (Unique Parameter ID).
Every UPID is mapped to its own manufacturer specific IDN to access it via the sercos service channel.

UPIDs are mapped according to the following table:

| UPID | IDN (dec) | IDN (hex) |
|---|---|---|
| 0xHBLB | P-Y-0000.HBdec.LBdec | 0xHBLB0000 + (0x00008000 + 0x0000Y000) |

HB: High Byte (hexadecimal), HBdec: High Byte (decimal)
LB: Low Byte (hexadecimal), LBdec: Low Byte (decimal)

Y = 0: RAM value of a UPID is accessed
Y = 1: ROM value of a UPID is accessed
Y = 2: RAM and ROM value of a UPID is accessed (only applicable when writing UPIDs)

(Note: Not every UPID has a RAM and a ROM Value)

Example:
Access RAM value (Y=0) of UPID 0x13A6 (Maximal Current of Control Parameter Set A)
**UPID = 0x13A6**
**IDN (dec) = P-0-0000.19.166**
**IDN (hex) =** 0x13A60000 + (0x00008000 + 0x00000000) = **0x13A68000**

### 13.5.1 Supported UPID Functions via IDN Access over the Sercos Service Channel

Use Schneider functions *FC_SercosReadServiceData & FC_SercosWriteServiceData* (see Schneider help)

| UPID Access | Data Service Type (i_wType) |
|---|---|
| Read UPID value (RAM or ROM value) | 5 (read/write user data) |
| Write UPID value (RAM and/or ROM value) | 5 (read/write user data) |
| Get minimum value of UPID | 3 (read/write min. value) |
| Get maximum value of UPID | 4 (read/write max. value) |

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

**Advanced:**
By writing anything to **IDN P-0-0099 (hex: 16#00008063)** a drive reboot can be performed (e.g., to load and activate written ROM parameters).
**Attention**: The Sercos line/ring will be interrupted until the drive has completed the reboot.

### 13.5.2 Read RAM Parameter Example

To read the RAM value of a LinMot parameter the function *FC_SercosReadServiceData* (see Schneider help) can be used.

```
PROGRAM SR_Main
VAR
	bStartReadIDN: BOOL := 0;
	diRetVAL: DINT;
	i_stAxisId := DRV_SercDrv.stLogicalAddress;
	wIDN: DWORD := 16#1BF38000; // UPID 1BF3h (Max Read Out Motor Temp, Offset = -50, Scale =
	0.9803…°C)
	dwIDNDataIN: DWORD ; // read data value
	wNumBytetoRead: WORD:= 4; // read 4 bytes of data
	uiReadDataLen: UINT; // actual length of read data
	uiMaxReadDataLen: UINT; // actual maximum length of read data (i.e. max possible length of a string)

END_VAR

...
// ReadIDN
IF bStartReadIDN THEN

	RetVAL:=FC_SercosReadServiceData(
		i_stAxisId,
		wIDN,
		5,
		ADR(dwIDNDataIN),
		wNumBytetoRead,
		uiReadDataLen,
		uiMaxReadDataLen);

	StartReadIDN := 0;

END_IF
…
```

### 13.5.3 Write RAM Parameter Example

To write the RAM value of a LinMot parameter the function *FC_SercosWriteServiceData* (see Schneider help) can be used.

```
PROGRAM SR_Main
VAR
	bStartWriteIDN: BOOL := 0;
	diRetVAL: DINT;
	i_stAxisId := DRV_SercDrv.stLogicalAddress;
	wIDN: DWORD := 16#13A58000; // UPID 13A5h (Integrator Limit)
	dwIDNDataOUT: DWORD := 3000; // Set Integrator Limit to 3A (Scale 0.1 mA)
	wNumBytetoWrite: WORD := 4; // Write 4 bytes of data

END_VAR
...
// WriteIDN
IF bStartWriteIDN THEN  // If StartWriteIDN = TRUE parameter will be read

	RetVAL:=FC_SercosWriteServiceData(
		i_stAxisId,
		wIDN,
		5,
		ADR(dwIDNDataOUT),
		wNumBytetoWrite);

	bStartWriteIDN := 0;
END_IF
…
```

## 13.6 Enable/Disable Drive Warnings

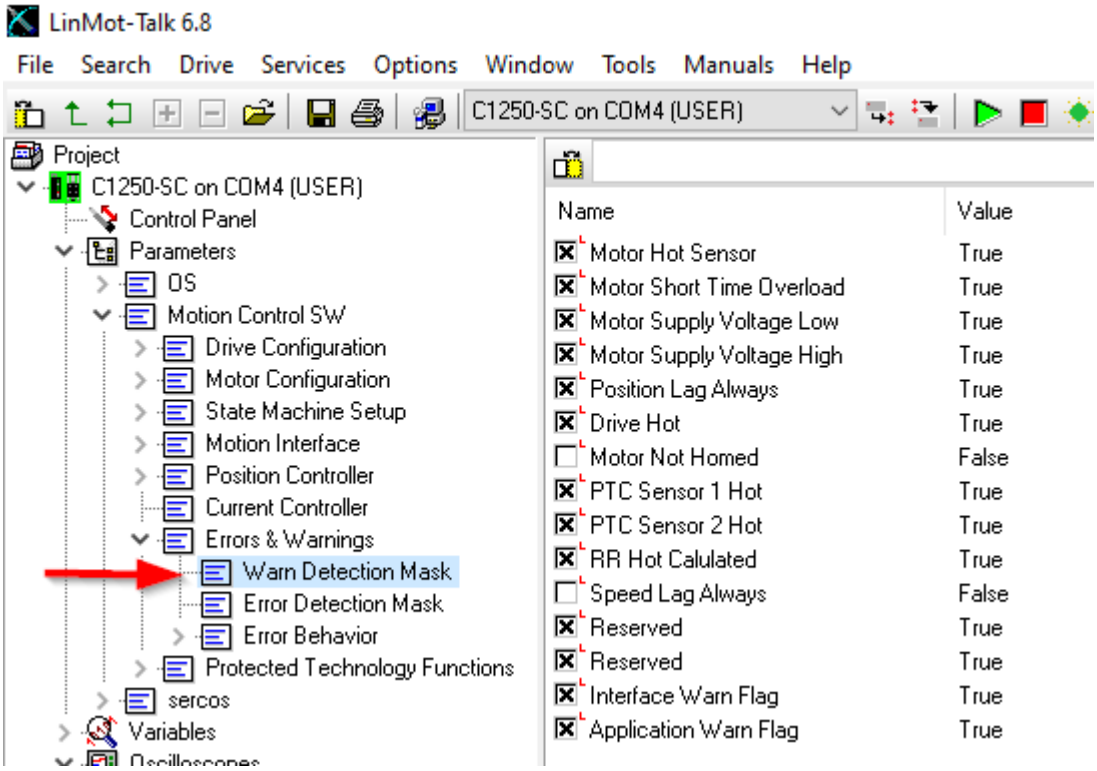
By default, any SERCOS Slave C2D warning (8165) or SERCOS C2Dman.specific warning (8166) will cause the PLC to stop the axes. The LinMot drive can generate several warnings *Motor Not Homed*, *Speed Lag Always* and so on.

To avoid this behaviour on the Schneider PLC, these diagnostic messages can be deactivated using *FC_DiagConfigSet2* (see Schneider help for more information).

In the example project (see 13.1 Overview) in the *Init_Action* of the program *SR_MainMaschine* the two diagnostic codes 8165 and 8166 are set to diagnostic class 1:

```
IF (SystemConfiguration.G_ifSercos.State = 4) THEN
    //Warnings, see help
    diResultDiagConfigSet8165 := FC_DiagConfigSet2(i_diDiagCode:= 8165, i_diDiagClass:= 1, i_stLogAddr:= SERCDRV_TYPE, i_diDiagSubClass:= 1);
    diResultDiagConfigSet8166 := FC_DiagConfigSet2(i_diDiagCode:= 8166, i_diDiagClass:= 1, i_stLogAddr:= SERCDRV_TYPE, i_diDiagSubClass:= 1);

    xInitDone :=  TRUE;
END_IF
```

Otherwise it is also possible to disable warnings on the drive side:




**Attention:**
Be aware that some of the warnings of a LinMot drive can indicate an upcoming error. So, make sure you have good reasons to disable warnings on the drive, e.g., *Motor Hot Sensor* or *Drive Hot.*

# 14 Drive Profile: CODESYS SoftMotion, CiA402

## 14.1 Overview

This chapter shows how a LinMot drive with *DS402 (CoE) / CiA402* interface (e.g., C1250-DS-XC-1S) can be integrated and setup in a CODESYS SoftMotion environment (on a Raspberry Pi 2).

**Download:**
Example projects can be downloaded from:
http://download.linmot.com/plc_lib/examples/CODESYS_DS (named *LM_DS_Demo_CODESYS_RaspPi...*)

Components used:

- LinMot C1250-DS-XC-1S-000 (article number 0150-2416) with firmware 6.6 Build 20170522
- Raspberry Pi 2, Model B, target system version 3.5.10.20
- CODESYS Control for Raspberry PI 3.5.10.20 Package
- CODESYS V3.5 SP10 Patch 3

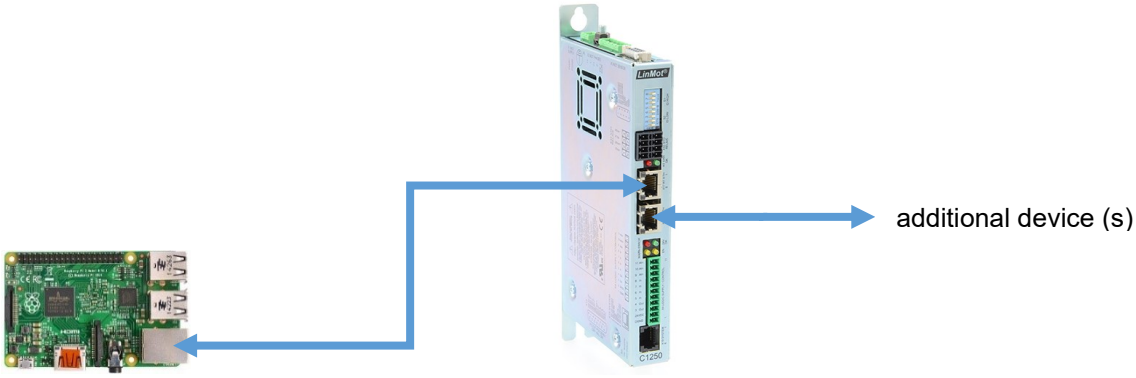



Image Source: http://www.raspberrypi.org/

EtherCAT is the real-time Ethernet network originally developed by Beckhoff. The LinMot acts as Slave in this network and is implemented with the standard ASIC ET1100 from Beckhoff.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

## 14.2 Configuration and Connection of the LinMot Drive

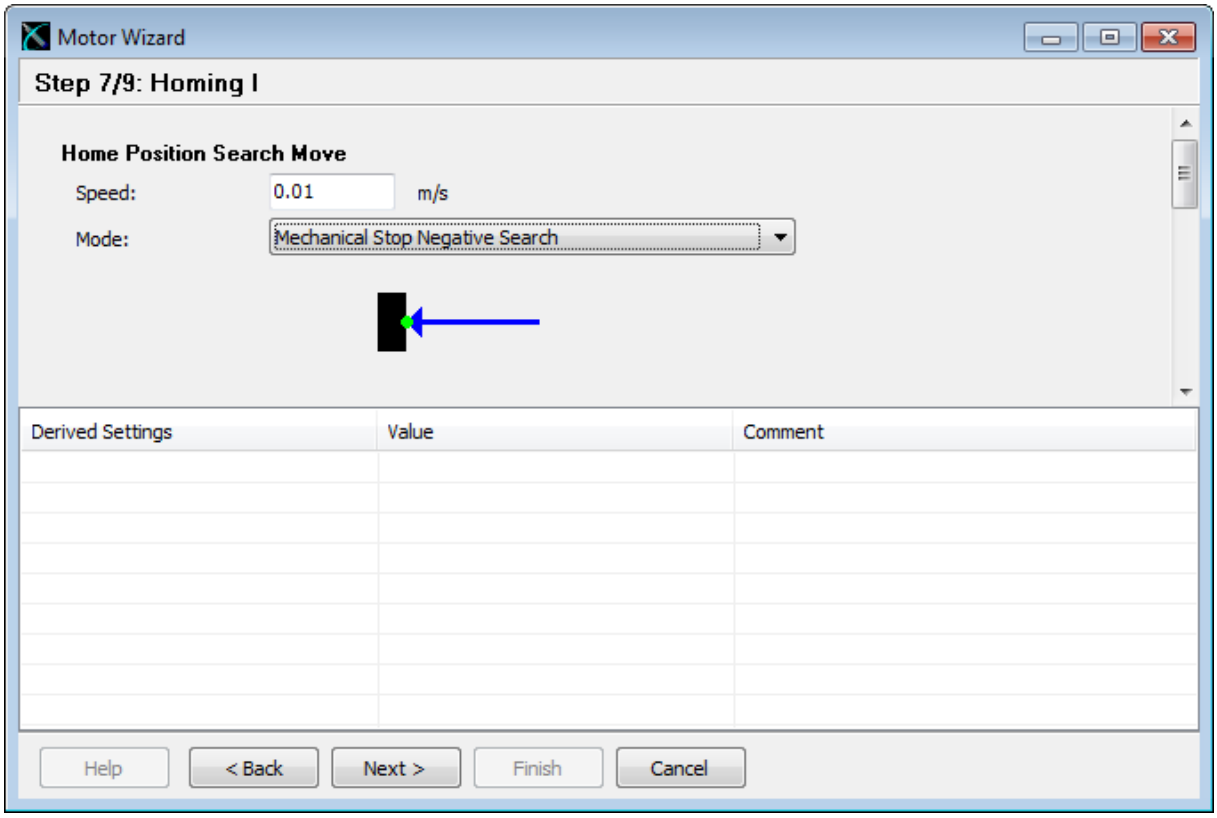
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 14.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select a homing mode in the Motor Wizard as with the DS (DS402 / CiA402) interface a drive-based homing is possible (e.g. *Mechanical Stop Negative Search*):




See Appendix I: Basic Position Control Loop Tuning

### 14.2.2 XML File

Install the XML file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

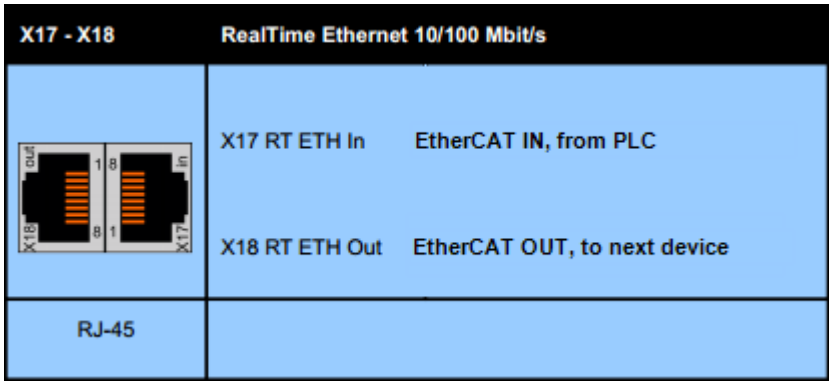
- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML\
- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML\ (-MI drives)

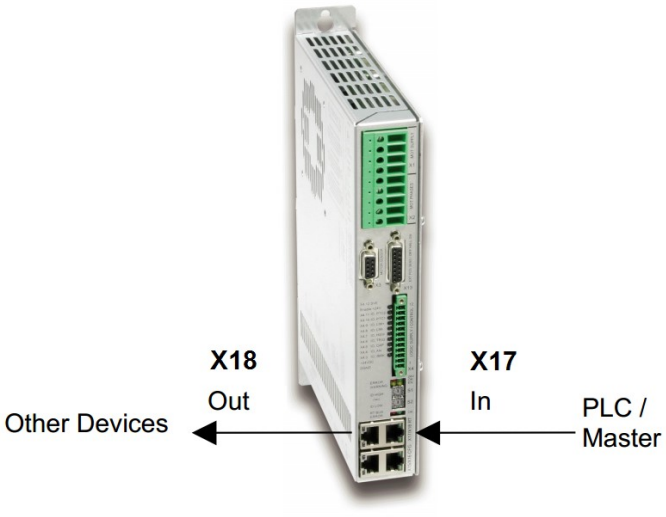
**XML file names:**

| | |
|---|---|
| NTIL_CiA402_Servos_Vx_xrx.xml | LinMot DS drives SG6-7 |
| NTIL_CiA402_SG5_Servos_Vx_xrx.xml | LinMot DS drives SG5 |
| NTIL_CiA402_Servos_MI_Vx_xrx.xml | LinMot MI drives SG6 |

### 14.2.3 EtherCAT Connection

The drive is connected to the EtherCAT network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-DS-UC drive. On all other LinMot drives supporting EtherCAT DS the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.



## 14.3 PLC Setup

### 14.3.1 EtherCAT Device Description File XML

The EtherCAT device description file (XML) is located by default in the LinMot-Talk installation path:
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML

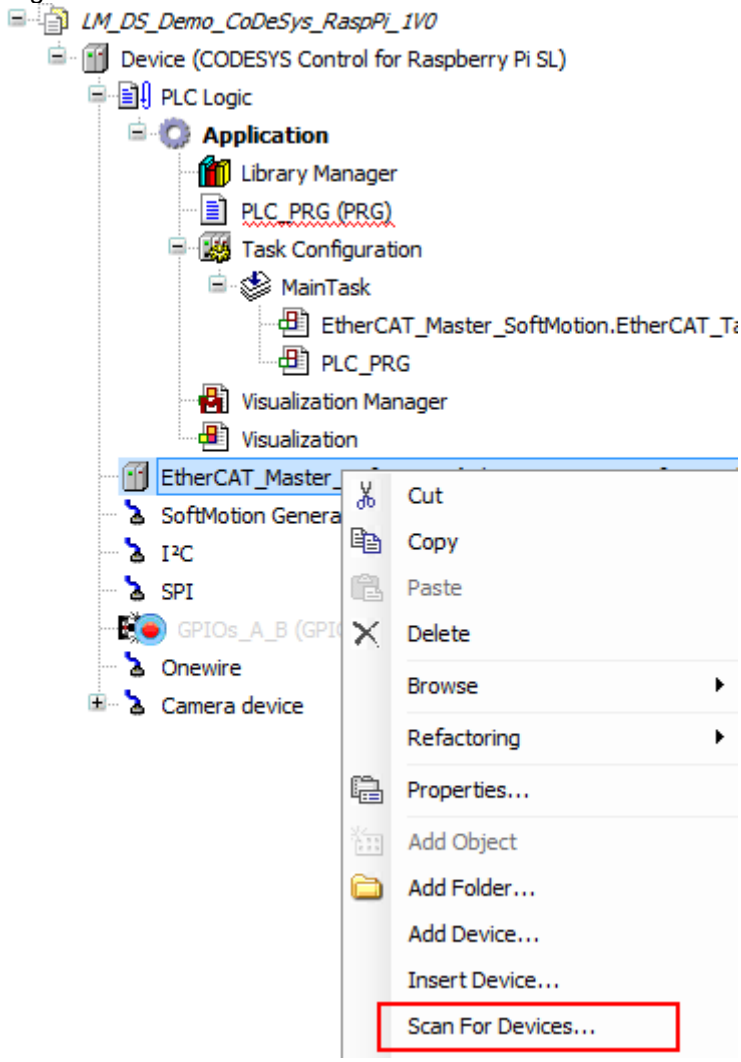
Install the XML in CODESYS by selecting “Tools > Device Repository”.

### 14.3.2 Add and setup the LinMot drive

Follow these steps to add the LinMot drive to your CODESYS project.

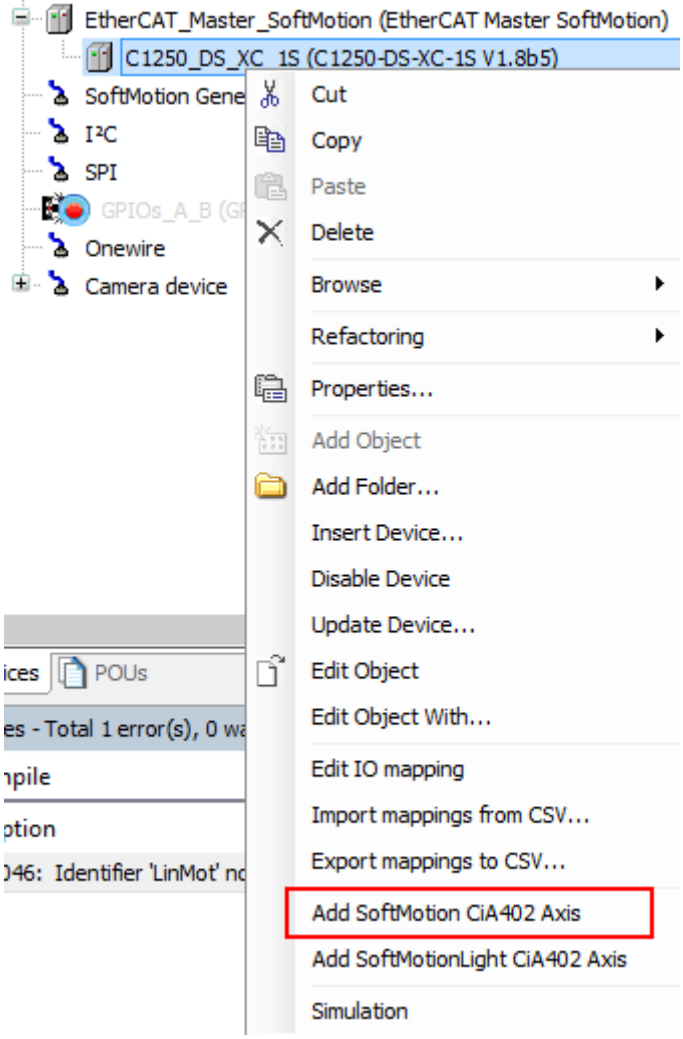
1. If not already existing add an *EtherCAT Master SoftMotion* by right-click on the PLC and select *Add Device…*

2. Right-Click on the *EtherCAT Master SoftMotion* and select Scan for Devices…



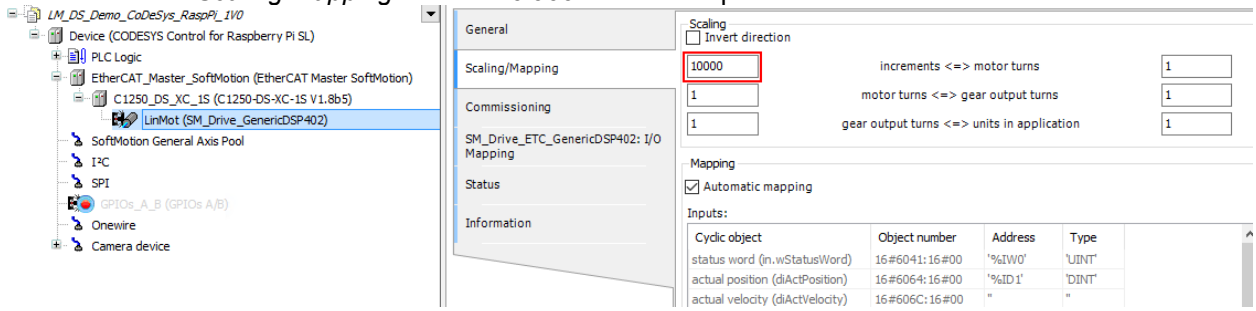

In the following window select Copy All Devices to Project

3. Right-Click on the added drive (e.g. C1250_DS_XS_1S…) and select *Add SoftMotion CiA402 Axis*




Select Ok in the following pop up.

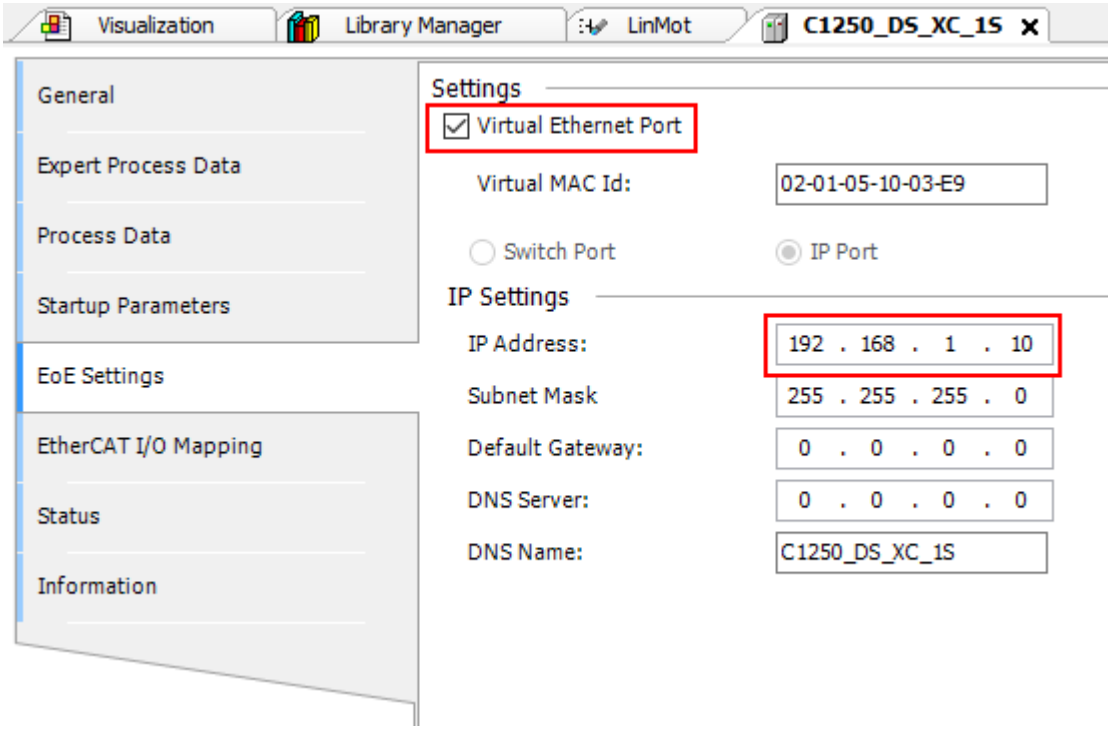
4. Move to the tab *Scaling/Mapping* and set 10’000 increments per 1 motor turn




You are now ready to use the axis.

### 14.3.3 Check EoE settings

On supported drives (C1250-..., C1450-… & E1450-...) please check the EoE (Ethernet over EtherCAT) settings.
You can enable the *Virtual Ethernet Port* and assign an IP to be able to login over EtherCAT into the drive using LinMot-Talk.




Be aware that the PLC must route the requests from LinMot-Talk over Ethernet to the EtherCAT network.

**Note:**
EoE can be used to login into the drive with LinMot-Talk directly over EtherCAT.
E.g., to change drive settings or for monitoring and tracing.
If not required, this feature can be disabled.

LinMot drives do NOT support the DHCP mode yet. A fix IP must be assigned or the EoE feature must be completely disabled (disable Virtual Ethernet Port).

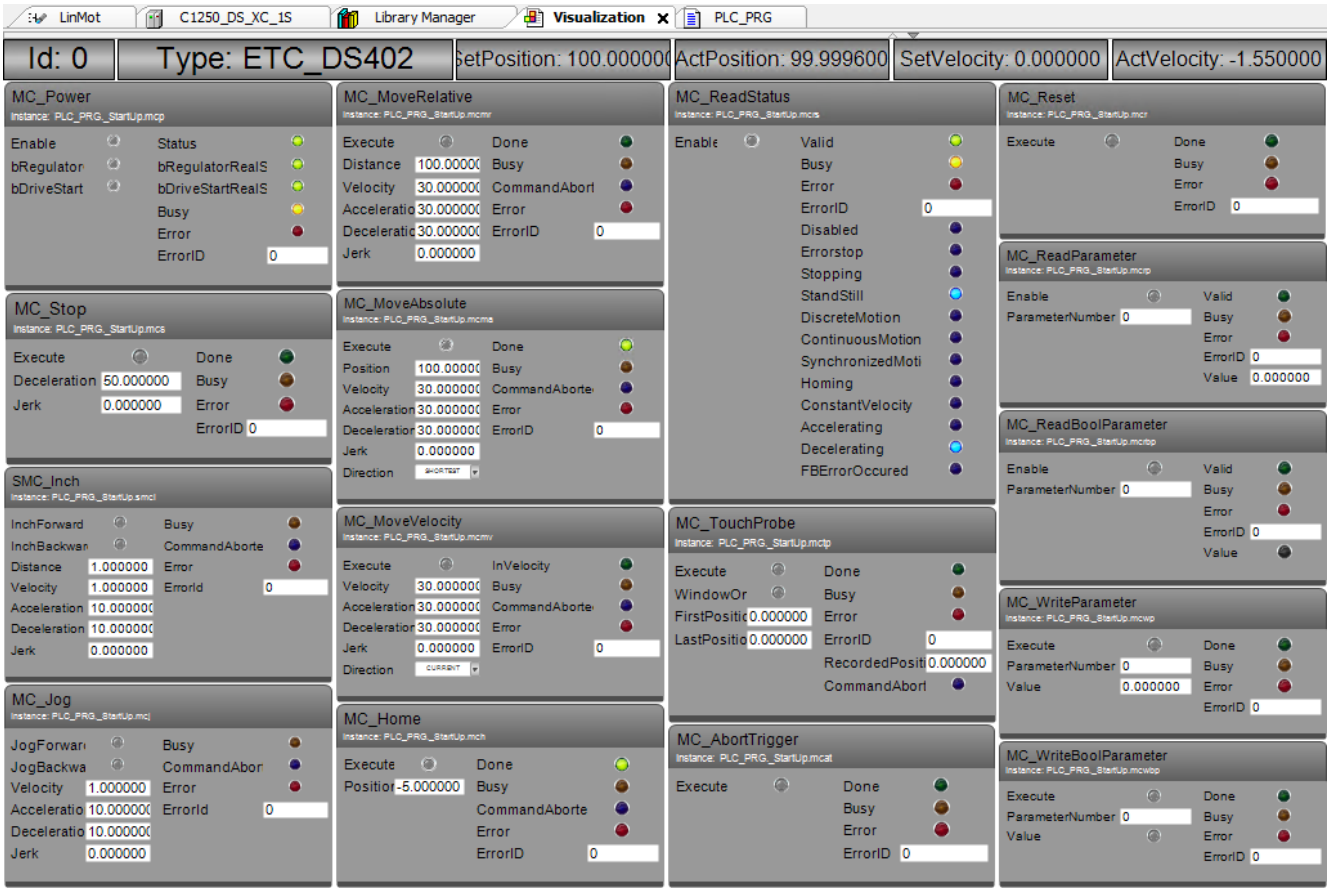
### 14.4 Homing

To home the axis using the drive-controlled homing procedure the function block MC_Home (library SM3_Basic) can be used.

Executing this function block will start the homing procedure defined with the Motor Wizard of LinMot-Talk (e.g., Mechanical Stop Negative Search). The Position input of the function block defines the home position of the axis, e.g., the mechanical stop.

## 14.5 Visualization

In the example project the VISU_NEW_SMC_StartupDrive (SM3_Basic) template is used



## 14.6 Parameter Access

Parameters can be modified using their UPIDs (Unique Parameter ID) via CoE.
To use a UPID command the function blocks MC_ReadParameter and MC_WriteParameter can be used.

The Sub-Index specifies the command which is performed

| SDO Service | Index | SubIndex | Description |
|---|---|---|---|
| Read | 2000h+UPID | 0x01 | Parameter UPID read RAM value |
| Write | 2000h+UPID | 0x01 | Parameter UPID write RAM value |
| Read | 2000h+UPID | 0x02 | Parameter UPID read ROM value |
| Write | 2000h+UPID | 0x02 | Parameter UPID write ROM value |
| Read | 2000h+UPID | 0x03 | Parameter UPID read minimal value |
| Read | 2000h+UPID | 0x04 | Parameter UPID read maximal value |
| Read | 2000h+UPID | 0x05 | Parameter UPID read default value |
| Write | 2000h+UPID | 0x06 | Parameter UPID write RAM and ROM value |
| Write | 2000h | 0x07 | Parameter Default OS SW instance |
| Write | 2000h | 0x08 | Parameter Default MC SW instance |
| Write | 2000h | 0x09 | Parameter Default Intf SW instance |
| Write | 2000h | 0x0A | Parameter Default Appl SW instance |
| Write | 2000h | 0x0B | Reset device |

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

The input *ParameterNumber* of the MC_... function blocks must be of type DINT and contains the data length the index and the subindex.

E.g., *ParameterNumber* := -DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength), 24) +
SHL(UINT_TO_DWORD(uiIndex), 8) + usiSubIndex);

```
PROGRAM PLC_PRG
VAR
    _StartUp: SM3_Basic.SMC_StartupDrive;

    //Parameter Access
    MC_ReadParameter1:  MC_Readparameter;
    MC_WriteParameter1: MC_Writeparameter;
    usiDataLength: USINT := 4; // Length 4 Bytes
    uiIndex: UINT; //= 16#2000 + UPID number of the LinMot parameter. E.g. UPID 16#13a6 => Index := 16#2000 + 16#13a6 = 16#33a6
    usisubIndex: USINT := 1; // 1 = RAM Value, 2 = ROM Value, further information in the drive interface manual
END_VAR
```

```
// SM3_Basic StartupDrive
_StartUp(Axis:= LinMot);

// Parameter Access
MC_ReadParameter1.ParameterNumber := -DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength), 24) + SHL(UINT_TO_DWORD(uiIndex), 8) + usisubIndex);
MC_WriteParameter1.ParameterNumber := -DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength), 24) + SHL(UINT_TO_DWORD(uiIndex), 8) + usisubIndex);

MC_ReadParameter1(Axis:=LinMot);
MC_WriteParameter1(Axis:=LinMot);
```

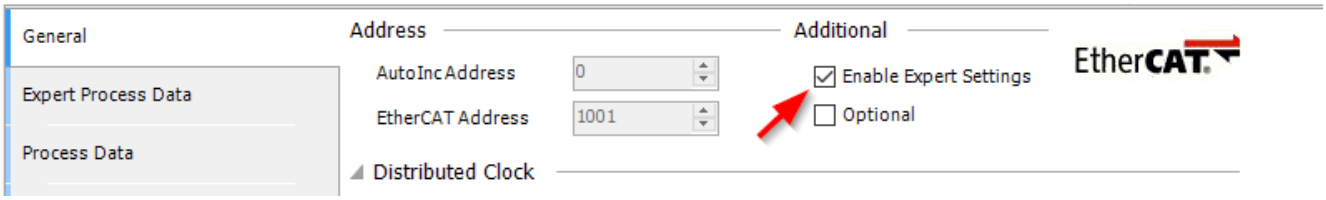
Example:

To write 1A to the RAM value of the parameter Maximal Current:

| | | | |
|---|---|---|---|
| Name: | **Maximal Current** | DataLength = **4** |
| UPID: | **13A6h** | Index **= 16#2000 + 16#13a6 = 16#33a6** |
| Type: | **SInt32 = DINT** | SubIndex **= 1** |
| Scale: | **0.001 A** | Value **= 1A / 0.001A (scale) = 1'000** |

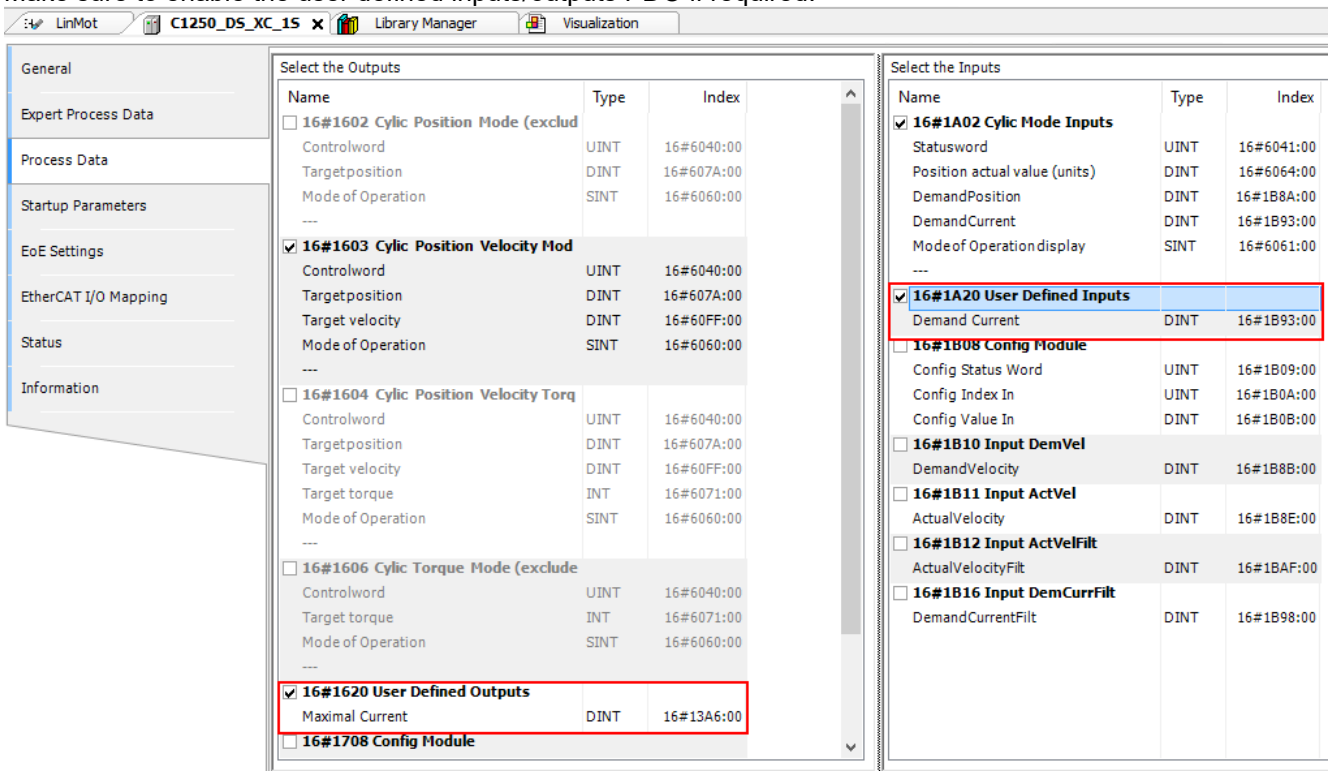
## 14.7 Add Additional Parameters or Variables to the Process Data

To be able to modify the Process Data make sure to enable the expert settings:

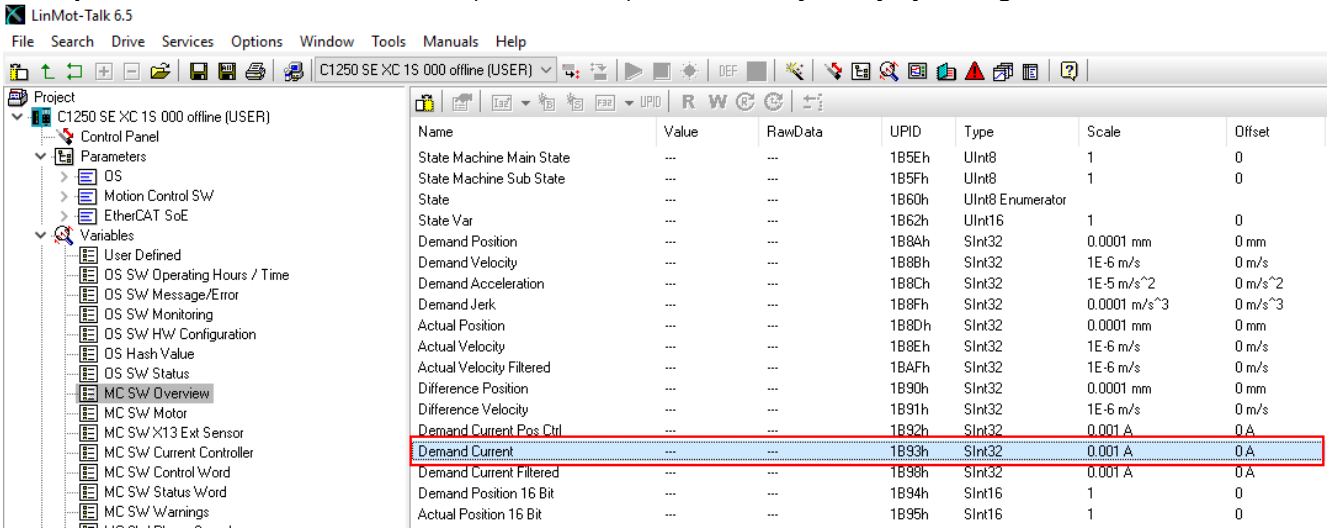



It is possible to add additional parameters or variables of the drive to the Process data PDO for cyclical write or read access.

Make sure to enable the user defined inputs/outputs PDO if required:

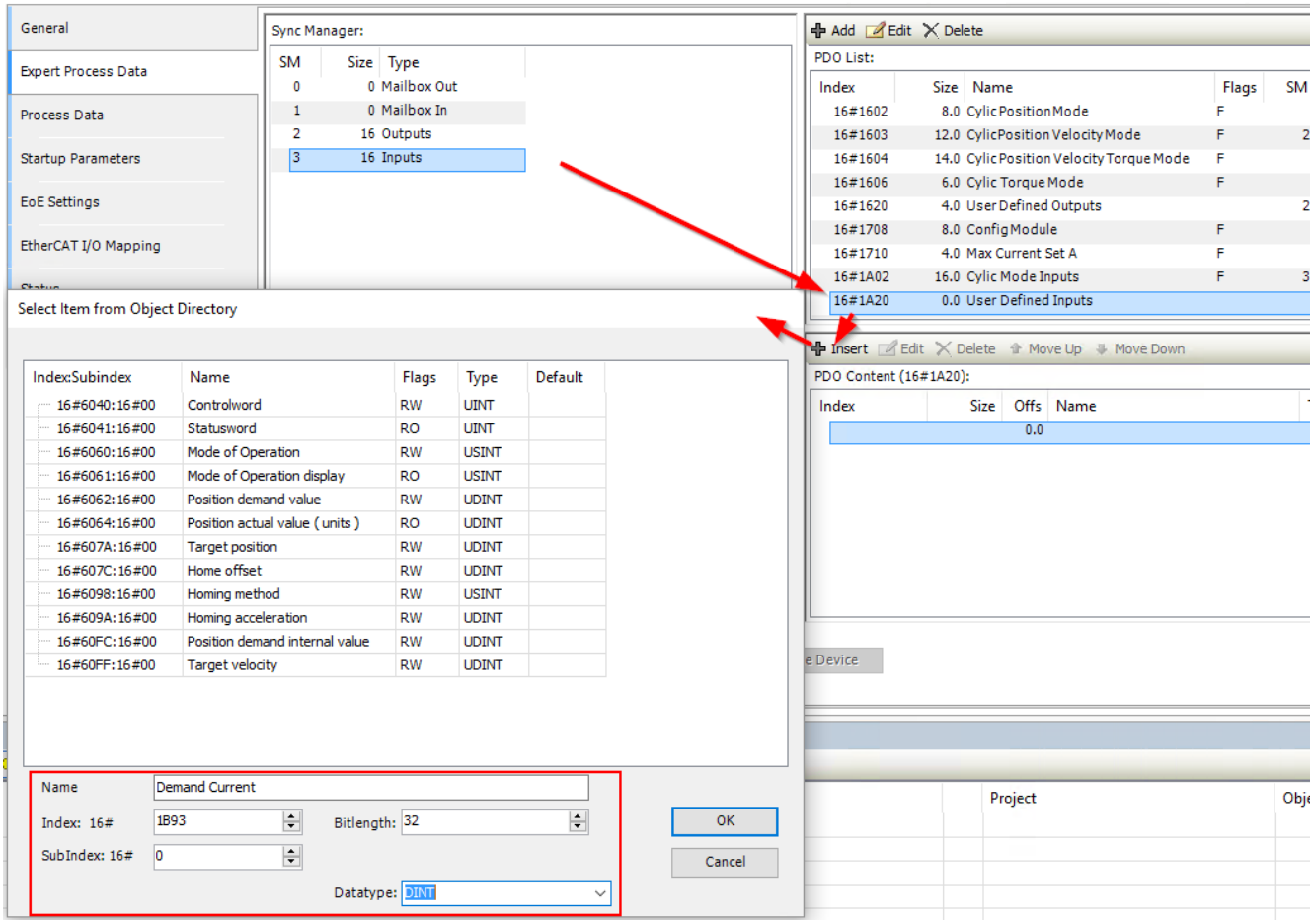


**Example 1:** Read the *Demand Current (UPID 1B93h)* of the drive cyclically by adding it to AT 1:

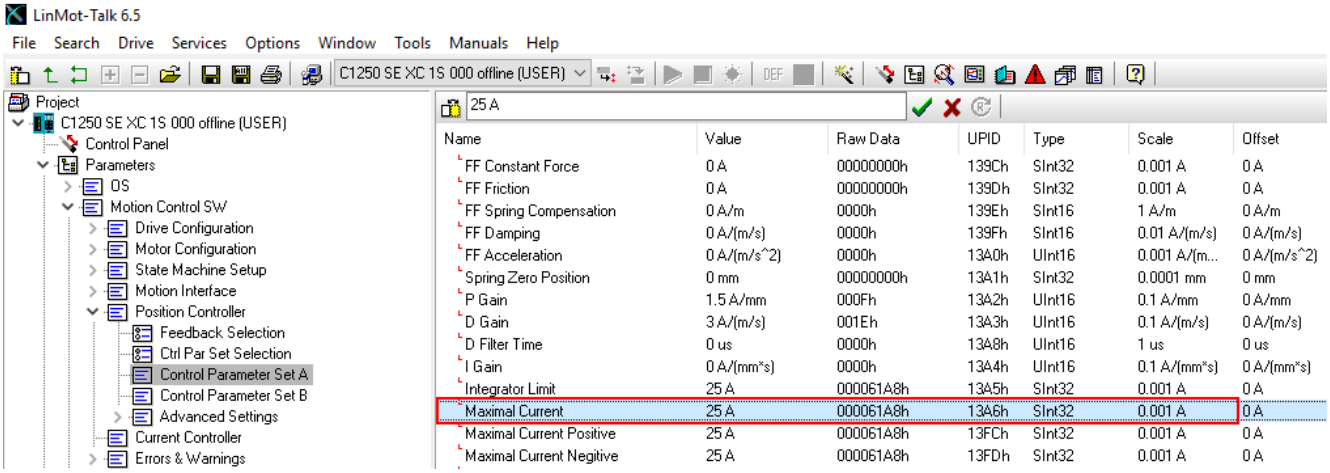



**Name:** Demand Current
**UPID:** 1B93h
**Type:** SInt32 = DINT
**Scale:** 0.001 A

1. In the Expert Process Data select *User Defined Inputs* from the PDO List.
2. Press the *Insert* button
3. In the *Select Item From Object Dictionary* window enter the name, the UPID as *Index* and set the *Datatype* to DINT (as the Demand Current is of type SINT32)
4. Click OK to finish

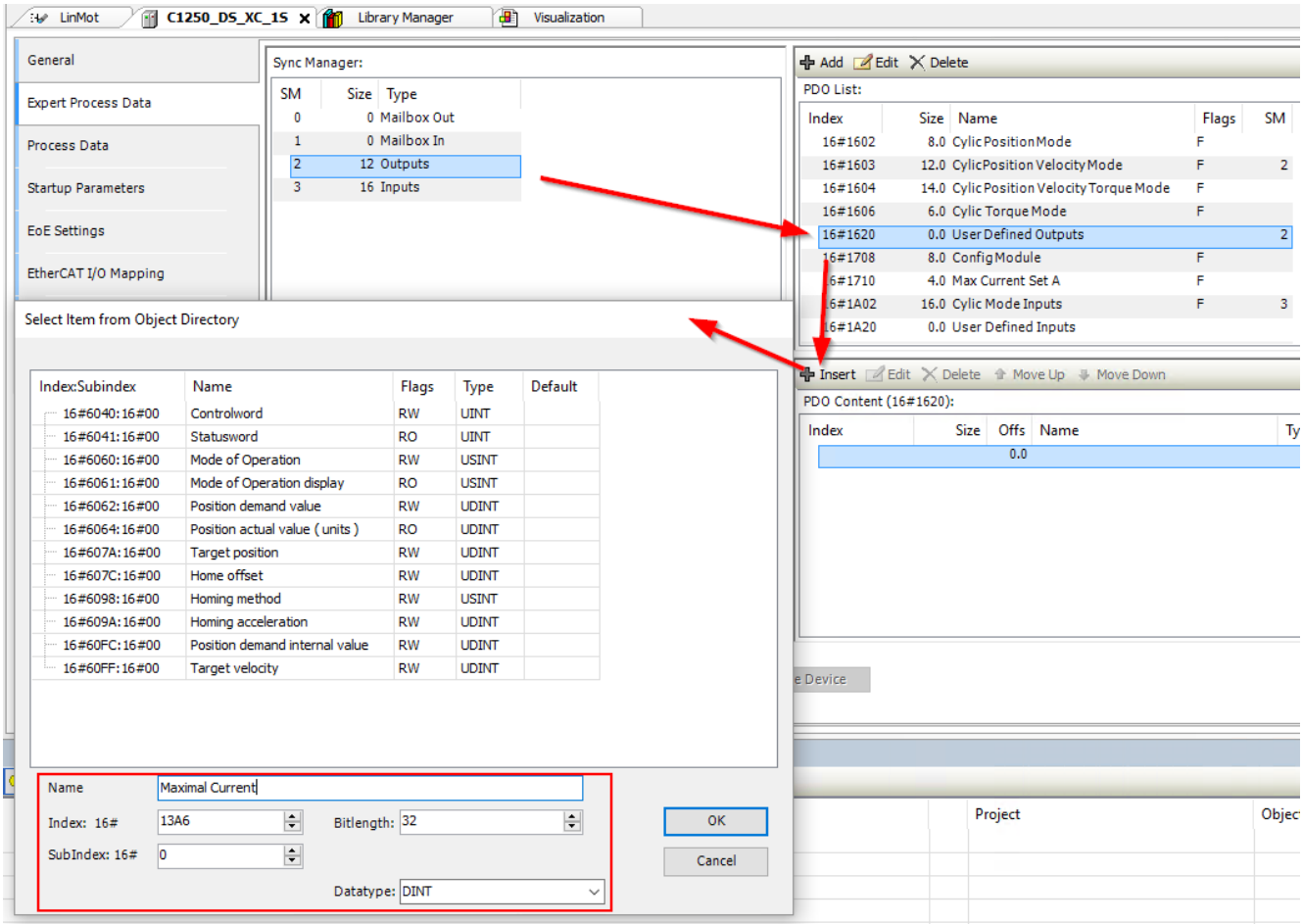


**Example 2:** Write the *Maximal Current (UPID 13A6h)* of the drive cyclically by adding it to the process data:




**Name:** Maximal Current
**UPID:** 13A6h
**Type:** SInt32 = DINT
**Scale:** 0.001 A

1. In the Expert Process Data select *User Defined Outputs* from the PDO List.
2. Click the *Insert* button
3. In the *Select Item From Object Dictionary* window enter the name, the UPID as *Index* and set the *Datatype* to DINT (as the Maximal Current is of type SINT32)
4. Click OK to finish



# 15 Drive Profile: B&R NC Motion, CiA402

## 15.1 Overview

This chapter shows how a LinMot drive with *Powerlink* interface (e.g., C1250-PL-XC-1S) can be integrated and setup in a B&R environment as DS402 motion axis (CiA402).

