Automation Systems

Controller-based Automation

EtherCAT®

Communication manual EN

Lenze

Contents

Contents

Contents

Contents

1 About this documentation

This documentation ...

- contains detailed information about the commissioning, configuration, and diagnostics of the EtherCAT® bus system as part of the Lenze automation system "Controller-based Automation".
- is part of the "Controller-based Automation" manual collection. It consists of the following sets of documentation:

Documentation type	Subject
Product catalogue	Controller-based Automation (system overview, sample topologies) Lenze Controllers (product information, technical data)
System manuals	Visualisation (system overview/sample topologies)
Communication manuals **Online helps**	Bus systems • Controller-based Automation EtherCAT® • Controller-based Automation CANopen® • Controller-based Automation PROFIBUS® • Controller-based Automation PROFINET®
Reference manuals **Online helps**	Lenze Controllers: • Controller 3200 C • Controller c300 • Controller p300 • Controller p500
Software manuals **Online helps**	Lenze Engineering Tools: • »PLC Designer« (programming) • »Engineer« (parameter setting, configuration, diagnostics) • »VisiWinNET® Smart« (visualisation) • »Backup & Restore« (data backup, recovery, update)

More technical documentation for Lenze components

Further information on Lenze products which can be used in conjunction with Controller-based Automation can be found in the following sets of documentation:

Design / configuration / technical data	
☐	**Product catalogues** • Controller-based Automation • Controllers • Inverter Drives/Servo Drives
Installation and wiring	
🕮	**Mounting instructions** • Controllers • Communication cards (MC-xxx) • I/O system 1000 (EPM-Sxxx) • Inverter Drives/Servo Drives • Communication modules
☐	**Hardware manuals** • Inverter Drives/Servo Drives
Parameterisation / configuration / commissioning	
☐	**Online help/reference manuals** • Controllers • Inverter Drives/Servo Drives • I/O system 1000 (EPM-Sxxx)
☐	**Online help/communication manuals** • Bus systems • Communication modules
Sample applications and templates	
☐	**Online help / software manuals and reference manuals** • i700 application sample • Application Samples 8400/9400 • FAST Application Template • FAST technology modules

Symbols:

🕮 Printed documentation
☐ PDF file / online help in the Lenze Engineering tool

Tip!

Current documentation and software updates with regard to Lenze products can be found in the download area at:

www.lenze.com

Target group

This documentation is intended for persons who commission and maintain a Controller-based automation system by means of a Lenze Controller and the »PLC Designer« engineering tool.

Information regarding the validity

The information provided in this documentation is valid for the Lenze automation system "Controller-based Automation" from release 3.0.

Screenshots/application examples

All screenshots in this documentation are application examples. Depending on the firmware version of the field devices and the software version of the Engineering tools installed (e.g. »PLC Designer«), screenshots in this documentation may differ from the representation on the screen.