**Download:**
The demo project can be downloaded from:
http://download.linmot.com/plc_lib/examples/BR/DS402 (named *LM_DS402_Demo_AS4...*)

Components used:

- LinMot C1250-PL-XC-1S-000 (article number 0150-1885)
  Firmware 6.10 Build 20210521
- B&R X20CP1584 (1.9.0.0)
- Automation Studio V 4.9.2.46 SP#, Automation Runtime C4.90, mapp Motion **5.12.3**

**Attention:**
Drive firmware **6.10 Build 20210521 or later** is required to use the DS402 / CiA402 functionality

In the *B&R Help Explorer* check chapter ***McDS402Ax*** for more details about the DS402 / CiA402 integration of 3rd party drives.
Press Ctrl + G to open by Guid: **5935c8d1-cc4a-43f2-aead-783ff4f9f6fc**

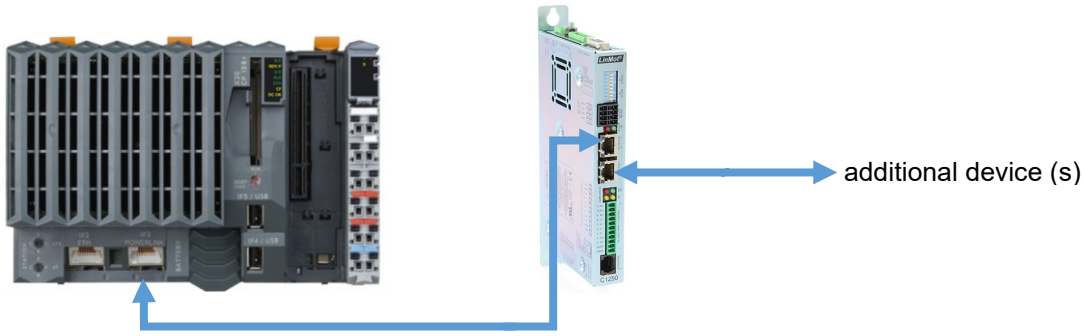



Image Source: http://www.br-automation.com/

The LinMot POWERLINK devices are Ethernet POWERLINK CNs (Controlled Nodes) with the following parameters (according to EPSG-DS-301-V1.1.0):

| Device Properties | Value |
|---|---|
| Minimal POWERLINK cycle time | 250 µs |
| Size of isochronous transmit buffer (maximal size of isochronous frames) | 64 Byte |
| Size of isochronous receive buffer (maximal size of isochronous frames) | 64 Byte |
| Overall buffer size available for isochronous data | 128 Byte |
| PReq to PRes latency (CN isochronous reaction time) | ~ 1 µs |
| SoA to ASnd latency (CN asynchronous reaction time) | ~ 1 µs |
| Maximum asynchronous MTU | 300 Byte |
| Ability to support multiplexed isochronous access | No |
| Asynchronous SDO transfer method (UDP/IP and/or POWERLINK ASnd) | POWERLINK ASnd only |

For further information on Ethernet POWERLINK please visit: http://www.ethernet-powerlink.org

## 15.2 Preconditions to use DS402 / CiA402

- Only SG6 devices are supported (C1250, C1251, C1450 & E1450)
- Drive firmware must be >= 6.10 Build 20210521
- Use of “00000156_xxxxxPL_DS402.xdd” as device description file when configuring the drive in the PLC

**Attention:**
After adding the axes, it is recommended to rebuild the project and to **restart both the PLC and drives** to have a proper start-up. Ignoring this may prevent being able to power on the axis.

## 15.3 Supported operating modes

- Mode 1: Profile Position Mode (pp)
- Mode 3: Profile Velocity Mode (pv)
- Mode 6: Homing Mode (hm)
- Mode 7: Interpolated Position Mode (ip)
- Mode 8: Cyclic synchronous position mode (csp)
- Mode 9: Cyclic synchronous velocity mode (csv)

## 15.4 Configuration and Connection of the LinMot Drive

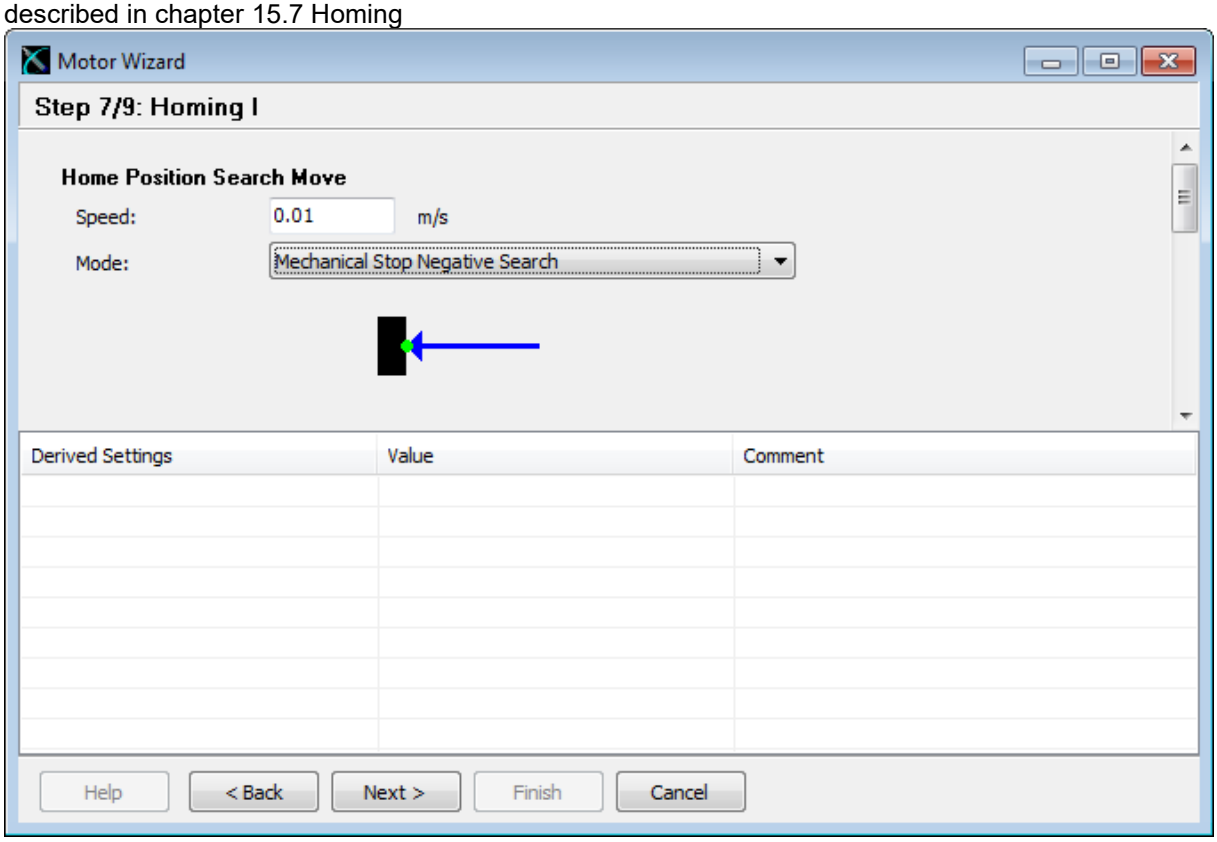
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 15.4.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select a homing mode in the Motor Wizard if you want to use the drive-based homing as described in chapter 15.7 Homing




See Appendix I: Basic Position Control Loop Tuning

### 15.4.2 XDD File

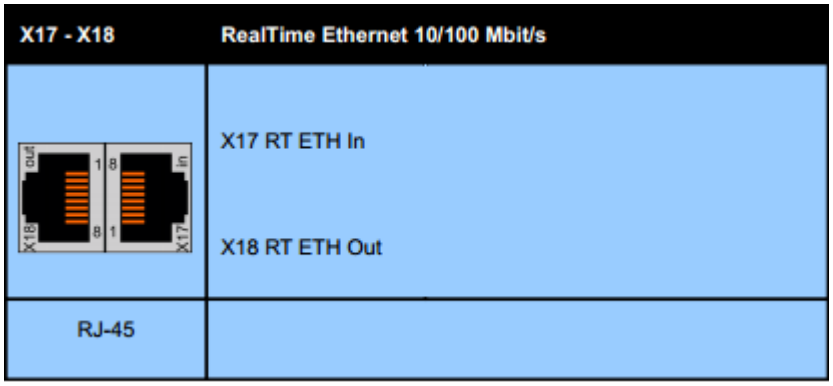
Install the XDD file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software.
It is located by default:

- Powerlink: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink\XDD\AS4
- Powerlink: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink_Nx\XDD\AS4 (-MI drives)

e.g., *00000156_C1250PL_DS402.xdd*

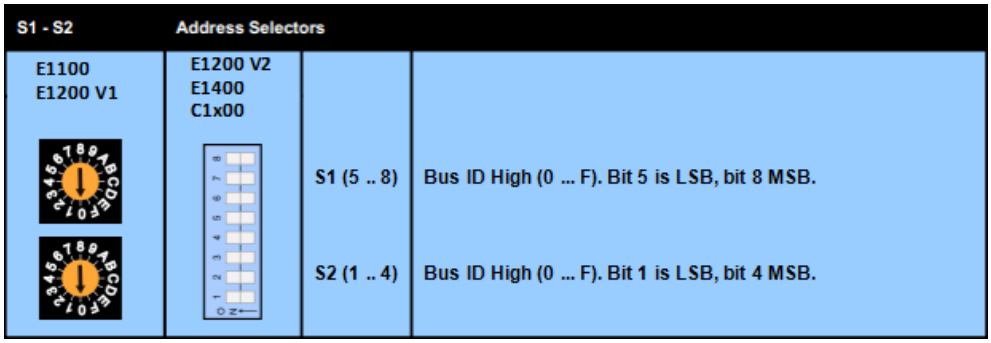
### 15.4.3 Powerlink Connection

The drive is connected to the Powerlink network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-PL-UC drive. On all other LinMot drives supporting Powerlink the ports are named the same (X17 & X18), but they may be placed differently on the drive housing.




### 15.4.4 Node ID

The Node ID of the LinMot drive is set using the hardware address selectors.



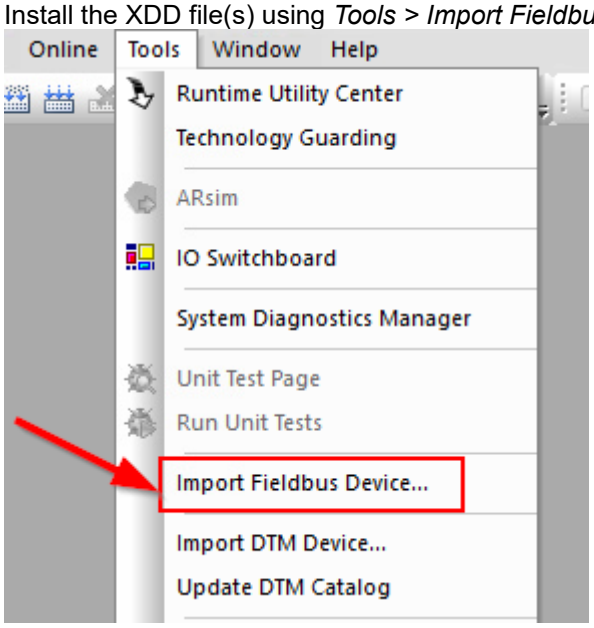

E.g.: Node ID 18 dec = 12 hex = 0001 0010 bin

=> S1 = 1 hex = 0001 bin (Dip Switch 5 = ON), S2 = 2 hex = 0010 bin (Dip Switch 2 = ON)

## 15.5 Automation Studio Physical View

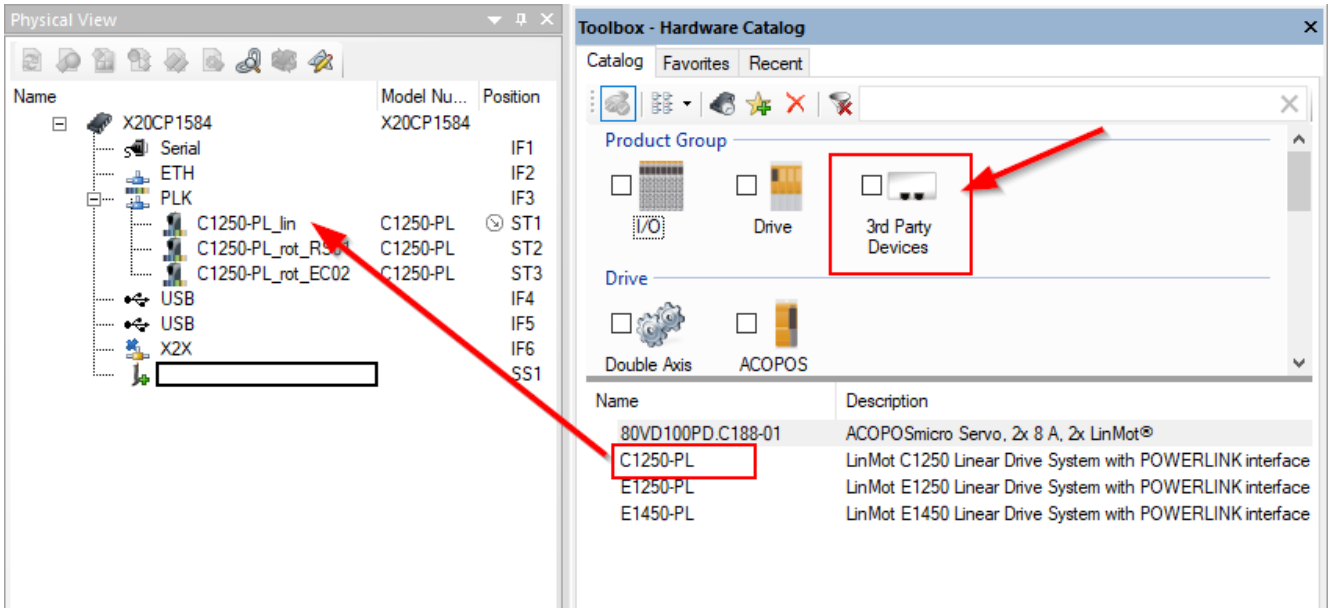
### 15.5.1 Add the LinMot drive to the Powerlink network

Install the XDD file(s) using *Tools* > *Import Fieldbus Device…* in Automation Studio.




The required XDD-file can be found in the following folder (default):
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\POWERLINK\XDD\AS4
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink_Nx\XDD\AS4

In the Hardware Catalog search for the LinMot drive (e.g., C1250...) and drag&drop it onto the Powerlink interface (PLK).
Alternatively select the Product Group *3rd Party Devices* > all installed LinMot drives are shown.

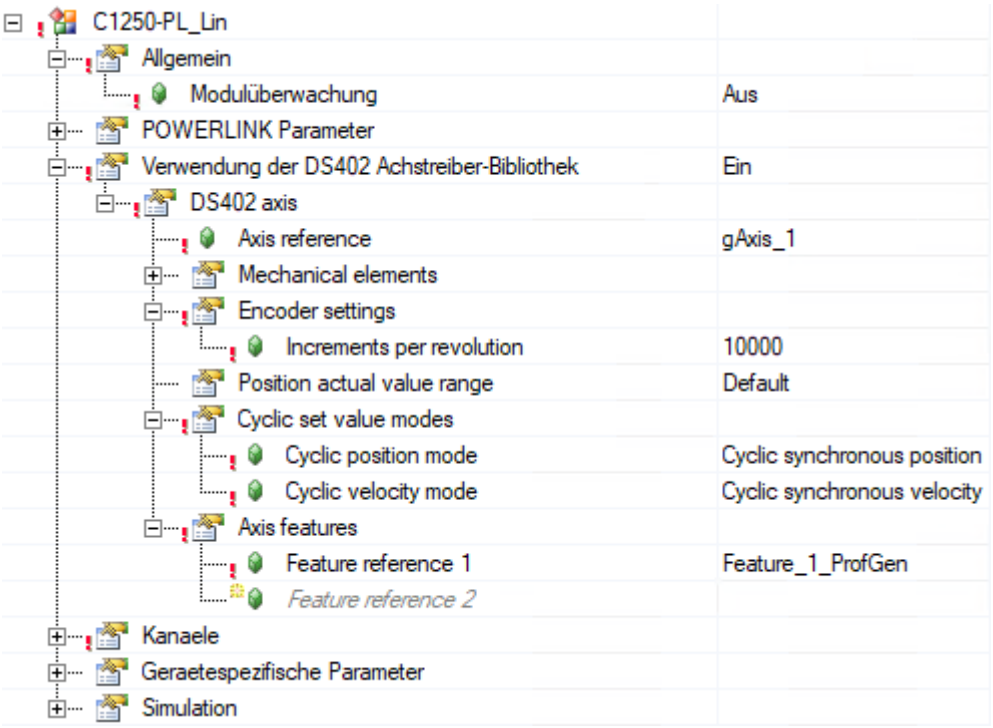



Set the Node ID according to your requirements.

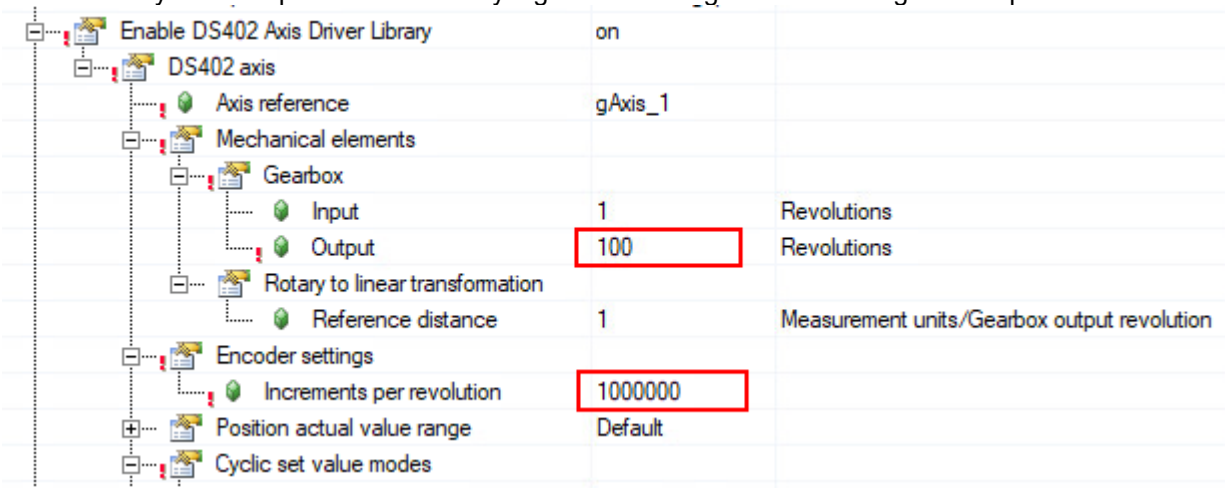
### 15.5.2 Configuration

In the configuration of the LinMot slave module (right-click on device > *Configuration*) the following setting must be done. Attention: The “Axis reference” can only be set when the axis is already defined (see 15.6.1)

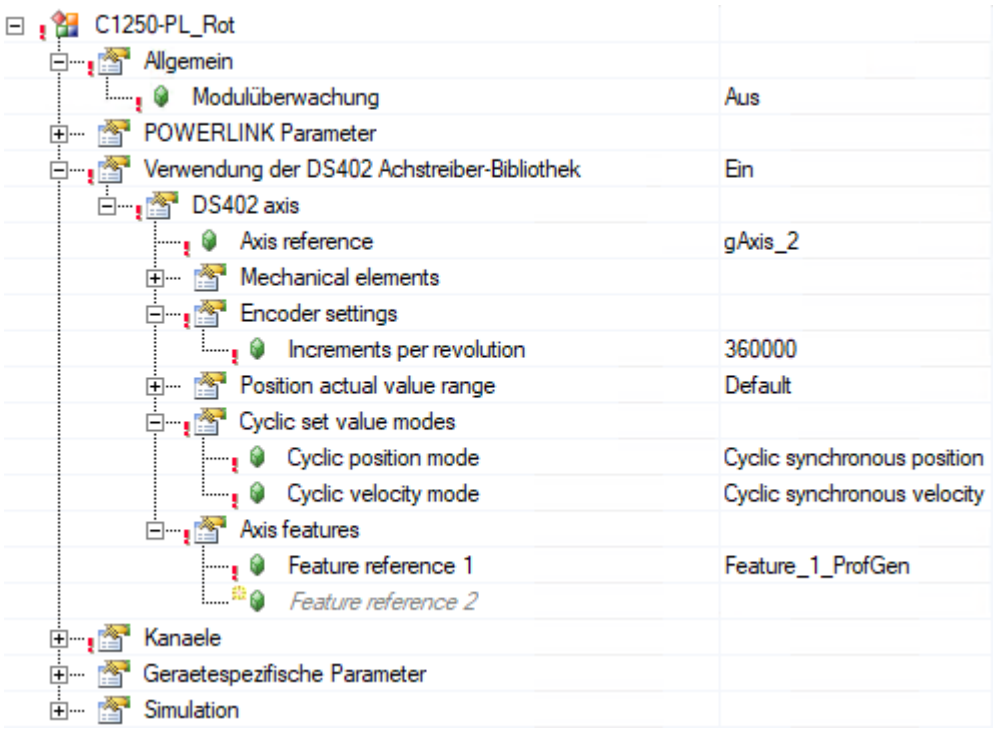
#### 15.5.2.1 Linear Axis (10’000 increments per revolution):




In case the dynamic requirements are very high the following encoder settings will help



#### 15.5.2.2 Rotary Axis PR01 or PR02 (360'000 increments per revolution):

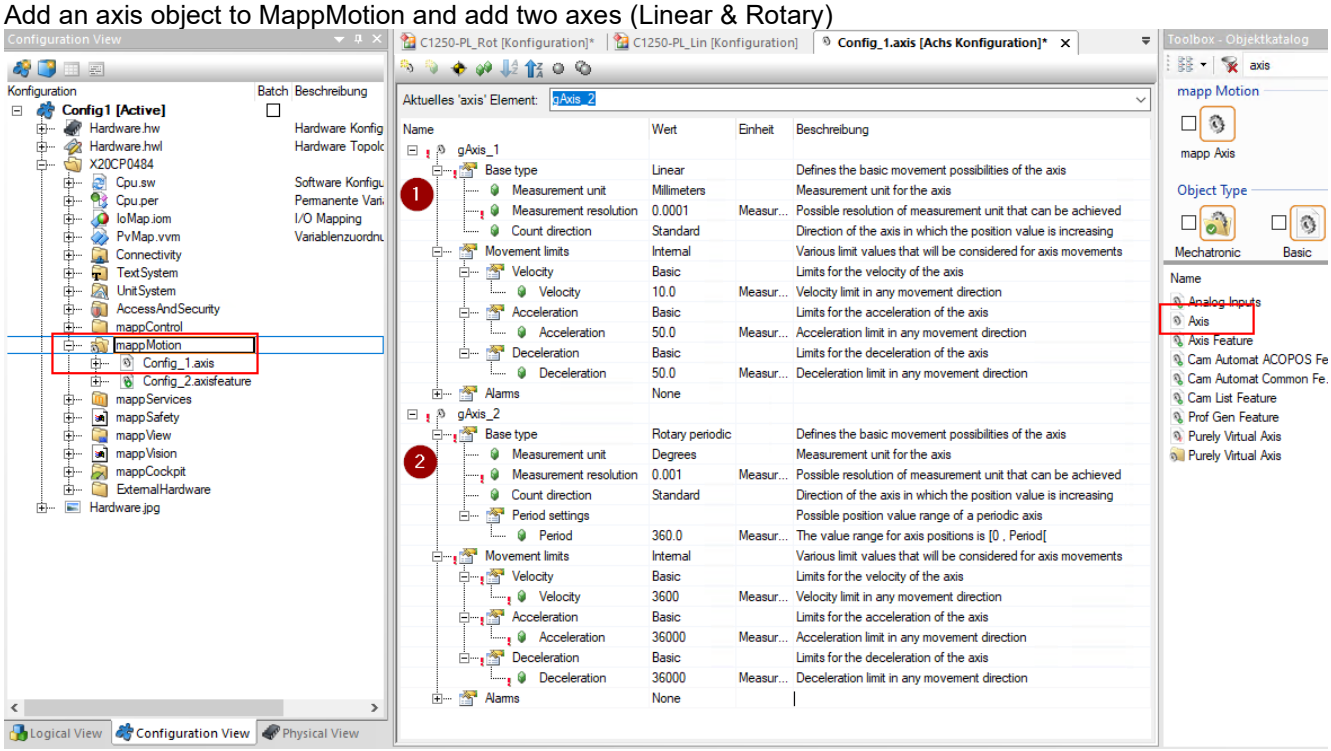



**Note:**
In this example the feature "*Feature_ProfGen*" is used. It is only required if CNC, CAM, or other synchronized motion is used.

## 15.6 Automation Studio Configuration View

### 15.6.1 Add axis objects

Add an axis object to MappMotion and add two axes (Linear & Rotary)




**Note:**
A **LinMot linear motor** has an internal position resolution of 0.1 um. Therefore the “Measurement resolution” for the unit Millimetres is **0.0001**.

A **LinMot PR01/PR02 rotary motor** has by default 360’000 ticks/increments per motor revolution. Therefore, for the “Measurement resolution” for the unit Degrees is 360° / 360’000 = **0.001**.

## 15.7 Homing

Within the DS402 / CiA402 profile a drive based / manufacturer specific homing is possible. To setup the homing the function block *MC_BR_InitHome_DS402Ax* is used.

Only manufacturer specific homing modes are supported.

When using the MC_BR_InitHome_DS402Ax function block, set "DriveSpecificHoming" to "mcSWITCH_ON". "DriveSpecificHomingModeMode" configures the HomingMode like the UPID 13C4h does:

| Name | Value | Raw Data |
|---|---|---|
| Actual Position | Off | 0000h |
| Mechanical Stop Negative Search | On | 0001h |
| Mechanical Stop Positive Search | Off | 0002h |
| Limit Switch Negative | Off | 000Eh |
| Limit Switch Positive | Off | 000Fh |
| Home Switch Negative | Off | 0003h |
| Home Switch Positive | Off | 0010h |
| Home Switch And Mechanical Stop Negative | Off | 0004h |
| Home Switch And Mechanical Stop Positive | Off | 0005h |
| Home Switch And Limit Switch Negative | Off | 0006h |
| Home Switch And Limit Switch Positive | Off | 0007h |
| Index On X13 And Mechanical Stop Negative | Off | 0008h |
| Index On X13 And Mechanical Stop Positive | Off | 0009h |
| Index On X13 And Limit Switch Negative | Off | 000Ah |
| Index On X13 And Limit Switch Positive | Off | 000Bh |
| Index On X13 Negative Search | Off | 000Ch |
| Index On X13 Positive Search | Off | 000Dh |
| No Drive Homing | Off | 0011h |

Select the homing mode you have setup with the Motor Wizard of LinMot-Talk.

Position offset, velocity and acceleration should also be configured to the users' needs.

### 15.7.1 Linear Motor

A possible setup for homing mode “Mechanical Stop Negative Search” when the drive-based homing should be modified could look as follows.
Just uncomment all *MC_BR_InitHome…* and the *MC_Home_0.HomingMode* lines.

The code as it is shown below is directly executing the homing sequence as it is setup within the Motor Wizard of LinMot-Talk.

```
PROGRAM _INIT

    gAxis_1; // Touch PV

    // MC_BR_InitHome_DS402Ax initialisations
    // This is only required if one doesn't want to use Homing Mode setup in the drive
    //  MC_BR_InitHome_DS402Ax_0.Axis := ADR(gAxis_1);
    //  MC_BR_InitHome_DS402Ax_0.HomingParameters.DriveSpecificHoming := mcSWITCH_ON;
    //  MC_BR_InitHome_DS402Ax_0.HomingParameters.DriveSpecificHomingMode := 1; // Homing Mode "Mechanical Stop Negative Search"
    //  MC_BR_InitHome_DS402Ax_0.HomingParameters.Position := -10;
    //  MC_BR_InitHome_DS402Ax_0.HomingParameters.StartVelocity := 10; // 10 mm/s
    //  MC_BR_InitHome_DS402Ax_0.HomingParameters.HomingVelocity := 10; // 10 mm/s
    //  MC_BR_InitHome_DS402Ax_0.HomingParameters.Acceleration := 100;

    // MC_Power initialisations
    MC_Power_0.Axis := ADR(gAxis_1);

    // MC_Home initialisations
    MC_Home_0.Axis := ADR(gAxis_1);
    MC_Home_0.HomingMode := mcHOMING_DEFAULT;
    //MC_Home_0.HomingMode := mcHOMING_INIT; // This is only required if one doesn't want to use Homing Mode setup in the drive
```

### 15.7.2 Rotary Motor

LinMot rotary motors (RS01 & EC02) have a single turn absolute encoder.
Therefore, as homing mode *ncHOME_OFFSET* can be used.

**Note:**
It is recommended to home axes with absolute encoders first before switching the motor on.

## 15.8 Parameter access

### 15.8.1 SDO commands

For SDO access the B&R library *AsEPL* is required.
Function blocks: *EplSDORead() & EplSDOWrite()*

The Index specifies the LinMot parameter (UPID) to be addressed. It is UPID + 16#2000.
The Sub-Index specifies the command which is performed

| Index | Sub-Index | Description | Access Type |
|---|---|---|---|
| 2000h -5FFFh | | LinMot Parameter Commands | |
| UPID Commands | | | |
| | 01h | RAM Value of UPID | Read & Write |
| | 02h | ROM Value of UPID | Read & Write |
| | 03h | Min Value of UPID | Read Only |
| | 04h | Max Value of UPID | Read Only |
| | 05h | Default Value of UPID | Read Only |
| | 06h | RAM / ROM Write of UPID | Write Only |
| System Commands | | | |
| | 07h | Set ROM to default (OS) | Write Only |
| | 08h | Set ROM to default (MC) | Write Only |
| | 09h | Set ROM to default (Interface) | Write Only |
| | 0Ah | Set ROM to default (Application) | Write Only |
| | 0Bh | Reset drive | Write Only |
| | 35h | Stop MC and APPL Software | Write Only |
| | 36h | Start MC and APPL Software | Write Only |

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

Possible variable declaration:

| | |
|---|---|
| instLMAxis01_SdoWrite | EplSDOWrite |
| instLMAxis01_SdoRead | EplSDORead |
| udiDataWrite | UDINT |
| udiDataRead | UDINT |

Possible function block calls where IF3 is the Powerlink interface:

```
// SDO read
instLMAxis01_SdoRead(pDevice:=ADR('IF3'), pData:=ADR(udiDataRead));

// SDO write
instLMAxis01_SdoWrite(pDevice:=ADR('IF3'), pData:=ADR(udiDataWrite));
```

**Example read parameter:**

Read the RAM value of the parameter Maximal Current:

**Name:** Maximal Current
**UPID:** 16#13A6
**Type:** SInt32 = DINT = 4 Bytes
**Scale:** 0.001 A

**pDevice** := pointer to POWERLINK interface, e.g., ADR('IF3')
**node** := **1** (Node ID of the LinMot drive)
**index** := **13222 = 16#33a6** = 16#2000 + 16#13a6
**subindex** := **1** (access RAM value)
**pData** := pointer to data buffer, e.g., ADR(udiDataRead)
**datalen** := **4** (always 4 Bytes! Even if the parameter to read is less)

| | | | | |
|---|---|---|---|---|
| instLMAxis01_SdoRead | EplSDORead | local | | |
| enable | BOOL | | | TRUE |
| pDevice | UDINT | | | 69733729 |
| node | USINT | | | 1 |
| index | UINT | | | 13222 |
| subindex | USINT | | | 1 |
| pData | UDINT | | | 69735842 |
| datalen | UDINT | | | 4 |
| status | UINT | | | 65535 |
| errorinfo | UDINT | | | 16#0000_0000 |
| readlen | UDINT | | | 4 |
| intern | EPL_I_TYPE | | | |
| udiDataRead | UDINT | local | | 15000 |

**Example write parameter:**

Write 1A to the RAM value of the parameter Maximal Current:

**Name:** Maximal Current
**UPID:** 16#13A6
**Type:** SInt32 = DINT = 4 Bytes
**Scale:** 0.001 A > value to be sent = 1'000

**pDevice** := pointer to POWERLINK interface, e.g., ADR('IF3')
**node** := **1** (Node ID of the LinMot drive)
**index** := **13222 = 16#33a6** = 16#2000 + 16#13a6
**subindex** := **1** (access RAM value)
**pData** := pointer to data buffer, e.g., ADR(udiDataWrite)
**datalen** := **4** (always 4 Bytes! Even if the parameter to read is less)

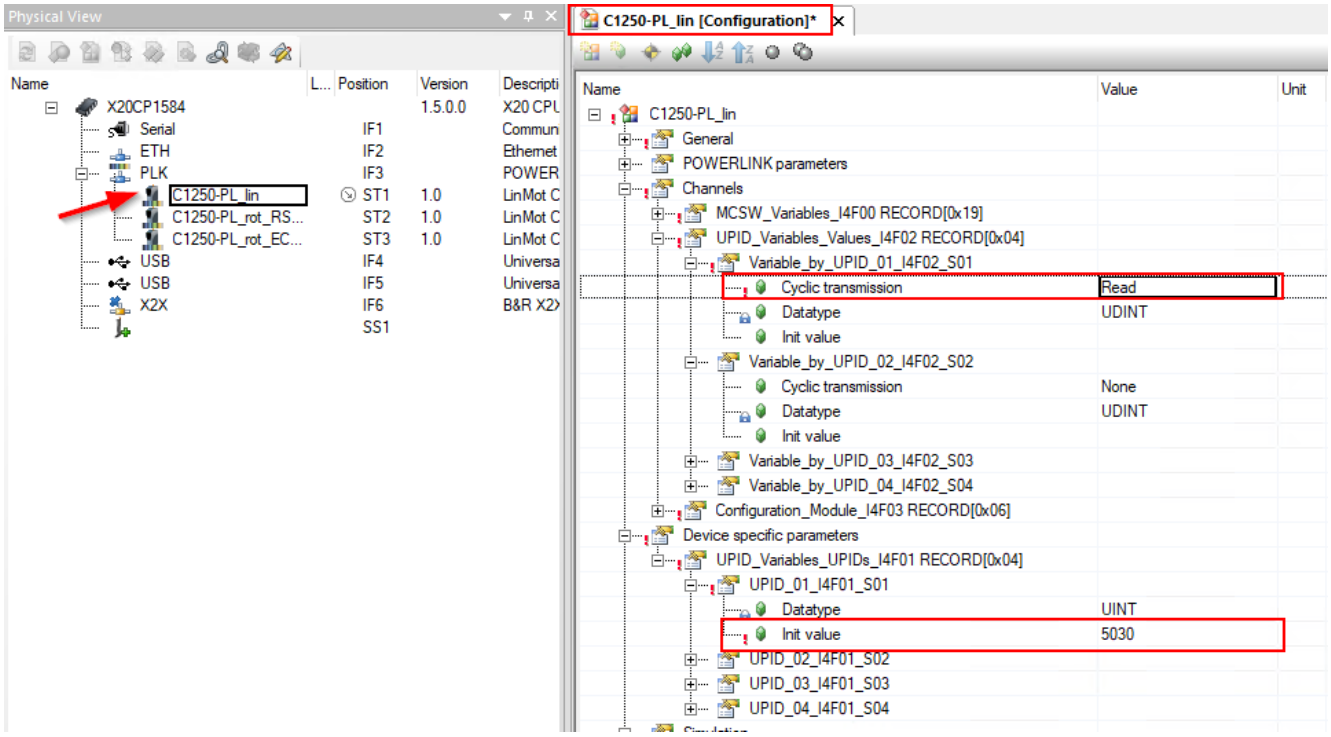
| | | | | |
|---|---|---|---|---|
| instLMAxis01_SdoWrite | EplSDOWrite | local | | |
| enable | BOOL | | | TRUE |
| pDevice | UDINT | | | 69733783 |
| node | USINT | | | 1 |
| index | UINT | | | 13222 |
| subindex | USINT | | | 1 |
| pData | UDINT | | | 69735838 |
| datalen | UDINT | | | 4 |
| status | UINT | | | 65535 |
| errorinfo | UDINT | | | 0 |
| intern | EPL_I_TYPE | | | |
| udiDataWrite | UDINT | local | | 1000 |

## 15.9 Add Additional Parameters or Variables to the Process Data

It is possible to add up to 4 parameters or variables to the cyclic PDO of the drive. This allows fast read/write access.

Right-click on the drive and select *Configuration*.

- Set the ***Cyclic transmission*** to either *Read* (for read access from the drive) or *Write* (for write access to the drive) in the “Channels”.
- As ***Init value*** in the “Device specific parameters” set the UPID number (e.g., Maximal Current has UPID 16#13A6 > 5030 decimal) of the parameter you want to access.




Afterwards you can assign a PV in the I/O configuration of the device.

# 16 Drive Profile: B&R NC Motion, SDC interface

## 16.1 Overview

This chapter shows how a LinMot drive with *Powerlink* interface (e.g., C1250-PL-XC-1S) can be integrated and setup in a B&R environment as motion axis (ACP10 > SDC > LinMot).

**Download:**
The demo project can be downloaded from:
http://download.linmot.com/plc_lib/libraries/BR/SDC/ (named *LM_SDC_Demo_...*)

The general library for LinMot drives is available from:
http://download.linmot.com/plc_lib/libraries/BR/ (v2.10.0 or later required)

Components used:

- LinMot C1250-PL-XC-1S-000 (article number 0150-1885)
  Firmware 6.7 Build 20190311 & 6.8 Build 20190315
- B&R X20CP1584
- Automation Studio V 4.3.8.58 SP#, Automation Runtime G4.34, ACP10 ARNC0 **5.2.1**
- Automation Studio V 4.4.6.71 SP#, Automation Runtime C4.45, ACP10 ARNC0 **5.4.1**

In the *B&R Help Explorer* check chapter *ACP10SDC* for more details about SDC.
It contains all important information about requirements and setup of SDC NC objects.

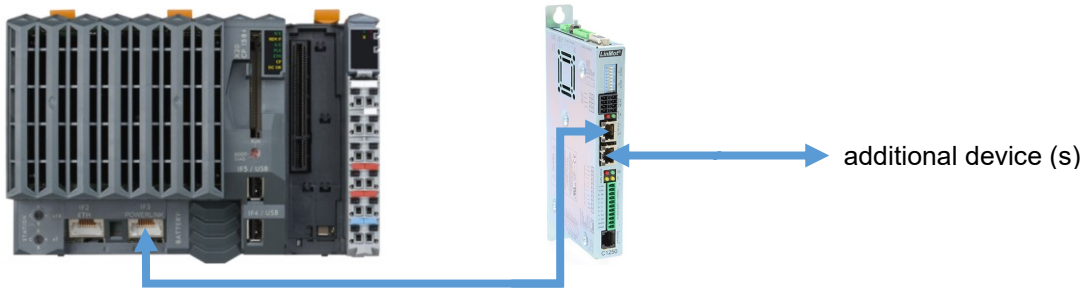



Image Source: http://www.br-automation.com/

The LinMot POWERLINK devices are Ethernet POWERLINK CNs (Controlled Nodes) with the following parameters (according to EPSG-DS-301-V1.1.0):

| Device Properties | Value |
|---|---|
| Minimal POWERLINK cycle time | 200 µs |
| Size of isochronous transmit buffer (maximal size of isochronous frames) | 64 Byte |
| Size of isochronous receive buffer (maximal size of isochronous frames) | 64 Byte |
| Overall buffer size available for isochronous data | 128 Byte |
| PReq to PRes latency (CN isochronous reaction time) | ~ 1 µs |
| SoA to ASnd latency (CN asynchronous reaction time) | ~ 1 µs |
| Maximum asynchronous MTU | 300 Byte |
| Ability to support multiplexed isochronous access | No |
| Asynchronous SDO transfer method (UDP/IP and/or POWERLINK ASnd) | POWERLINK ASnd only |

For further information on Ethernet POWERLINK please visit:
http://www.ethernet-powerlink.org

## 16.2 Compatibility LinMot_SDC <> ACP10 ARNC0

The following table lists the dependencies of the LinMot_SDC library.

| LinMot_SDC | < compatible with > | ACP10 ARNC0 |
|---|---|---|
| 1.00.1 | <-> | 5.2.0 to 5.2.9 |
| 1.00.2 | <-> | 5.4.0 to 5.4.9 |

## 16.3 Configuration and Connection of the LinMot Drive

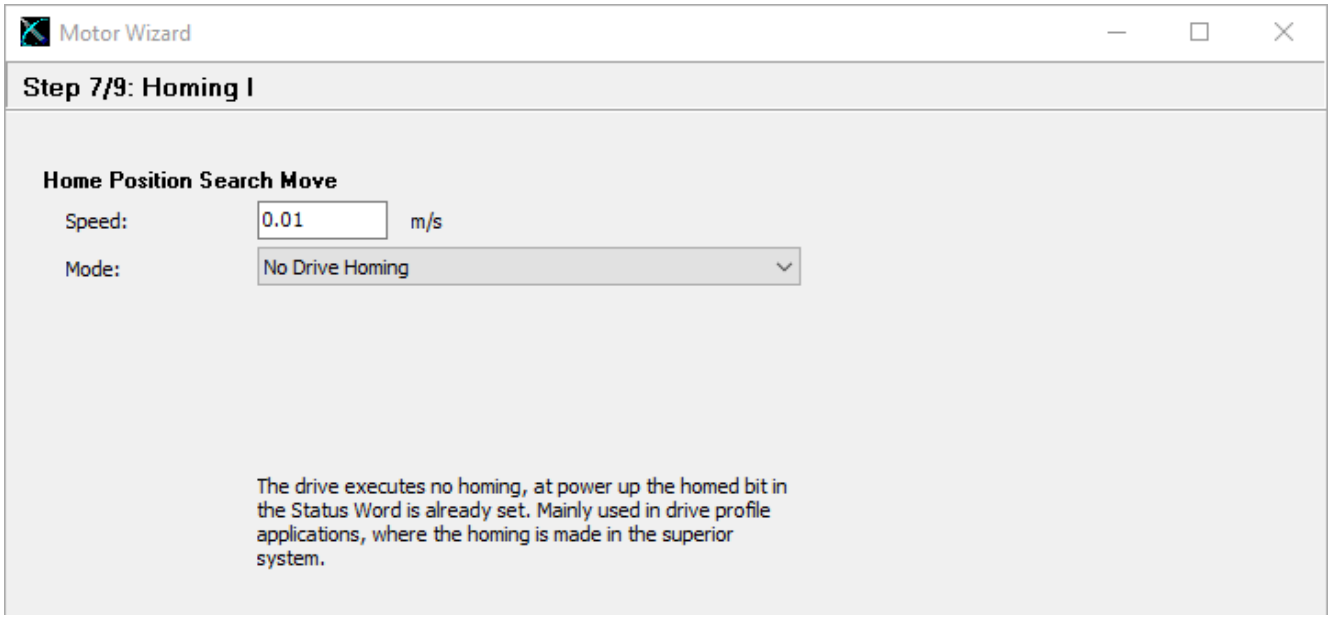
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 16.3.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select “No Drive Homing” as homing mode in the Motor Wizard as with the SDC interface the motor is homed within the PLC.




See Appendix I: Basic Position Control Loop Tuning

### 16.3.2 XDD File

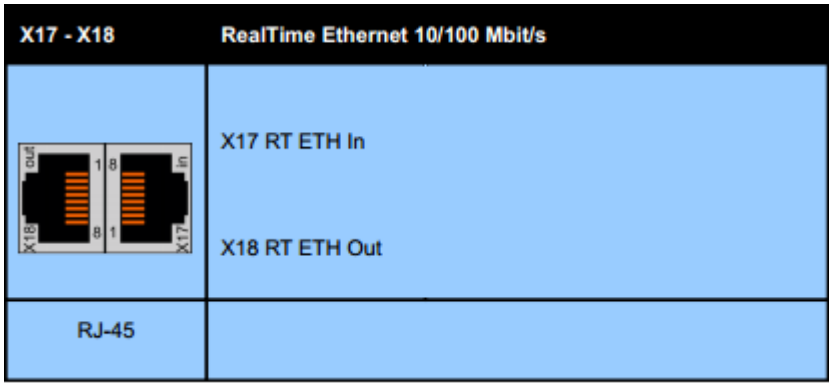
Install the XDD file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software.
It is located by default:

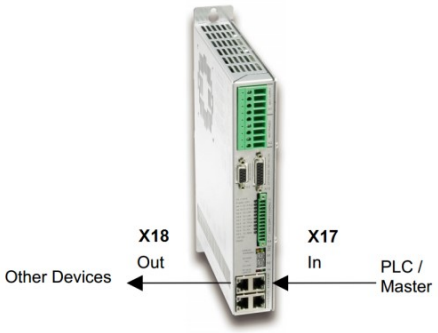
- Powerlink: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink\XDD
- Powerlink: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink_Nx\XDD (-MI drives)

There are separate XDD files available for either AS3 or AS4.

### 16.3.3 Powerlink Connection

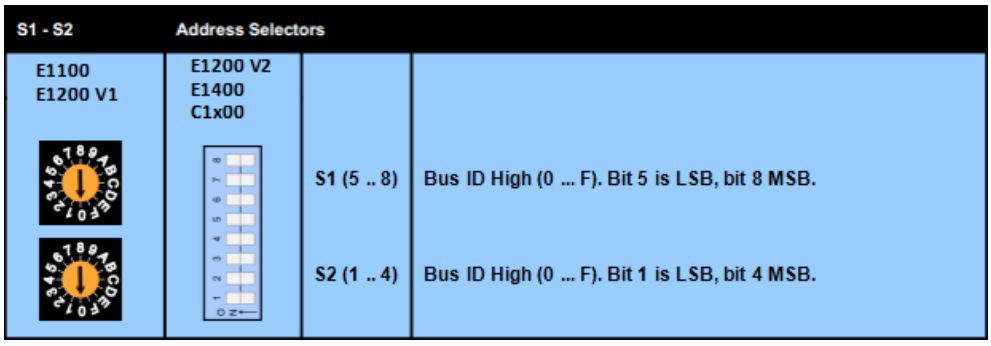
The drive is connected to the Powerlink network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-PL-UC drive. On all other LinMot drives supporting Powerlink the ports are named the same (X17 & X18) but they may be placed differently on the drive housing.




### 16.3.4 Node ID

The Node ID of the LinMot drive is set using the hardware address selectors.



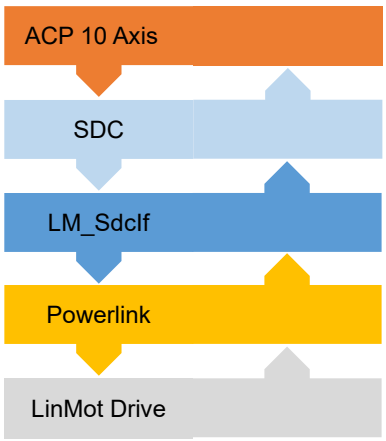

E.g.: Node ID 18 dec = 12 hex = 0001 0010 bin

=> S1 = 1 hex = 0001 bin (Dip Switch 5 = ON), S2 = 2 hex = 0010 bin (Dip Switch 2 = ON)

## 16.4 Short Description of the LinMot SDC interface function block

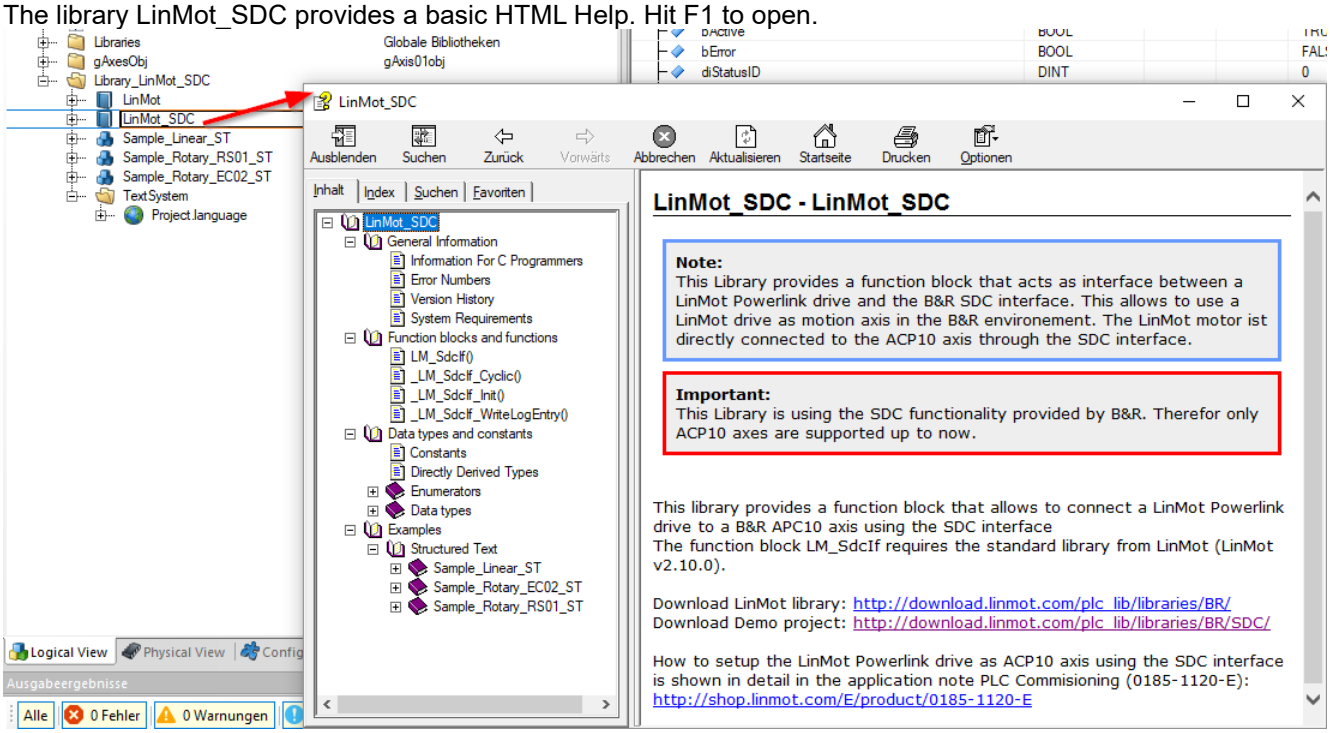
### 16.4.1 Principle of Operation

The *LM_SdcIf* function block acts as interface between the LinMot drive and the B&R SDC (Smart Device Controller) interface. On top of the SDC interface the ACP10 Manager is located that manages the ACP10 axes.



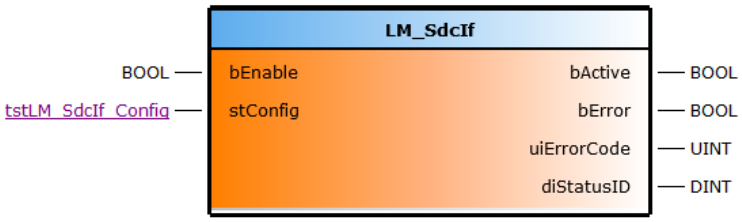

### 16.4.2 HTML Help

The library LinMot_SDC provides a basic HTML Help. Hit F1 to open.



### 16.4.3 LM_SdcIf (Interface Function Block)

This is the function block that acts as interface between SDC and LinMot drive.




### 16.4.4 enCallType

The function block LM_SdcIf has two call types.

- enLM_SDCIF_CALLTYPE_INIT (value 1):
  With this type the function block LM_SdcIf is called in the Init section of the PLC program to setup all required internal variables

- enLM_SDCIF_CALLTYPE_CYCLE (value 2):
  With this type the function block LM_SdcIf is called in the cyclic section of the PLC program. This is the normal operation mode.

### 16.4.5 stConfig

This structure contains the parameters required for a proper operation of the LM_SdcIf function block.

| Name | Data Type | Default Value | Description | |
|---|---|---|---|---|
| pszLogName | UDINT | 0 | Log name to be used / created. Set 0 if not logger should be used / created | |
| pszAxisName | UDINT | 0 | ACP10 axis name. | |
| pszSDCHWName | UDINT | 0 | Name of SDC HW without **_HW** | *Name of SDC HW* |
| *pszSDCDrvIf32Name* | *UDINT* | *0* | *Name of SDC Drv32If* **(Attention: max length of string is 34 characters!)** | |
| *pszSDCEncIf32Name* | *UDINT* | *0* | *Name of SDC Enc32If* **(Attention: max length of string is 34 characters!)** | |
| *pszSDCDiDoIf32Name* | *UDINT* | *0* | *Name of SDC DiDoIf* **(Attention: max length of string is 34 characters!)** | |
| *pszSDCLmName* | *UDINT* | *0* | *Name of LinMot tstLM_LinMot* | |
| udiSM_SCALE_LOAD_MOTREV | UDINT | 1 | Same as used in the ACOPOS Parameter Table | |
| udiSM_SCALE_LOAD_UNITS | UDINT | 10000 | Same as used in the ACOPOS Parameter Table | |
| udiSM_SCALE_STEP | UDINT | 10000 | Same as used in the ACOPOS Parameter Table | |
| diResReductionFactor | DINT | 10 | Resolution reduction (Default 10 for LinMot linear motors. Use 1 for other motors) | |
| diEncReductionFactor | DINT | 10 | Encoder reduction (Default 10 for LinMot linear motors. Use 1 for other motors) | |
| tEnableDelay | TIME | T#10ms | Delay velocity calculation during enable | |

**Note:**
The default values of the parameters in the table with blue background color are set to be used with a LinMot linear motor. This default values and the settings done in chapter 16.8 result in 1um per Unit and 10um actual position resolution.

**Attention:**
The orange parameters in italic font are optional. In the examples of this manual they are not used. In this manual, the names of the individual SDC components are derived from *pszSDCHWName* as shown in chapter 16.5 point 3.
Nevertheless, it is possible to set unique names for each SDC component and the LinMot axis type (tstLM_axis) and assign them to the parameters above (orange).

## 16.5 Prerequisites

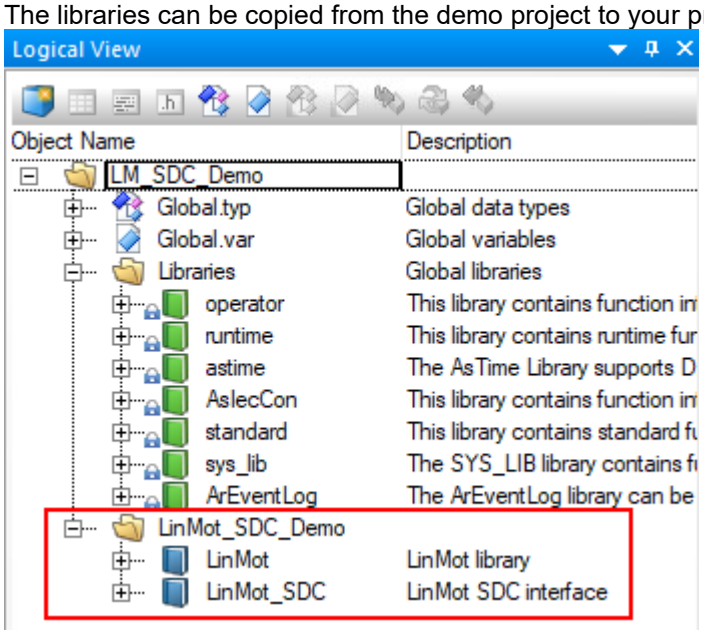
**Attention:**
There are some prerequisites that must be fulfilled before the following chapters can be successfully completed.

1. The project you are using must already have the PLC type setup. Otherwise please follow the B&R wizard for creating a new project

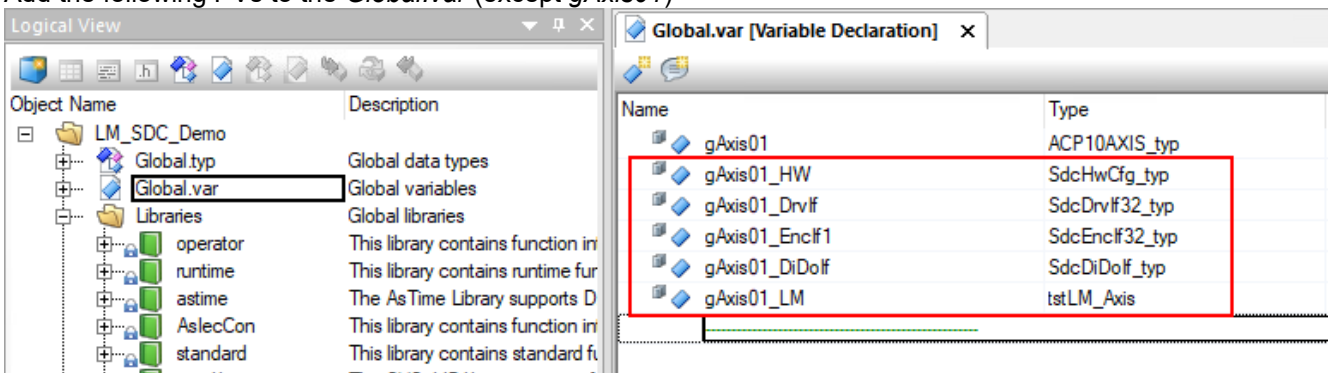
2. The libraries from LinMot (*LinMot* v2.10.0 or later & *LinMot_SDC*) must be installed in the project.
   Check compatibility with ACP10 ARNC0 in **chapter 16.2 Compatibility LinMot_SDC <> ACP10 ARNC0**

   You may want to create a separate package for testing (E.g., LinMot_SDC_Demo).

   The libraries can be copied from the demo project to your project:




3. Add the following PVs to the *Global.var* (except *gAxis01*)



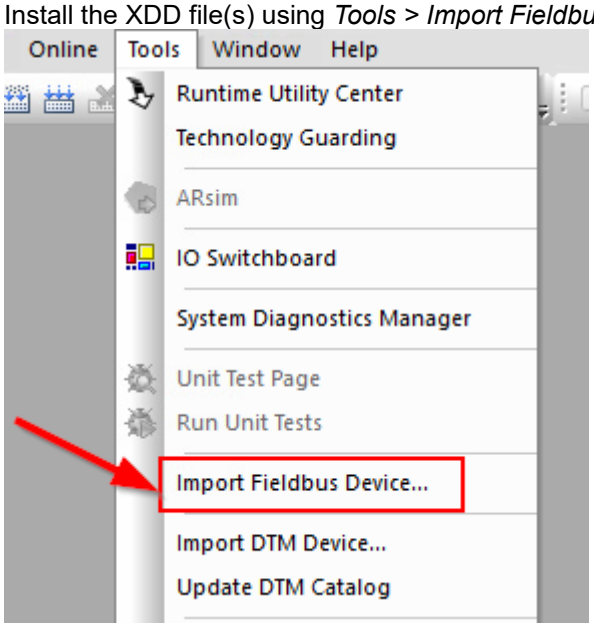

**Attention:**
Don't add the PV ***gAxis01*** (ACP10Axis_typ) yet. It will be added in chapter 16.6.5.

**Attention:**
The names of the PVs are very important! As shown above the PVs beside the ACP10AXIS (gAxis01) must be named *xxx_HW, xxx_DrvIf, xxx_EncIf1, xxx_DiDoIf, xxx_LM* where xxx is always the same for the same axis (e.g., *gAxis01*).

## 16.6 Automation Studio Physical View

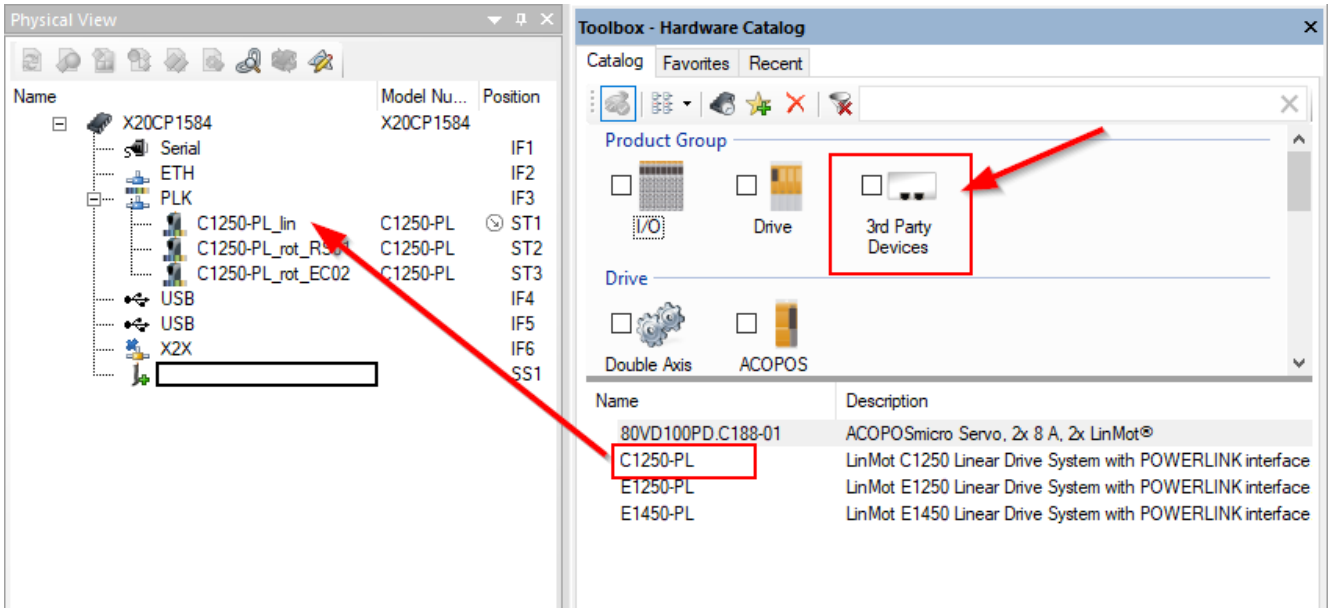
### 16.6.1 Add the LinMot drive to the Powerlink network

Install the XDD file(s) using *Tools* > *Import Fieldbus Device…* in Automation Studio.




The required XDD-file can be found in the following folder (default):
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\POWERLINK\XDD
C:\Program Files\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\Powerlink_Nx\XDD

In the Hardware Catalog search for the LinMot drive (e.g., C1250...) and drag&drop it onto the Powerlink interface (PLK).
Alternatively select the Product Group *3rd Party Devices* > all installed LinMot drives are shown.

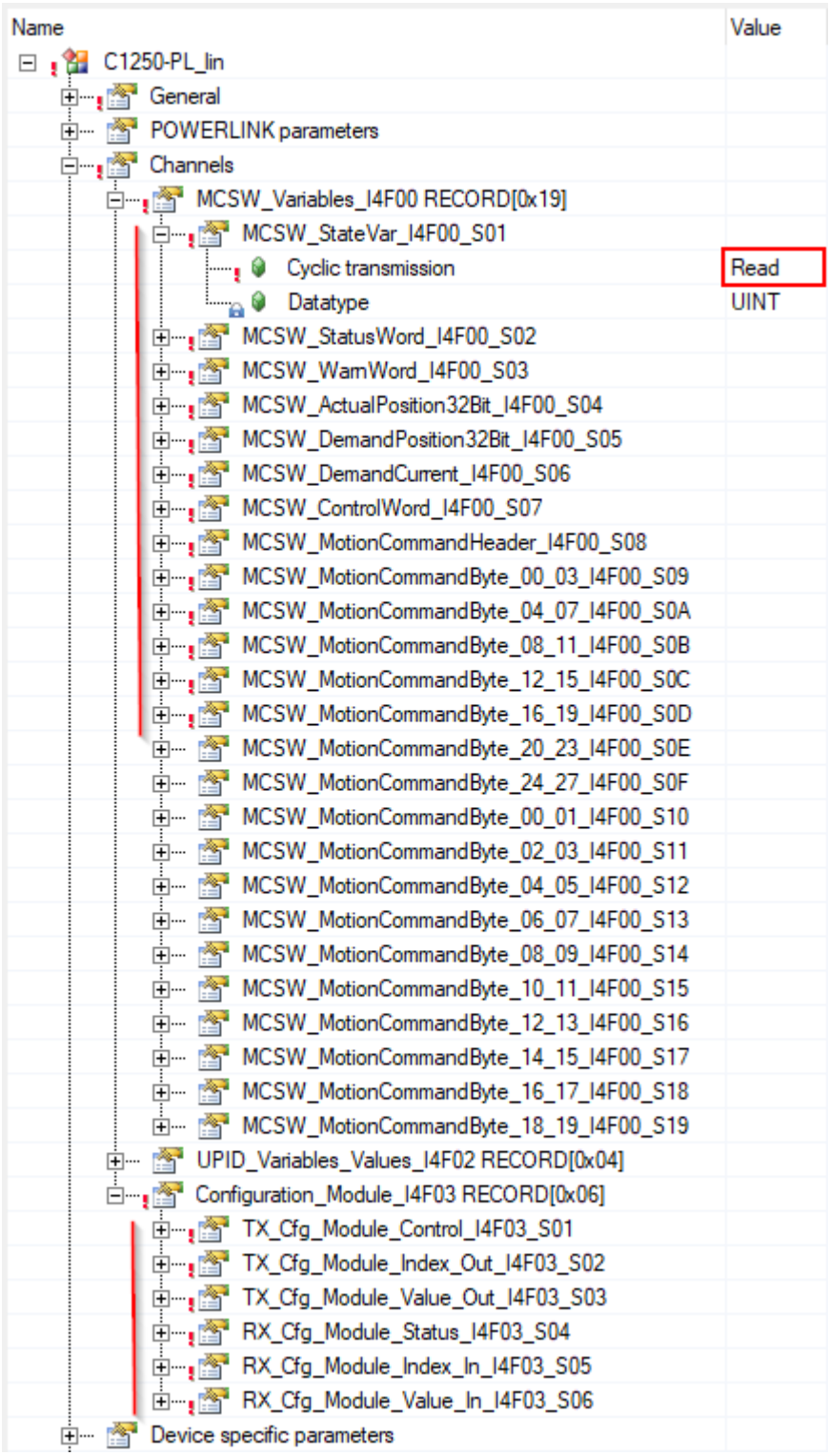



Set the Node ID according to your requirements.

### 16.6.2 Configuration

In the configuration of the LinMot slave module (right-click on device > *Configuration*) the following channels must be selected:

Set the **Cyclic transmission** of all channels marked with a red exclamation mark to **Read** or to **Write** respectively.



### 16.6.3 I/O Mapping

Right-click on the device and open *I/O mapping.* The inputs and outputs of the modules are mapped to the axis structure (datatype tstLM_Axis).

In the actual example the used PV *gAxis01_LM* is defined in the *Global.var* as data type *tstLM_Axis:*

Global.var [Variable Declaration]

| Name | Type |
|---|---|
| gAxis01 | ACP10AXIS_typ |
| gAxis01_HW | SdcHwCfg_typ |
| gAxis01_DrvIf | SdcDrvIf32_typ |
| gAxis01_EncIf1 | SdcEncIf32_typ |
| gAxis01_DiDoIf | SdcDiDoIf_typ |
| gAxis01_LM | tstLM_Axis |

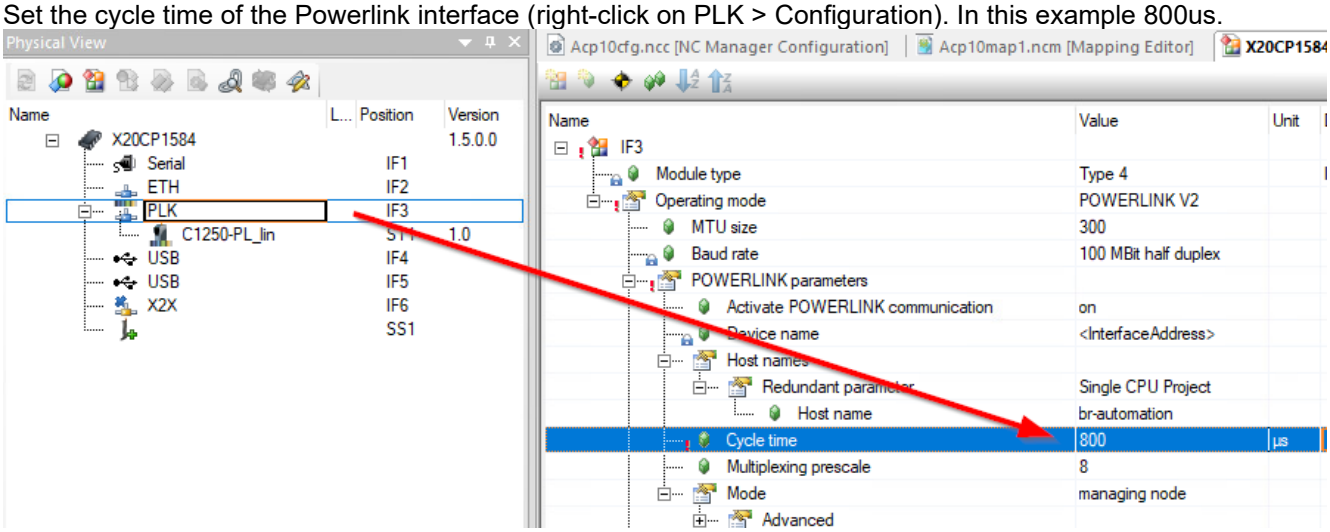
Map the channels to the process variables as shown below

Global.var [Variable Declaration] | C1250-PL_lin [I/O Mapping]

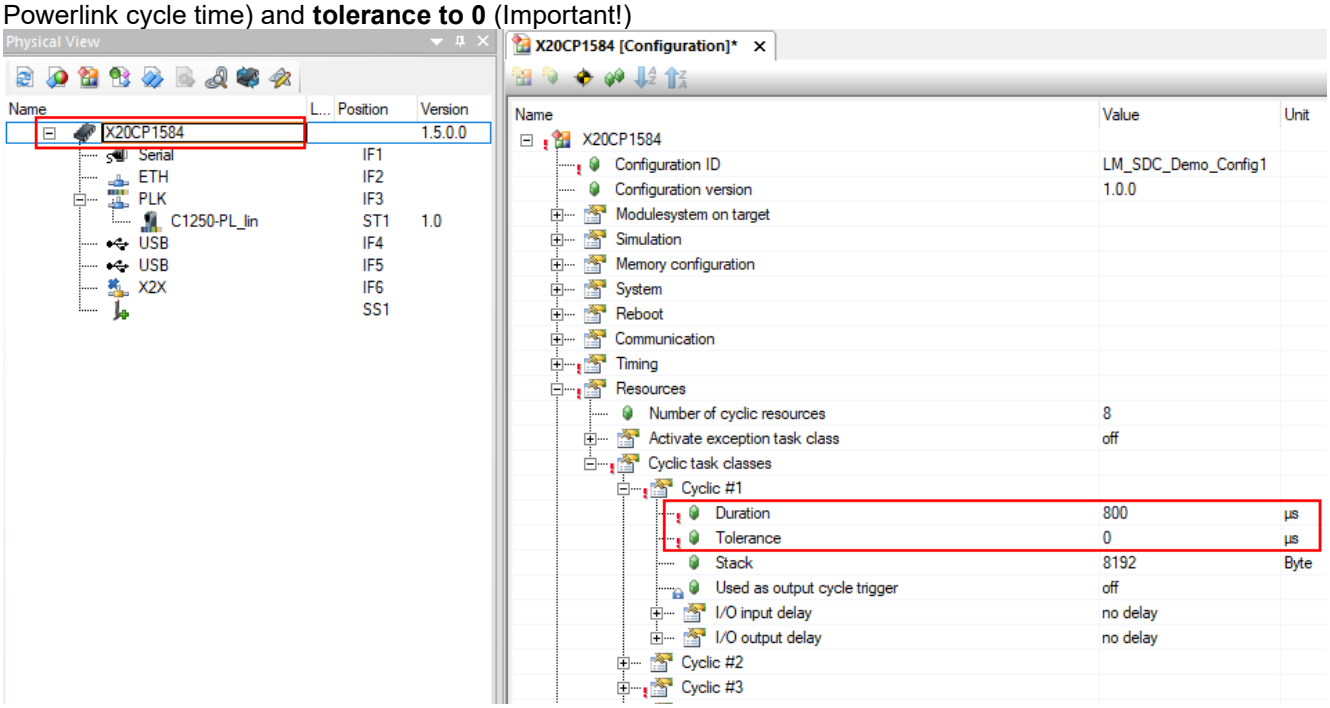
| Channel Name | Process Variable | Data Type |
|---|---|---|
| ModuleOk | | BOOL |
| MCSW_StateVar_I4F00_S01 | ::gAxis01_LM.DrvToPlc.StateVar | UINT |
| MCSW_StatusWord_I4F00_S02 | ::gAxis01_LM.DrvToPlc.StatusWord | UINT |
| MCSW_WarnWord_I4F00_S03 | ::gAxis01_LM.DrvToPlc.WarnWord | UINT |
| MCSW_ActualPosition32Bit_I4F00_S04 | ::gAxis01_LM.DrvToPlc.ComActualPosition | DINT |
| MCSW_DemandPosition32Bit_I4F00_S05 | ::gAxis01_LM.DrvToPlc.ComDemandPosition | DINT |
| MCSW_DemandCurrent_I4F00_S06 | ::gAxis01_LM.DrvToPlc.ComActualCurrent32 | DINT |
| MCSW_ControlWord_I4F00_S07 | ::gAxis01_LM.PlcToDrv.ControlWord | UINT |
| MCSW_MotionCommandHeader_I4F00_S08 | ::gAxis01_LM.PlcToDrv.MCHeader | UINT |
| MCSW_MotionCommandByte_00_03_I4F00_S09 | ::gAxis01_LM.PlcToDrv.MCParaDWord_00_03 | UDINT |
| MCSW_MotionCommandByte_04_07_I4F00_S0A | ::gAxis01_LM.PlcToDrv.MCParaDWord_04_07 | UDINT |
| MCSW_MotionCommandByte_08_11_I4F00_S0B | ::gAxis01_LM.PlcToDrv.MCParaDWord_08_11 | UDINT |
| MCSW_MotionCommandByte_12_15_I4F00_S0C | ::gAxis01_LM.PlcToDrv.MCParaDWord_12_15 | UDINT |
| MCSW_MotionCommandByte_16_19_I4F00_S0D | ::gAxis01_LM.PlcToDrv.MCParaDWord_16_19 | UDINT |
| TX_Cfg_Module_Control_I4F03_S01 | ::gAxis01_LM.PlcToDrv.CfgControlWord | UINT |
| TX_Cfg_Module_Index_Out_I4F03_S02 | ::gAxis01_LM.PlcToDrv.CfgIndexOut | UINT |
| TX_Cfg_Module_Value_Out_I4F03_S03 | ::gAxis01_LM.PlcToDrv.CfgValueOut | UDINT |
| RX_Cfg_Module_Status_I4F03_S04 | ::gAxis01_LM.DrvToPlc.CfgStatusWord | UINT |
| RX_Cfg_Module_Index_In_I4F03_S05 | ::gAxis01_LM.DrvToPlc.CfgIndexIn | UINT |
| RX_Cfg_Module_Value_In_I4F03_S06 | ::gAxis01_LM.DrvToPlc.CfgValueIn | UDINT |

### 16.6.4 Powerlink Interface and PLC Configuration

Set the cycle time of the Powerlink interface (right-click on PLK > Configuration). In this example 800us.




In the PLC configuration (right-click on X20CP1584 > Configuration) set Cyclic #1 **duration to 800us** (same as Powerlink cycle time) and **tolerance to 0** (Important!)

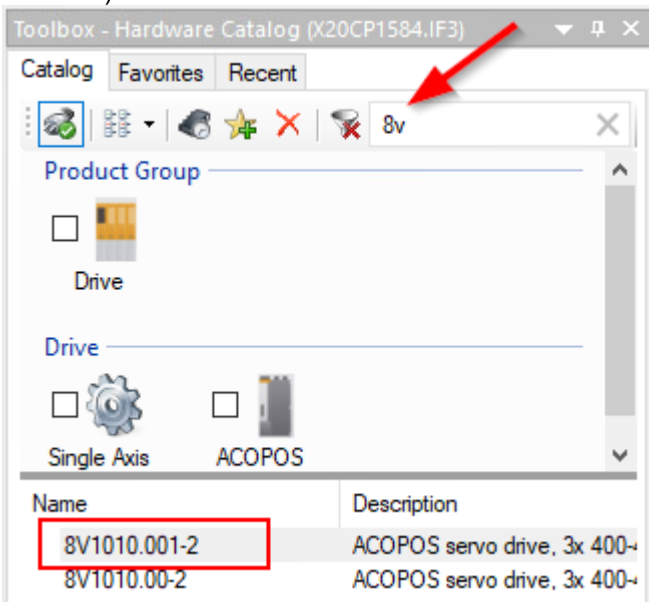



**Note:**
Other cycle times are possible too. Just make sure that Cyclic #1 *Duration* and Powerlink *Cycle Time* are set the same (e.g. 2000us).

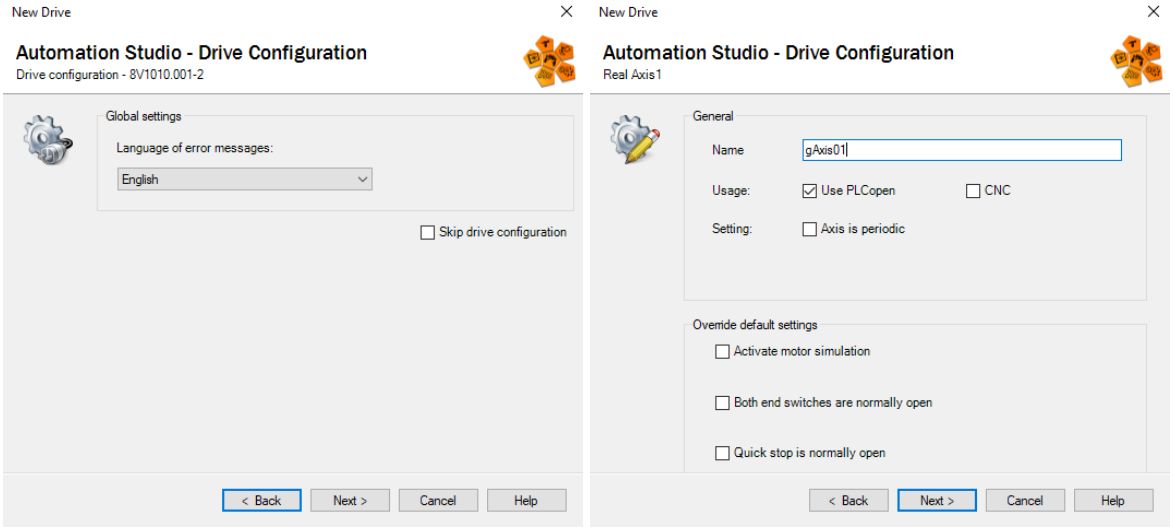
### 16.6.5 Add a B&R drive to generate all NC object specific requirements automatically

In the *B&R Help Explorer* check chapter *ACP10SDC > Using the Drive Configuration wizard*.
It explains how to add the required components.
As alternative it is also possible to do all steps manually as described in the B&R Help Explorer.

1. Search for 8V1010.001-2 in the *Hardware Catalog* and then double-click (or drag&drop on the Powerlink interface)

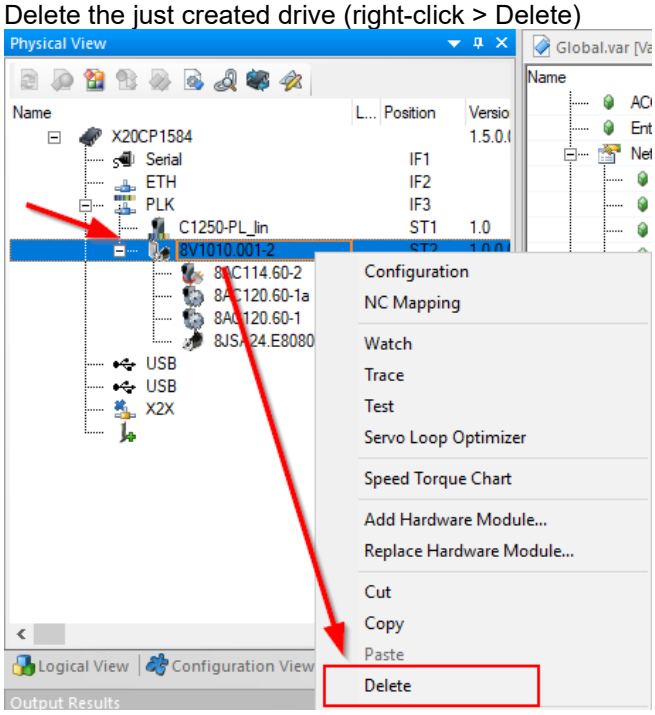



2. Click through the Drive Configuration Wizard without changing anything until *Global Settings*. Then select the language, set a name for the axis (gAxis01 in this example) and click Finish at the end.




Now all required libraries for NC objects are added to the project. Also, the *NC manager configuration* and the *NC mapping* are created.

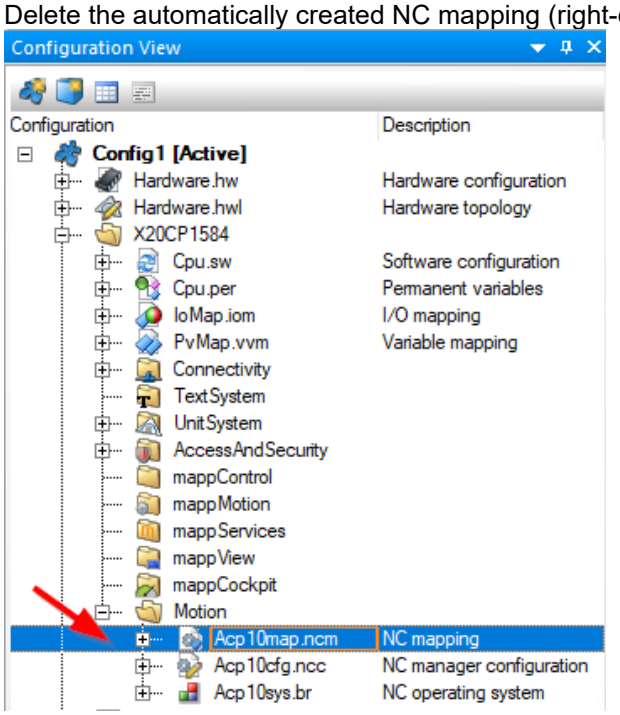
3. Delete the just created drive (right-click > Delete)



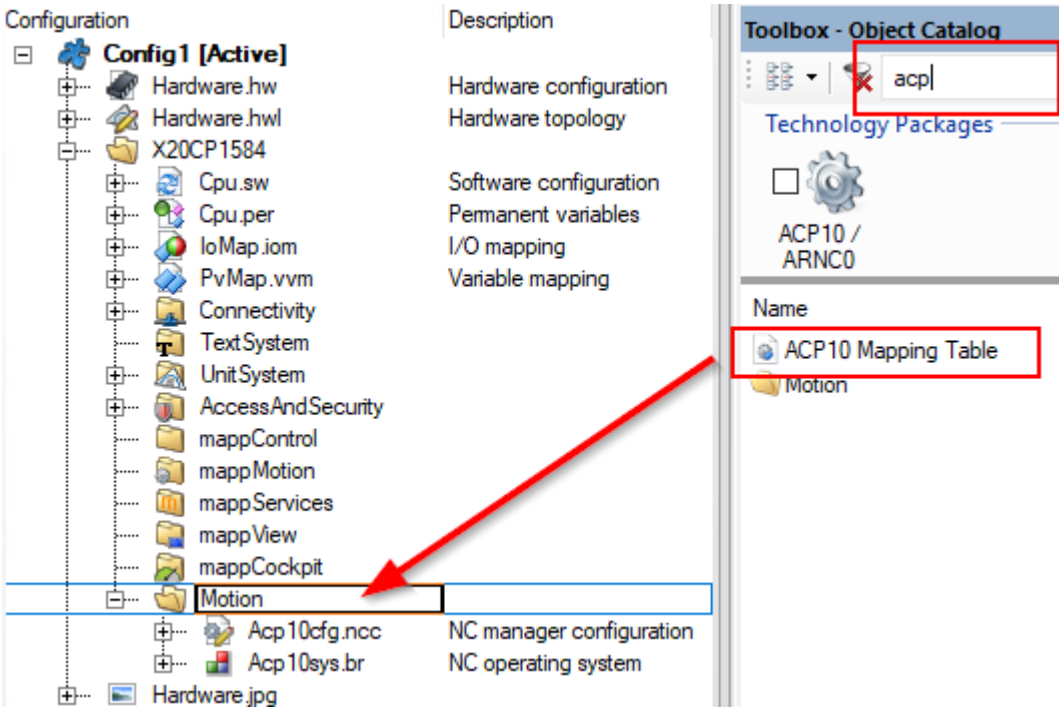
## 16.7 Automation Studio Configuration View

### 16.7.1 Setup NC Manager and NC Mapping

Delete the automatically created NC mapping (right-click > Delete)

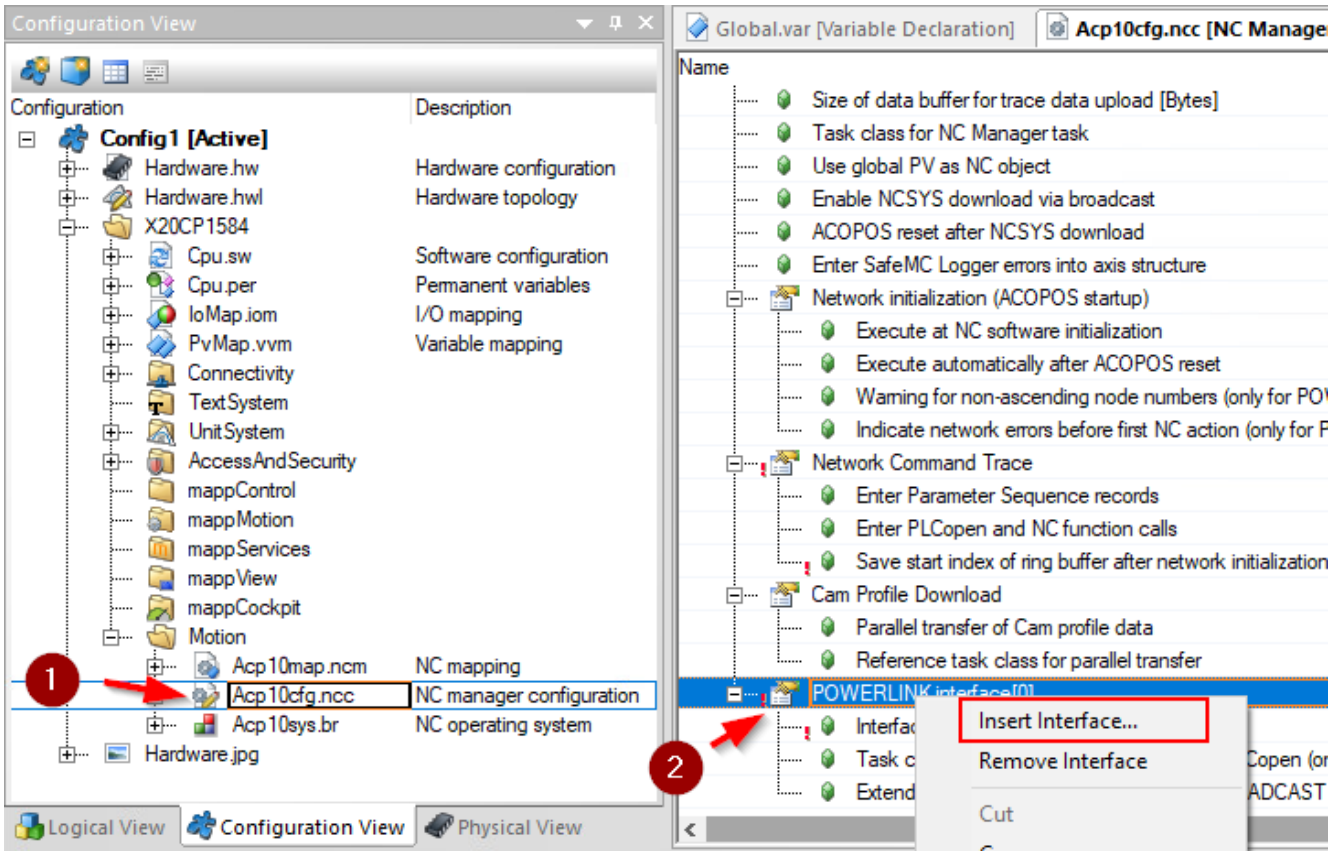



Add a new NC mapping by drag&drop *ACP10 Mapping Table* from the Toolbox – Object Catalog

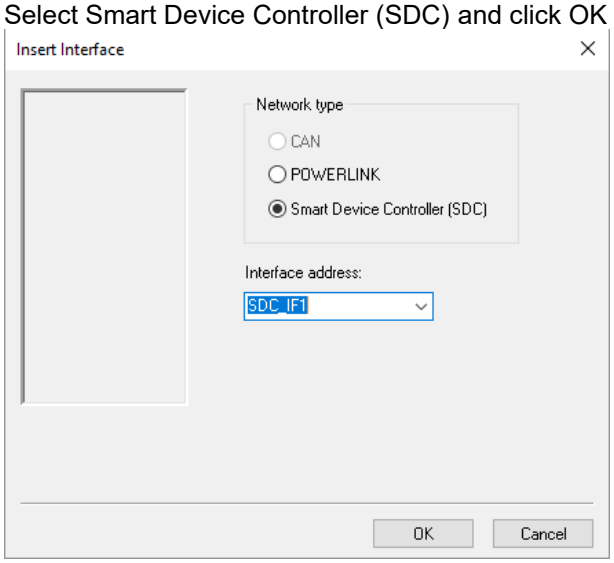



**Attention:**
Assign a name that is different to *Acp10map.ncm* or you will later not be able to add a new NC object. E.g.name it *Acp10map1.ncm*

Double-click on the NC manager configuration. Then right-click on the POWERLINK interface (or anywhere else in the NC manager configuration) and select “Insert Interface”

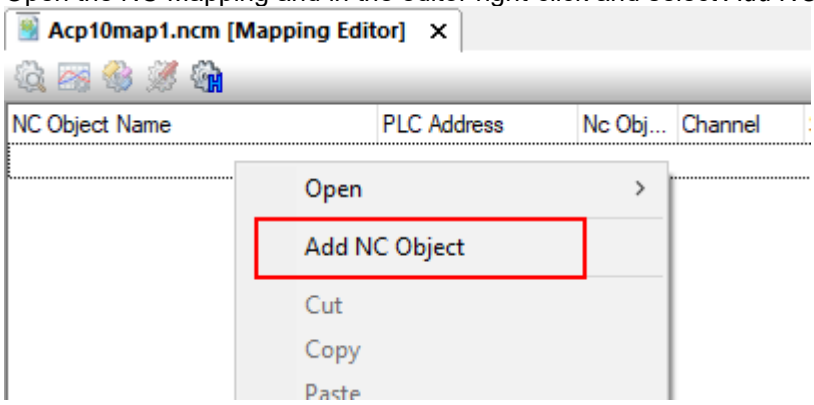



Select Smart Device Controller (SDC) and click OK



### 16.7.2 Add new NC Object

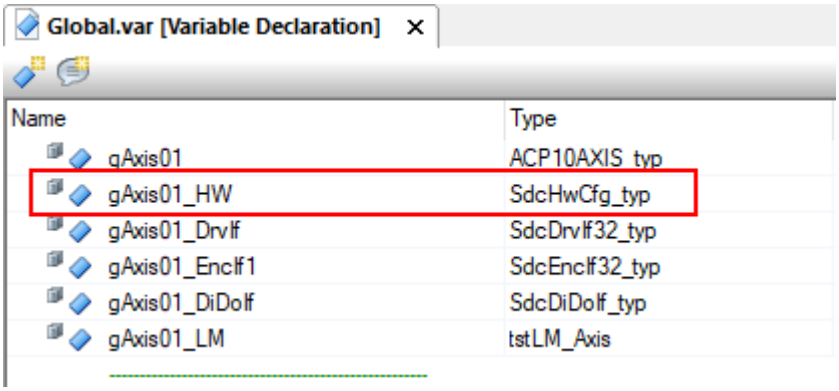
Open the NC Mapping and in the editor right-click and select *Add NC Object*




In the new empty line enter the following data:

| NC Object Name | PLC Address | Nc Object Type | Channel | Simulation | NC INIT Parameter | ACOPOS Parameter | Additional Data |
|---|---|---|---|---|---|---|---|
| gAxis01 | SDC_IF1.ST1 | ncAXIS | 1 | Off | gAxis01i | gAxis01a | SDC_AX_HwPv="gAxis01_HW" |

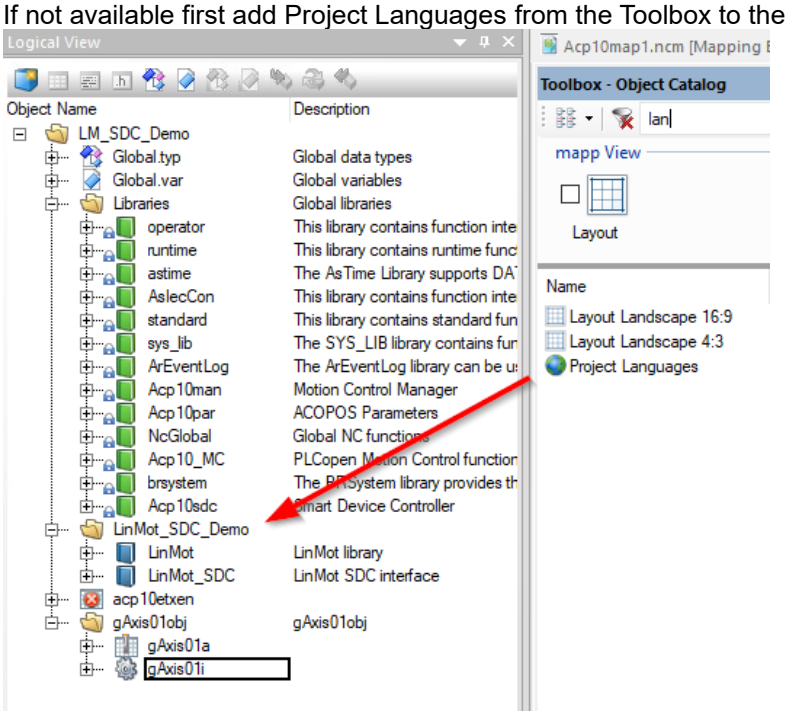
**NC Object Name:** Name of the ACP10Axis in Global.var. In this example *gAxis01*
**PLC Address:** Select the SDC interface, e.g. SDC_IF1.STx (Where x is a number. Here 1)
**Nc Object Type:** ncAxis
**Channel:** 1
**Simulation:** Off
**NC INIT Parameter:** Init Parameter table for this axis. In this example *gAxis01i*
**ACOPOS Parameter:** ACOPOS Parameter table for this axis. In this example *gAxis01a*
**Additional Data:** Name of SDC HW PV. In this example *gAxis01_HW* as defined in *Global.var*



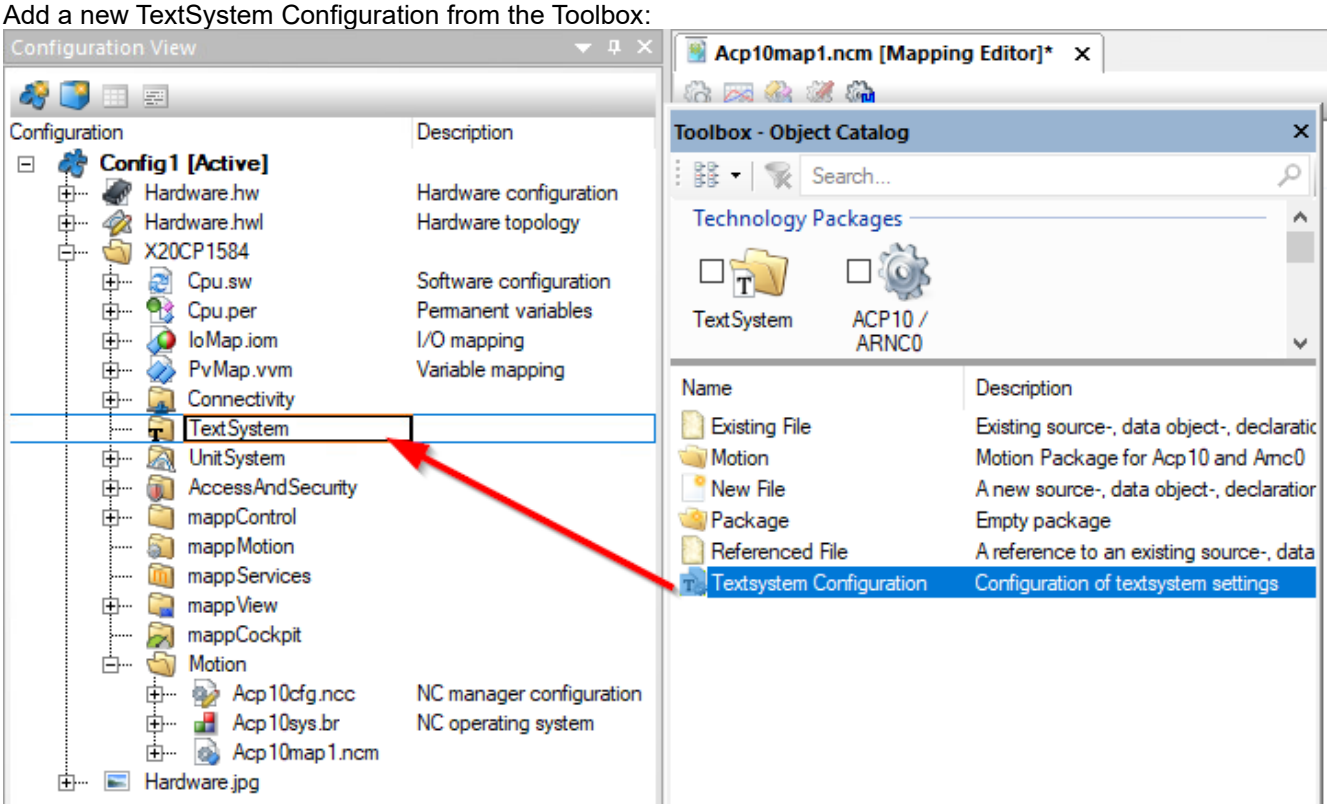
### 16.7.3 Add TextSystem

**Note:**
If you don't want to use the logger functionality this chapter can be skipped.

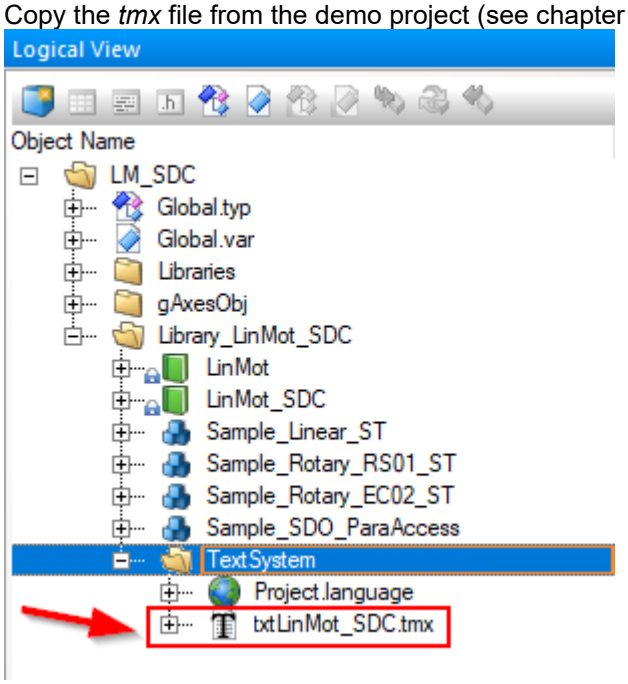
If not available first add Project Languages from the Toolbox to the project (in Logical View) and set it up:




Add a new TextSystem Configuration from the Toolbox:



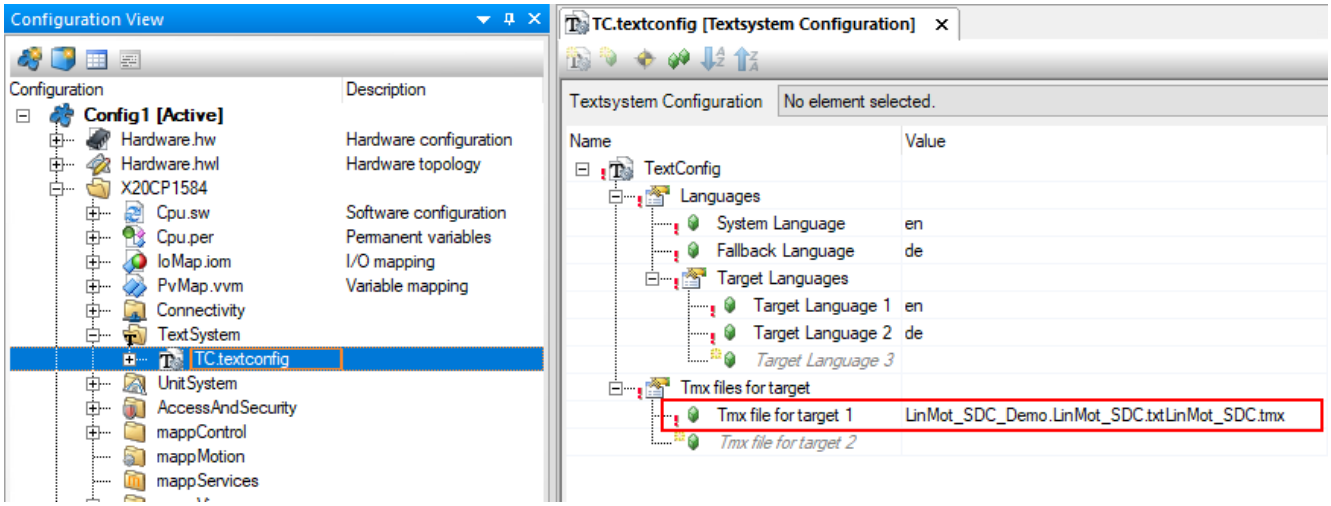
Copy the *tmx* file from the demo project (see chapter 16.1) to your project:




Open the TextSystem Configuration and set it up by selecting the *.tmx file just copied.