1.1 Document history

Version			Description
9.0	11/2020	TD06	Update for the "Controller-based Automation" 3.17 Lenze automation system • New: L_ETC_MMCReadAddress (FB) (📖 167) • General corrections
8.0	04/2019	TD06	New: Checking the wiring by means of the "Cable check" function (📖 60)
7.4	05/2017	TD17	New: Start parameters of the Inverter Drives 8400 motec (📖 111)
7.3	11/2016	TD17	Update for the Lenze automation system "Controller-based Automation" 3.14 • Chapter Controller-based Automation: Central motion control (📖 15)
7.2	05/2016	TD17	Update for the Lenze automation system "Controller-based Automation" 3.13 • Chapter Restarting the EtherCAT fieldbus (📖 179): Notes added.
7.1	10/2015	TD17	Update for the "Controller-based Automation" 3.12 Lenze automation system • Corrections in chapter Modular machine configuration (📖 114) • Messages: Invalid Input Configuration / Invalid Output Configuration (📖 199) in chapter Error scenarios (📖 195) supplemented.
7.0	05/2015	TD17	Update for the "Controller-based Automation" 3.10 Lenze automation system • Chapter Modular machine configuration (📖 114) new. • Chapter Error messages for modular machine configuration (📖 212) new
6.5	01/2015	TD17	Update for the Lenze automation system "Controller-based Automation" 3.9 • L_ETC_STATE (📖 178) supplemented. • Message: Invalid SyncManager Configuration (📖 198) in chapter Error scenarios (📖 195) supplemented.
6.4	04/2014	TD17	Revision on the Lenze automation system "Controller-based Automation" 3.8
6.3	11/2013	TD17	Update for the Lenze automation system "Controller-based Automation" 3.6
6.2	03/2013	TD17	Update for the Lenze automation system "Controller-based Automation" 3.5
6.1	12/2012	TD17	New function block L_ETC_GetEmergency (FB) (📖 152) in the "Controller-based Automation" 3.4 Lenze automation system
6.0	11/2012	TD17	• General corrections • New layout
5.0	08/2012	TD17	• Revision on the Lenze automation system "Controller-based Automation" 3.3 • Information on the Servo-Inverter i700 added. • Information on the ECS servo system and »GDC« removed.
4.2	12/2011	TD17	• Update for the Lenze automation system "Controller-based Automation" 3.2 • Chapter L_IODrvEtherCAT function library (📖 126) updated.
4.1	03/2011	TD17	• Special features for the ECS servo system added. • References to Lenze sample projects for EtherCAT logic field devices (device application + PLC program) added. ▸ Commissioning of the system (📖 40)
4.0	10/2010	TD17	Commissioning and configuration with the Lenze »PLC Designer« V3.x
3.0	10/2009	TD17	General update
2.0	05/2009	TD17	General update
1.0	09/2008	TD17	First edition

1.2 Conventions used

This documentation uses the following conventions to distinguish between different types of information:

Type of information	Highlighting	Examples/notes
Spelling of numbers		
Decimal	Normal spelling	Example: 1234
Decimal separator	Point	The decimal point is always used. For example: 1234.56
Hexadecimal	0x[0 ... 9, A ... F]	Example: 0x60F4
Binary • Nibble	0b[0, 1]	Example: '0b0110' Example: '0b0110.0100'
Text		
Program name	» «	PC software Example: Lenze »Engineer«
Window	*italics*	The *Message window*... / The dialog box *Options*...
Variable names		By setting *bEnable* to TRUE...
Control element	**bold**	The **OK** button... / The **Copy** command... / The **Properties** tab... / The **Name** input field...
Sequence of menu commands		If several commands must be used in sequence to carry out a function, the individual commands are separated by an arrow. Select **File → Open** to...
Shortcut	**<bold>**	Use **<F1>** to open the online help.
		If a shortcut is required for a command to be executed, a "+" has been put between the key identifiers: With **<Shift>+<ESC>** ...

Program code	`Courier`	`IF var1 < var2 THEN`
		`  a = a + 1`
		`  END IF`

Keyword	**`Courier bold`**	
Hyperlink	underlined	Optically highlighted reference to another topic. Can be activated with a mouse-click in this documentation.
Icons		
Page reference	(📖 9)	Optically highlighted reference to another page. Can be activated with a mouse-click in this documentation.
Step-by-step instructions		Step-by-step instructions are indicated by a pictograph.