To do so right-click on “Tmx file for target 1” and select *Add New Dynamic Node*.

Click in the empty Value field and choose *xxx.LinMot_SDC.txtLinMot_SDC.tmx*



## 16.8 Automation Studio Logical View: Linear Motor Example

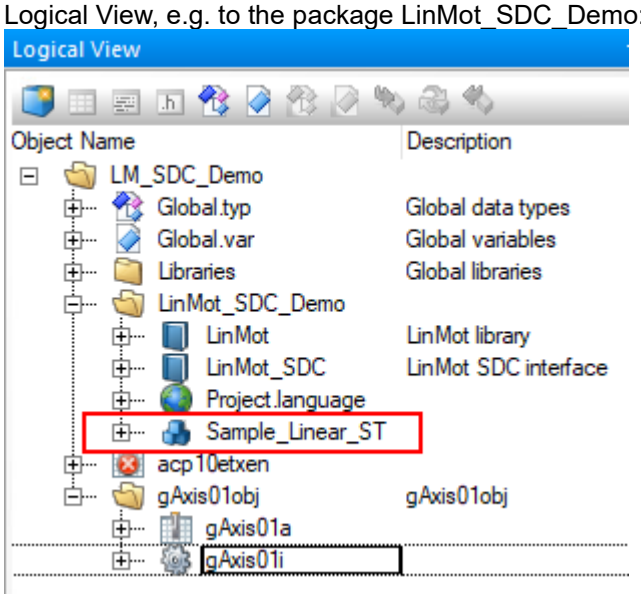
This chapter shows the integration of a LinMot linear motor.

By default, a LinMot linear drive has a maximum resolution of 0.1um of the actual and demand positions. The SDC encoder interface has the limitation that it must be able to check two times per task/Powerlink cycle the actual encoder value of the motor. With 10'000 Units per motor revolution (=1mm) and a task/Powerlink cycle time of 800us this limits the maximum reachable velocity to around 400mm/s which is slow for a LinMot linear motor.

Therefore, the resolution is changed to 1um by setting the SCALE_ENC_INCR and the parameter *diEncReductionFactor* of the LinMot SDC interface function block accordingly. This allows maximum velocities of up to 4'000 mm/s which is enough for most applications.

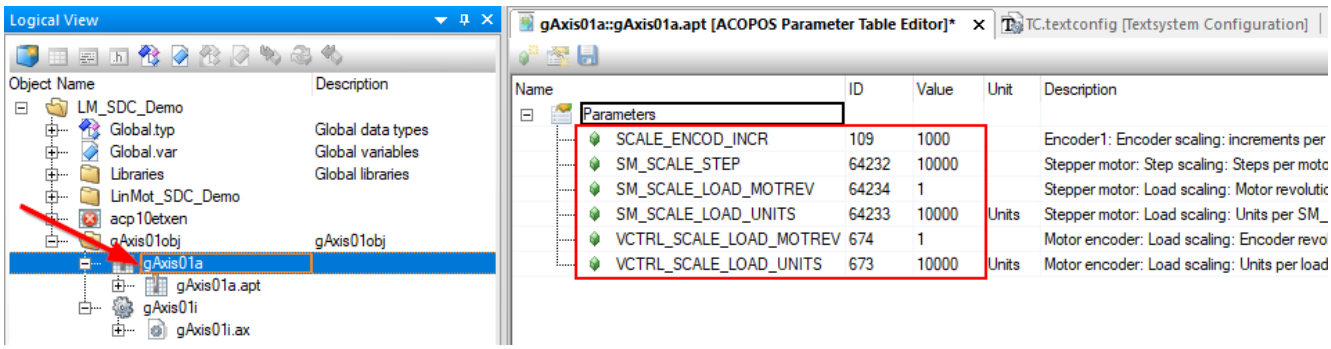
### 16.8.1 Add the example program to the project

Copy the sample program *Sample_Linear_ST* from the demo project (see chapter 16.1) and past it to the Logical View, e.g. to the package LinMot_SDC_Demo:




### 16.8.2 ACOPOS Parameter Table

For a LinMot linear motor the parameters must be set exactly as shown below. This will result in a resolution of 1um per Unit.



### 16.8.3 Init Parameter Table

Set all digital input active levels to *ncACTIV_HI*.
Set *Units* in *Encoder_if* to 10000 (> 1um per Unit with LinMot linear motor)
Set *t_predict* and *t_total* to 5 times Powerlink cycle time each > 800us * 5 = 0.004 s

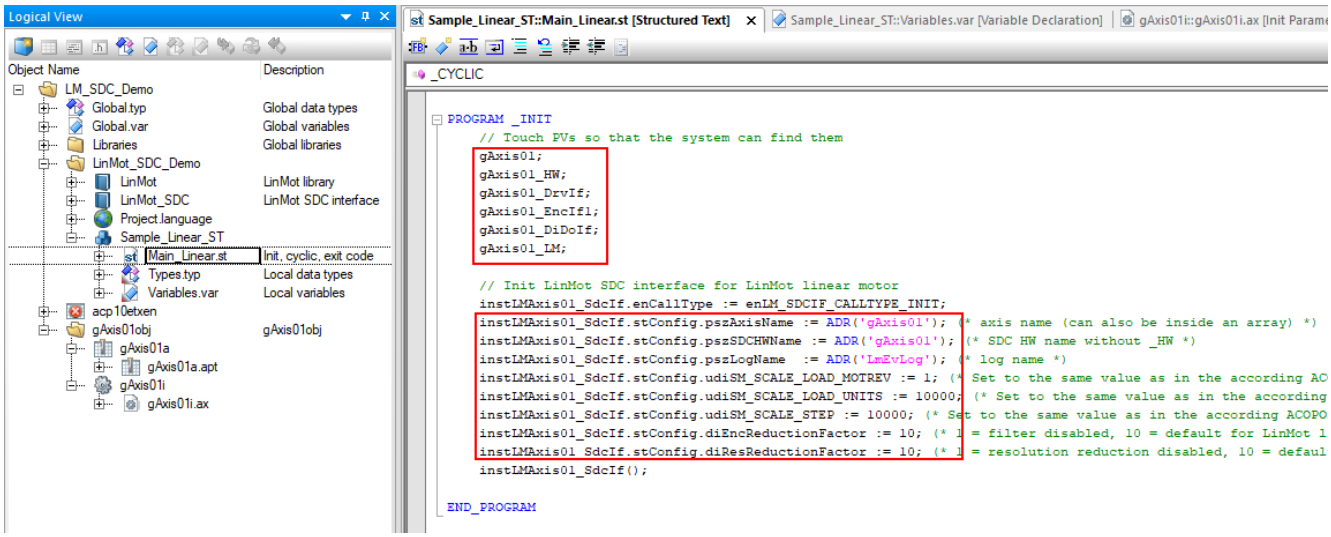
| Name | Value | Unit | Description |
|---|---|---|---|
| ACP10AXIS_typ | | | |
| dig_in | | | Digital Inputs |
| level | | | Active Input Level |
| reference | ncACTIV_HI | | Reference switch |
| pos_hw_end | ncACTIV_HI | | Positive HW end switch |
| neg_hw_end | ncACTIV_HI | | Negative HW end switch |
| trigger1 | ncACTIV_HI | | Trigger1 |
| trigger2 | ncACTIV_HI | | Trigger2 |
| encoder_if | | | Encoder Interface |
| parameter | | | Parameters |
| count_dir | ncSTANDARD | | Count direction |
| scaling | | | Scaling |
| load | | | Load |
| units | 10000 | Units | Units at the load |
| rev_motor | 1 | | Motor revolutions |
| limit | | | Limit value |
| controller | | | Controller |
| mode | ncPOSITION | | Mode |
| position | | | Position Controller |
| kv | 50 | 1/s | Proportional amplification |
| tn | 0 | s | Integral action time |
| t_predict | 0.004 | s | Prediction time |
| t_total | 0.004 | s | Total time |
| p_max | 1.0e30 | Units/s | Maximum proportional action |
| i_max | 0 | Units/s | Maximum integral action |

As the limits depend on the application and the motor type used it is recommended to start with some moderate values.

| | | | |
|---|---|---|---|
| limit | | | Limit value |
| parameter | | | Parameters |
| v_pos | 1000000.0 | Units/s | Speed in positive direction |
| v_neg | 1000000.0 | Units/s | Speed in negative direction |
| a1_pos | 5000000.0 | Units/s² | Acceleration in positive direction |
| a2_pos | 5000000.0 | Units/s² | Deceleration in positive direction |
| a1_neg | 5000000.0 | Units/s² | Acceleration in negative direction |
| a2_neg | 5000000.0 | Units/s² | Deceleration in negative direction |
| t_jolt | 0 | s | Jolt time |
| t_in_pos | 0 | s | Settling time before message 'In Position' |
| pos_sw_end | 2000000000 | Units | Positive SW end |
| neg_sw_end | -2000000000 | Units | Negative SW end |
| ds_warning | 2000.0 | Units | Lag error limit for display of a warning |
| ds_stop | 5000.0 | Units | Lag error limit for stop of a movement |
| a_stop | 1.0e30 | Units/s² | Acceleration limit for stop of a movement |
| dv_stop | 0 | 1/s | Speed error limit for stop of a movement |
| dv_stop_mode | ncOFF | | Mode for speed error monitoring |

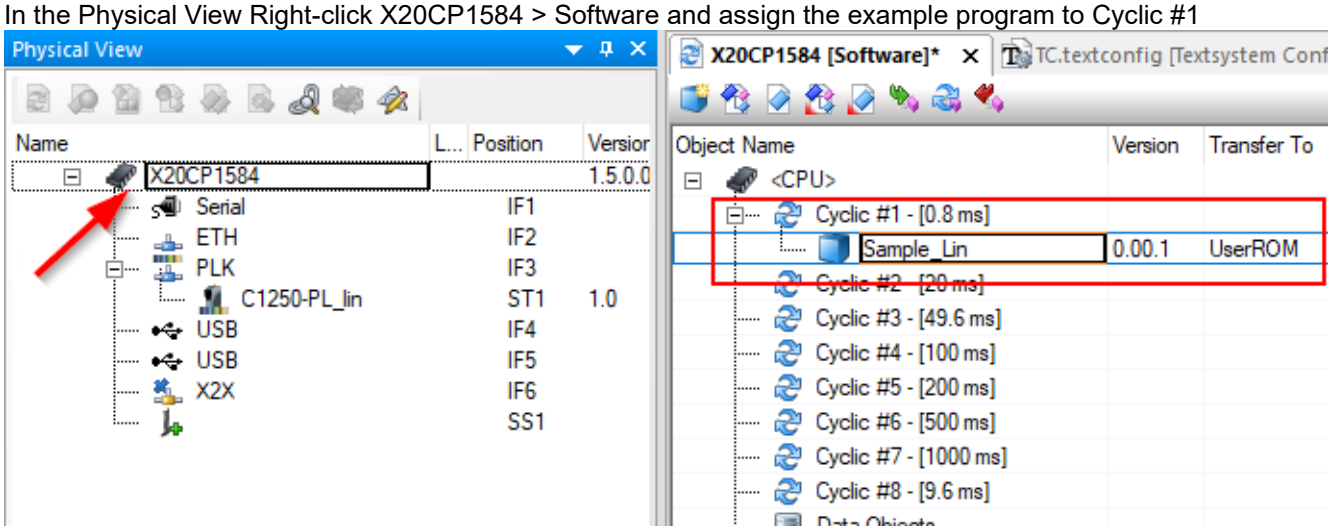
### 16.8.4 Check Init Parameters of the SDC function block

Open Main_Linear.st and check if the axis, SDCHW and log names as well as the other parameters in stConfig are set correctly.




### 16.8.5 Assign the example program to Cyclic #1

In the Physical View Right-click X20CP1584 > Software and assign the example program to Cyclic #1



## 16.9 Automation Studio Logical View: Rotary Motor Example

This chapter shows the integration of a rotary motor. In this example a RS01 (rotary part of a LinMot PR01 linear rotary motor).

To add a second axis, repeat chapter 16.6 and 16.7 and use another name, e.g. *gAxis02*.

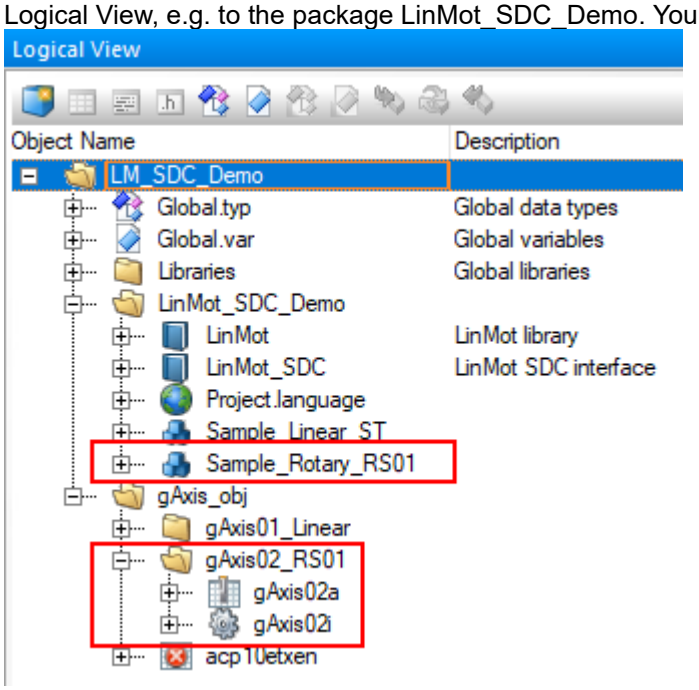
The mapping table with more axes may look as follows:

Global.typ [Data Type Declaration] | **Acp10map1.ncm [Mapping Editor]*** ×

| NC Object Name | PLC Address | Nc Object Type | Channel | Simulation | NC INIT Parameter | ACOPOS Parameter | Additional Data |
|---|---|---|---|---|---|---|---|
| gAxis01 | SDC_IF1.ST1 | ncAXIS | 1 | Off | gAxis01i | gAxis01a | SDC_AX_HwPv="gAxis01_HW" |
| gAxis02 | SDC_IF1.ST2 | ncAXIS | 1 | Off | gAxis02i | gAxis02a | SDC_AX_HwPv="gAxis02_HW" |

### 16.9.1 Add the example program to the project

Copy the sample program *Sample_Rotary_RS01* from the demo project (see chapter 16.1) and paste it to the Logical View, e.g. to the package LinMot_SDC_Demo. You may also want to copy the axes objects:



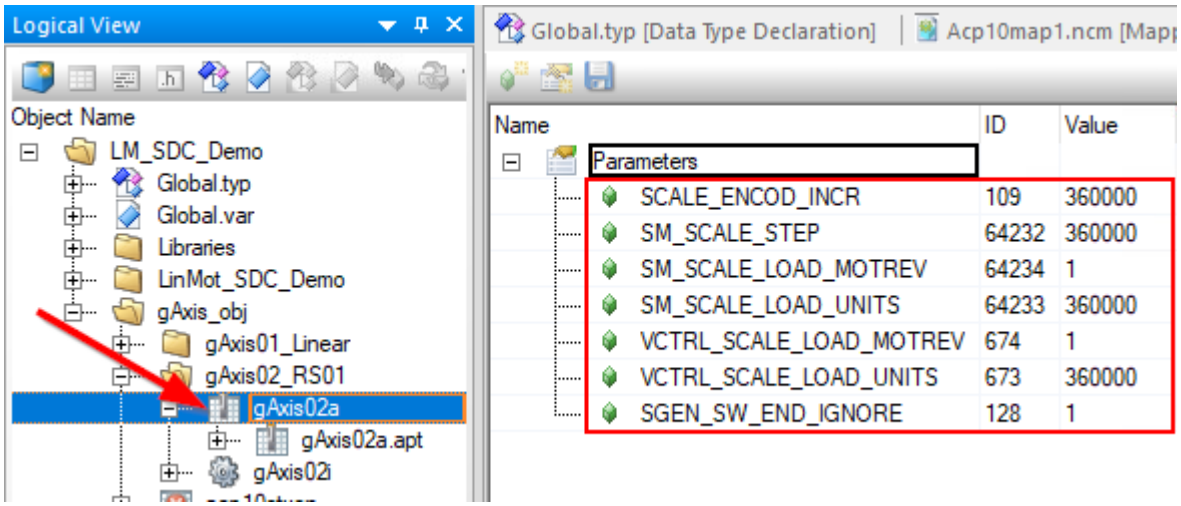
### 16.9.2 ACOPOS Parameter Table

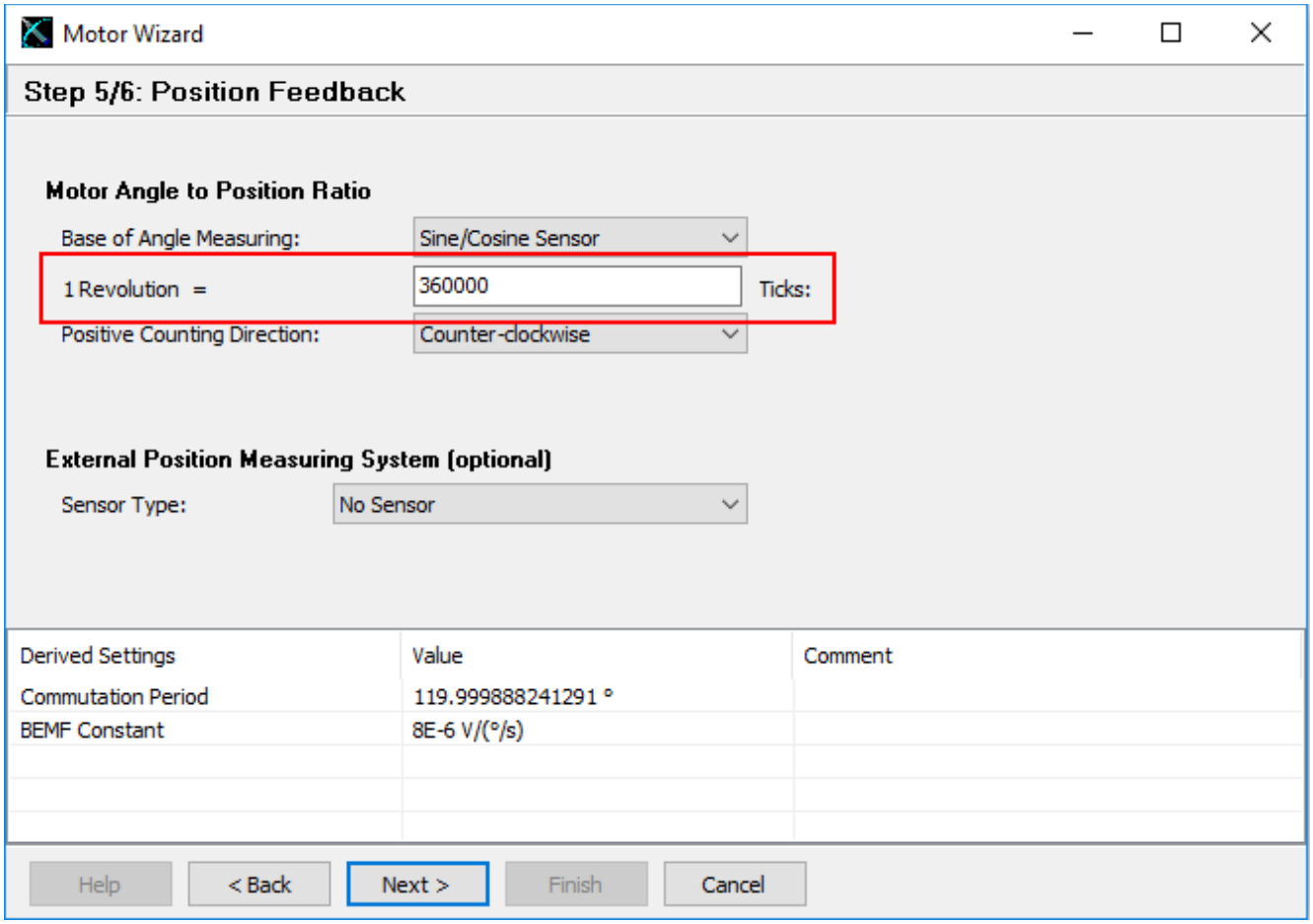
For a rotary motor the parameters must be set in relation to the settings on the LinMot drive.

The following examples will result in a resolution of **0.001 degree** per Unit.

The parameters ***SCALE_ENCOD_INCR*** and ***SM_SCALE_STEP*** must be set to the number of ticks shown in the LinMot-Talk motor wizard.

**Example 1: RS01 motor**

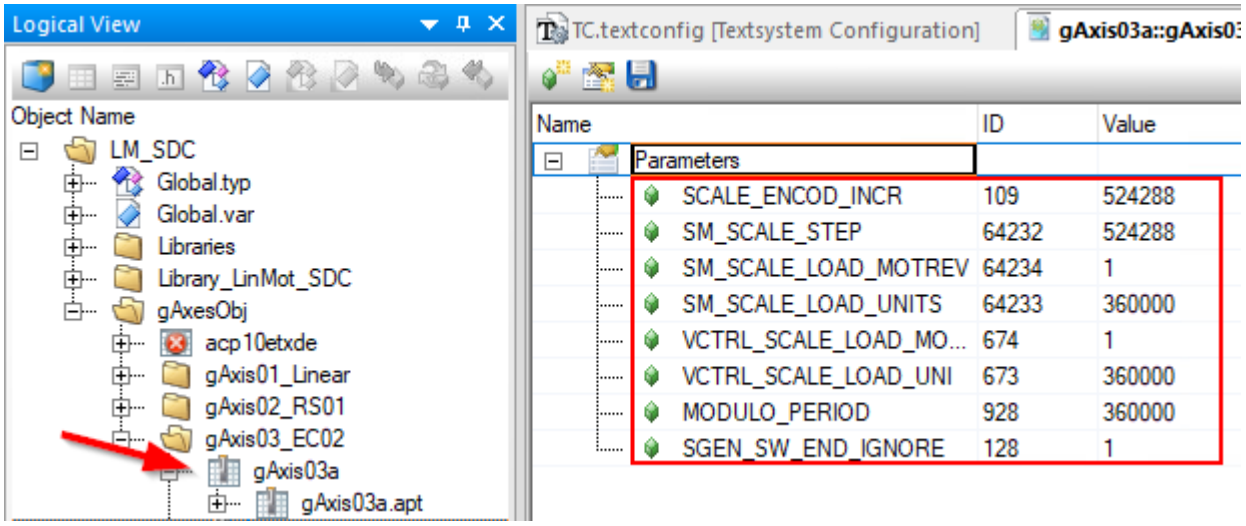


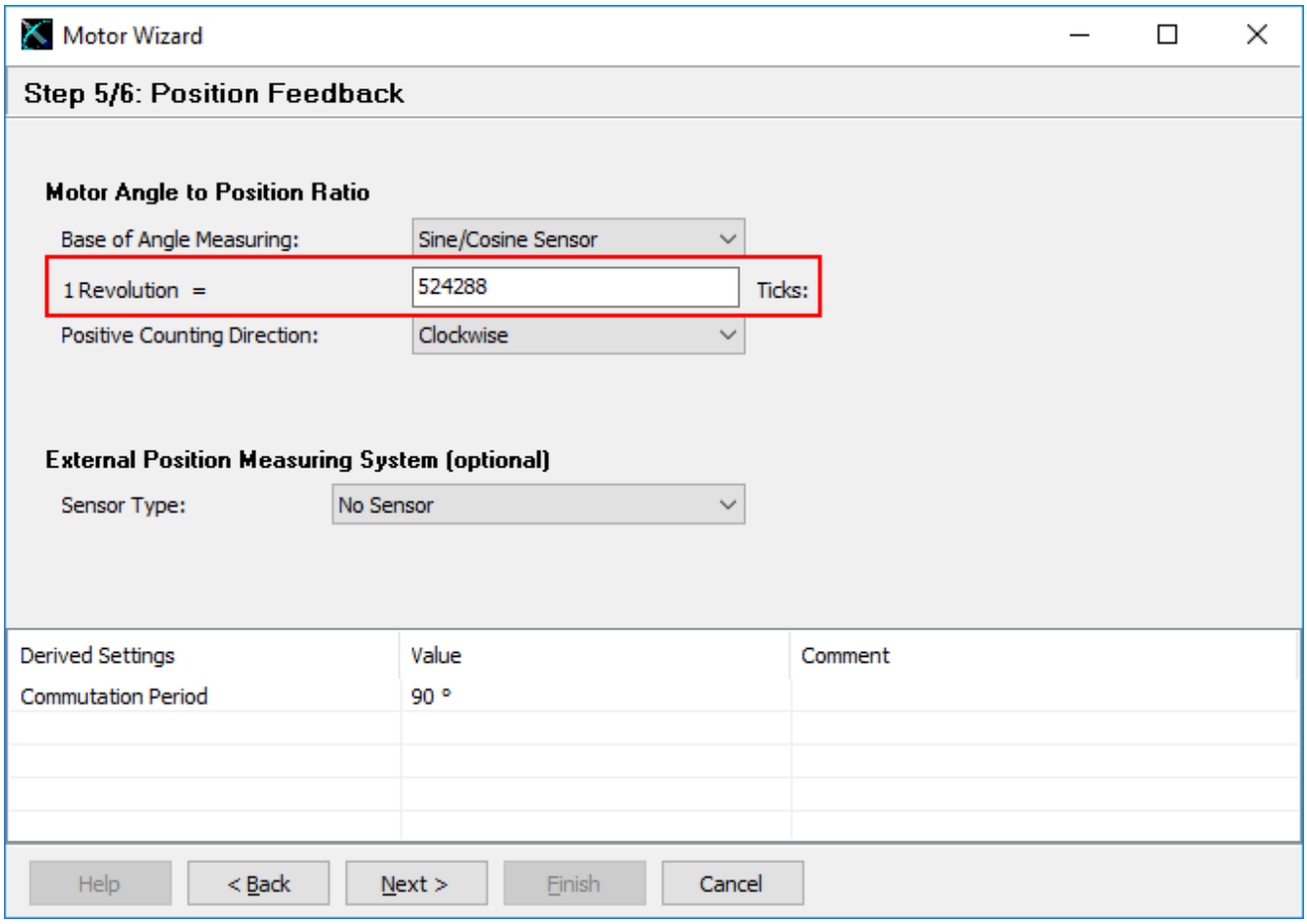



**Example 2: EC02 motor**

This motor has 524288 ticks per motor revolution. For this reason, the parameters ***SCALE_ENCOD_INCR*** and ***SM_SCALE_STEP*** must be set to 524288.

The unit is the same as before: 0.001 degree



### 16.9.3 Init Parameter Table

Set all digital input active levels to *ncACTIV_HI*.
Set *Units* in *Encoder_if* to 360000 (> 0.001 degree per Unit)
Set *t_predict* and *t_total* to 5 times Powerlink cycle time each > 800us * 5 = 0.004 s

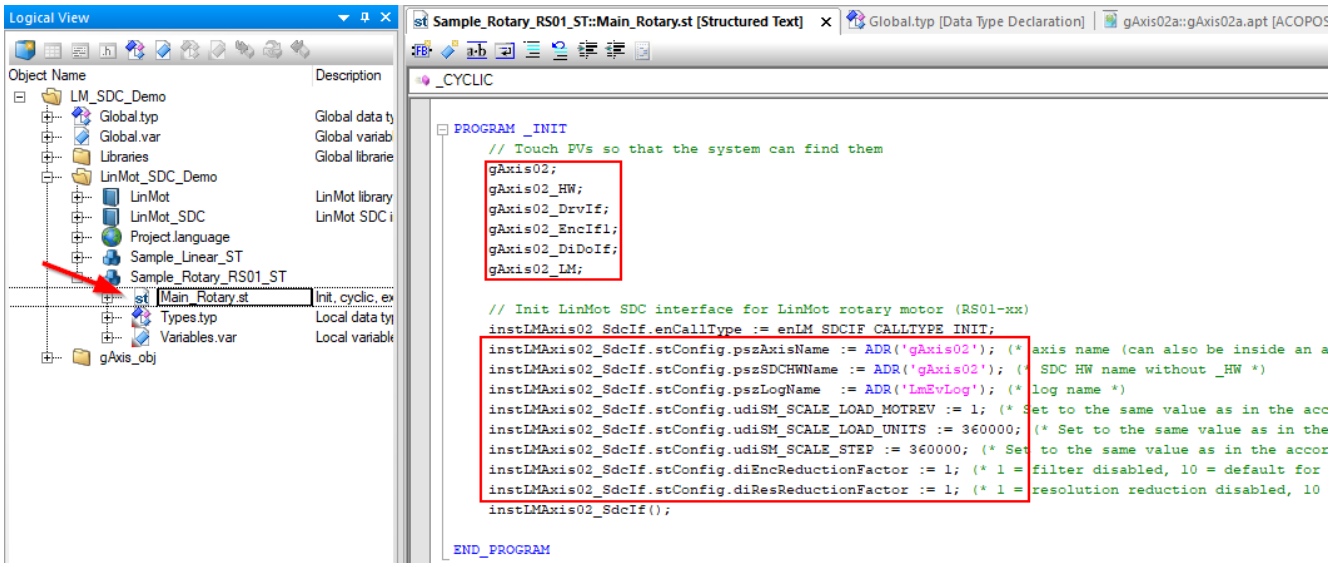
| Name | Value | Unit | Description |
|---|---|---|---|
| ACP10AXIS_typ | | | |
| dig_in | | | Digital Inputs |
| level | | | Active Input Level |
| reference | ncACTIV_HI | | Reference switch |
| pos_hw_end | ncACTIV_HI | | Positive HW end switch |
| neg_hw_end | ncACTIV_HI | | Negative HW end switch |
| trigger1 | ncACTIV_HI | | Trigger1 |
| trigger2 | ncACTIV_HI | | Trigger2 |
| encoder_if | | | Encoder Interface |
| parameter | | | Parameters |
| count_dir | ncSTANDARD | | Count direction |
| scaling | | | Scaling |
| load | | | Load |
| units | 360000 | Units | Units at the load |
| rev_motor | 1 | | Motor revolutions |
| limit | | | Limit value |
| controller | | | Controller |
| mode | ncPOSITION | | Mode |
| position | | | Position Controller |
| kv | 0.0 | 1/s | Proportional amplification |
| tn | 0.0 | s | Integral action time |
| t_predict | 0.004 | s | Prediction time |
| t_total | 0.004 | s | Total time |
| p_max | 0.0 | Units/s | Maximum proportional action |
| i_max | 0.0 | Units/s | Maximum integral action |

As the limits depend on the application and the motor type used it is recommended to start with some moderate values.

| Name | Value | Unit | Description |
|---|---|---|---|
| limit | | | Limit value |
| parameter | | | Parameters |
| v_pos | 3600000.0 | Units/s | Speed in positive direction |
| v_neg | 3600000.0 | Units/s | Speed in positive direction |
| a1_pos | 7200000.0 | Units/s² | Acceleration in positive direction |
| a2_pos | 7200000.0 | Units/s² | Deceleration in positive direction |
| a1_neg | 7200000.0 | Units/s² | Acceleration in negative direction |
| a2_neg | 7200000.0 | Units/s² | Deceleration in negative direction |
| t_jolt | 0.0 | s | Jolt time |
| t_in_pos | 0.0 | s | Settling time before message "In Position" |
| pos_sw_end | 2000000000 | Units | Positive SW end |
| neg_sw_end | -2000000000 | Units | Negative SW end |
| ds_warning | 10000.0 | Units | Lag error limit for display of a warning |
| ds_stop | 20000.0 | Units | Lag error limit for stop of a movement |
| a_stop | 1.0E+09 | Units/s² | Acceleration limit for stop of a movement |
| dv_stop | 0.0 | 1/s | Speed error limit for stop of a movement |
| dv_stop_mode | ncOFF | | Mode for speed error monitoring |

### 16.9.4 Check Init Parameters of the SDC function block

Open Main_Linear.st and check if the axis, SDCHW and log names as well as the other parameters in stConfig are set correctly.

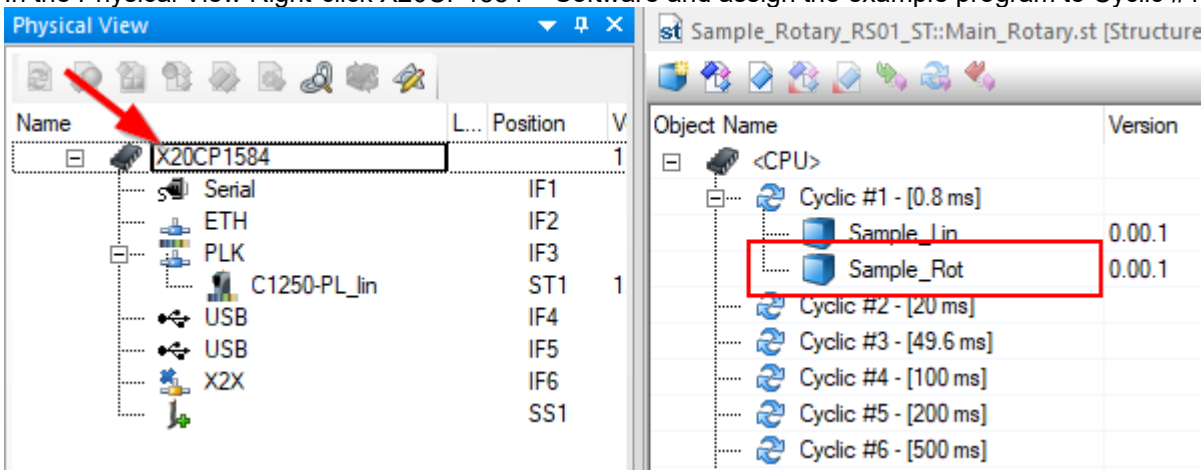



**Attention:**
The parameter ***SM_SCALE_STEP*** must be set as shown in chapter 16.9.2.

For a LinMot RS01 rotary motor the parameter ***SM_SCALE_STEP*** is set to 360000.
For a LinMot EC02 rotary motor the parameter ***SM_SCALE_STEP*** is set to 524288.
For any other motor please check the Position Feedback in the LinMot-Talk Motor Wizard.

### 16.9.5 Assign the example program to Cyclic #1

In the Physical View Right-click X20CP1584 > Software and assign the example program to Cyclic #1



## 16.10 Homing

### 16.10.1 Linear Motor

A possible setup for homing mode ncBlock_DS:

| | | | |
|---|---|---|---|
| move | | | Movement |
| stop | | | Stop Movement |
| homing | | | Homing procedure |
| parameter | | | Parameters |
| s | -1000 | Units | Reference position |
| v_switch | 5000.0 | Units/s | Speed for searching the reference switch |
| v_trigger | 5000.0 | Units/s | Trigger Speed |
| a | 50000.0 | Units/s² | Acceleration |
| mode | ncBLOCK_DS | | Mode |
| edge_sw | ncPOSITIVE | | Edge of reference switch |
| start_dir | ncNEGATIVE | | Start direction |
| trigg_dir | ncPOSITIVE | | Trigger direction |
| ref_pulse | ncOFF | | Reference pulse |
| fix_dir | ncOFF | | Fixed direction |
| tr_s_block | 1.0 | Rev. | Distance for blocking the activation of 'triggering reference pulse' |
| torque_lim | 0.0 | Nm | Torque limit for homing on block |
| ds_block | 1000.0 | Units | Lag error for block detection |
| ds_stop | 2000.0 | Units | Lag error for stop of a movement |

### 16.10.2 Rotary Motor

LinMot rotary motors (RS01 & EC02) have a single turn absolute encoder.
Therefore, as homing mode *ncHOME_OFFSET* can be used.

| | | | |
|---|---|---|---|
| move | | | Movement |
| mode | ncOFF | | Mode |
| detail | ncOFF | | Detail |
| stop | | | Stop Movement |
| homing | | | Homing procedure |
| init | ncTRUE | | Homing procedure initialized |
| status | | | Status |
| parameter | | | Parameters |
| s | 0 | Units | Reference position |
| v_switch | 5000 | Units/s | Speed for searching the reference switch |
| v_trigger | 5000 | Units/s | Trigger Speed |
| a | 50000 | Units/s² | Acceleration |
| mode | ncHOME_OFFSET | | Mode |
| edge_sw | ncPOSITIVE | | Edge of reference switch |
| start_dir | ncNEGATIVE | | Start direction |
| trigg_dir | ncPOSITIVE | | Trigger direction |
| ref_pulse | ncOFF | | Reference pulse |
| fix_dir | ncOFF | | Fixed direction |
| tr_s_block | 0.1 | Rev. | Distance for blocking the activation of 'triggering reference pulse' |
| torque_lim | 0 | Nm | Torque limit for homing on block |
| ds_block | 1000 | Units | Lag error for block detection |
| ds_stop | 2000 | Units | Lag error for stop of a movement |

**Note:**
It is recommended to home axes with absolute encoders first before switching the motor on.

### 16.10.3 Reference Switch (mcHOME_ABS_SWITCH)

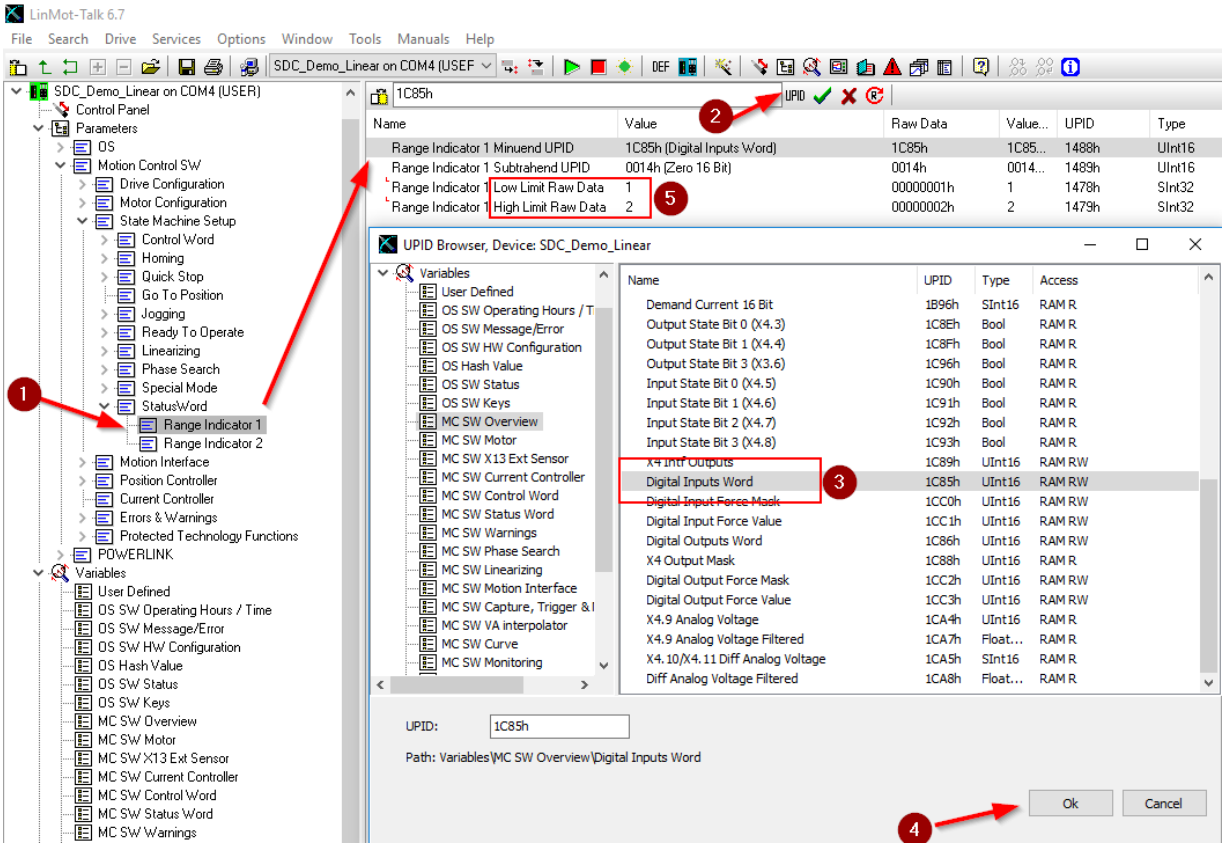
**Attention:**
Up to **LinMot drive firmware 6.7** this mode works only when on the drive only the digital input X4.5 is used. If any other input on X4 is set or used for other functions, then this method does not work with this firmware.

**Attention:**
From **LinMot drive firmware 6.8 Build 20190315** this method works too with boolean parameters. E.g. for X4.5 set the parameter *Range Indicator 1 Minuend UPID* to *1C90h* (Inputs State Bit 0 (X4.5)) instead of *1C85h*. See screenshot below.

It is possible to connect a hardware switch to the LinMot drives X4 inputs (e.g. X4.5). The state of the digital input is then passed through to *iReference* of the SDC DI/DO Interface.

Setup the reference switch in the LinMot drive:

- For LinMot drive firmware 6.7:




- For LinMot drive firmware 6.8 Build 20190315 and newer replace the value in *Range Indicator 1 Minuend UPID* to *1C90h* (Inputs State Bit 0 (X4.5)) instead of *1C85h*

As example we want to use the following homing mode
(see B&R Automation Studio Help > search for mcHOME_ABS_SWITCH):

```
edge_sw = ncNEGATIVE
trigg_dir = ncPOSITIVE
fix_dir = ncOFF
```


Possible setup of the homing mode:

| | | | |
|---|---|---|---|
| homing | | | Homing procedure |
| init | ncTRUE | | Homing procedure initialized |
| status | | | Status |
| parameter | | | Parameters |
| s | 0 | Units | Reference position |
| v_switch | 1000 | Units/s | Speed for searching the reference switch |
| v_trigger | 5000 | Units/s | Trigger Speed |
| a | 50000 | Units/s² | Acceleration |
| mode | ncABS_SWITCH | | Mode |
| edge_sw | ncNEGATIVE | | Edge of reference switch |
| start_dir | ncNEGATIVE | | Start direction |
| trigg_dir | ncPOSITIVE | | Trigger direction |
| ref_pulse | ncOFF | | Reference pulse |
| fix_dir | ncOFF | | Fixed direction |
| tr_s_block | 1 | Rev. | Distance for blocking the activation of 'triggering reference pulse' |
| torque_lim | 0 | Nm | Torque limit for homing on block |
| ds_block | 1000 | Units | Lag error for block detection |
| ds_stop | 2000 | Units | Lag error for stop of a movement |

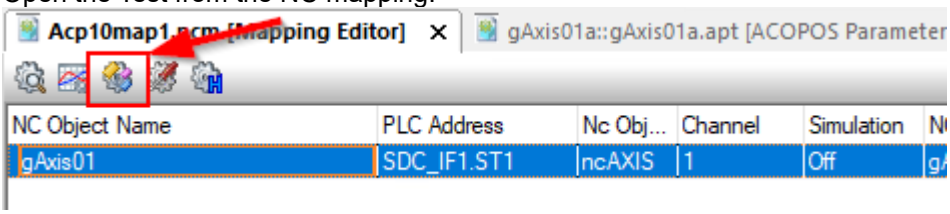
## 16.11 Test

After the LinMot drive is integrated as shown in the chapters before, the project can be compiled and downloaded to the PLC.

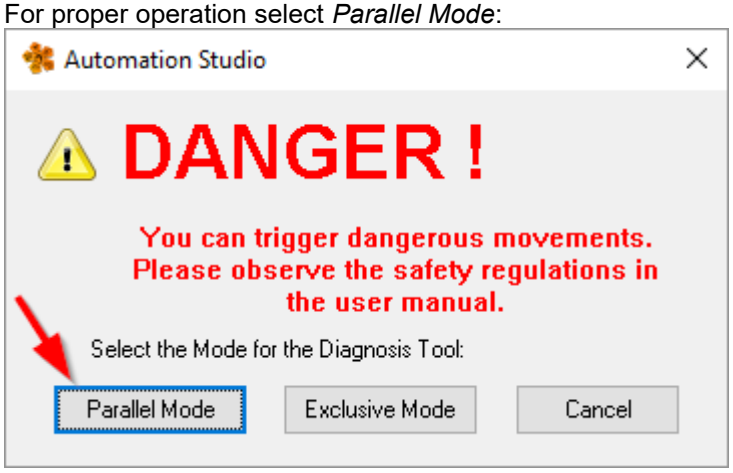
**Attention:**
It is recommended to rebuild the project and to **restart both the PLC and drives** to have a proper start-up. Ignoring this may prevent being able to power on the axis.

For testing purposes, the Test tool in Automation Studio can be used.

Open the Test from the NC mapping:

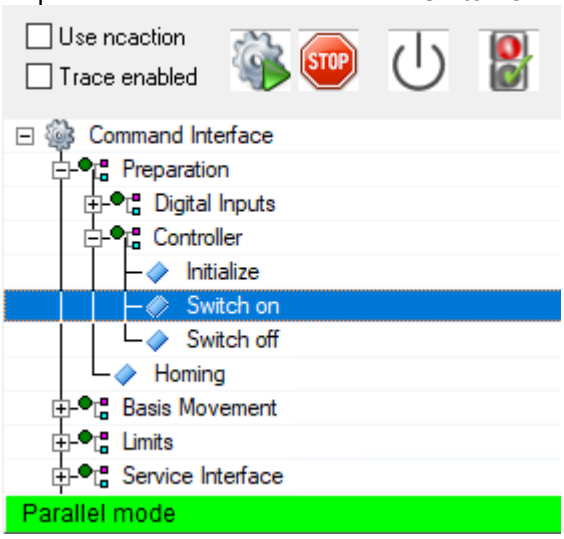



For proper operation select *Parallel Mode*:




### 16.11.1 Switch On

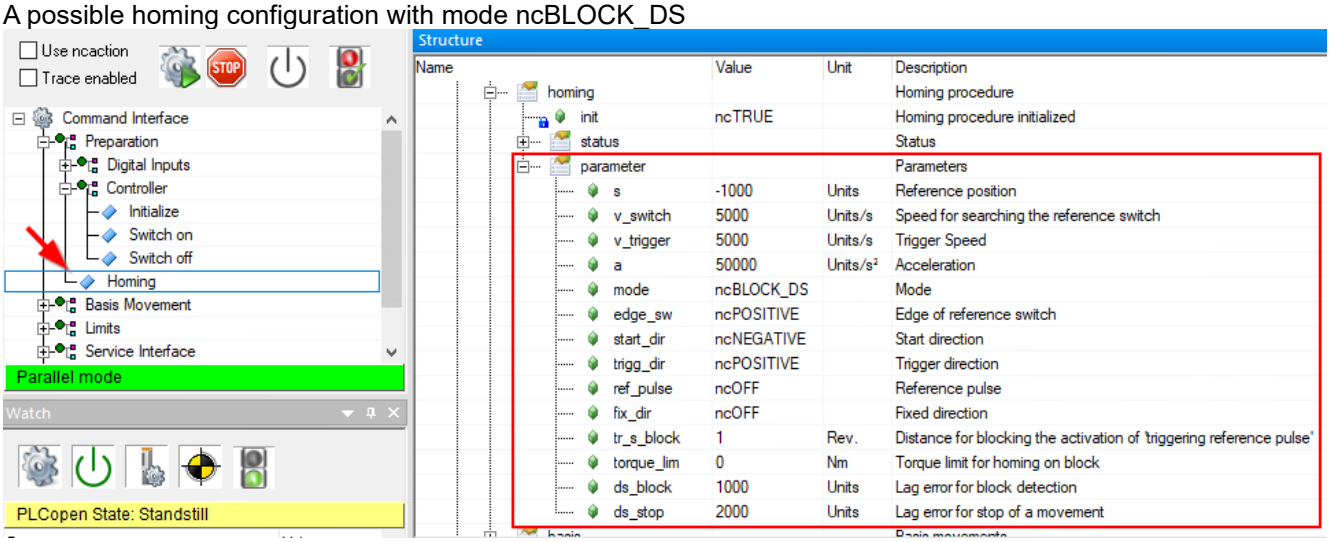
To power on the motor execute the *Switch On* command:



### 16.11.2 Homing

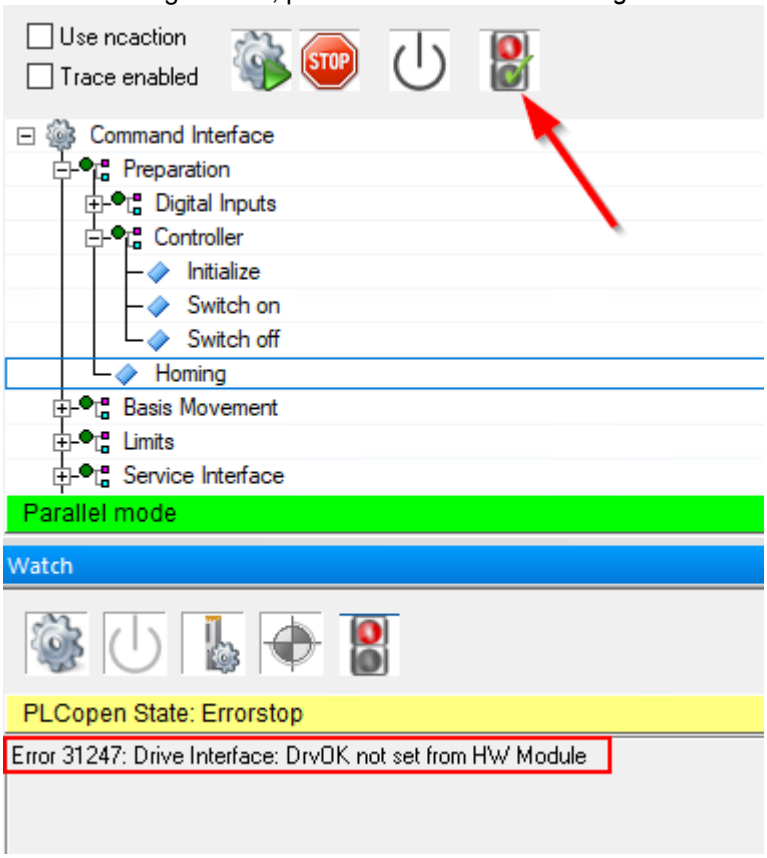
To home the axis (in this example a linear motor to hard stop) use the Homing command.

A possible homing configuration with mode ncBLOCK_DS




### 16.11.3 Error Acknowledge

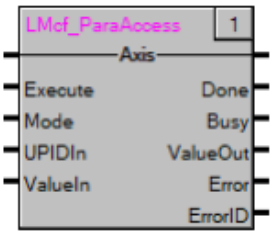
To acknowledge errors, press the button *Acknowledge Error* as long as an error is shown in the Watch:



## 16.12 Parameter Access

### 16.12.1 Parameter access by LMcf_ParaAccess function block

Every parameter and variable in a LinMot system has its own UPID (Unique Parameter ID).
Read and write access is possible using the LMcf_ParaAccess function block from the library LinMot.




| Supported Modes | | |
|---|---|---|
| Value | Used Inputs | Description |
| 0 | UPIDIn | Read ROM Value of Parameter by UPID |
| 1 | UPIDIn | Read RAM Value of Parameter by UPID |
| 2 | UPIDIn, ValueIn | Write ROM Value of Parameter by UPID |
| 3 | UPIDIn, ValueIn | Write RAM Value of Parameter by UPID |
| 4 | UPIDIn, ValueIn | Write RAM and ROM Value of Parameter by U |
| 5 | UPIDIn | Get minimal Value of Parameter by UPID |
| 6 | UPIDIn | Get maximal Value of Parameter by UPID |
| 7 | UPIDIn | Get default Value of Parameter by UPID |

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the life-time of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

**Note:**
Further documentation about this function block can be found in the LinMot library manual.
See chapter 16.1.

### 16.12.2 Parameter access by SDO commands

Alternatively, also access by SDO is possible. For SDO access the B&R library *AsEPL* is required.
Function blocks: *EplSDORead() & EplSDOWrite()*

The Index specifies the LinMot parameter (UPID) to address. It is UPID + 16#2000.
The Sub-Index specifies the command which is performed

| Index | Sub-Index | Description | Access Type |
|---|---|---|---|
| 2000h -5FFFh | | LinMot Parameter Commands | |
| UPID Commands | | | |
| | 01h | RAM Value of UPID | Read & Write |
| | 02h | ROM Value of UPID | Read & Write |
| | 03h | Min Value of UPID | Read Only |
| | 04h | Max Value of UPID | Read Only |
| | 05h | Default Value of UPID | Read Only |
| | 06h | RAM / ROM Write of UPID | Write Only |
| System Commands | | | |
| | 07h | Set ROM to default (OS) | Write Only |
| | 08h | Set ROM to default (MC) | Write Only |
| | 09h | Set ROM to default (Interface) | Write Only |
| | 0Ah | Set ROM to default (Application) | Write Only |
| | 0Bh | Reset drive | Write Only |
| | 35h | Stop MC and APPL Software | Write Only |
| | 36h | Start MC and APPL Software | Write Only |

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the life-time of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

Possible variable declaration:

```
instLMAxis01_SdoWrite   EplSDOWrite
instLMAxis01_SdoRead    EplSDORead
udiDataWrite            UDINT
udiDataRead             UDINT
```

Possible function block calls where IF3 is the Powerlink interface:

```
// SDO read
instLMAxis01_SdoRead(pDevice:=ADR('IF3'), pData:=ADR(udiDataRead));

// SDO write
instLMAxis01_SdoWrite(pDevice:=ADR('IF3'), pData:=ADR(udiDataWrite));
```

**Example read parameter:**

Read the RAM value of the parameter Maximal Current:

**Name:** Maximal Current
**UPID:** 13A6h
**Type:** SInt32 = DINT = 4 Bytes
**Scale:** 0.001 A

**pDevice** := pointer to POWERLINK interface, e.g. ADR('IF3')
**node** := **1** (Node ID of the LinMot drive)
**index** := **13222 = 16#33a6** = 16#2000 + 16#13a6
**subindex** := **1** (access RAM value)
**pData** := pointer to data buffer, e.g. ADR(udiDataRead)
**datalen** := **4** (always 4 Bytes! Even if the parameter to read is less)

| | | | | |
|---|---|---|---|---|
| instLMAxis01_SdoRead | EplSDORead | local | | |
| enable | BOOL | | | TRUE |
| pDevice | UDINT | | | 69733729 |
| node | USINT | | | 1 |
| index | UINT | | | 13222 |
| subindex | USINT | | | 1 |
| pData | UDINT | | | 69735842 |
| datalen | UDINT | | | 4 |
| status | UINT | | | 65535 |
| errorinfo | UDINT | | | 16#0000_0000 |
| readlen | UDINT | | | 4 |
| intern | EPL_I_TYPE | | | |
| udiDataRead | UDINT | local | | 15000 |

**Example write parameter:**

Write 1A to the RAM value of the parameter Maximal Current:

**Name:** Maximal Current
**UPID:** 13A6h
**Type:** SInt32 = DINT = 4 Bytes
**Scale:** 0.001 A > value to be sent = 1'000

**pDevice** := pointer to POWERLINK interface, e.g. ADR('IF3')
**node** := **1** (Node ID of the LinMot drive)
**index** := **13222 = 16#33a6** = 16#2000 + 16#13a6
**subindex** := **1** (access RAM value)
**pData** := pointer to data buffer, e.g. ADR(udiDataWrite)
**datalen** := **4** (always 4 Bytes! Even if the parameter to read is less)

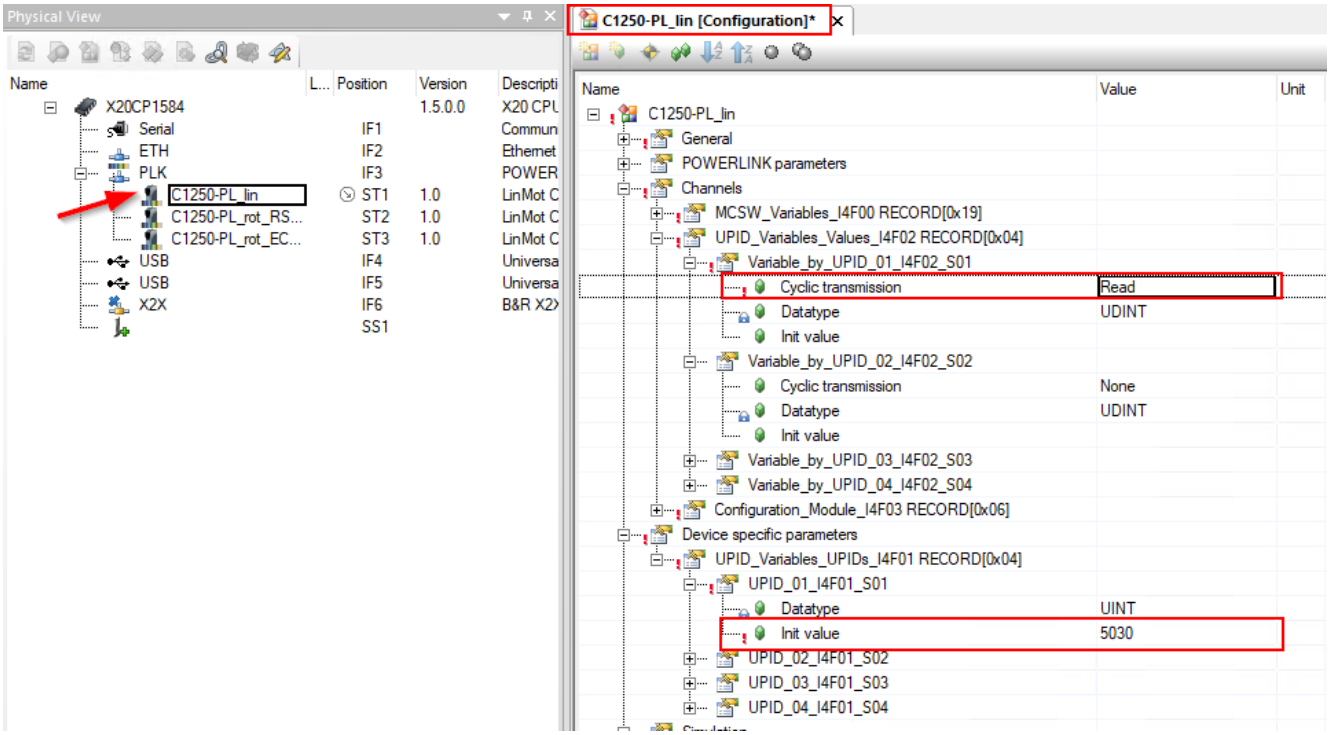
| | | | | |
|---|---|---|---|---|
| instLMAxis01_SdoWrite | EplSDOWrite | local | | |
| enable | BOOL | | | TRUE |
| pDevice | UDINT | | | 69733783 |
| node | USINT | | | 1 |
| index | UINT | | | 13222 |
| subindex | USINT | | | 1 |
| pData | UDINT | | | 69735838 |
| datalen | UDINT | | | 4 |
| status | UINT | | | 65535 |
| errorinfo | UDINT | | | 0 |
| intern | EPL_I_TYPE | | | |
| udiDataWrite | UDINT | local | | 1000 |

## 16.13 Add Additional Parameters or Variables to the Process Data

It is possible to add up to 4 parameters or variables to the cyclic PDO of the drive. This allows fast read/write access.

Right-click on the drive and select *Configuration*.

- Set the ***Cyclic transmission*** to either *Read* (for read access from the drive) or *Write* (for write access to the drive) in the “Channels”.
- As ***Init value*** in the “Device specific parameters” set the UPID number (e.g. Maximal Current has UPID 16#13A6 > 5030 decimal) of the parameter you want to access.




Afterwards you can assign a PV in the I/O configuration of the device.

## 16.14 Errors (LM_SdcIf outputs)

### 16.14.1 bError

This output indicates that an error occurred either in the *LM_SdcIf* function block itself, in the used libraries (LinMot, Sys_Lib or ArEventLog) or in the LinMot drive. The source is shown by the *diStatusID* output.

### 16.14.2 diStatusID

diStatusID shows the actual status of the LM_SdcIf function block. The following values are valid.

| Parameter | Value | Description |
|---|---|---|
| enLM_SDCIF_ERR_OK | 0 | No Event ID: No error |
| enLM_SDCIF_INF_FUB_ENABLE_FALSE | 1610612736 | Function block not enabled |
| enLM_SDCIF_WRN_LIB_LINMOT | -1610612736 | Library LinMot: Warning |
| enLM_SDCIF_ERR_LIB_CONFIG_AXNAME | -536870912 | Object at parameter "pszAxisName" or "pszSDCHWName" not found or null. Or PV names (pszSDC...) too long (>34 characters) |
| enLM_SDCIF_ERR_LIB_SYS_LIB | -536870911 | Library Sys_Lib: Error |
| enLM_SDCIF_ERR_LIB_LINMOT | -536870910 | Library LinMot: Error |
| enLM_SDCIF_ERR_LIB_AREVENTLOG | -536870909 | Library ArEventLog: Error |

### 16.14.3 uiErrorCode

If the LinMot drive has an error this is indicated by:

- diStatusID value of -536870910 (enLM_SDCIF_ERR_LIB_LINMOT)
- uiErrorCode output greater than 0 (zero)

A list of possible uiErrorCode values can be found in either the MS_SW manual or within LinMot-Talk (Menu > Manuals > Errors > MC Layer).

It is possible to read the error message (English only) as STRING using the function block *LMcf_GetErrorText* from the library LinMot.

| inst_LMGetErrorText | LMcf_GetErrorTxt | |
|---|---|---|
| Execute | BOOL | TRUE |
| ErrorCode | UINT | 11 |
| Done | BOOL | TRUE |
| Busy | BOOL | FALSE |
| ErrorText | STRING[80] | 'Err: Pos Lag Always Too Big' |
| Error | BOOL | FALSE |
| ErrorID | UINT | 0 |
| Axis | UDINT | 65872452 |

**Note:**
Further documentation about this function block can be found in the LinMot library manual.
See chapter 16.1.

# 17 Drive Profile: OMRON Sysmac Studio, CoE DS402

## 17.1 Overview

This chapter shows how LinMot drives with *DS402 (CoE) / CiA402* interface (e.g., C1250-DS-XC-1S) can be integrated and setup in an OMRON N-Series (NJ, NX, NY) environment.

For this example, ***Sysmac Studio V1.50*** and ***LinMot-Talk V6.11*** were used.

**Download:**
Example projects can be downloaded from:
http://download.linmot.com/plc_lib/examples/Omron_DS/ (named: *prjLinMotDS402_fwV6.10_Nx1.smc2)*


Image Source: http://www.omron.com/

EtherCAT is the real-time Ethernet network originally developed by Beckhoff. The LinMot drives acts as slaves in this network and are implemented with either the standard ASIC ET1100 from Beckhoff or Hilscher NETX.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

**Note:**
In this chapter the following Modes of Operation of the DS402 / CiA402 profile are used:
6 = homing mode
8 = cyclic synchronous position (csp)
10= cyclic synchronous torque (cst)

The modes are switched by setting object 6060h (Mode of operation) to the according value.
The drive shows the active mode in object 6061h (Mode of operation display).

## 17.2 Configuration and Connection of the LinMot Drive

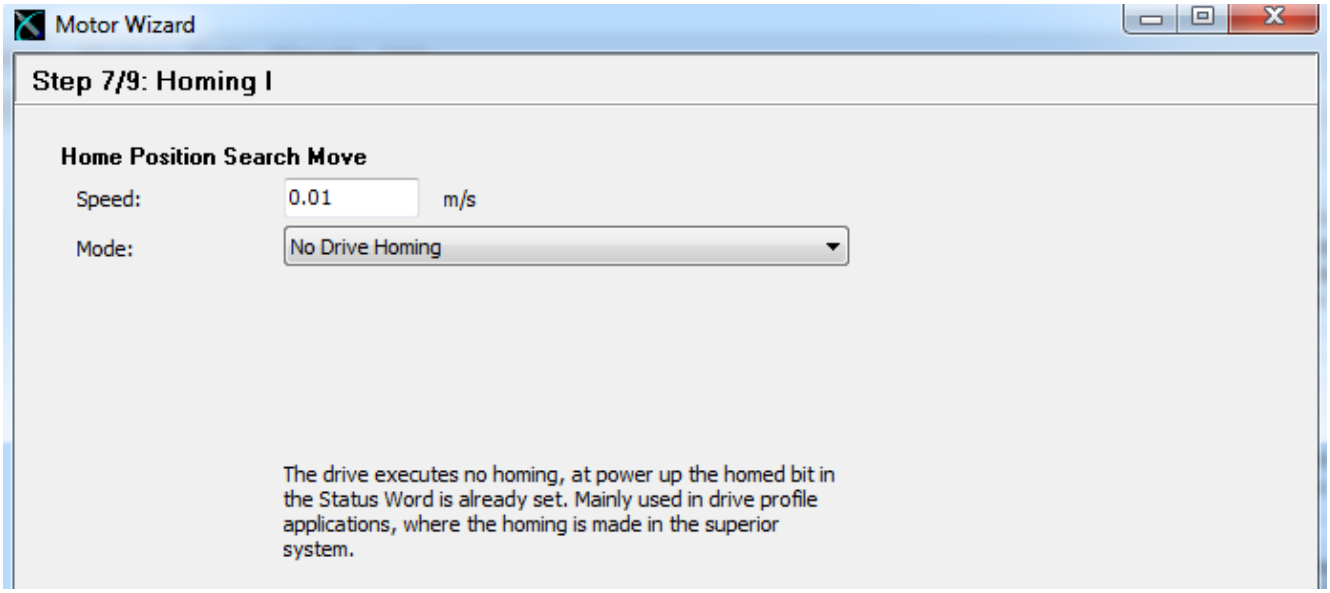
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 17.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Only select a homing mode in the Motor Wizard in case of a drive-based homing (e.g., *Mechanical Stop Negative Search*). Otherwise, *No Drive Homing* must be selected.




See Appendix I: Basic Position Control Loop Tuning

For ***Drive Based Homing*** please refer to chapter ***17.5***

### 17.2.2 XML File

Install the XML file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML\
- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML\ (-MI drives)

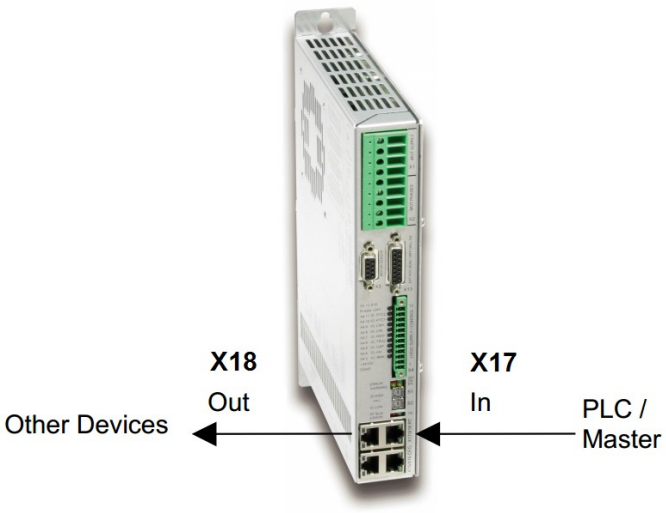
**XML file names:**

| | |
|---|---|
| NTIL_CiA402_Servos_Vx_xrx.xml | LinMot DS drives SG6-7 |
| NTIL_CiA402_SG5_Servos_Vx_xrx.xml | LinMot DS drives SG5 |
| NTIL_CiA402_Servos_MI_Vx_xrx.xml | LinMot MI drives SG6 |

### 17.2.3 EtherCAT Connection

The drive is connected to the EtherCAT network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-DS-UC drive. On all other LinMot drives supporting EtherCAT DS402 / CiA402 the ports are named the same (X17 & X18), but they may be placed differently on the drive housing.

| X17 - X18 | RealTime Ethernet 10/100 Mbit/s | |
|---|---|---|
|  | X17 RT ETH In | (OMRON: EtherCAT IN, from PLC) |
| | X18 RT ETH Out | (OMRON: EtherCAT OUT, to next device) |
| RJ-45 | | |



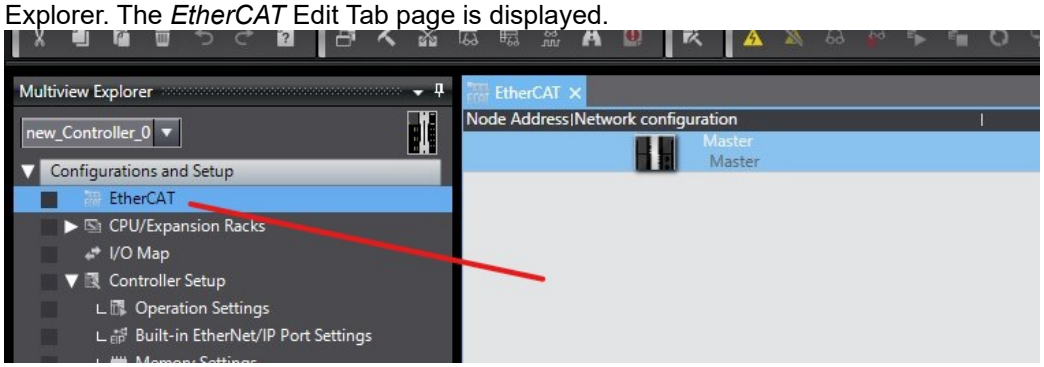
## 17.3 PLC Setup

### 17.3.1 Install EtherCAT Device Description File XML

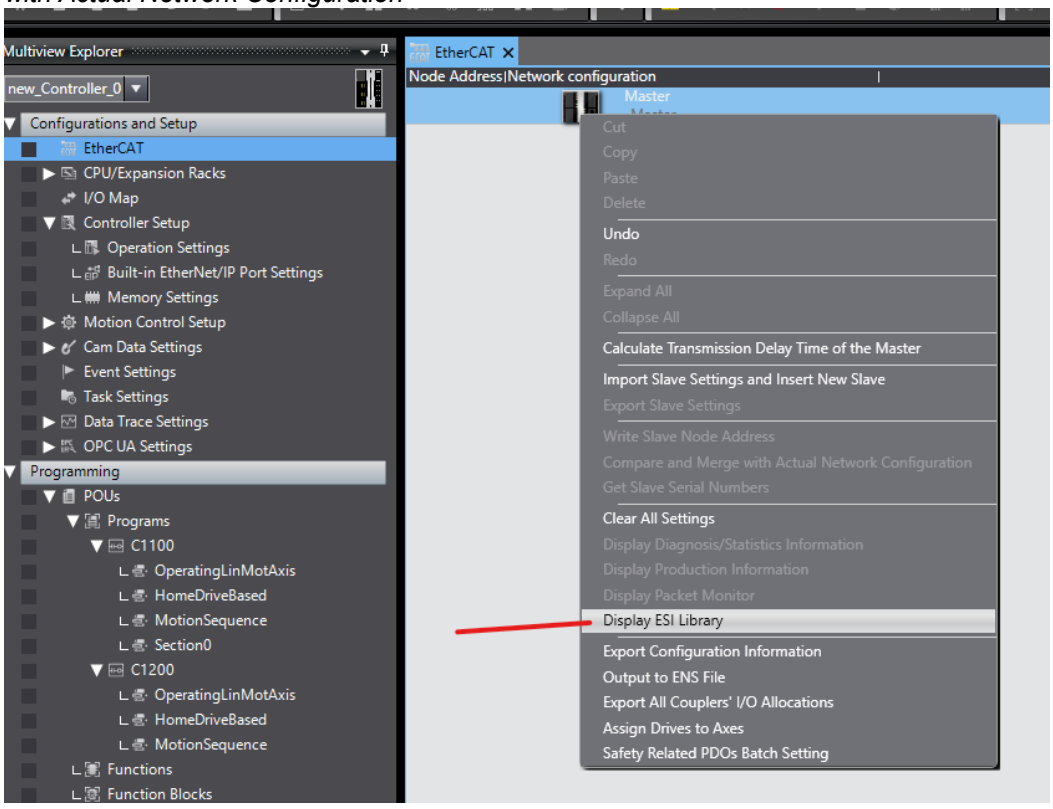
Find the most recent *ESI* file as described in chapter ***17.2.2 XML File***.

Install this file as follows:

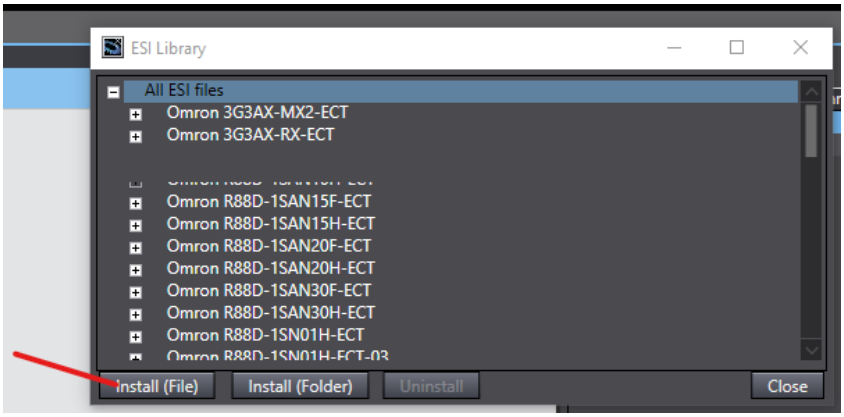
1. When *Offline* with the PLC, double-click *EtherCAT* under *Configuration and Setup* in the Multiview Explorer. The *EtherCAT* Edit Tab page is displayed.

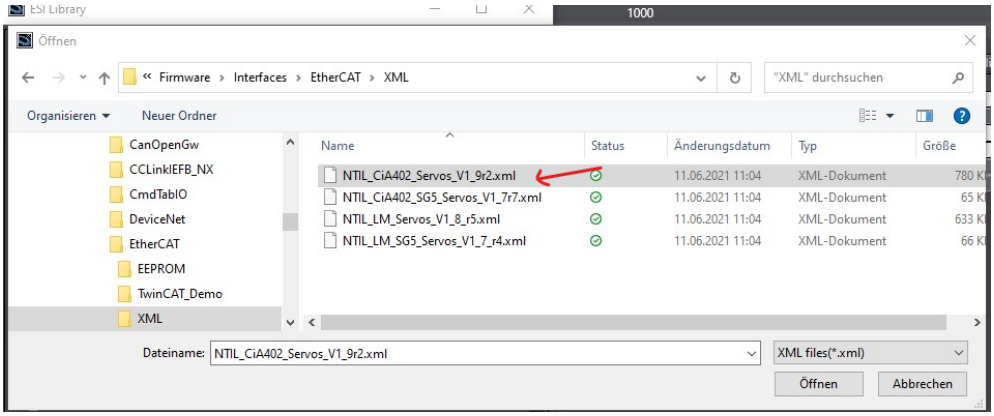



2. Scan EtherCAT network by right-click *Master* in the *Network configuration* and select *Compare and Merge with Actual Network Configuration*



3. Install XML file




### 17.3.2 Define EtherCAT node address on LinMot-Device

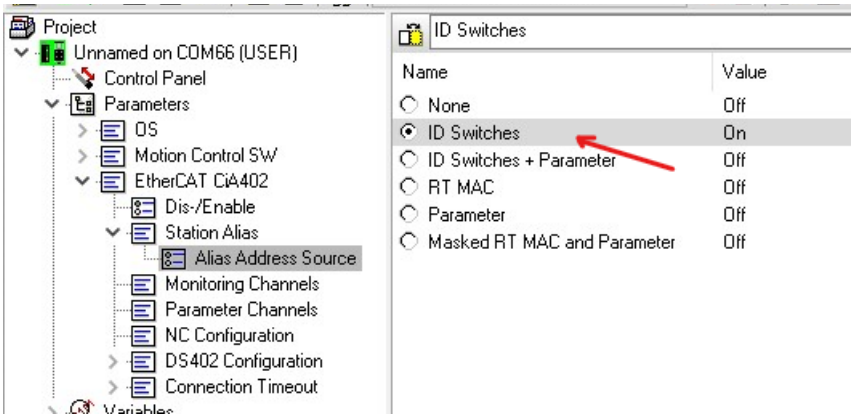
The accepted address range depends on the used PLC.
NJ, NX1: 1...192
NX7/NY: 1...512

OMRON does not support implicit addressing. But ***Node address "0"*** can be addressed by Sysmac Studio or with additional function blocks in the user program.

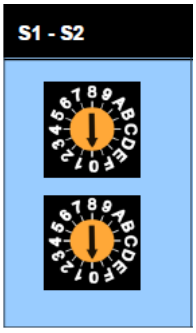
#### 17.3.2.1 Set node address by address selector on device

This is the most popular way, because it is user-friendly to maintenance people.

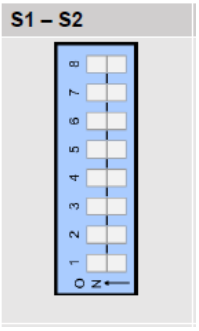
1. Select *ID Switches*



2. Depending on the LinMot-Drive the address selector is designed as Hex- or DIP-Switch.




Hex-Switch

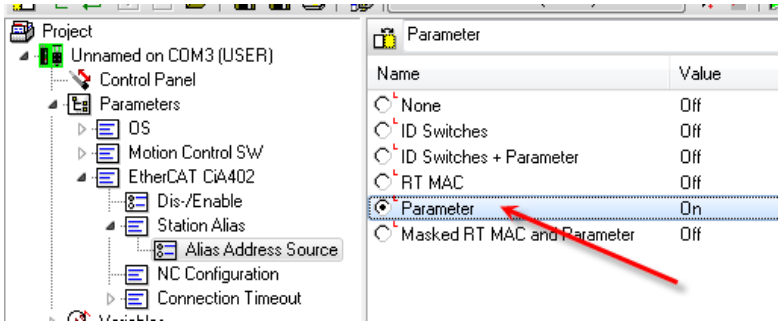



DIP-Switch

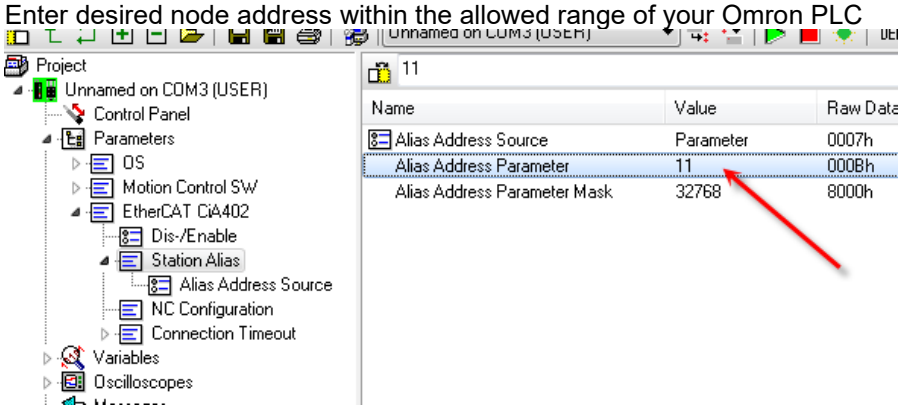
#### 17.3.2.2 Set node address by parameter on device

The node address can be defined by parameter.

3. Select *Parameter*




4. Enter desired node address within the allowed range of your Omron PLC

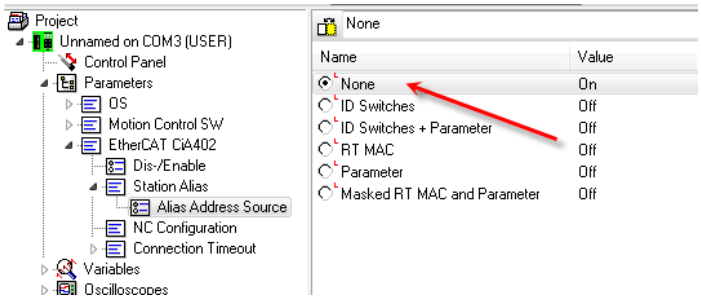


#### 17.3.2.3 Write node address from Sysmac Studio

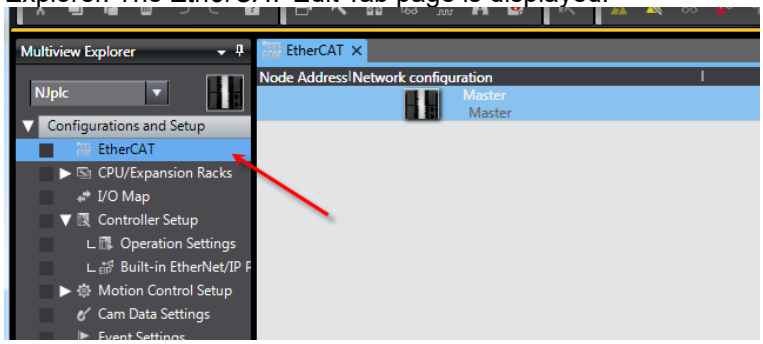
The node address can be written to the slave from *Sysmac Studio* (OMRON Software license required!).

This feature is currently not supported for **-MI** Drives

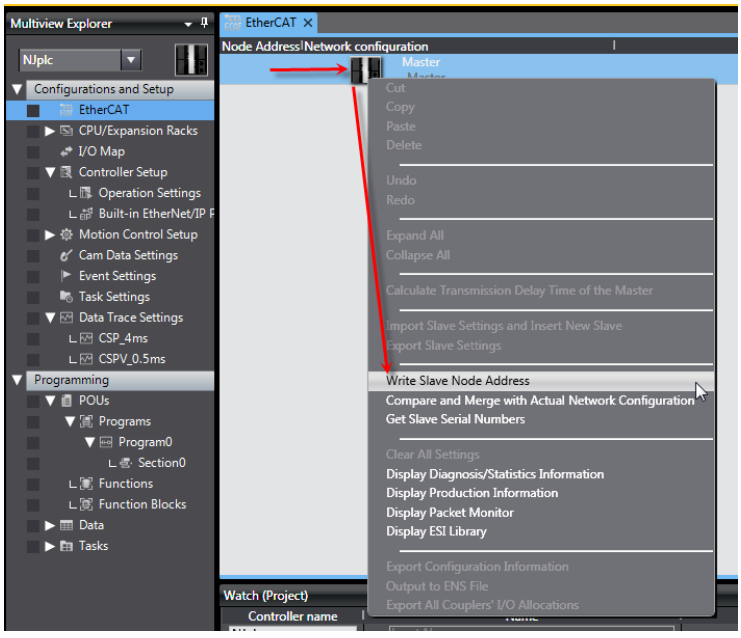
1. Select *None* and reboot drive



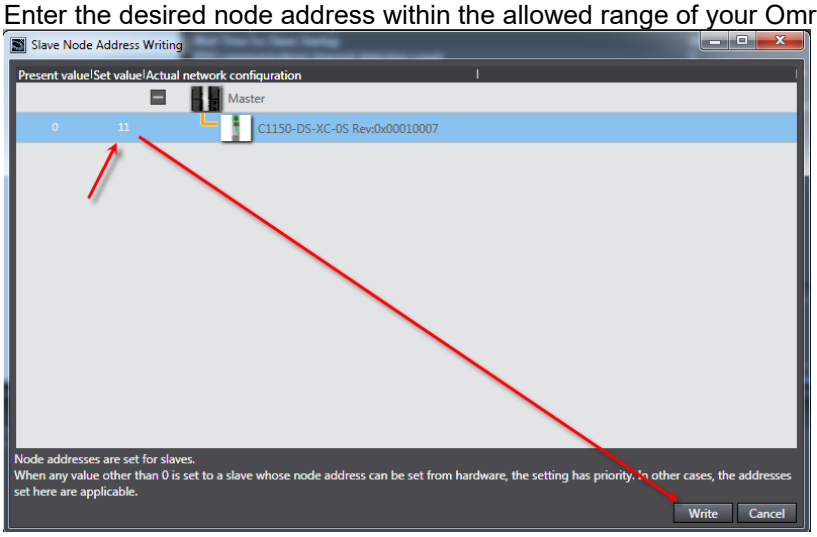

2. When *Online* with the PLC double-click *EtherCAT* under *Configuration and Setup* in the Multiview Explorer. The *EtherCAT* Edit Tab page is displayed.




3. Open the writing-tool by right-click *Master* in the *Network configuration* and select *Write Slave Node Address*



4. Enter the desired node address within the allowed range of your Omron PLC and reboot drive.

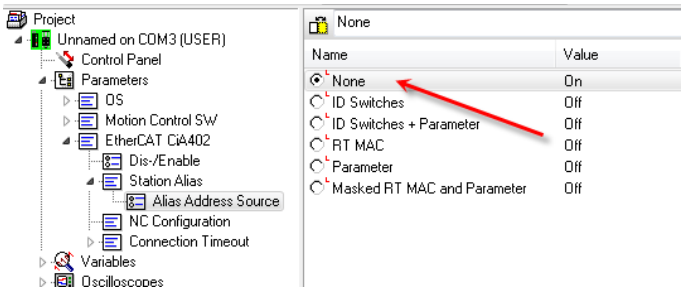



### 17.3.2.4 Set node address by function block

This feature is currently not supported for **-MI** Drives

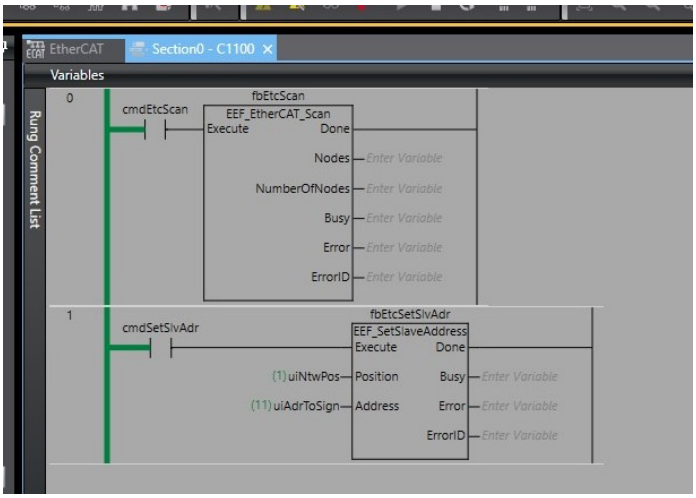
Omron supports a library with extended functions for EtherCAT: EthercatExtendedFunctions.slr

1. Select *None* and reboot drive




2. Use function block ***EEF_SlaveAddress*** to set a node address to the desired drive and reboot the drive.

   You may use function block *EEE_EtherCAT_Scan* to check the EtherCAT network before and after writing a slave address.

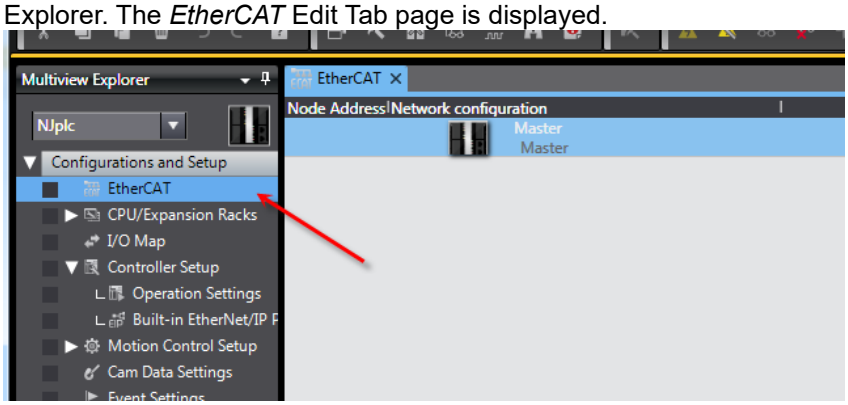



The OMRON library *EthercatExtendedFunctions.slr* can be requested by your local Omron support. It is an additional Library made by Omron ATC in Barcelona.

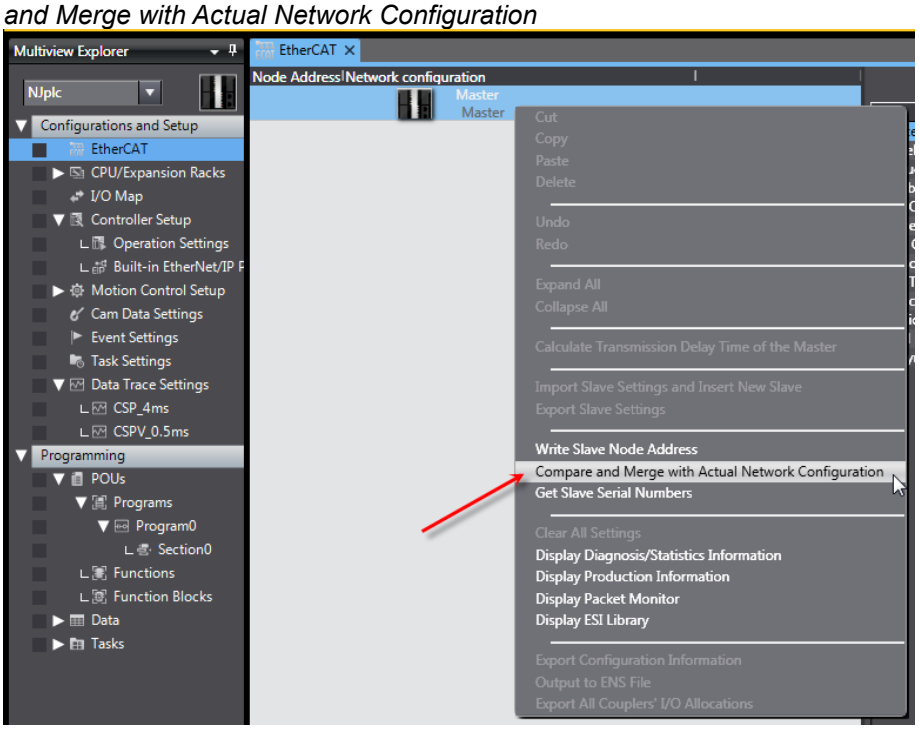
### 17.3.3 Scanning the EtherCAT network for devices

Follow these steps to add LinMot drives to your OMRON PLC.

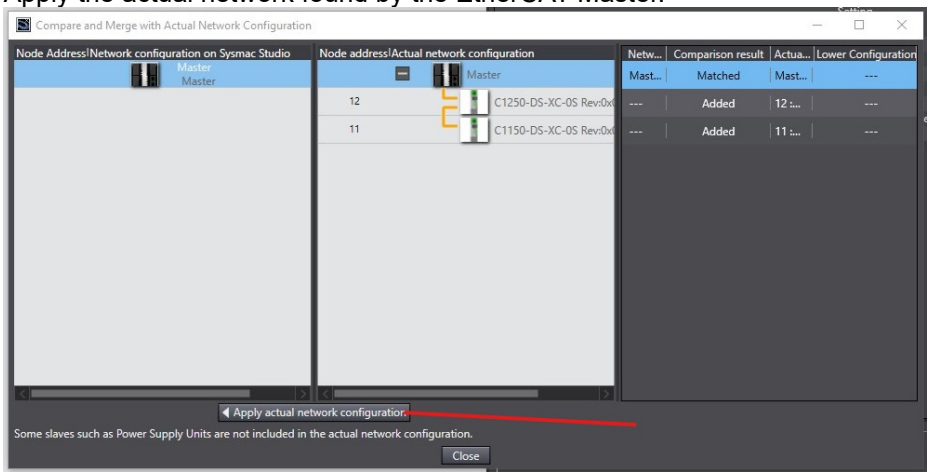
1. When *Online* with the PLC double-click *EtherCAT* under *Configuration and Setup* in the Multiview Explorer. The *EtherCAT* Edit Tab page is displayed.