1.3 Terminology used

Term	Meaning
Code	Parameter for parameterising or monitoring the field device. The term is also referred to as "index" in common usage.
CoE	CANopen over EtherCAT
Controllers	The Controller is the central component of the Lenze automation system which controls the motion sequences by means of the operating system. The Controller communicates with the field devices (inverters) via the fieldbus.
DC	Distributed clocks (distributed synchronised clocks)
Engineering PC	The Engineering PC and the Engineering tools installed serve to configure and parameterise the system "Controller-based Automation". The Engineering PC communicates with the controller via Ethernet.
Engineering tools	Software solutions for easy engineering in all phases which serve to commission, configure, parameterise and diagnose the Lenze automation system. ▸ Lenze Engineering tools (📖 30)
ETG	EtherCAT Technology Group (EtherCAT user organisation)
FAST	The **Lenze FAST** is installed on the Lenze Controller by default in the **"FAST Runtime"** version with **"FAST Motion"** for the central control of PLC applications.
FB	Function block (contained in a function library)
Fieldbus node	Devices integrated in the bus system as, for instance, Controller and inverter
Field device	
FoE	File Access over EtherCAT
FUN	Function (contained in a function library)
Index	Each object has a unique index for addressing purposes. In this documentation the index is represented as a hexadecimal value and is identified by a prefixed "0x", e.g. "0x1000".
Inverter	Generic term for Lenze frequency inverters, servo inverters
PLC	Programmable Logic Controller
Object	"Container" for one or more parameters with which you can parameterise or monitor the fieldbus node.
PLC	Programmable logic controller (PLC)
Subcode	If a code contains several parameters, they are stored in "subcodes". This manual uses a slash "/" as a separator between code and subcode (e.g. "C00118/3"). Is usually referred to as "subindex".
Subindex	If a code contains several parameters, they are stored in so-called "subindexes". In this documentation, the slash "/" is used as a separator between an index and subindex, e.g. "0x1018/1".
Bus systems	
CAN	CAN (Controller Area Network) is an asynchronous, serial fieldbus system.
CANopen	CANopen® is a CAN-based communication protocol. The Lenze system bus (CAN on board) works with a subset of this communication protocol. CANopen® is a registered community trade mark of the CiA® (CAN in Automation e. V.) CAN user organisation.
EtherCAT	EtherCAT® (**Ether**net for **C**ontroller and **A**utomation **T**echnology) is an Ethernet-based fieldbus system which fulfils the application profile for industrial real-time systems. EtherCAT® is a registered trademark and patented technology, licenced by Beckhoff Automation GmbH, Germany.
ETHERNET	Ethernet specifies the software (protocols) and hardware (cables, plugs, etc.) for wired data networks. In the form of "Industrial Ethernet", the Ethernet standard is used in industrial production systems. On the basis of IEEE 802.3, standard Ethernet is specified by the Institute of Electrical and Electronics Engineers (IEEE), USA.

Term	Meaning
	EtherNet/IP™ (EtherNet Industrial Protocol) is an Ethernet-based fieldbus system that uses Common Industrial Protocol™ (CIP™) to exchange data. EtherNet/IP™ and Common Industrial Protocol™ (CIP™) are brand labels and patented technologies, licensed by the ODVA (Open DeviceNet Vendor Association) user organisation, USA.
	PROFIBUS® (Process Field Bus) is a commonly used fieldbus system for automating machines and production plants. PROFIBUS® is a registered trademark and patented technology licensed by the PROFIBUS & PROFINET International (PI) user organisation.
	PROFINET® (Process Field Network) is a real-time capable fieldbus system based on Ethernet. PROFINET® is a registered trademark and patented technology licensed by the PROFIBUS & PROFINET International (PI) user organisation.

1.4 Definition of the notes used

The following signal words and symbols are used in this documentation to indicate dangers and important information:

Safety instructions

Layout of the safety instructions:

Pictograph and signal word!

(characterise the type and severity of danger)

Note

(describes the danger and gives information about how to prevent dangerous situations)

Pictograph	Signal word	Meaning
	Danger!	**Danger of personal injury through dangerous electrical voltage** Reference to an imminent danger that may result in death or serious personal injury if the corresponding measures are not taken.
	Danger!	**Danger of personal injury through a general source of danger** Reference to an imminent danger that may result in death or serious personal injury if the corresponding measures are not taken.
STOP	**Stop!**	**Danger of property damage** Reference to a possible danger that may result in property damage if the corresponding measures are not taken.

Application notes

Pictograph	Signal word	Meaning
	Note!	Important note to ensure trouble-free operation
	Tip!	Useful tip for easy handling
		Reference to another document

2 Safety instructions

Please observe the safety instructions in this documentation when you want to commission an automation system or a plant with a Lenze Controller.

The device documentation contains safety instructions which must be observed!

Read the documentation supplied with the components of the automation system carefully before you start commissioning the Controller and the connected devices.

Danger!

High electrical voltage

Injury to persons caused by dangerous electrical voltage

Possible consequences

Death or severe injuries

Protective measures

Switch off the voltage supply before working on the components of the automation system.

After switching off the voltage supply, do not touch live device parts and power terminals immediately because capacitors may be charged.

Observe the corresponding information plates on the device.

Danger!

Injury to persons

Risk of injury is caused by ...

- unpredictable motor movements (e.g. unintended direction of rotation, too high velocities or jerky movement);
- impermissible operating states during the parameterisation while there is an active online connection to the device.

Possible consequences

Death or severe injuries

Protective measures

- If required, provide systems with installed inverters with additional monitoring and protective devices according to the safety regulations valid in each case (e.g. law on technical equipment, regulations for the prevention of accidents).
- During commissioning, maintain an adequate safety distance to the motor or the machine parts driven by the motor.

Stop!

Damage or destruction of machine parts

Damage or destruction of machine parts can be caused by ...

- Short circuit or static discharges (ESD);
- unpredictable motor movements (e.g. unintended direction of rotation, too high velocities or jerky movement);
- impermissible operating states during the parameterisation while there is an active online connection to the device.

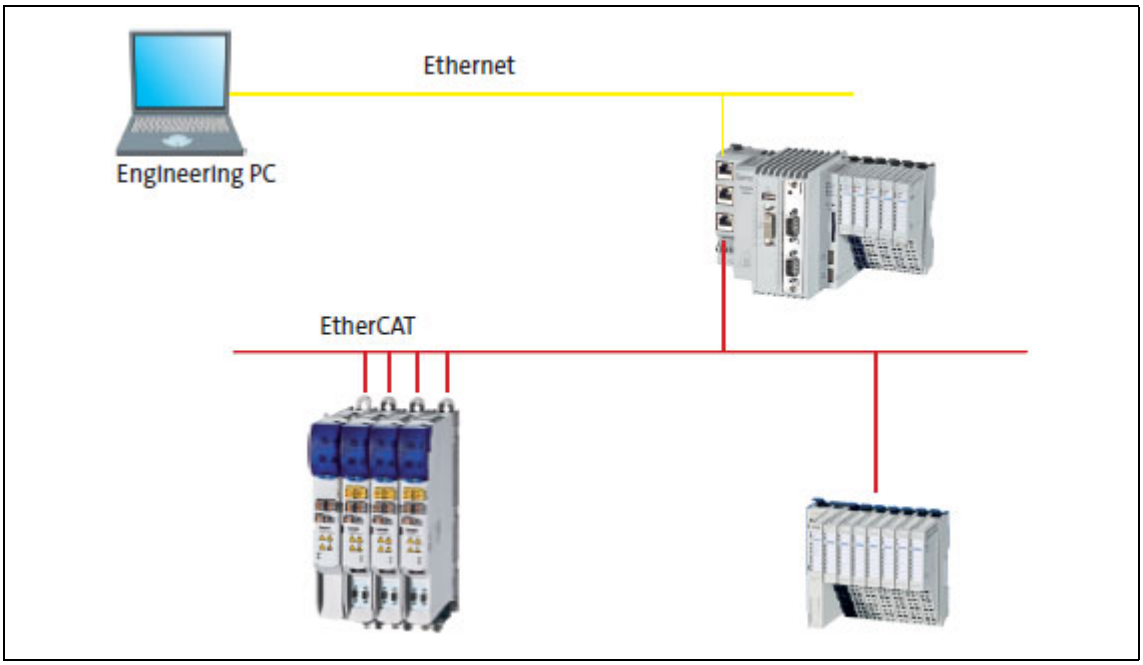
Protective measures

- Always switch off the voltage supply before working on the components of the automation system.
- Do not touch electronic components and contacts unless ESD measures were taken beforehand.
- If required, provide systems with installed inverters with additional monitoring and protective devices according to the safety regulations valid in each case (e.g. law on technical equipment, regulations for the prevention of accidents).

3 Controller-based Automation: Central motion control

The Lenze "Controller-based Automation" system serves to create complex automation solutions with central motion control. Here, the Controller is the control centre of the system.