2. Scan the EtherCAT network by right-click on the *Master* in the *Network configuration* and select *Compare and Merge with Actual Network Configuration*




3. Apply the actual network found by the EtherCAT Master.



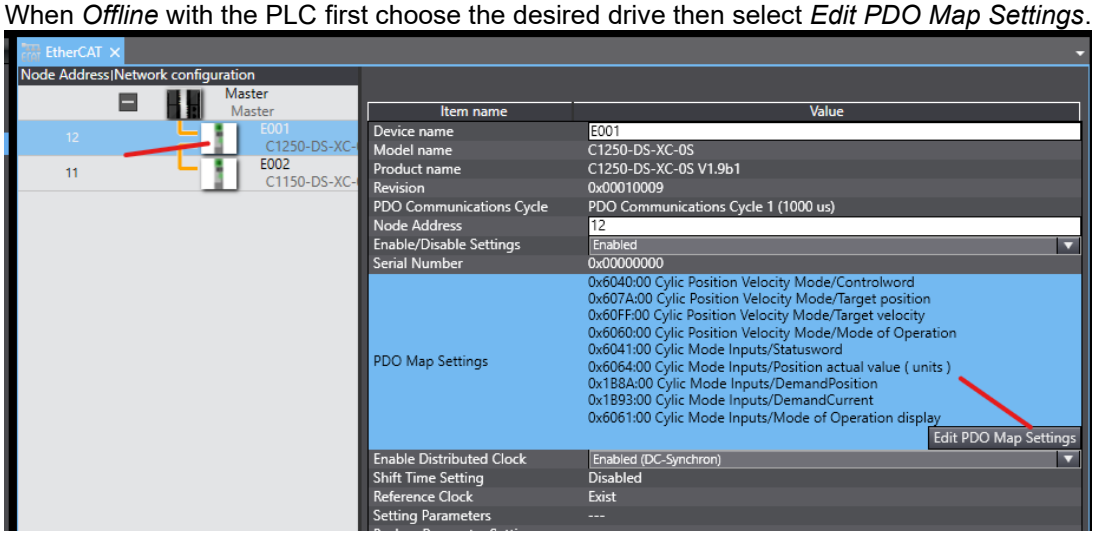
### 17.3.4 Edit PDO mapping

The following steps show how to setup the PDO mapping.

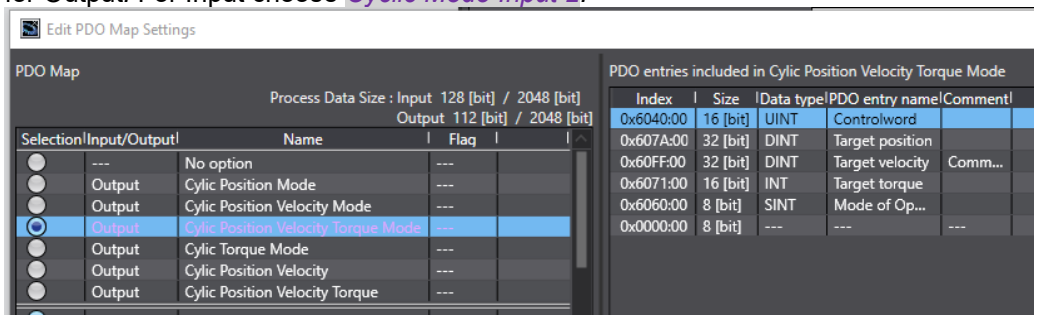
For standard (minimal) functionality, no change of the PDO mapping is required.

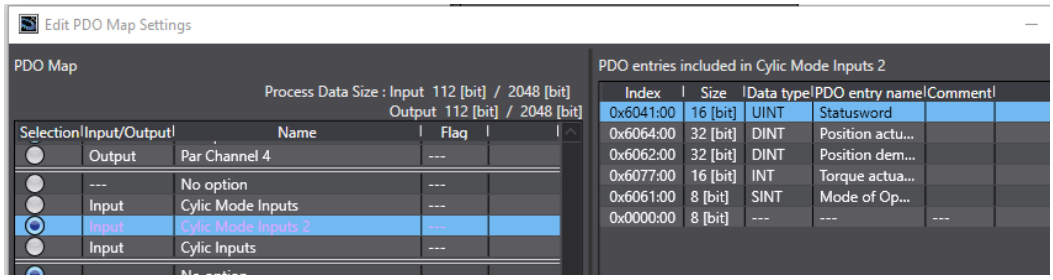
The less PDO's are mapped the less bandwidth is required - and therefor the performance of EtherCAT fieldbus will be higher.

1. Open PDO Map Dialog:
   When *Offline* with the PLC first choose the desired drive then select *Edit PDO Map Settings*.



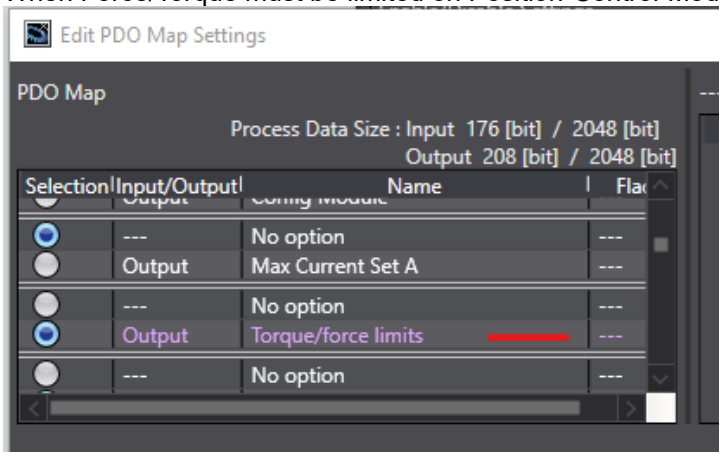

2. For full functionality (Position- AND Force/Torque-Control) choose *Cyclic Position Velocity Torque Mode* for Output. For Input choose *Cyclic Mode Input 2*.



3. For cyclic PDO transmission of any drive value/parameter choose *Par Channel X* for Output and *Mon Channel X* for Input.
   Please refer to chapter 17.8.1 for more information.

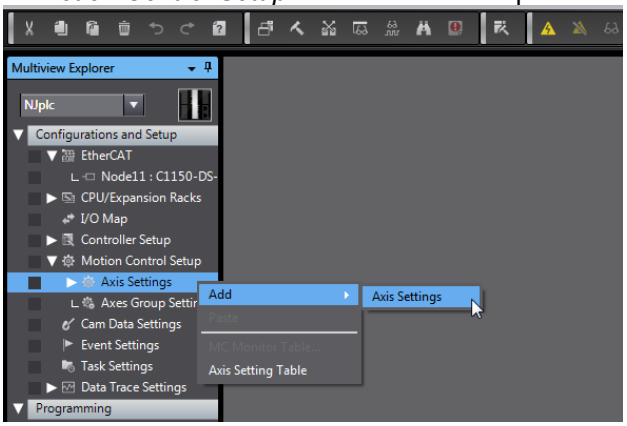
4. When Force/Torque must be limited on Position-Control-Mode, choose in addition *Torque/Force limits*.



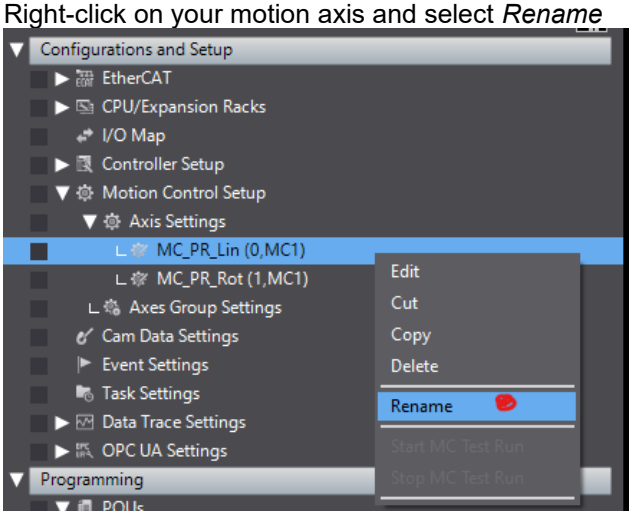

### 17.3.5 Create a motion axis and link it to your physical LinMot axis

Follow these steps to use the LinMot axis as a motion axis on your OMRON PLC.

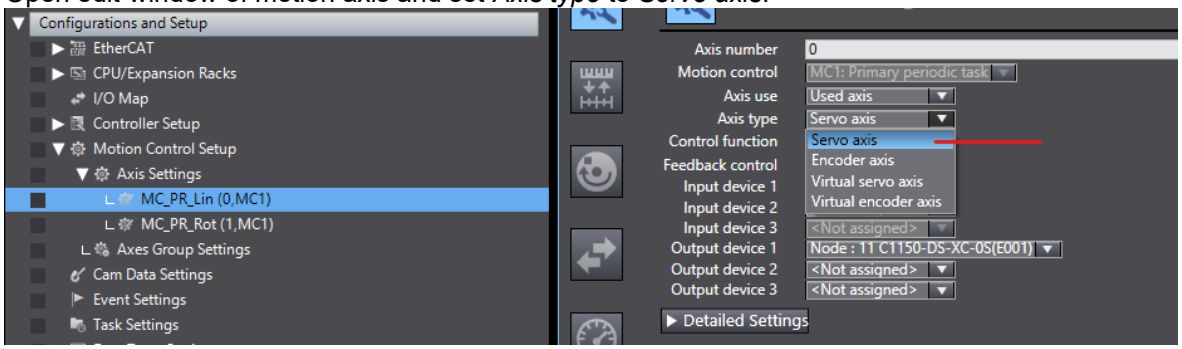
5. Create a motion axis:
   When *Offline* with the PLC right-click *Axis Setting → Add → Axis Setting* under *Configuration and Setup → Motion Control Setup* in the Multiview Explorer.



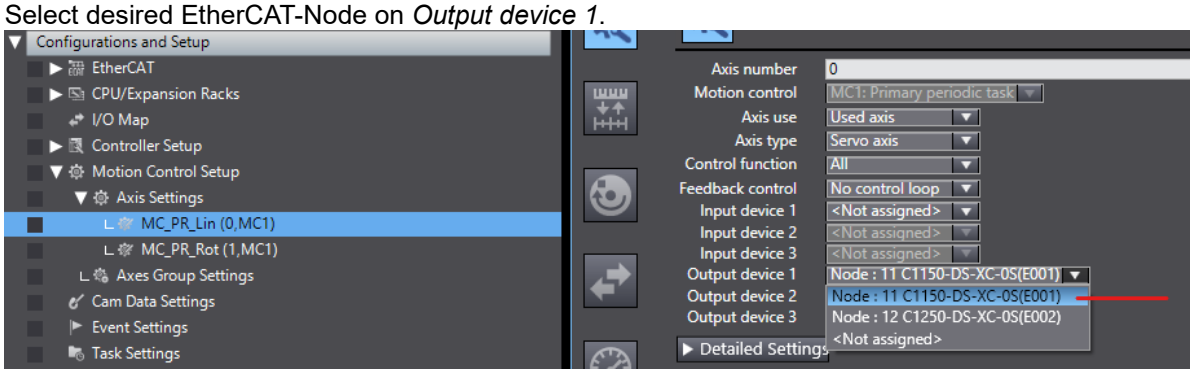

6. First assign a meaningful name to the axis.
   Right-click on your motion axis and select *Rename*



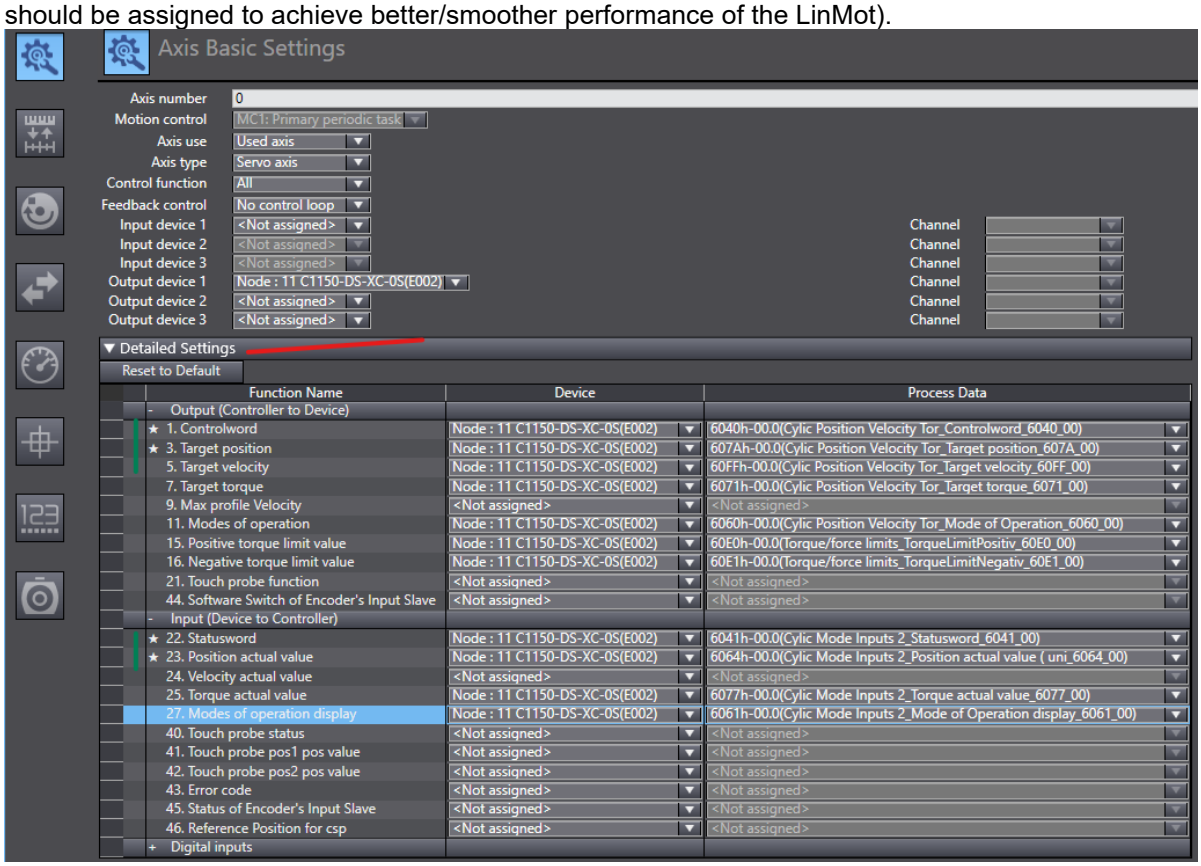
7. Set axis type.
   Open edit-window of motion axis and set *Axis type* to *Servo axis*.




8. Link physical axis to the motion axis:
   Select desired EtherCAT-Node on *Output device 1*.



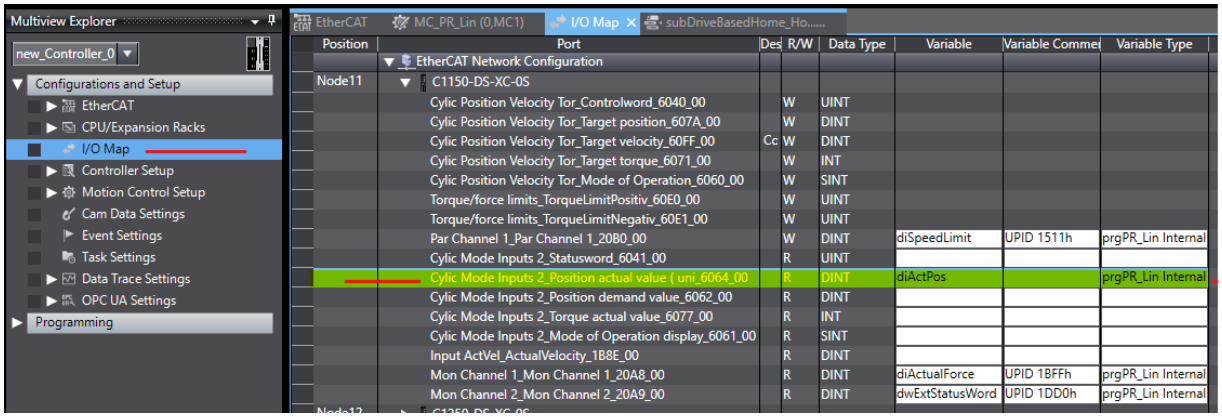

9. Assign PDO's of the axis
   Select *Detailed Settings* and assign Out- and Inputs as shown below.
   (Entries marked with a star are the minimal entries required from Sysmac. In addition, *Target velocity* should be assigned to achieve better/smoother performance of the LinMot).



10. OPTIONAL: When using drive-based-homing, assign the Actual-Position of Drive in the PDO mapping to synchronise it with the position value of the Motion-Axis on PLC.

    Open *Configuration and Setup → I/O MAP* in the Multiview Explorer. Select desired node and assign PDO 6064h as shown below.

    For detail description refer to chapter 17.5 (drive-based-homing).

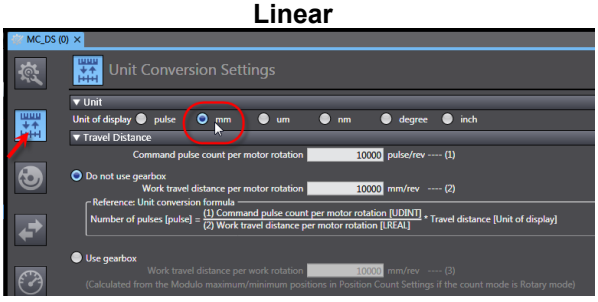



### 17.3.6 Setting up the motion axis

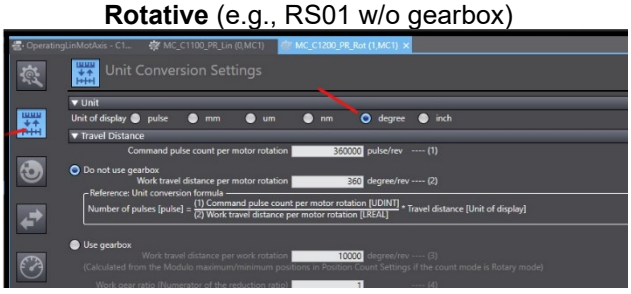
The following steps show how to basically setup the motion axis.

1. Set axis unit:
   Choose *Unit Conversion Settings* in the Edit-Window of your axis and set *Unit* to “mm”.

**Linear**



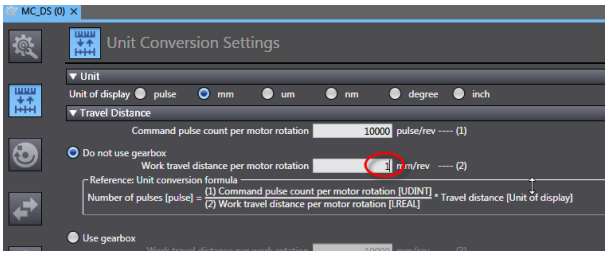

**Rotative** (e.g., RS01 w/o gearbox)



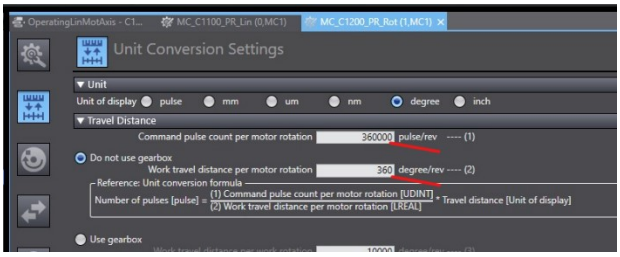

2. Set axis scaling:
   Set *Travel Distance* to standard value *10’000:1* [Ticks/mm] on linear axis or *360’000:360* [Ticks/rotation] on rotary axis.

For rotary motors, choose your desired work travel distance according to your mechanical system (gearbox, belt transmission, pinion etc.)

Linear

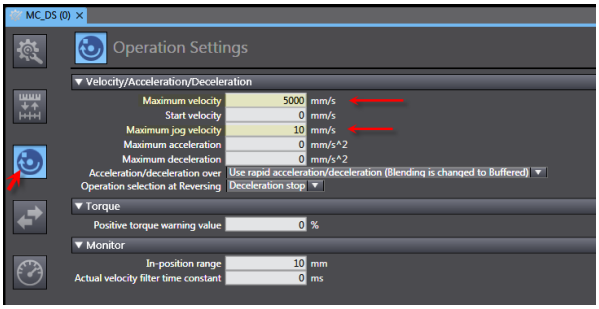



Rotative (e.g., RS01 w/o gearbox)

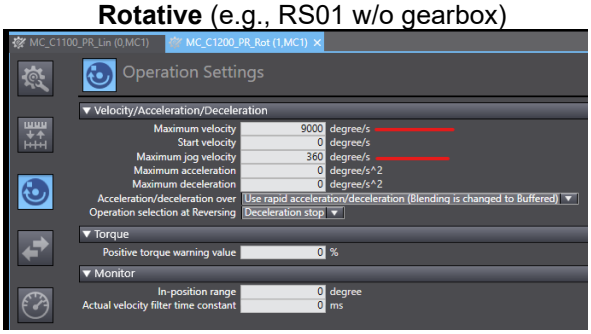


3. Set axis velocity limits:
   Choose *Operation Settings* and set your desired values at least into *Velocity* fields.

**Linear**




**Rotative** (e.g., RS01 w/o gearbox)

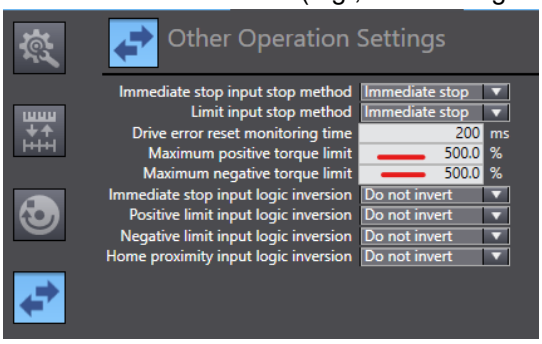



4. Set axis torque limits:
   In case force/torque must be limited in Position-Control-Mode, set torque limit to **500.0%**.
   Use motion instruction ***MC_SetTorqueLimit*** to limit force/torque of your motor.

Maximal value (equals to 500.0%) can be read out of motor:
*LinMot-Talk / Variables / MC SW Motor / Maximal Force*
*Maximal Torque*

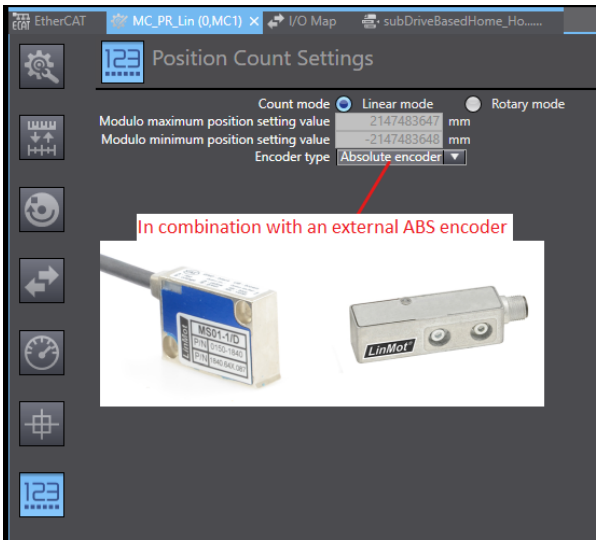
→ Max. value is defined by each individual motor and may vary slightly from datasheet.

**Linear** and/or **Rotative** (e.g., RS01 w/o gear)

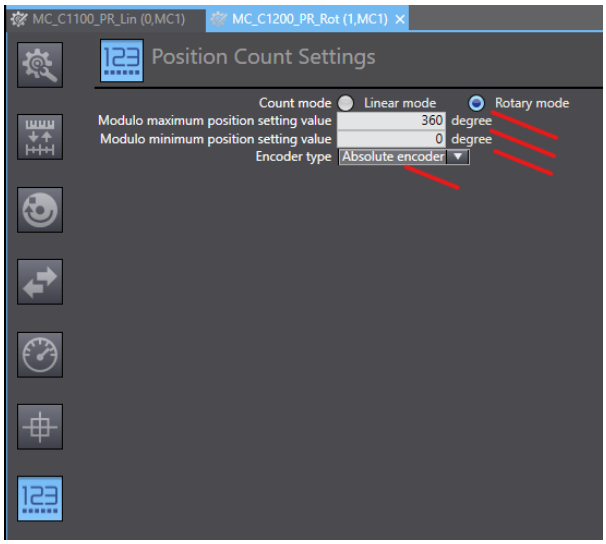



5. Configure ABS encoder and Modulo:
   In case of using a LinMot rotary motor (with single turn ABS encoder) or an external ABS encoder for linear motors, *Absolute encoder* must be selected. To use modulo functionality, select *Rotary mode*.

**Linear** with external ABS encoder



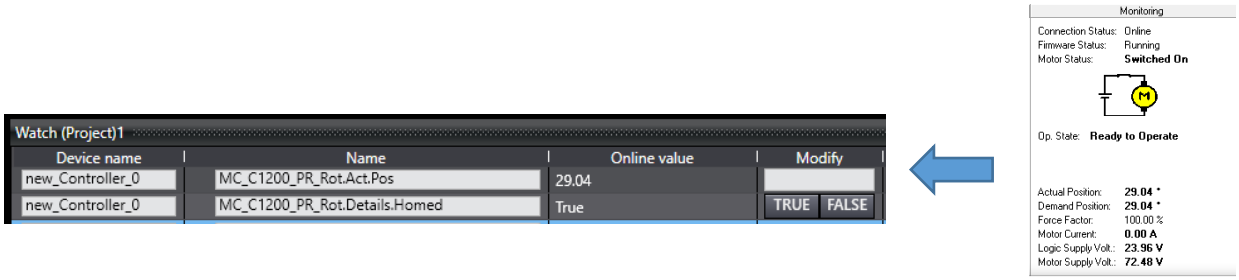

**Rotative** (e.g., RS01 w/o gearbox)



6. Transfer program to your PLC:
   Go Online and *Transfer*


## 17.4 Homing procedure on Sysmac with ABS Encoder

When using ABS encoder, there is no need for any homing procedure.



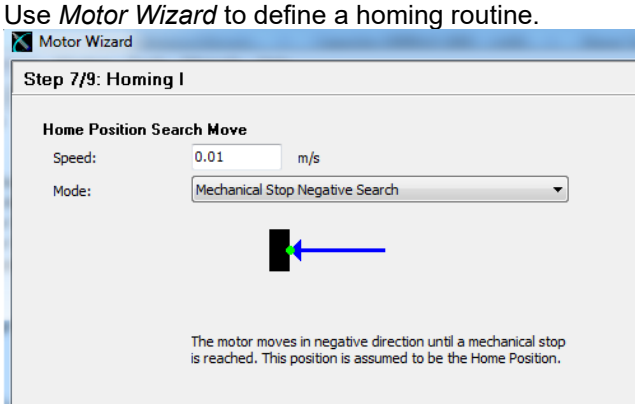

## 17.5 Drive Based Homing (INC Encoder)

In general, the easiest way to home a linear motor is to use homing routines predefined in the servo drive.

### 17.5.1 Settings on LinMot drive

The following setting are required on the drive.

1. Use *Motor Wizard* to define a homing routine.



### 17.5.2 Function blocks required on Sysmac

To operate a drive-based homing the following function blocks are used.

#### 17.5.2.1 EC_CoESDORead

With this function block, the status word of the physical LinMot axis will be read.

Variables to declare:

| | Data type | Value |
|---|---|---|
| Function block | EC_CoESDORead | |
| SDO Parameter | _sSDO_ACCESS | Index := 16#3D51, Subindex := 1 |
| Read Data | ARRAY[0..3] OF WORD | Located in [0] |

#### 17.5.2.2 EC_CoESDOWrite

With this function block, the homing procedure will be executed and stopped.

Variables to declare:

| | Data type | Value |
|---|---|---|
| Function block | EC_CoESDOWrite | |
| SDO Parameter | _sSDO_ACCESS | Index := 16#33F7, Subindex := 1<br>Index := 16#341F, Subindex := 1 |
| Write Data | WORD | 16#0 (FALSE) or 16#1 (TRUE) |
| Write Data Size | UINT | 2 (Bytes to write) |

#### 17.5.2.3 MC_ResetFollowingError

With this function block, the position lag on the motion axis will be reset.

Variables to declare:

| | Data type | Initial Value |
|---|---|---|
| Function block | MC_ResetFollowingError | |

#### 17.5.2.4 MC_HomeWithParameter

With this function block, the PLC motion axis will be homed.

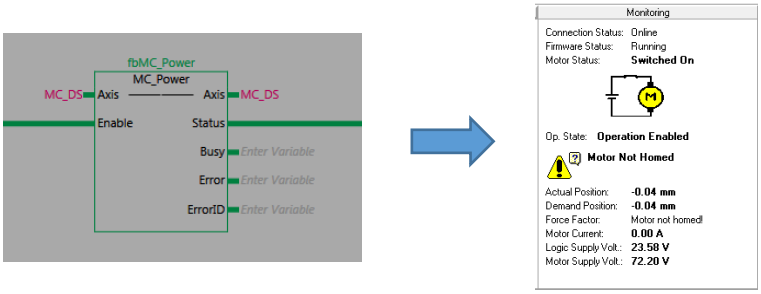
Variables to declare:

| | Data type | Initial Value |
|---|---|---|
| Function block | MC_HomeWithParameter | |
| Homing Parameter | _sHOMING_REF | (HomingMode := _mcHomePreset) others irrelevant |

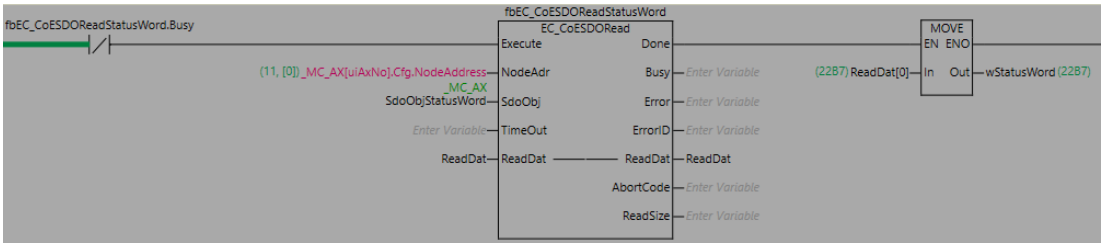
### 17.5.3 Homing procedure on Sysmac (motors with INC Encoder)

This section describes the *Drive Based Homing* sequence.

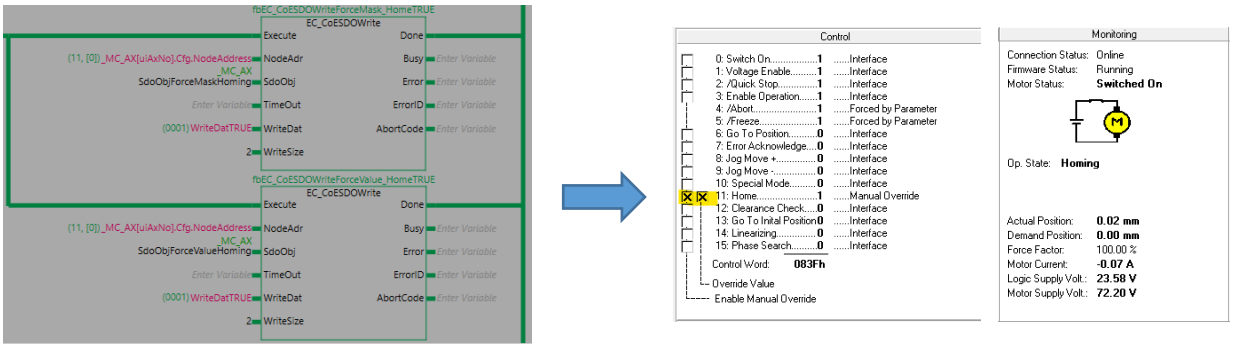
1. Use MC_Power instruction to make axis ready to operate.



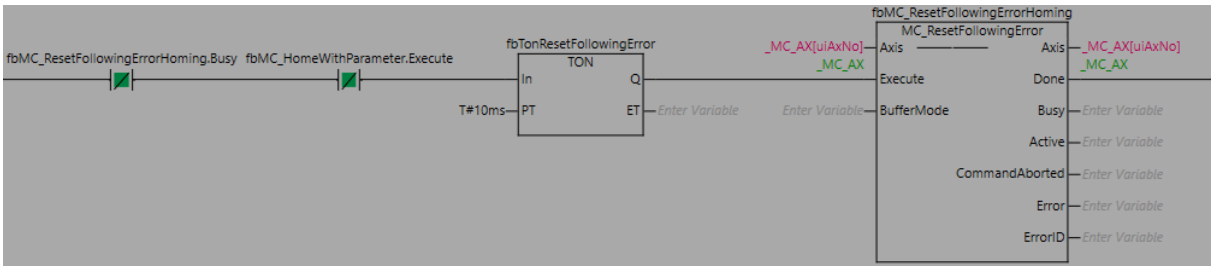

2. Read Status-Word of LinMot drive while homing to check Homed Bit.



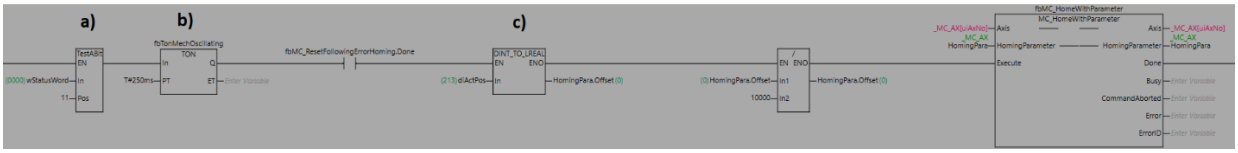

3. Execute homing by setting the *Home* within the *Control Word* to TRUE.

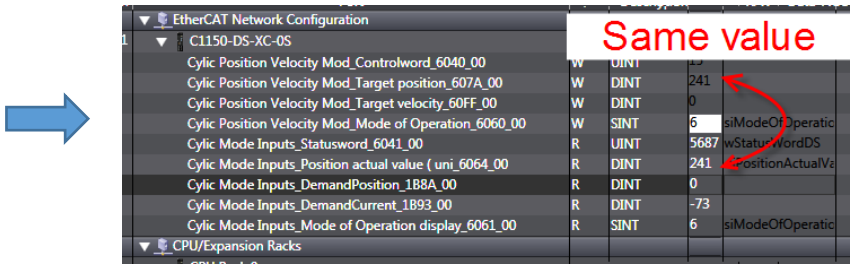



4. Clear the following-error of motion axis during *Drive Based Homing* otherwise malfunction can occur (e.g., when homing long stroke).
   It must be cleared, because Sysmac does not update the axis position during *Drive Based Homing*.

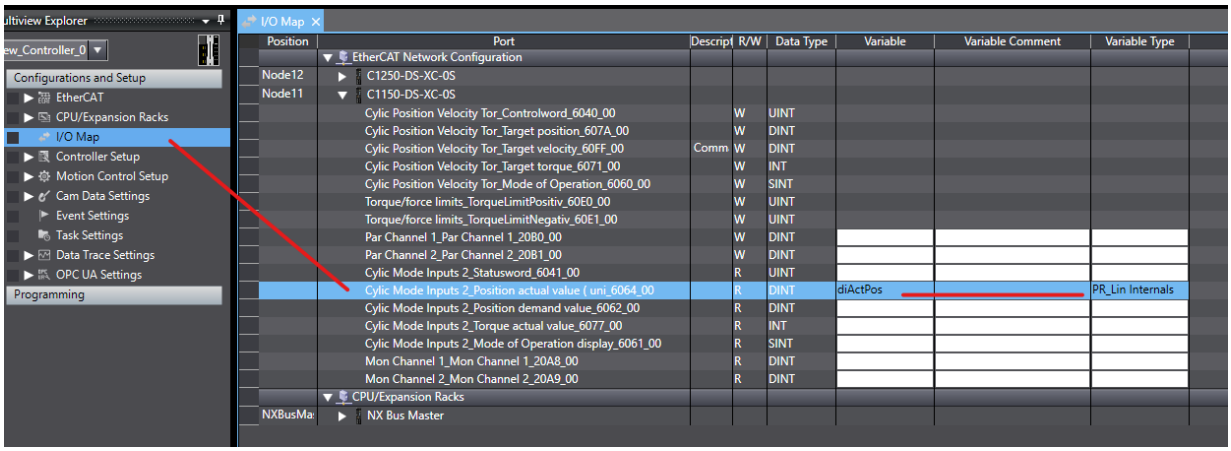


5. a) Observe Homed Bit in Status-Word of LinMot drive.
   b) When axis is homed, may wait a short time, until the axis does not oscillate due mechanical construction.
   c) Use Actual-Position[i] of LinMot drive as offset for MC_Homing on PLC.

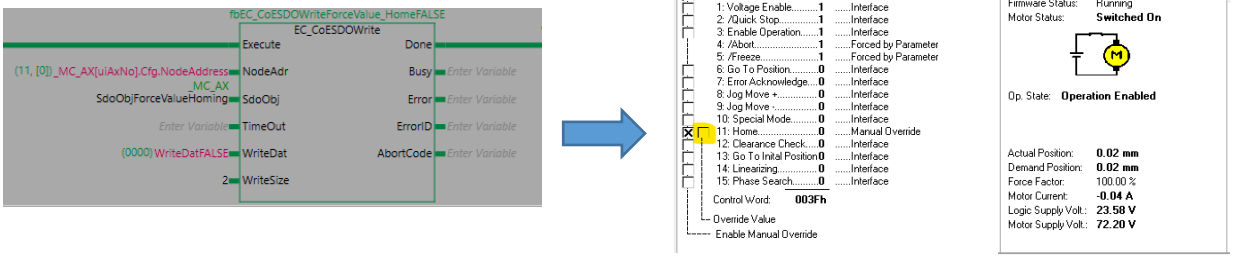



[i] Map a variable in I/O Map to have Actual-Position available. Also refer to chapter 17.3.5 10)

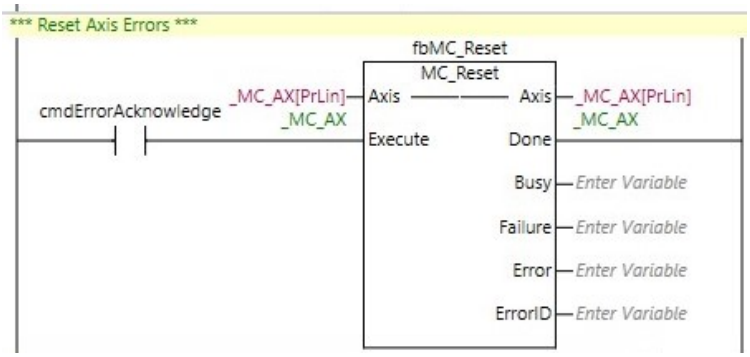



6. Reset homing by setting the *Home* within the *Control Word* to FALSE.



## 17.6 Drive reset to clear axis error

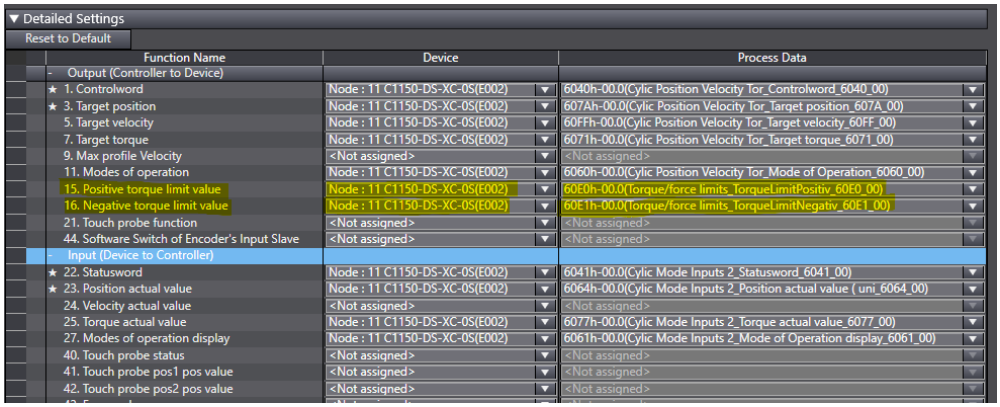
To acknowledge an error of a DS402 / CiA402 drive, use the standard motion instruction ***MC_Reset***.

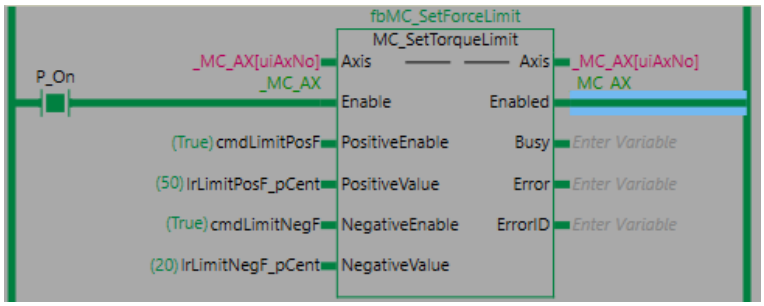



## 17.7 Control of force / torque

### 17.7.1 Limiting force or torque output in Position-Control-Mode

To limit output torque/force of a DS402 / CiA402 servo drive, use standard motion instruction ***MC_SetTorqueLimit***. Please make sure, the necessary PDOs are mapped.

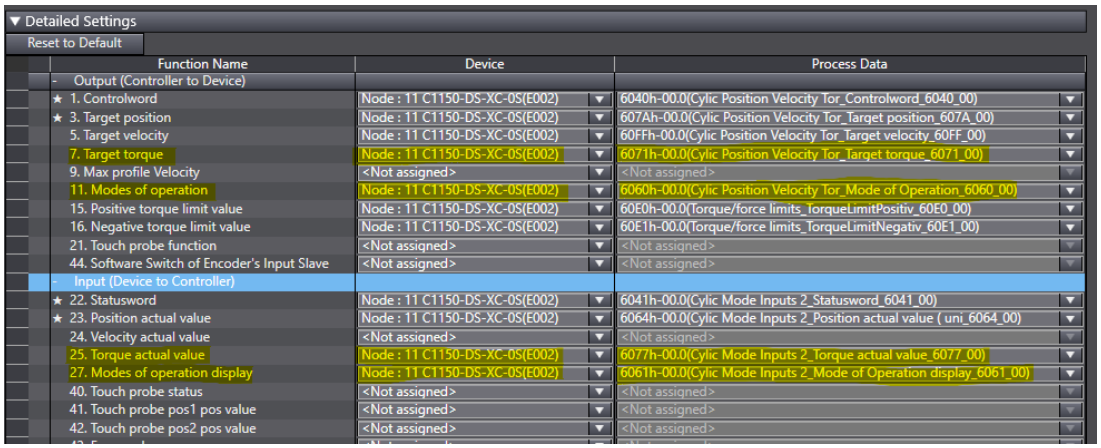


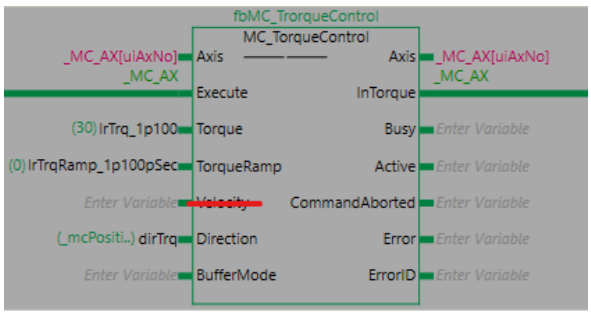




**Value 500.0% equals to max. force or torque.**

### 17.7.2 Control force or torque output in Torque-Control-Mode (CST Mode 10)

To control output torque/force of a DS402 servo drive, use standard motion instruction ***MC_TorqueControl***.
Please make sure, necessary PDOs are mapped, and parameter settings on drive are done correctly.



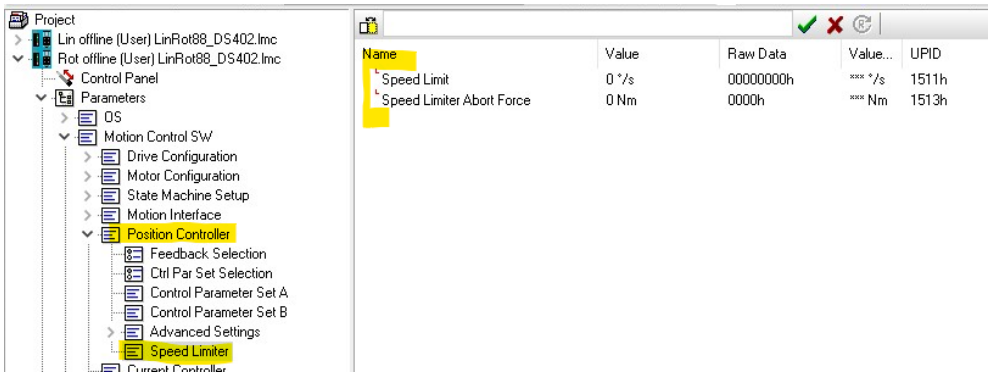




**Torque 500.0% equals to max. force or torque.**

The Velocity input of MC_TorqueControl is currently not supported for LinMot drives.
Use the Speed Limiter instead

#### 17.7.2.1 Speed Limiter for Torque/Force-Control-Mode

To prevent overspeed while controlling force or torque, use the Speed Limiter.

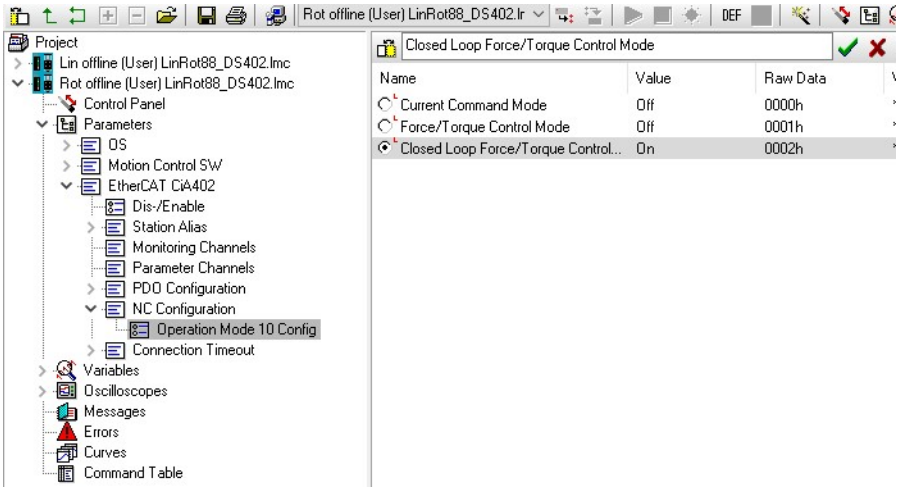


When the motor speed reaches *Speed Limit*, the drive switches to Position-Controlled-Mode.
The selected Control parameter Set (A or B) will become active!

Switching from Force/Torque- to Position-Mode and vice versa while the motor is moving, will always cause a bumpy behaviour.
Smoother reaction can be achieved by adjusting PID values of the corresponding control loops.

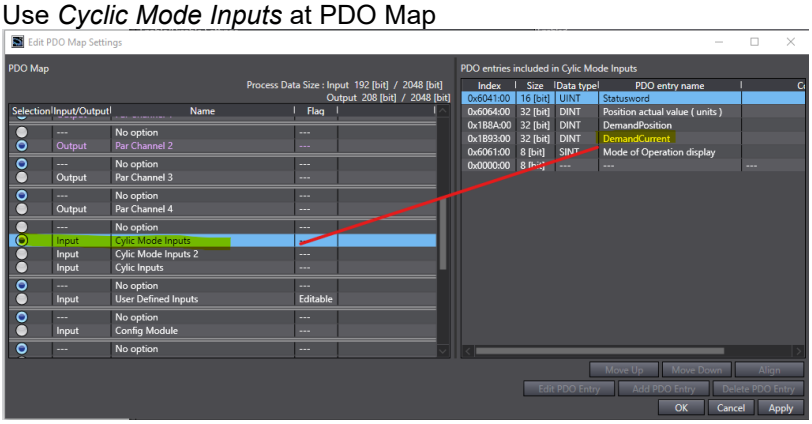
### 17.7.2.2 Torque/Force-Control-Mode in CST Mode 10

If using an additional force or torque sensor, Closed-Loop Control Mode must be selected on drive.