System structure of Controller-based Automation



[3-1] **Example:** EtherCAT with Controller 3231 C, I/O system 1000 and Servo-Inverter i700

Lenze provides especially coordinated system components:

- Engineering software
  The Lenze Engineering tools (📖 30) on your Engineering PC (Windows® operating system) serve to parameterise, configure and diagnose the system. The Engineering PC communicates with the Controller via Ethernet.
  The Lenze engineering tools are provided for download at:
  www.lenze.com → **Download → Software Downloads**
- Controllers
  The Lenze Controller is available as Panel Controller with integrated touch display and as Cabinet Controller in control cabinet design.
  Cabinet Controllers provide a direct coupling of the I/O system 1000 via the integrated backplane bus.
- Bus systems
  EtherCAT is the standard "on-board" bus system of the Controller-based Automation. EtherCAT enables the control of all nodes on one common fieldbus.
  Optionally, CANopen, PROFIBUS and PROFINET can be used as extended topologies.
  For Controllers 3200 C and p500, the Ethernet interfaces also allow for the use of EtherNet/IP.
  Controllers c300 and p300 are provided with an "on board" CANopen interface (in addition to EtherCAT).
- Inverter (e.g. Servo-Inverter i700)

"Application software" of the Lenze Controllers

The "application software" of the Lenze Controllers enables the control and/or visualisation of motion sequences.

FAST technology modules provide for an easy development of a modular machine control in the »PLC Designer«.

The following "Application software" versions are provided:

- "FAST Runtime"
  Sequence control takes place in the Controller (by means of logically linked control signals).
  Motion control takes place in the inverter.
- "FAST Motion"
  Sequence control and motion control take place in the Controller.
  The inverter solely serves as an actuating drive.
  Motion applications place special requirements on the cycle time and real-time capability of the bus system between the Controller and the subordinate fieldbus nodes. This is for instance the case if the nodes are to be traversed with each other in a synchronised fashion, or if position setpoints are to be transmitted.
- "Visualisation"
  The optional visualisation of the automation system can be used separately or in addition to "FAST Runtime" or "FAST Motion".
  For this purpose, an external monitor panel/display can be connected to the Cabinet Controller 3231 C/3241 C/3251 C.

Fieldbus communication

The Lenze controllers have different interfaces for fieldbus communication:

Range	Cabinet Controller		Panel Controller	
	c300	3200 C series	p300	p500
Interfaces (on board)				
Ethernet	1	2	1	2
EtherNet/IP	-		-	
EtherCAT	1 [1)]	1	1 [1)]	1
CANopen	1	-	1 [2)]	-
Optional interfaces (communication cards)				
CANopen MC-CAN2	-	●	-	● [2)]
PROFIBUS master MC-PBM	-	●	-	●
PROFIBUS slave MC-PBS	-	●	-	●
PROFINET device MC-PND	●	●	●	●
Ethernet MC-ETH	-	●	-	●
Serial interfaces MC-ISI	-	●	-	●

1) Only the master functionality is supported.

2) Up to release 3.9: "EL 100 CAN" driver / from release 3.10: "Lenze CAN driver"

Ethernet interface

The Ethernet interface serves to connect the Engineering PC or to create line topologies (no integrated switch for Controller c300/p300).

With Controllers 3200 C and p500, EtherNet/IP communication is also established via the Ethernet interfaces.

4 The Lenze automation system with EtherCAT

This chapter provides basic information about ...

- the structure of the Lenze automation system using the EtherCAT bus system;
- the Lenze Engineering tools required for commissioning;
- the interaction of the components.

4.1 Brief description of EtherCAT

Tip!

Detailed information on EtherCAT is provided on the Internet website of the EtherCAT Technology Group:

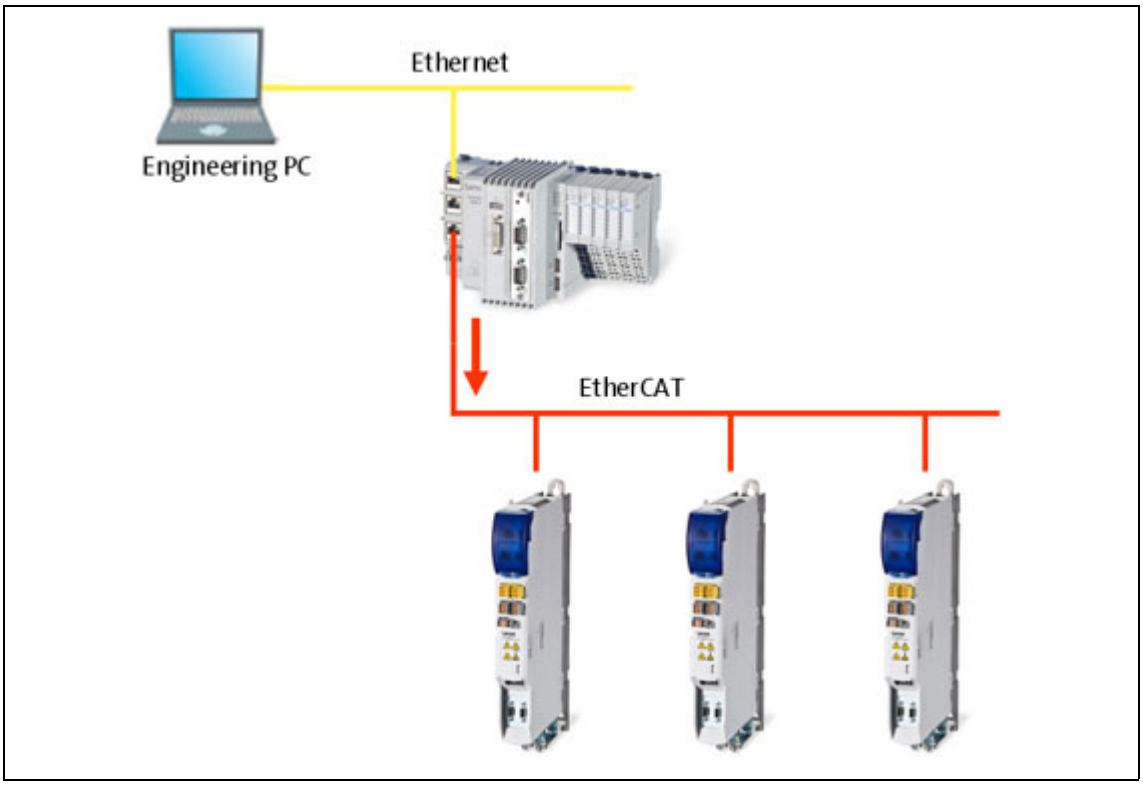
www.ethercat.org

Product features

- EtherCAT is a powerful bus system which is based on Ethernet.
- Due to its integrated synchronisation mechanisms based on "distributed clocks", EtherCAT possesses outstanding real time characteristics.
  ▸ Synchronisation with "Distributed clocks" (DC) (📖 37)
- EtherCAT provides a higher bandwidth compared to CANopen:
  - This enables motion and logic applications to be operated by the same fieldbus.
  - The number of the nodes to be controlled is higher.
  - The maximally possible bus length is longer.
- EtherCAT can access all field devices via a common interface. Therefore, unlike for the Lenze CANopen control technology, a division into Logic bus and Motion bus is not required.
- The "Modular Device Profile for IPC" (MDP) is based on the "Modular Device Profile Specification" of the EtherCAT Technology Group. All (software and hardware) components of the Lenze Controller or Embedded PC are divided into modules. The list of the modules available is generated dynamically, depending on the physically available components. The Lenze I/O system 1000 with the EPM-S130 head end supports the "Modular Device Profile".

4.1.1 Structure of the EtherCAT bus system

Basic structure



[4-1] **Example:** EtherCAT bus system with 3231 C controller and i700 servo inverter

Physical structure

An EtherCAT master can communicate with one or more nodes (slaves).

Internally, the EtherCAT bus has a ring topology. Since Ethernet cables are provided with a feed conductor and a return conductor within one cable, for the installer the topology seems to be a line. The last slave closes the ring.

Switches, hubs, or other infrastructure components known from the Ethernet standard must not be used because they impair the real-time performance.