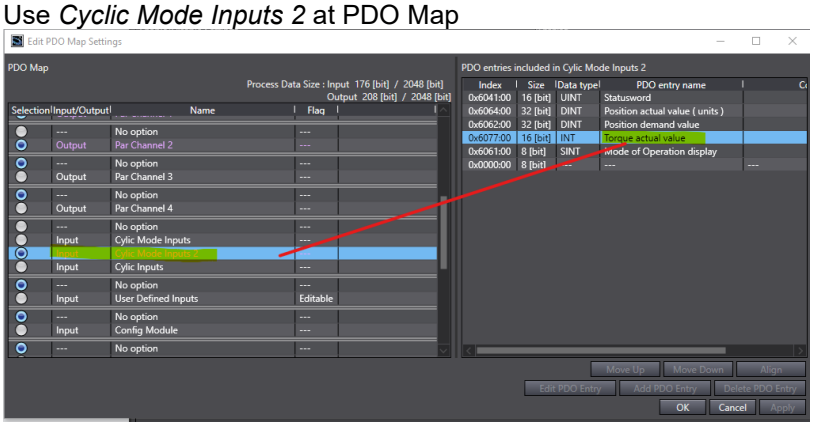

**0000h: Current Command Mode**
This mode is only for reverse compatibility with FW older than V6.10!

Use *Cyclic Mode Inputs* at PDO Map



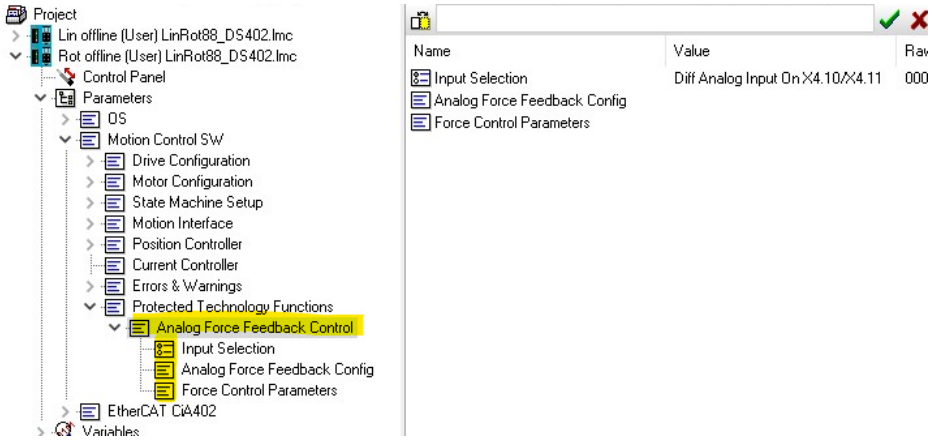
**0001h: Force/Torque Control Mode**
This mode corresponds to DS402.

Use *Cyclic Mode Inputs 2* at PDO Map



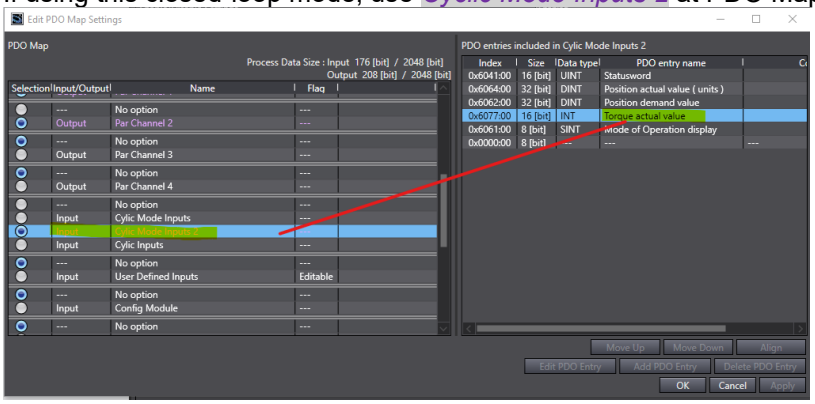

**0002h: Closed-Loop Force/Torque Control Mode**
In addition to *0001h: Force/Torque Control Mode*, this mode requires TF-Force Control (Technology Function).
LinMot-Article: 0150-2503

It is used in combination with an external torque or force sensor such as TS01 or TS02 on a PR-module.
These sensors must be wired directly to the drive.




Make sure all parameters in Analog Force Feedback Control are set correctly.

If using this closed-loop mode, use *Cyclic Mode Inputs 2* at PDO Map.



## 17.8 Parameter Access

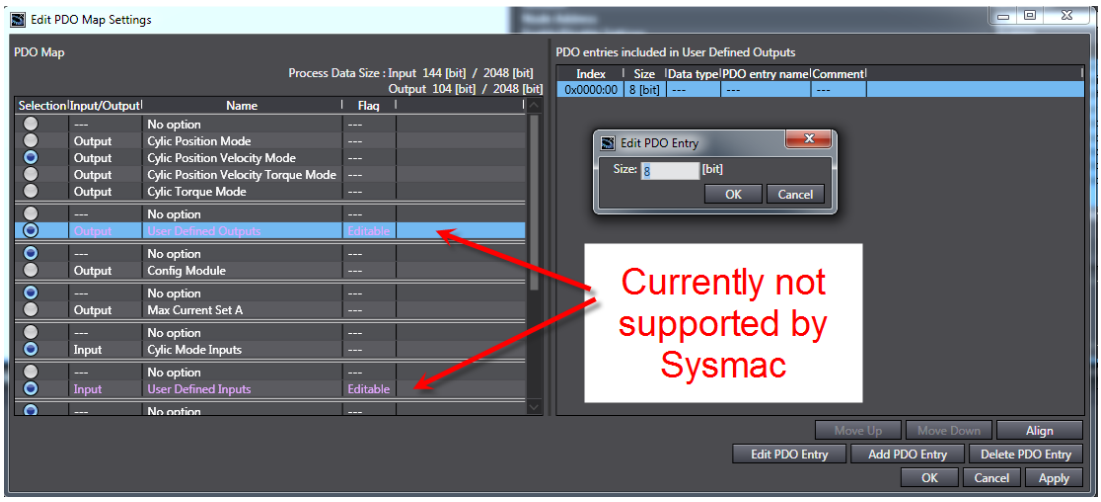
### 17.8.1 PDO Access

For cyclical write or read access of any parameter or variables, up to 4 channels each can be used.

OMRON does not support *User Defined In- or Outputs* for PDO. Please use up to 4 Parameter and Monitoring Channel each instead.
If no real-time access is required, use SDO for reading or writing values/parameters.

➔ Use parameter channels for cyclical write access.

➔ Use monitoring channels for cyclical read access.

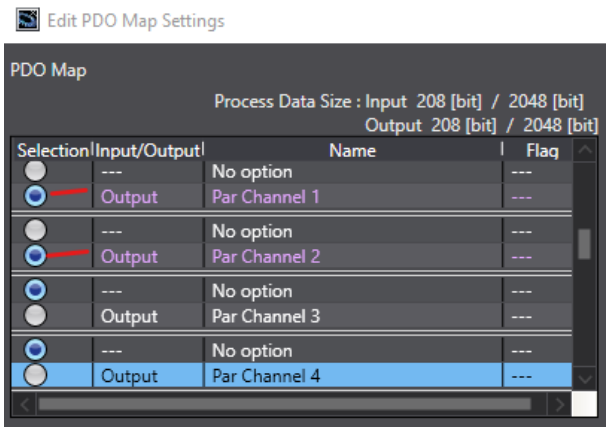


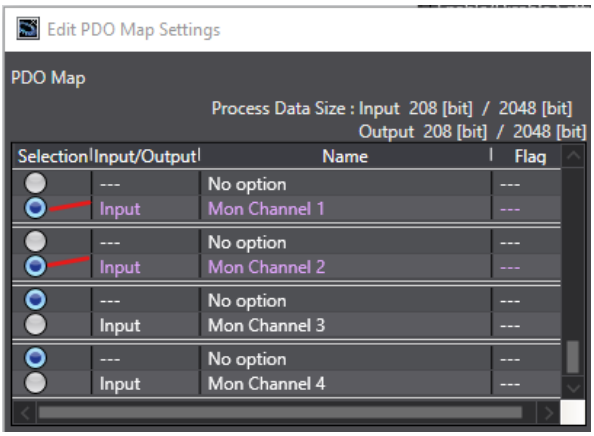

### 17.8.2 Example for user defined PDO access

In this example two channels each are used for a PR02-module (rotary motor for torque control).

#### 17.8.2.1 PDO Map

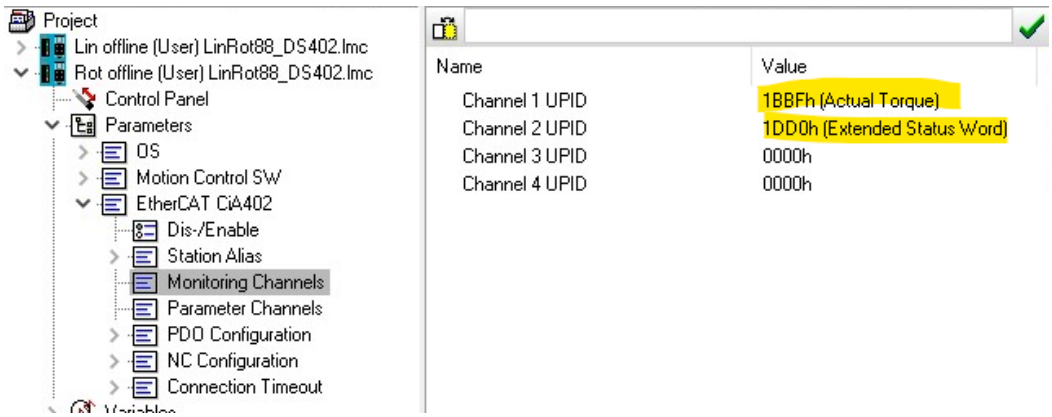
Use *Edit PDO Map Settings* at desired EtherCAT node.



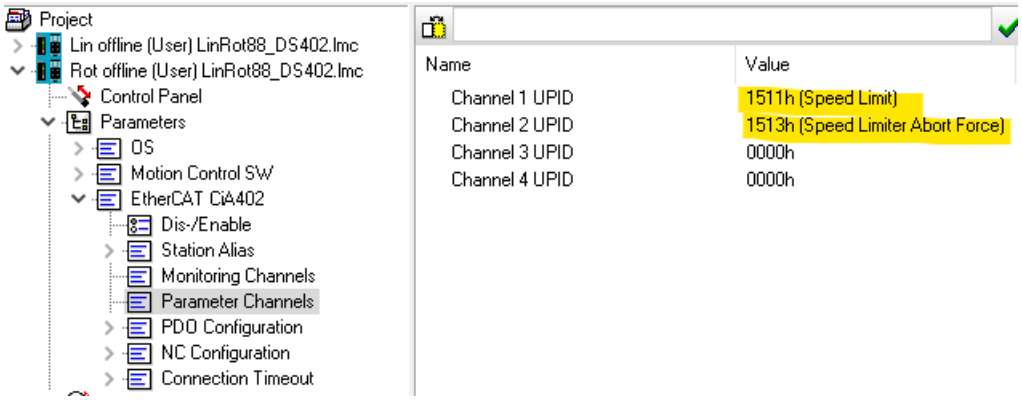



### 17.8.2.2 I/O Map

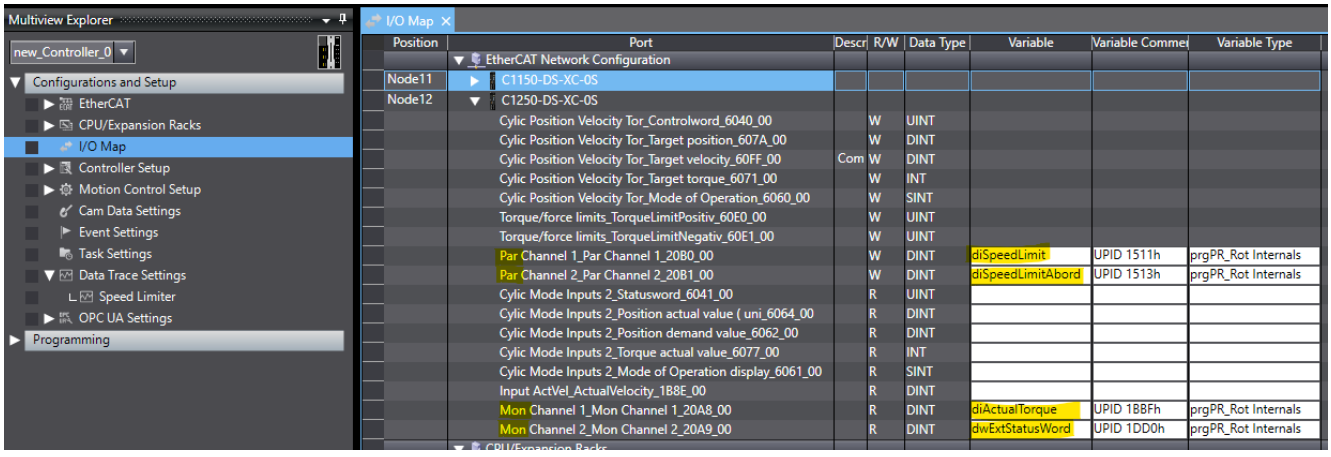
On the drive assign UPID *Actual Torque* to and *Extended Status Word* to the **Monitoring** Channels




And to the **Parameter** Channels assign UPID of *Speed Limit* and *Speed Limiter Abord Force*.



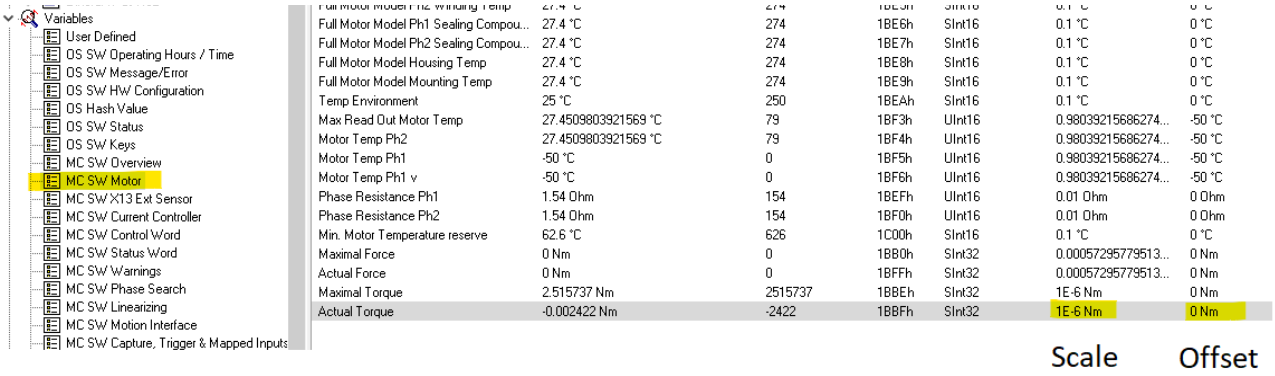

On Sysmac assign a DINT or a DWORD to the selected PDO channels.

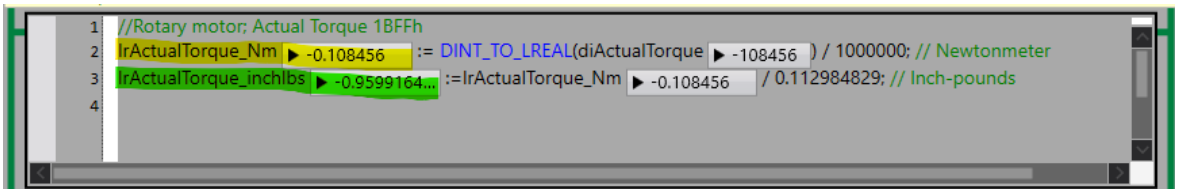


### 17.8.2.3 Scaling of mapped PDO

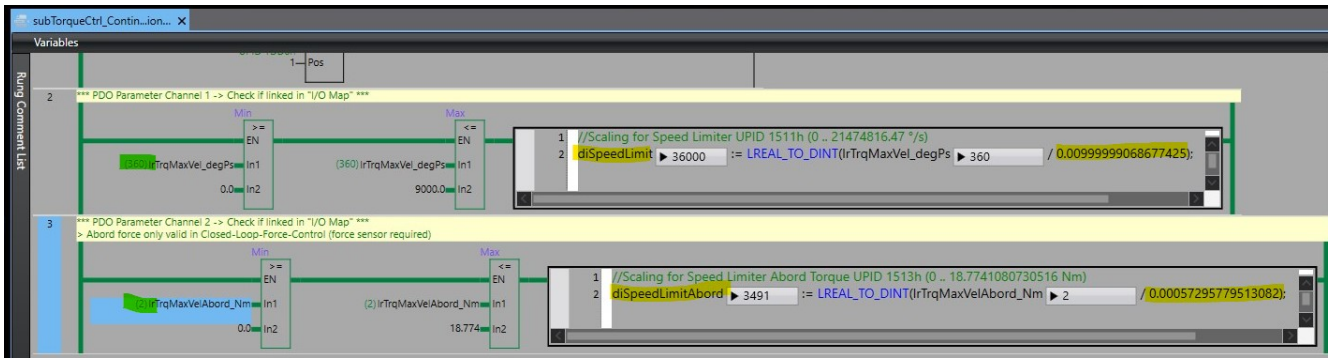
Check scale and offset of all used UPIDs and recalculate them on Sysmac

**Example *Actual Torque:***

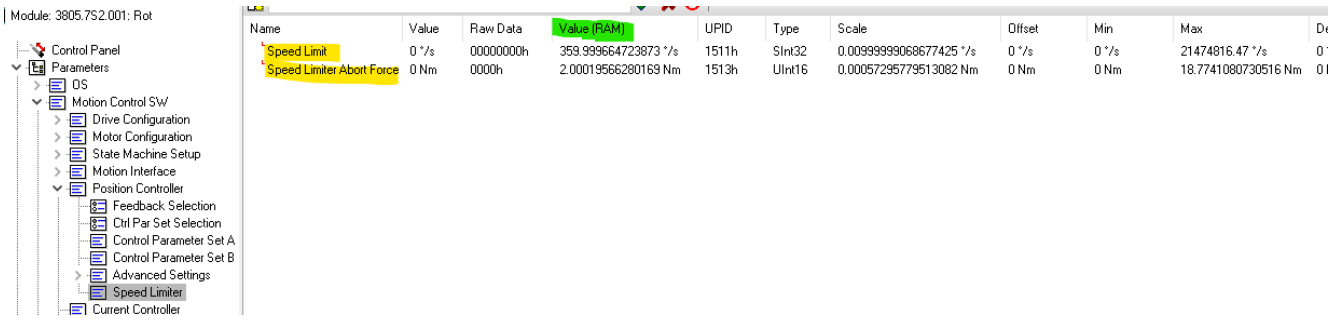


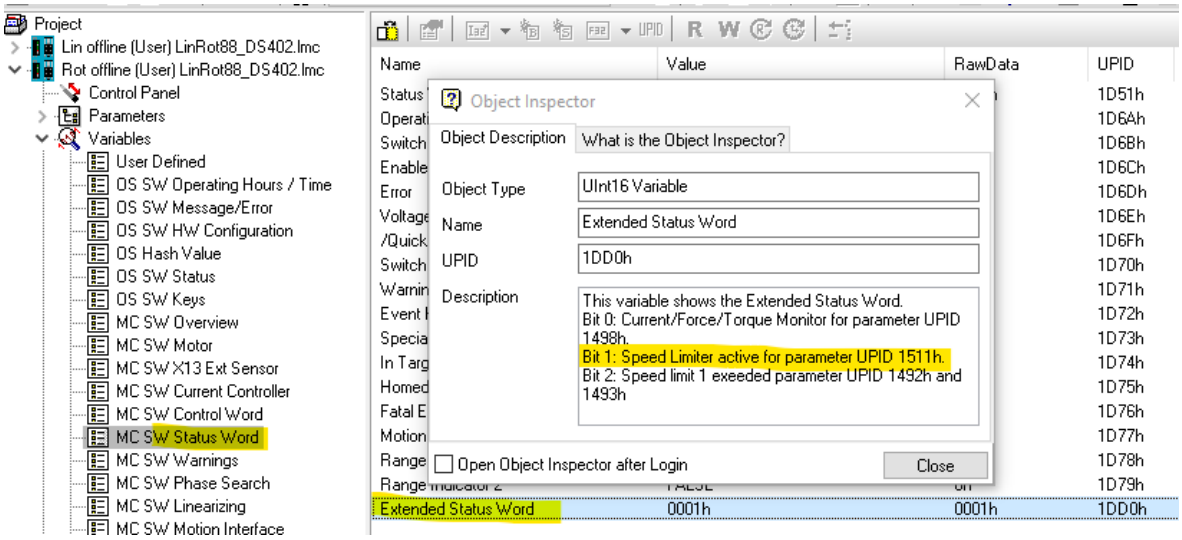

**Example *Speed Limit* and *Speed Limit Abort Force:***



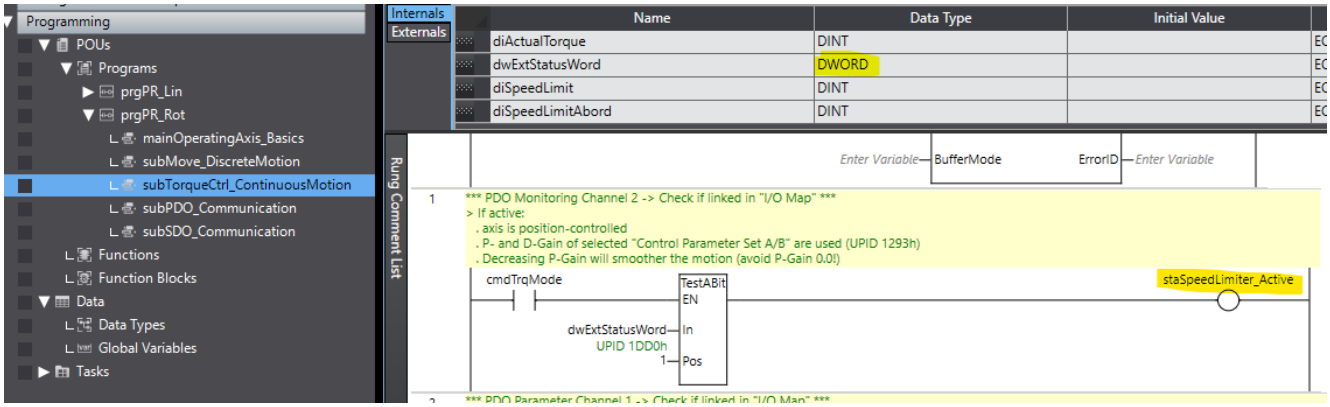

➔ Check the value of the parameter always in RAM



**Example *Speed Limiter Active:***




➔ For Bit variables DWORD instead of DINT can be used to make masking easier.

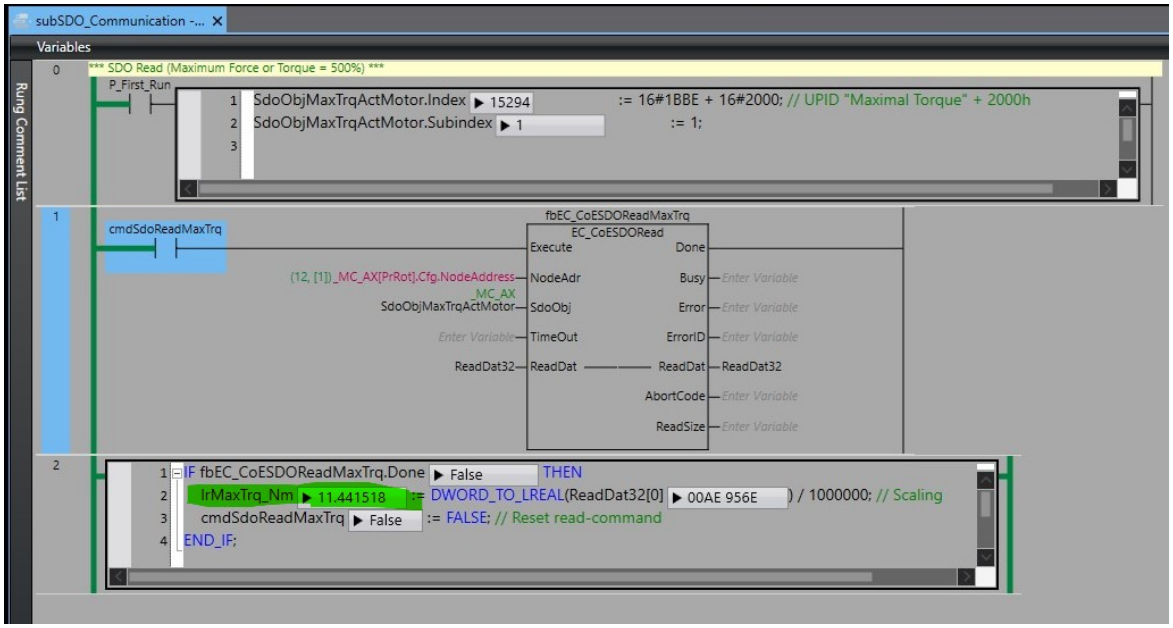


### 17.8.3 SDO Access

Parameters of the LinMot drive can be accessed using the Omron function block *EC_CoESDORead* and *EC_CoESDOWrite* which are part of Sysmac library.

More information about this function blocks can be found in the *Instructions Reference Manual (W502)* from Omron:
https://industrial.omron.eu/en/home (search for *W502)*

Example calls of this function blocks can be found in the example project (see chapter 17.1).




**NodeAdr:** EtherCAT-Slave node address

**SdoObj.Index:** Parameter address to be accessed = **16#2000** + UPID

**SdoObj.Subindex:** Parameter memory area to be accessed.
01: Read and write the **RAM** value of the UPID.
02: Read and write the **ROM** value of the UPID.

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

**Note:**
Use *EC_CoESDORead* and *EC_CoESDOWrite* from Omron to access drive parameters.

**Advanced: Stop / Start MC and Application SW and reboot drive**
The SDO-index 16#2000 allows

1. Stopping the MC and Appl SW for flash access or to setup the drive from the PLC
2. Starting the MC and Appl SW after flash access and to load changed ROM values to RAM,
3. Rebooting the drive (warm start).

Subindex 16#35 ▶ Stop MC and Application SW
Subindex 16#36 ▶ Start MC and Application SW
Subindex 16#0B ▶ Reboot drive

# 18 Drive Profile: TRIO, CiA402

## 18.1 Overview

This chapter shows how a LinMot drive with *DS402 (CoE) / CiA402* interface (e.g., C1250-DS-XC-1S) can be integrated and setup as axis in a TRIO motion controller.

**Download:**
An example project can be downloaded from:
http://download.linmot.com/plc_lib/examples/TRIO (named *LM_TRIO_DS_Demo...*)

**Attention:**
Drive firmware **6.10 Build 20210521 or later** is required to use all the functionality shown in this chapter.

Components used:

- LinMot C1250-DS-XC-1S-000 (article number 0150-2416) with firmware 6.10 Build 20210521
- TRIO MC4N ECAT (P904) v2.0309
- Motion Perfect v5.1.1

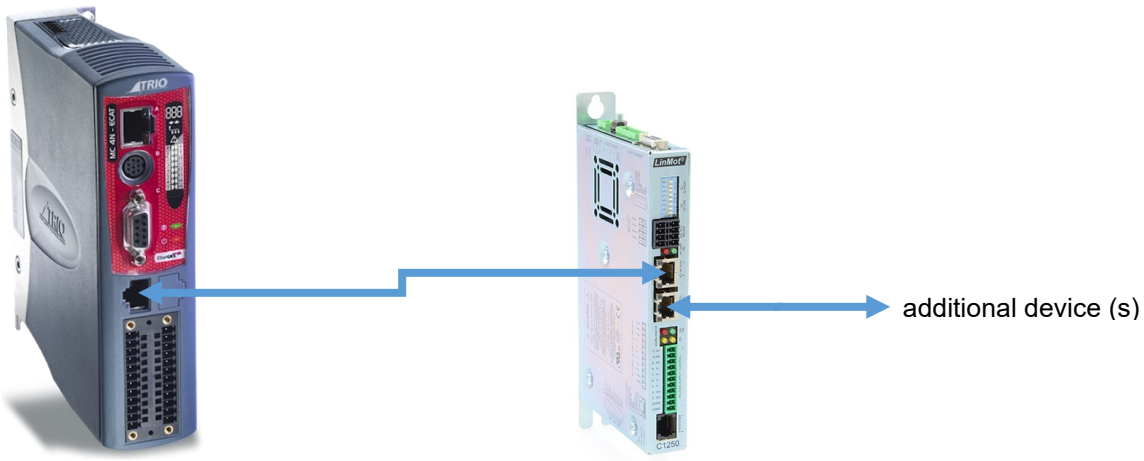



Image Source: http://www.triomotion.uk/

EtherCAT is the real-time Ethernet network originally developed by Beckhoff. The LinMot acts as Slave in this network and is implemented with the standard ASIC ET1100 from Beckhoff.

For further information on the EtherCAT fieldbus please visit:
http://www.ethercat.org/

## 18.2 Configuration and Connection of the LinMot Drive

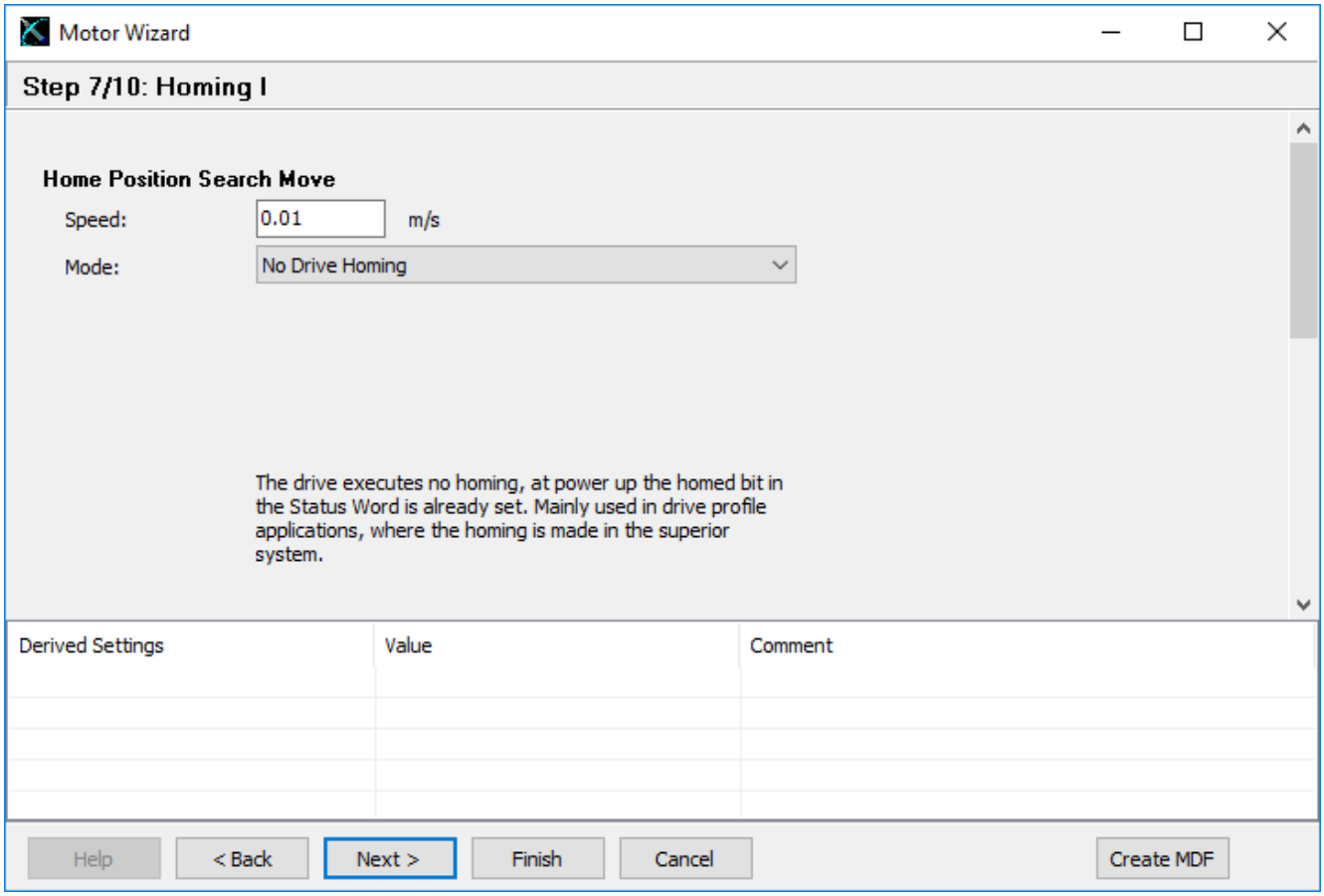
The LinMot drive is configured using the LinMot-Talk software:
http://www.linmot.com/download/linmot-talk-drive-configuration/

### 18.2.1 Motor Configuration

It is assumed that the motor connected to the drive is already configured.

Use the *Motor Wizard* inside LinMot-Talk to setup the motor (Menu -> Drive -> Motor Wizard):


Make sure that you select “No Drive Homing” in the Motor Wizard as with TRIO the homing of the axis is done in the PLC (DS402 mode of operation 6 is not yet supported).




See Appendix I: Basic Position Control Loop Tuning

### 18.2.2 XML File

Install the XML file that is part of the LinMot-Talk software/firmware you are using.
The most recent device files are always part of the newest LinMot-Talk software. They are located by default:

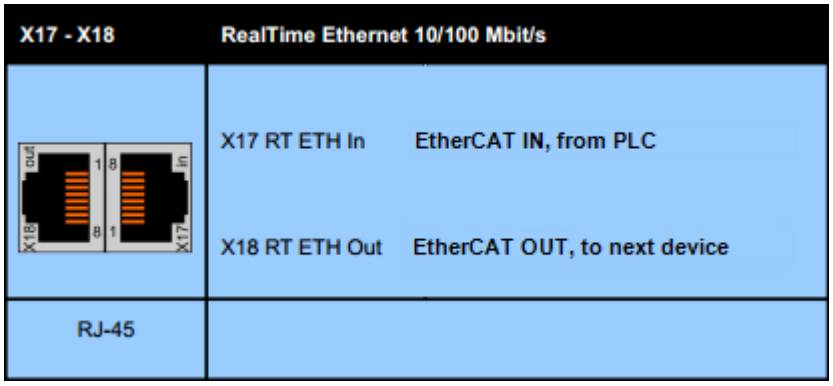
- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML\
- EtherCAT CoE: \\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML\ (-MI drives)

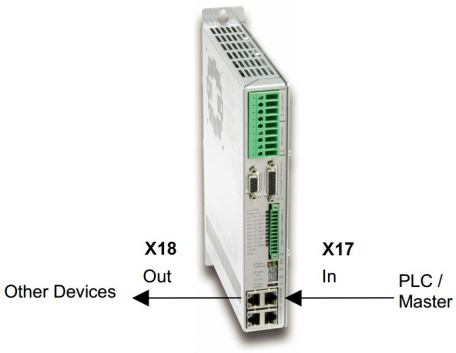
**XML file names:**

| | |
|---|---|
| NTIL_CiA402_Servos_Vx_xrx.xml | LinMot DS drives SG6-7 |
| NTIL_CiA402_SG5_Servos_Vx_xrx.xml | LinMot DS drives SG5 |
| NTIL_CiA402_Servos_MI_Vx_xrx.xml | LinMot MI drives SG6 |

### 18.2.3 EtherCAT Connection

The drive is connected to the EtherCAT network using the X17 (IN) & X18 (OUT) connectors.
The below pictures show the ports of an E1250-DS-UC drive. On all other LinMot drives supporting EtherCAT DS the ports are named the same (X17 & X18), but they may be placed differently on the drive housing.

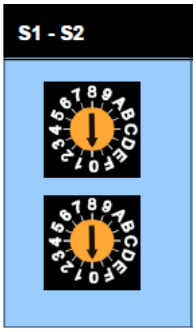


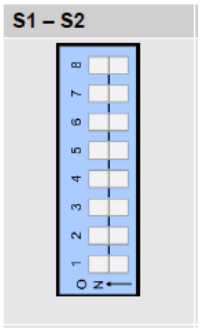

### 18.2.4 Set the node address / station alias by address selectors on the drive

This is the most popular way, because it is user-friendly to maintenance people.

Depending on the LinMot-Drive the address selectors are designed as Hex- or DIP-Switches.




Hex-Switches




DIP-Switches

## 18.3 Controller Setup

### 18.3.1 EtherCAT Device Description File (XML)

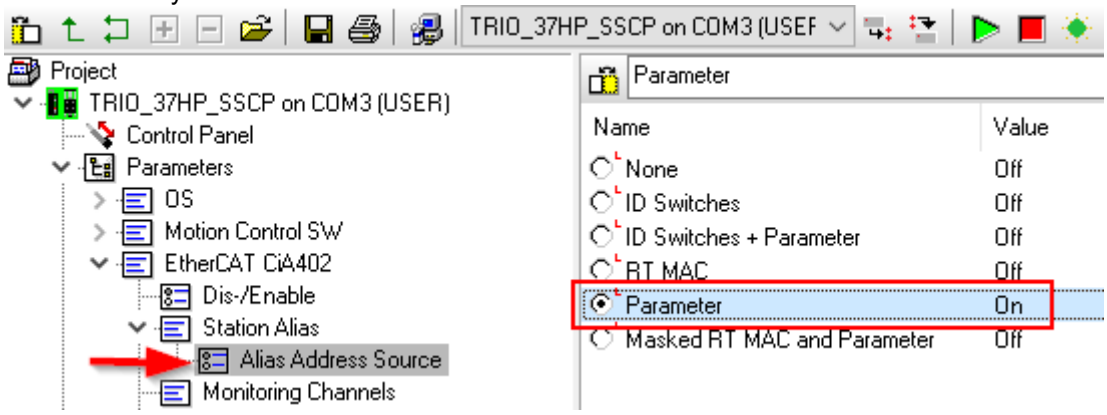
The EtherCAT device description file (XML) is located by default in the LinMot-Talk installation path:
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCAT\XML
C:\Program Files (x86)\LinMot\LinMot-Talk X.X Build XXXXXXXX\Firmware\Interfaces\EtherCat_Nx\XML

### 18.3.2 Add and setup the LinMot drive

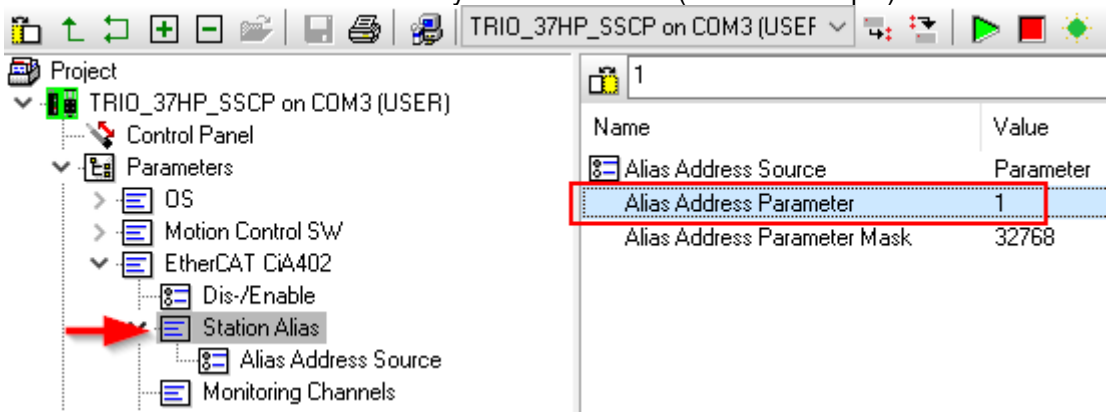
Follow these steps to add the LinMot drive to your Motion Perfect project.

1. Set the node address / station alias address of the LinMot drive to 1 for this example using the S1&S2 address selectors on the drive (if available > check chapter 18.2.4)

   Or alternatively in LinMot-Talk set the Alias Address Source of the drive to “Parameter”:

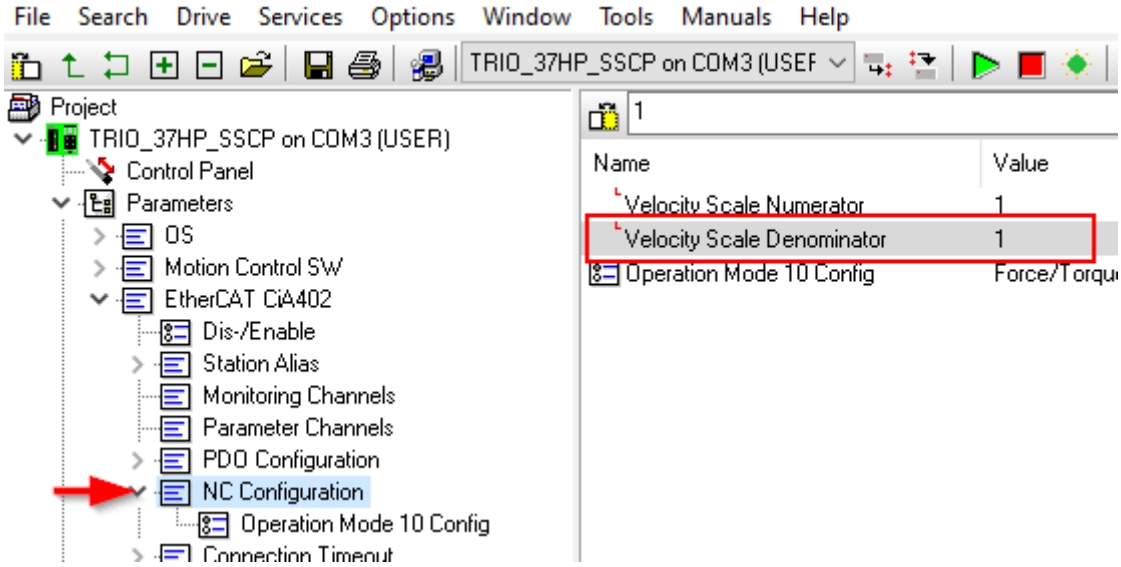



   And the “Alias Address Parameter” to your desired value (1 in this example):

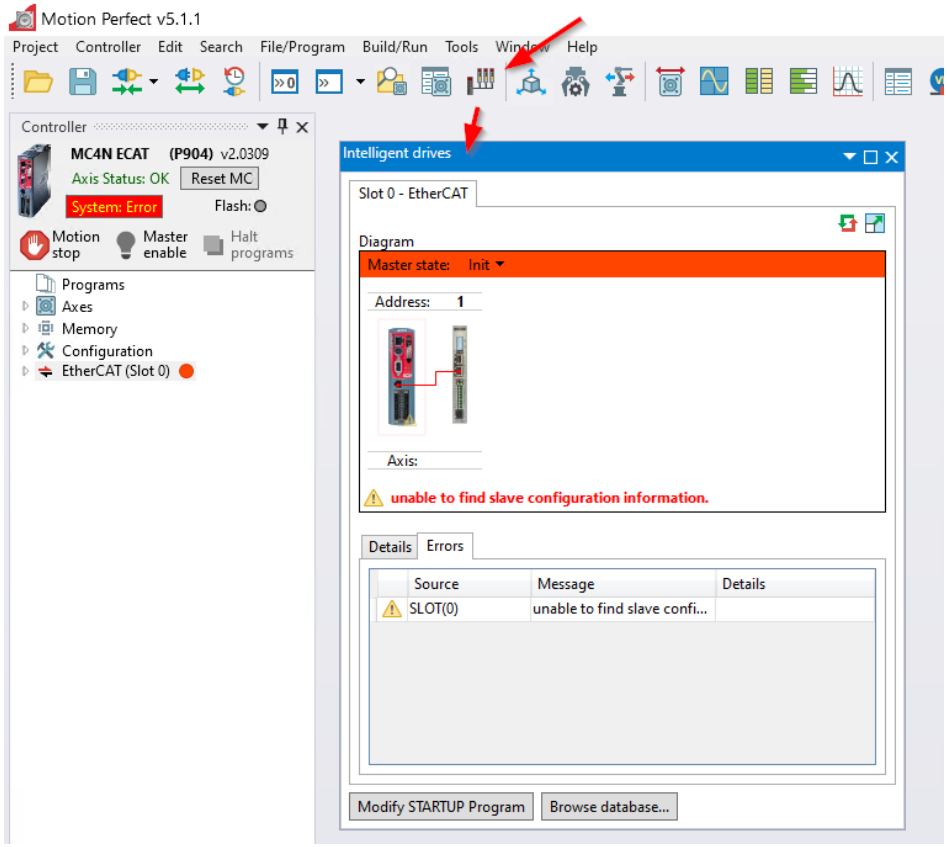



Attention: A drive reboot is required to activate the address after a change!

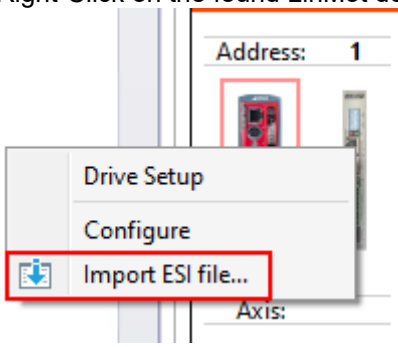
2. Set the “Velocity Scale Denominator” to **1**:



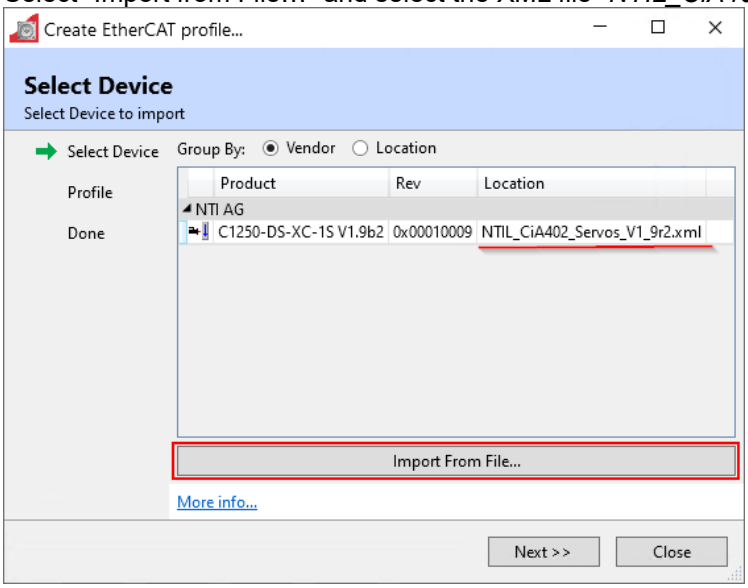

3. Open the “Intelligent drives” window:



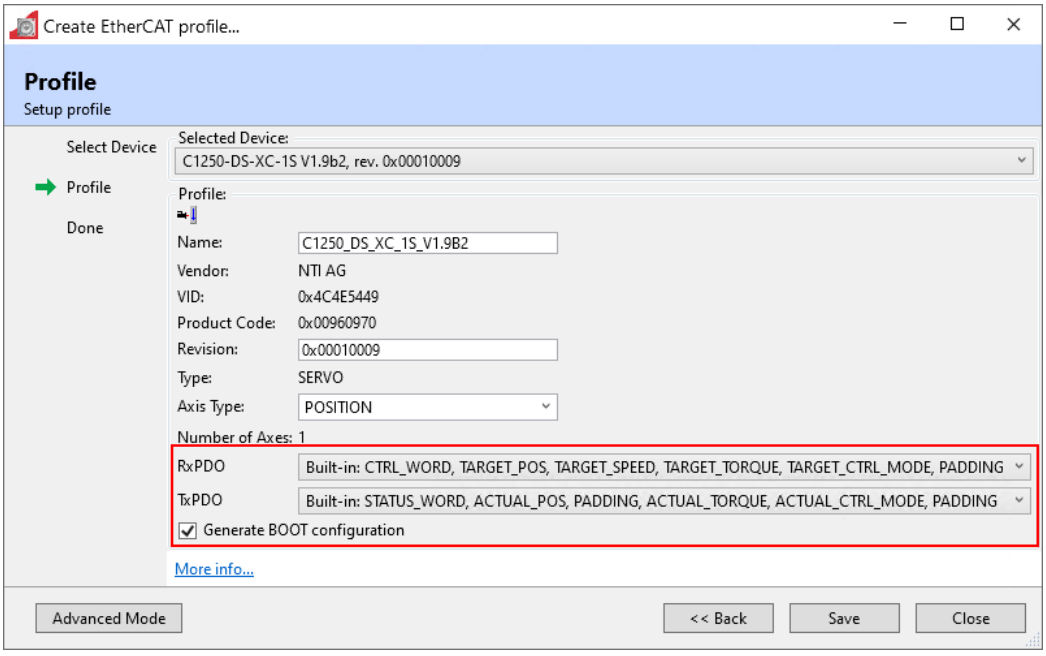
4. Right-Click on the found LinMot device (Address 1 in this case) and select *“Import ESI file…”*:




5. Select “Import from File…” and select the XML file “*NTIL_CiA402_Servos_...*”:

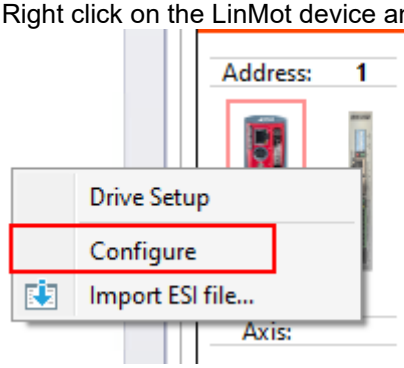



6. Choose the following **RxPDO** and **TxPDO** mappings, activate “Generate BOOT configuration” and click “Save”:

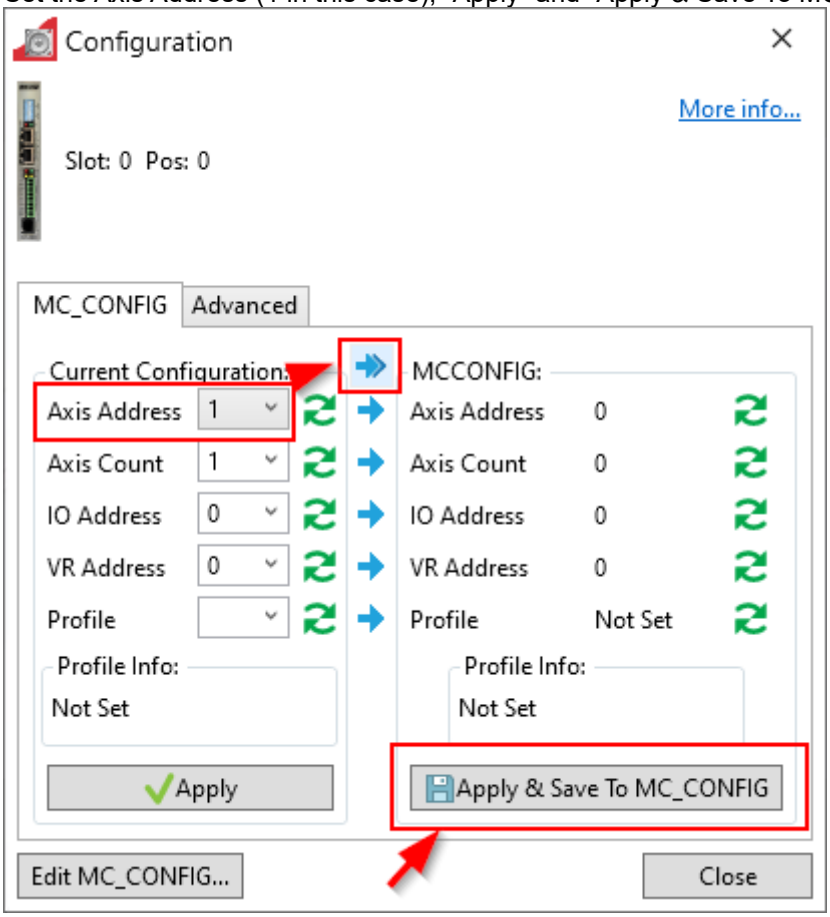



Close the window after successfully creating the EC_EXTEND file.

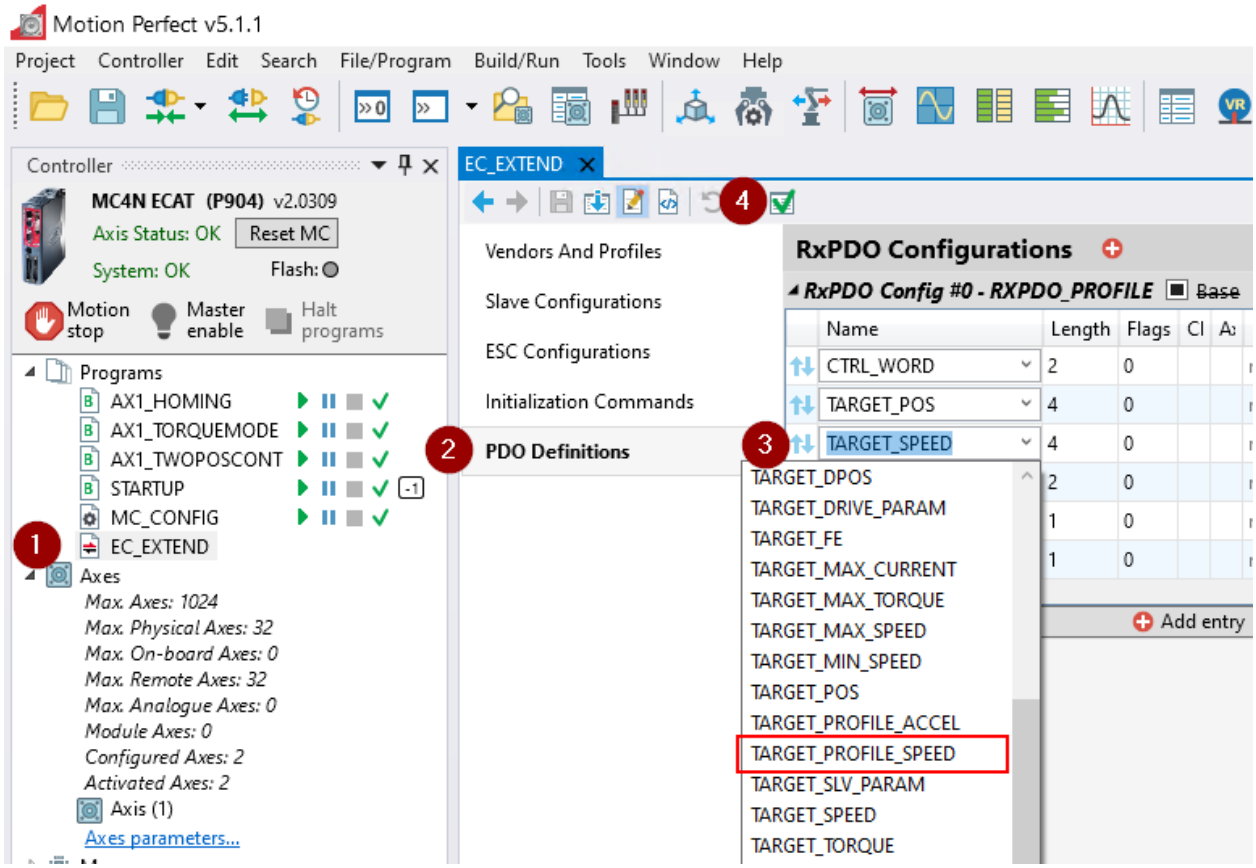
7. Right click on the LinMot device and select “Configure”:



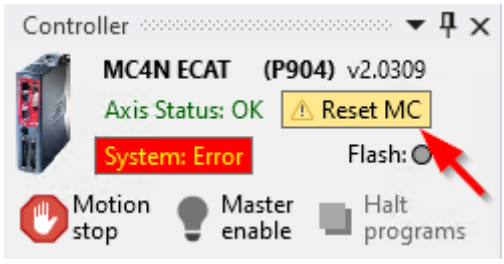

8. Set the Axis Address (1 in this case), “Apply” and “Apply & Save To MC_CONFIG”:



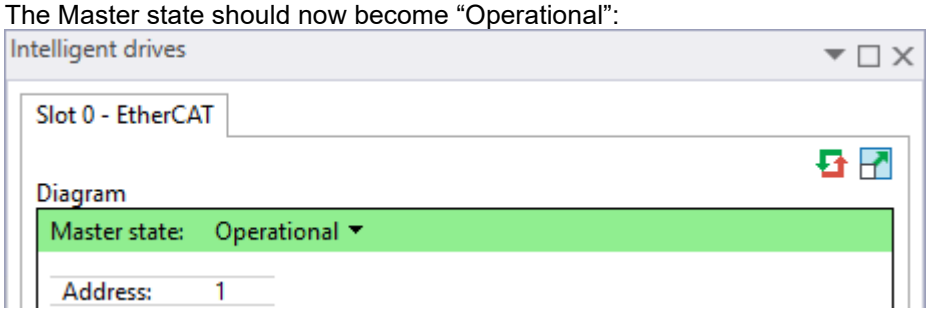
9. Open the EC_EXTEND file, click to enable editing, and assign the **TARGET_PROFILE_SPEED** instead of TARGET_SPEED in the RxPDO to have a smoother motion behaviour of the axis:




10. Reset the controller:




The Master state should now become “Operational”:



### 18.3.3 UNITS for Linear and Rotary Motors

To define the application units (mm, degrees, …) the drive parameters UNITS and ENCODER_RATIO can be used.

| Motor Type | Type | Application units | Counts per Rev | UNITS | ENCODER_RATIO |
|---|---|---|---|---|---|
| LinMot PS01 / PS02 | Linear | Millimetres [mm] | 10'000 | 10000 | - |
| LinMot RS01 (PR01 & PR02) | Rotary | Degrees [°] | 360'000 | 1000 | - |
| LinMot EC02 | Rotary | Degrees [°] | 524'288 | 1456* | (524160,524288)* |

***ENCODER_RATIO example for a EC02 motor (From the TrioBASIC help):**

Counts per rev: 524'288
Units per rev: 360 [°]

The highest value that is less than **524288** yet divides into **360** should be chosen. This is **524160** (524160 / **1456** = 360). This reduces the resolution from 0.00068664… to 0.00068681… degrees but enables you to program easily in degrees.

Therefor set the drive parameters as follows:

```
'LinMot EC02 with 524288 counts per motor rev.
ENCODER_RATIO(524160,524288)
UNITS = 1456 'AXIS calibrated IN degrees
```

## 18.4 Homing

There are several ways to home an axis.

The example project from chapter **18.1 Overview** contains a program called “AX1_Homing” that uses the DATUM and DEFPOS functions to implement a homing sequence to detect a hard stop.

```
'================================================================================
'Application based homing procedure (detect hard stop in negative direction)
'
'================================================================================

'Variables / Parameters
torque_limit = 2500 '=250.0% > Force/Torque Limit in 0.1% for hard stop detection,
'5000 = 500% = maximal force/torque of connected motor

home_position = 0.0 'Position that is set at hard stop
init_position = 10.0 'Position the axis moves to after detection

fe_limit_backup = 0.0 'Backup variable for FE_LIMIT


'Base: Axis 1
'================================================================================
BASE(1)

UNITS = 10000 'Set motor units. 10000inc = 1mm
'REP_DIST AXIS(axisx) = 360 ' make position modulo 0..360° for rotary axis
```

**Note:**
This is an application side homing sequence as TRIO does not support the DS402 drive-controlled homing mode (Mode of Operation 6) yet.

## 18.5 Parameter Access

Parameters can be modified using their UPIDs (Unique Parameter ID) via CoE.
To use a SDO service the **CO_READ_AXIS** and **CO_WRITE_AXIS** functions can be used (See TRIO Basic Help).

Syntax:

- **CO_READ_AXIS**(axis_number, index, subindex ,type [,vr_number])
- **CO_WRITE_AXIS**(axis_number, index, subindex, type, vr_number [,value])

The sub-index specifies the command which is performed

| SDO Service | CANopen Object index | CANopen Object sub-index | Description |
|---|---|---|---|
| Read | 2000h+UPID | 0x01 | Parameter UPID read RAM value |
| Write | 2000h+UPID | 0x01 | Parameter UPID write RAM value |
| Read | 2000h+UPID | 0x02 | Parameter UPID read ROM value |
| Write | 2000h+UPID | 0x02 | Parameter UPID write ROM value |
| Read | 2000h+UPID | 0x03 | Parameter UPID read minimal value |
| Read | 2000h+UPID | 0x04 | Parameter UPID read maximal value |
| Read | 2000h+UPID | 0x05 | Parameter UPID read default value |
| Write | 2000h+UPID | 0x06 | Parameter UPID write RAM and ROM value |
| Write | 2000h | 0x07 | Parameter Default OS SW instance |
| Write | 2000h | 0x08 | Parameter Default MC SW instance |
| Write | 2000h | 0x09 | Parameter Default Intf SW instance |
| Write | 2000h | 0x0A | Parameter Default Appl SW instance |
| Write | 2000h | 0x0B | Reset device |

**Attention ROM Access:**
Intense use of writing into the ROM memory can reduce the lifetime of the drive memory!
More details can be found in the *Drive Configuration Over Fieldbus (0185-1074)* user manual
> see chapter Documentation / User Manuals

Example Write:

To write 1A to the RAM value of the parameter Maximal Current:

| | | | |
|---|---|---|---|
| Name: | **Maximal Current** | Index | **= 16#2000 + 16#13a6 = 16#33a6** |
| UPID: | **13A6h** | Sub-index | **= 1** |
| Type: | **SInt32 = DINT = 4** | Value | **= 1A / 0.001A (scale) = 1'000** |
| Scale: | **0.001 A** | | |

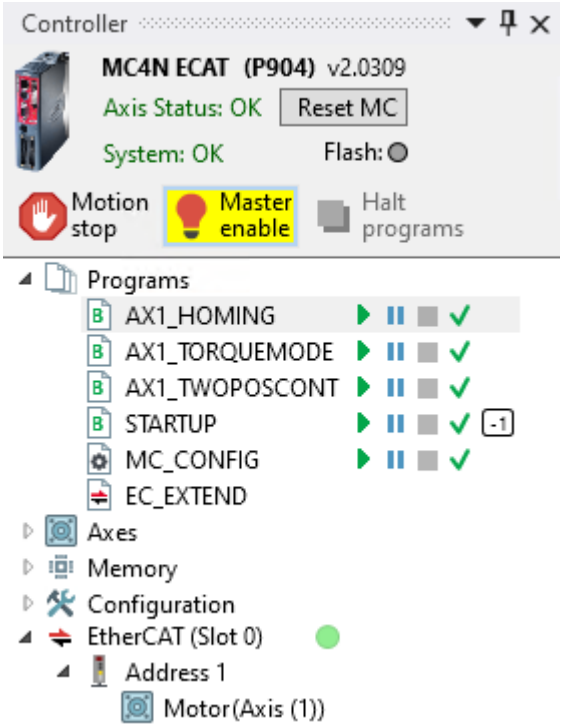
```
VR(200) = 1000 'Set Maximal Current UPID 13A6h to 1A = 1'000
CO_WRITE_AXIS(BASE, $33a6, 1, 4, 200)
```

Example Read:

```
'Read UPID 13A6 Maximal Current in Control Parameter Set A of the drive
CO_READ_AXIS(BASE, $33a6, 1 ,4 ,200)
motor_max_current_set_a = VR(200)
```

## 18.6 Examples

The example project from chapter **18.1 Overview** also contains programs to show a simple positioning cycle and a torque mode sequence.




Just set the correct BASE for the axis you want to use the program with:

```
'Base: Axis 1
'================================================================================
BASE(1)
```

# 19 Troubleshooting

The following points should be checked if there are any troubles getting the LinMot drive to connect to the fieldbus or using the LinMot PLC libraries.

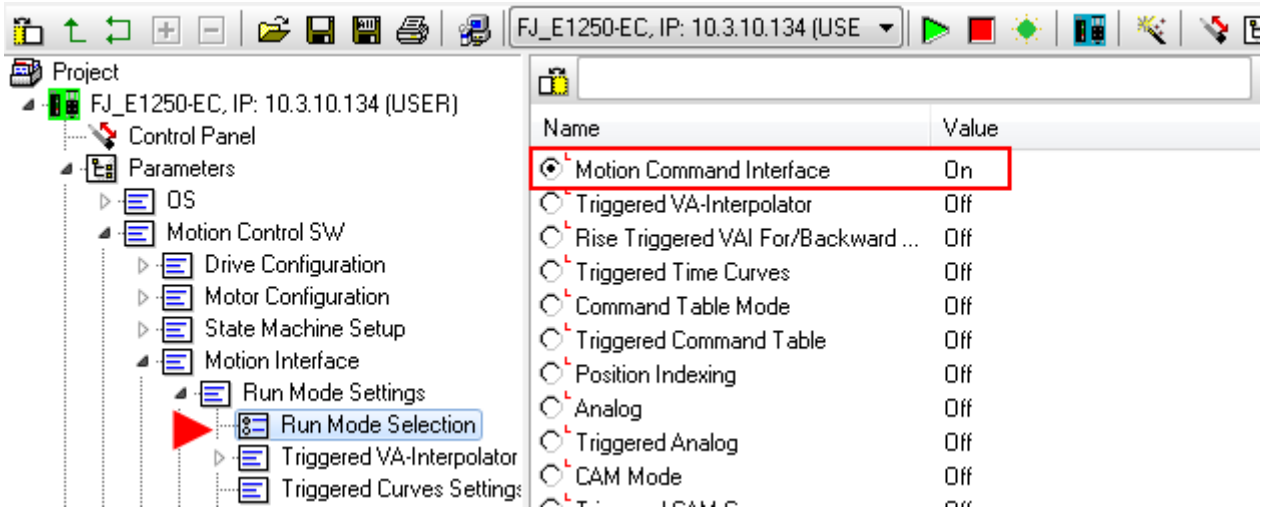
- Is the fieldbus connected to the correct connector of the LinMot drive?
  E.g., for EtherCAT or PROFINET X17 must be used as input (PLC → X17, X18 → next device)

- Is the address of the LinMot drive set correct? (Node address, MAC ID, IP)
  → Check the fieldbus variables inside LinMot-Talk

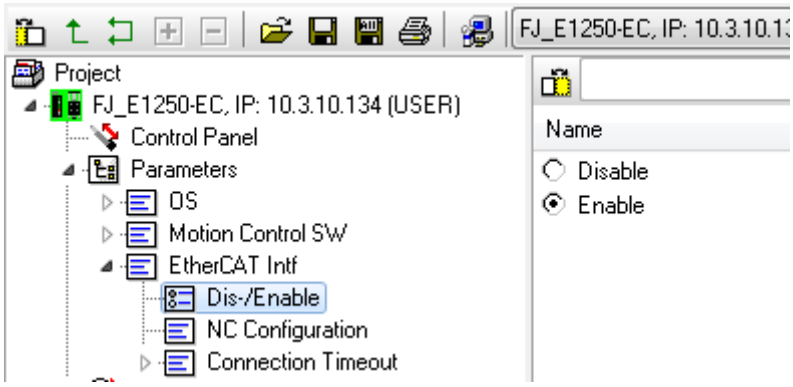
- Are all interface parameters on the drive set to the specified values of the library documentation?
  → Check with LinMot-Talk → Parameters → Interface

  By default, the LinMot libraries use factory default interface parameters on the LinMot drive.
  Except CANopen and LinRS.

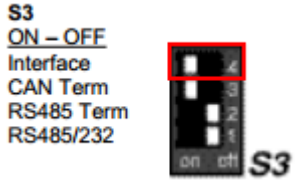
- Is the run mode of the drive set to “Motion Command Interface”?



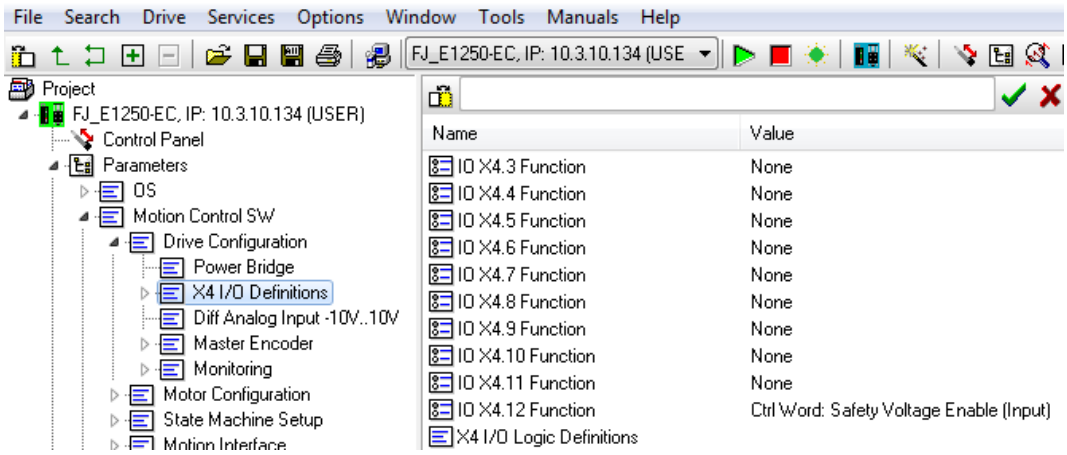

- Is the interface enabled?

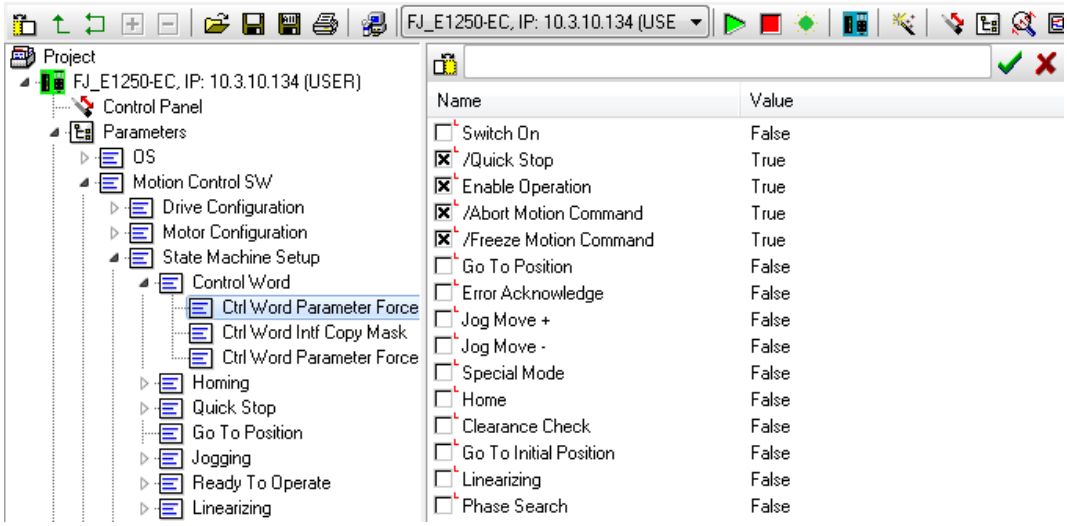



On E1100 series drives check if the interface dip switch is set to ON:



- Are there any digital inputs on X4 configured that are controlled by the PLC?
  Check if the I/O's are set correctly by the PLC or try to use the default settings and check if it works:

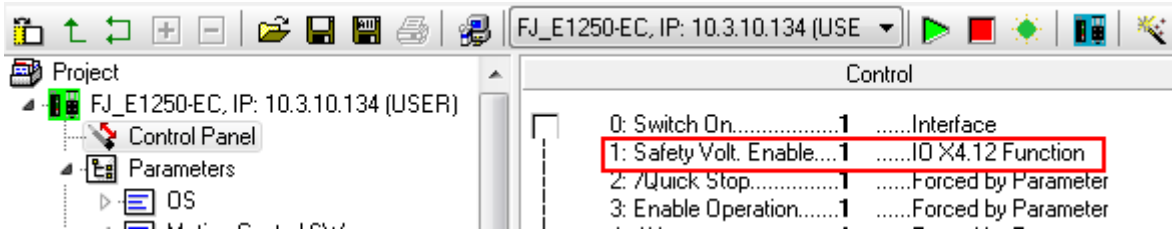



- If possible, use a clean configuration of the LinMot drive.
  1. Install / reinstall the firmware or set the drive to factory settings using LinMot-Talk
  2. Set up the motor using the motor wizard
  3. Set the fieldbus specific parameters (e.g., node address, IP, …)

  Now check again if the problem still exists

- Are the safety inputs powered before trying to switch on the motor?
  - Is the X4.12 input powered with 24V DC (SVE input on series E1100 and E1200 drives)?
  - Are the safety relays X33 powered (drives with -1S option)?



## Appendix I: Basic Position Control Loop Tuning

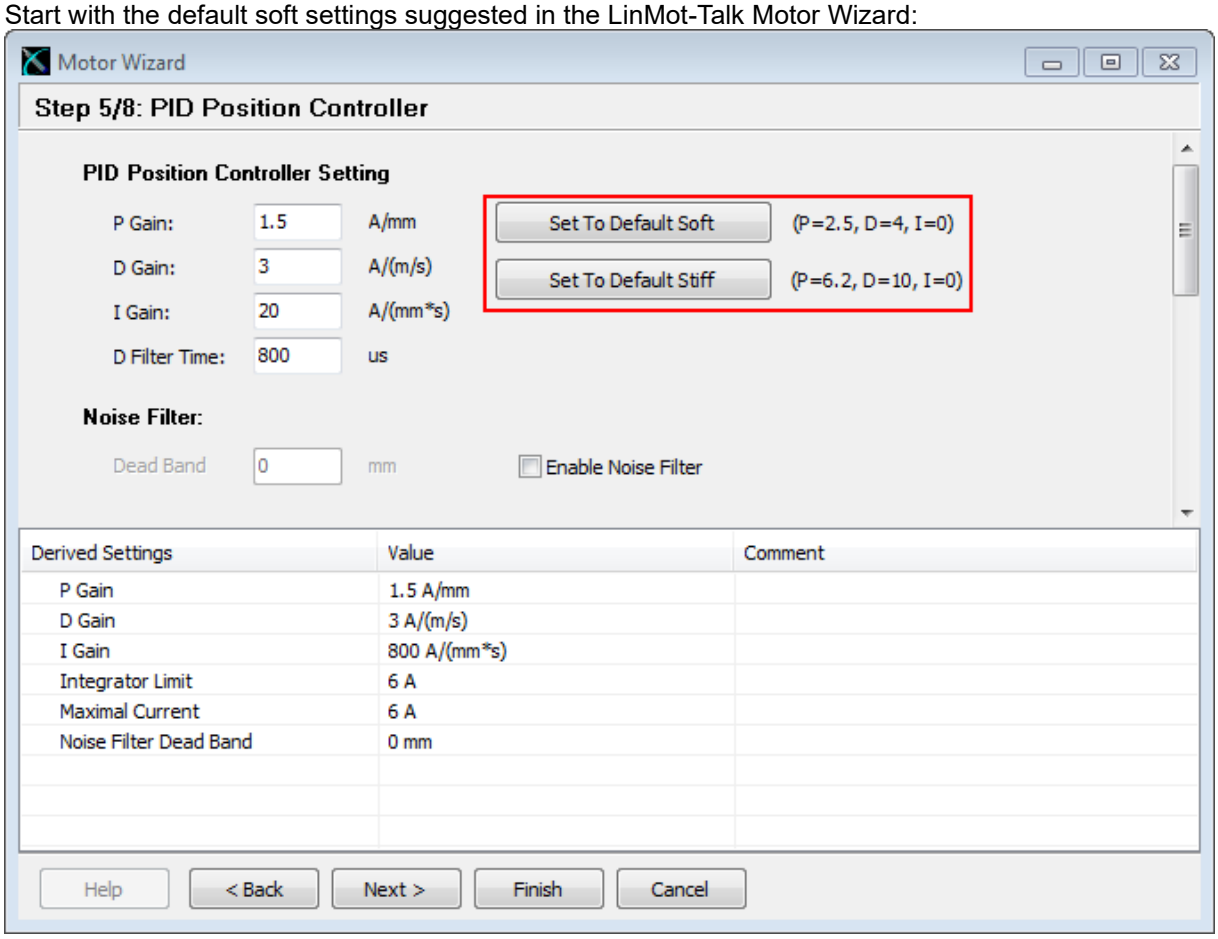
Tuning the position loop of a LinMot drive depends a lot on the application the motor is used for.

**Hint:** Further documentation

See application note about **position loop tuning**: 0185-1156 (EN / DE).
See application note about using the **LinMot oscilloscope**: 0185-0132 (EN / DE).

A basic setup of the position loop parameters could look as follows.

1. Start with the default soft settings suggested in the LinMot-Talk Motor Wizard:



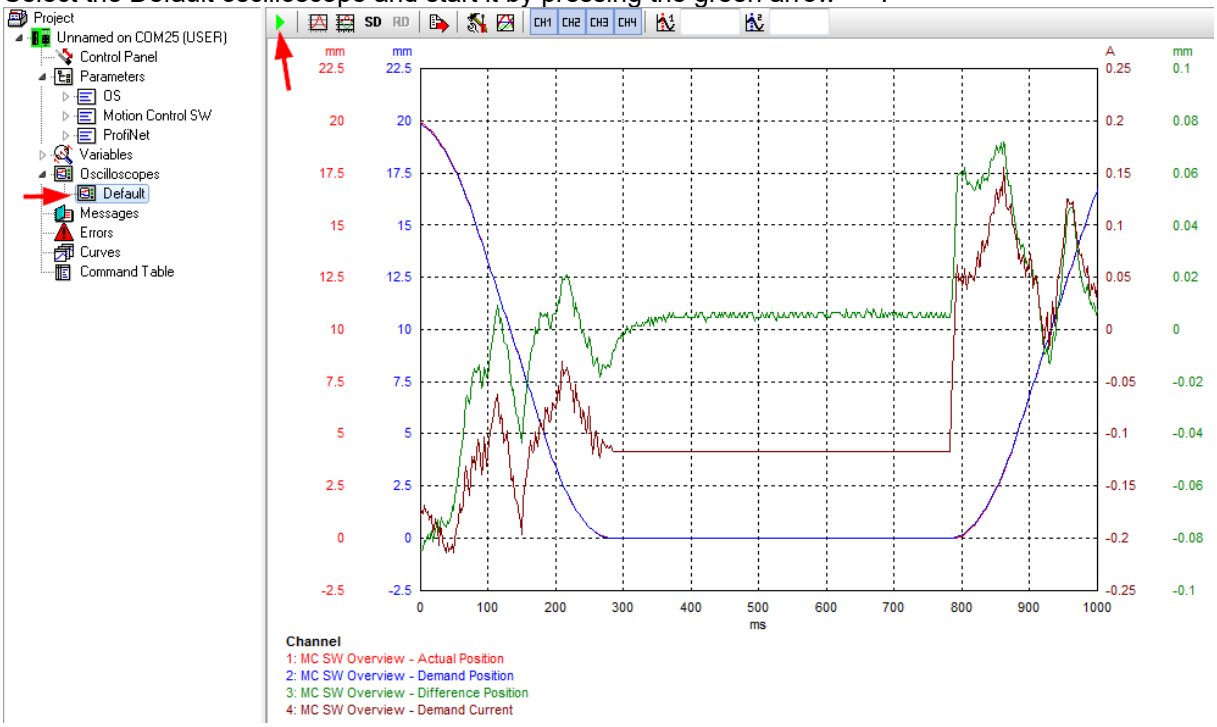

If the motor is mounted vertically or if a high accuracy at the target position is required set the *I Gain* of the position controller to a value between 20 to 50 A/(mm*s).

To reduce noise of the motor it is recommended to set the *D Filter Time* to 500us (C1200, C1100) or 800us (E1100, E1200)

It is highly recommended to disable the *Dead Band,* especially if a Drive Profile is used or in vertical applications. Uncheck *Enable Noise Filter*.

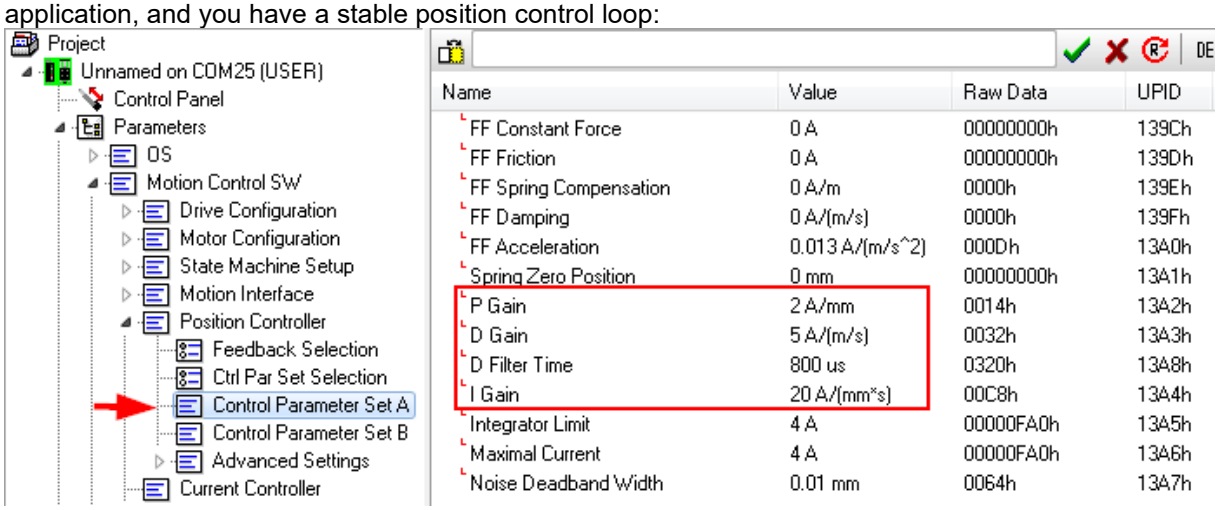
2. After finishing the Motor Wizard, rebooting the drive firmware, enabling, and homing the motor you can start moving the motor. With the built-in oscilloscope in LinMot-Talk you can trace the movement of the motor.

   Select the Default oscilloscope and start it by pressing the green arrow  .




   The green channel shows the *Difference Position* (Position Error) you may want to minimize.

3. Slowly increase the P, D and I Gains in the Control Parameter Set A until the accuracy fits your application, and you have a stable position control loop:




**Attention:**
When controlling the LinMot drive using a drive profile (e.g., SoE, CiA402, FSP_Drive, …) and velocity or acceleration jumps occur then try to set **all** FF … parameters (except *FF Constant Force*) in *Control Parameter Set A* to 0 (zero).

# Appendix II: Basic Closed Loop Torque/Force Control Loop Tuning

Tuning the closed loop torque/force control loop of a LinMot drive depends a lot on the application the motor is used for.

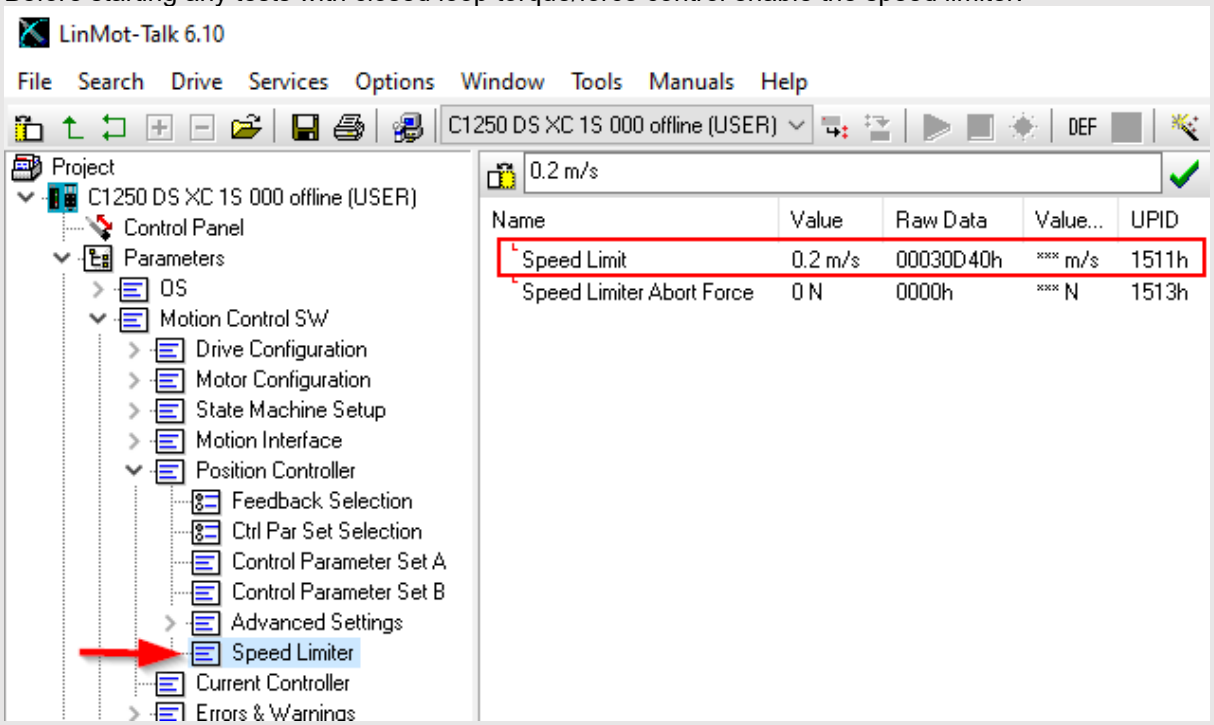
**Attention:**
To use closed loop force/torque control the technology function **TF-Force Control** (0150-2503) is required. Check the manual for more information: 0150-2503

**Attention:**
An application note about how to setup, tune and use closed loop torque/force control is available with document reference 0185-0144 (EN / DE).

Please follow that document to properly setup your system. This chapter here is thought to give a rough overview of where the parameters and variables related to closed loop torque/force control can be found.

Not following the above-mentioned document may lead to sever damage of your system.

**Attention:**
Before starting any tests with closed loop torque/force control enable the speed limiter!




This will help to protect you mechanics and sensors.

A basic setup of the torque/force control loop could look as follows:

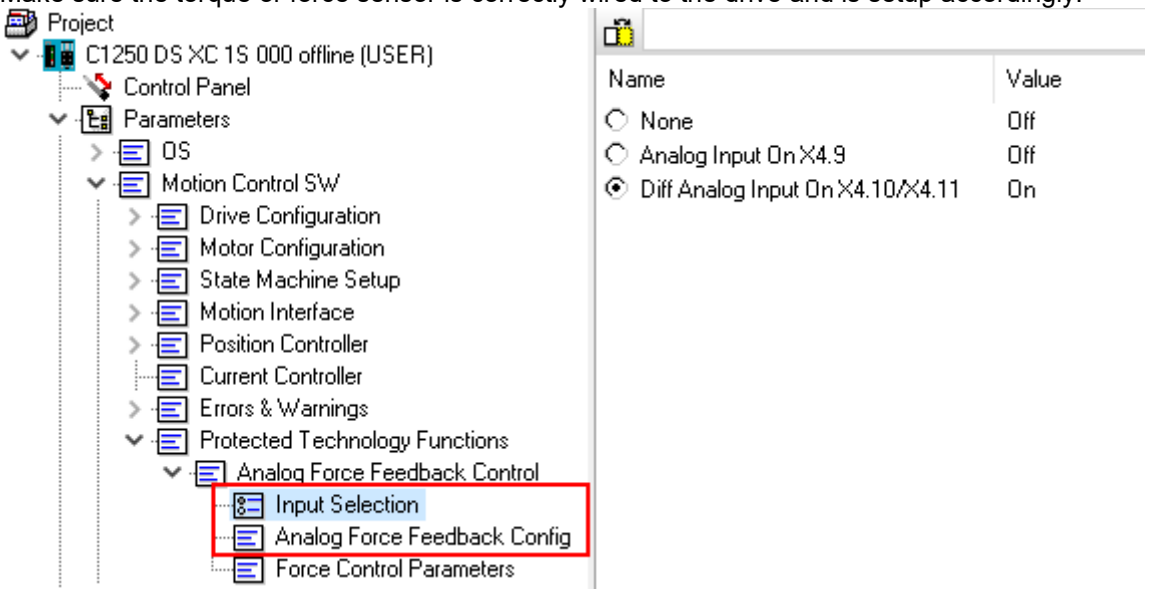
1. Enable and set the Speed Limiter as shown above
2. Make sure the *TF Force Control* license is activated (Drive > Set Access Code):

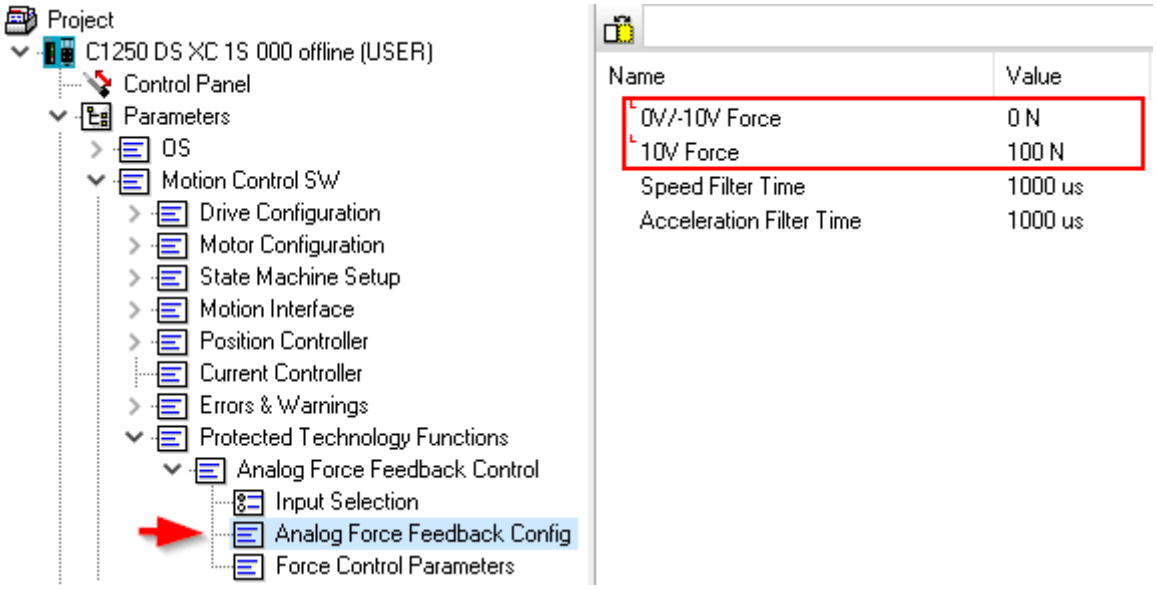
   Set Access Code

   **Active Keys:**

   | Name | Value | Access Code |
   |---|---|---|
   | Key1 | 5A93h - TF Force Control | 2BFD82D9h |
   | Key2 | 0000h - No Key | |

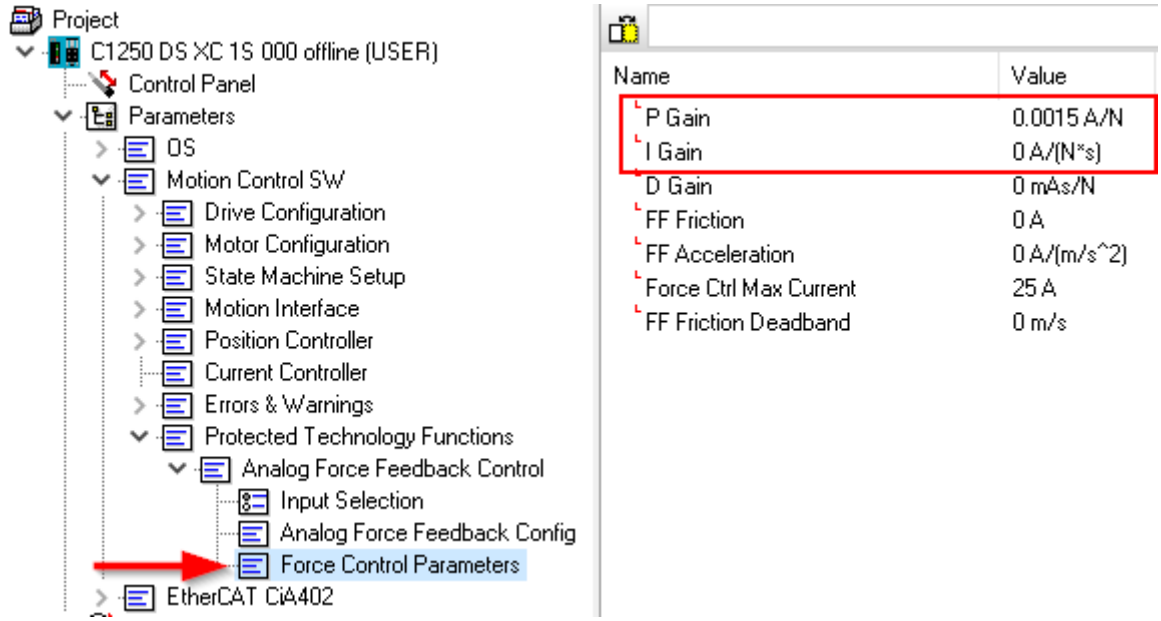
   If it is not activated please check the manual 0150-2503
3. Make sure the torque or force sensor is correctly wired to the drive and is setup accordingly:



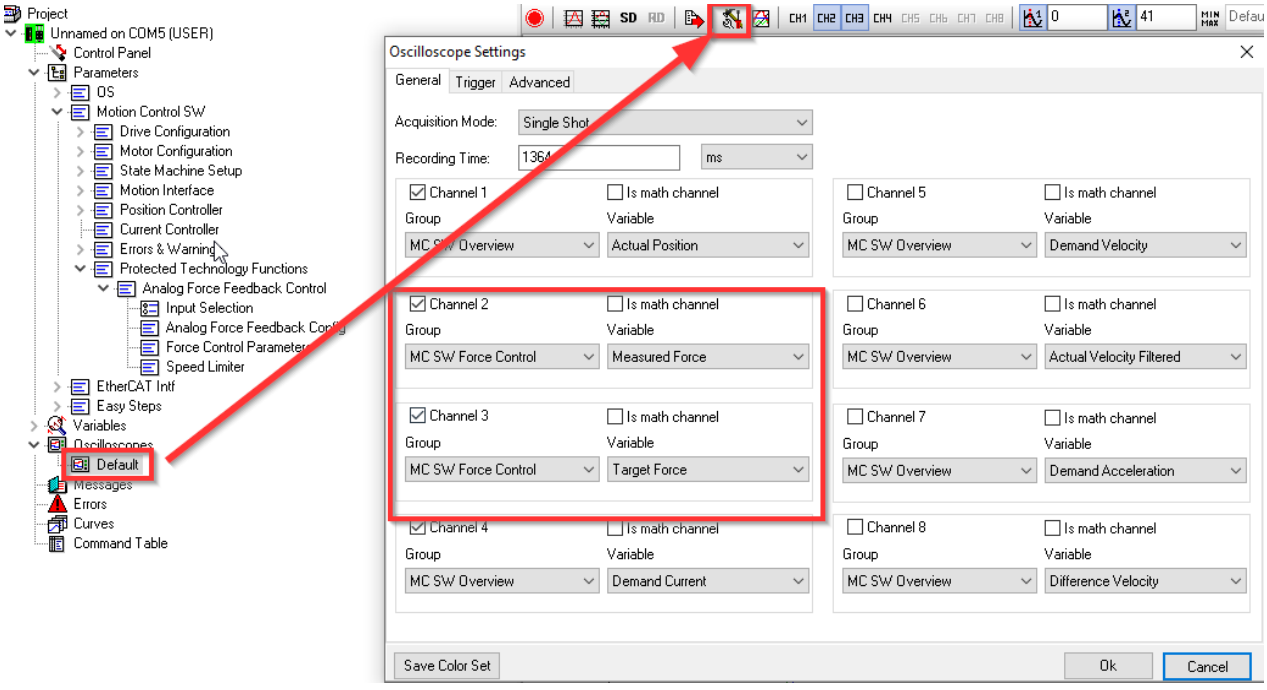



4. Start tuning the control loop. In most application it is enough to increase only slowly the I Gain.

   For linear motors (linear system of units) you can use 0.5 A/(N*s) as the step size for the I Gain, for rotary motors (rotary system of units) you can use 50 to 100 A/(Nm*s) as the step size.




5. Setup and start an oscilloscope (see manual 0185-0132 (EN / DE)) that contains the target and measured torque/force and start your process.




6. Check the result and repeat steps 4 and 5 until you reach the required control quality.

## Notes

## Contact & Support


**Europe / Asia Headquarters**

**NTI AG - LinMot & MagSpring**
Bodenaeckerstrasse 2
CH-8957 Spreitenbach
Switzerland

| | |
|---|---|
| Sales / Administration: | +41 56 419 91 91<br>office@linmot.com |
| Tech. Support: | +41 56 544 71 00<br>support@linmot.com |
| | +49 201 858940 10<br>support@linmot.de |
| Web: | https://www.linmot.com |

---

**North / South America Headquarters**

**LinMot USA Inc.**
N1922 State Road 120, Unit 1
Lake Geneva, WI 53147
USA

| | |
|---|---|
| Sales / Administration: | 262.743.2555<br>usasales@linmot.com |
| Tech. Support: | 262.743.2555<br>usasupport@linmot.com |
| Web: | https://www.linmot-usa.com |


---

Visit https://www.linmot.com/contact to find a distributor next to you.

Smart solutions are…