4.1.2 communication

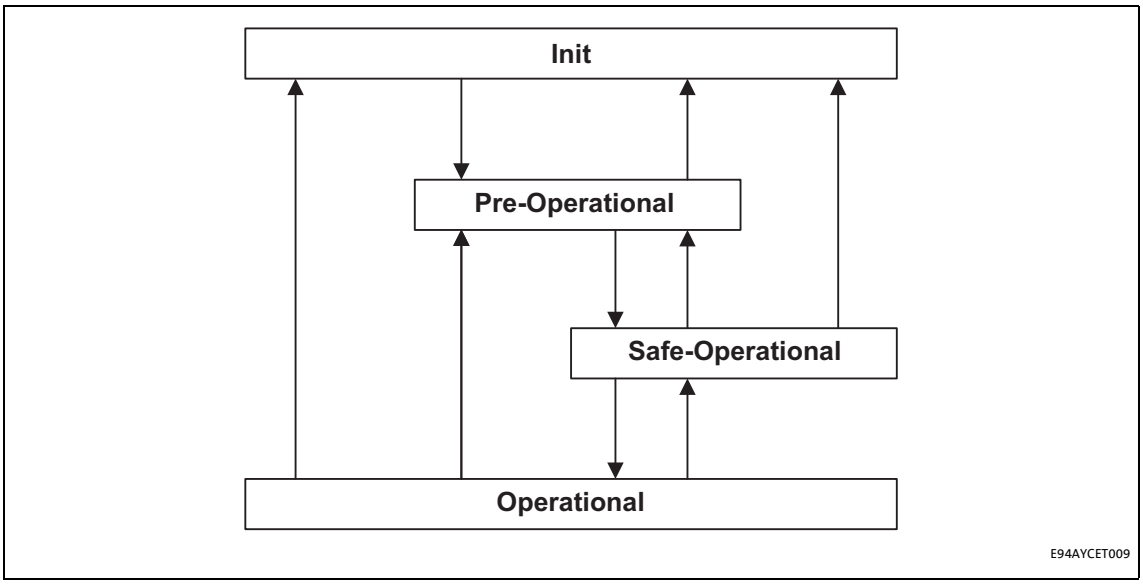
Compared with conventional Ethernet, the collision-free transfer of telegrams on the fieldbus makes EtherCAT a real-time capable bus system.

Communication is always initiated by the EtherCAT master, i.e. the Lenze Controller. A telegram sent by the master passes through all EtherCAT slaves. The last slave of the communication chain sends the telegram back to the EtherCAT master. On the way back, the telegram is directly sent to the EtherCAT master, without being processed in the slaves.

When EtherCAT is used, data are transferred in so-called "EtherCAT frames". The fieldbus nodes only remove the data intended for them and do so while the EtherCAT frame is passing through the device. Output data are inserted into the frame at the same time. Read and write access is always carried out on a small section of the overall EtherCAT frame – the datagrams. This means that a frame does not have to be received completely before it is processed. Each datagram is passed on with minimal delay.

4.1.2.1 The EtherCAT state machine

Before communication via EtherCAT is possible, the fieldbus is scanned by the EtherCAT state machine when the installation is being powered up. The following illustration shows the possible status changes from the point of view of an EtherCAT slave.



[4-2] EtherCAT state machine

Status	Description
Init	• Initialisation phase • No SDO/PDO communication with the slaves • Device can be detected by fieldbus scan
Pre-Operational	• The fieldbus is active. • SDO communication (mailbox communication) is possible. • No PDO communication

Safe-operational	• SDO communication (mailbox communication) is possible.
	• PDO communication:
	  • The input data in the process image are updated.
	  • The output data from the process image are not transferred to the slaves.

Operational	Normal operation for the master and the slaves, i.e. • SDO communication • PDO communication In this state, the slaves or the master are in the "Operational State". In this state the slaves and the EtherCAT BusCycleTask are synchronised with the DC distributed clock. • If synchronism could not be achieved, the machine is in "out-of-snc". ▸ Synchronisation with "Distributed clocks" (DC) (📖 37)

Note!

- The master and also the slaves remain in the **Operational state** even if the *xDistributedClockInSync* flag is set to "FALSE" due to the drift of the DC clocks.
  ▸ L_ETC_GetMasterDiagnostic (FB) (📖 155)
- A fieldbus scan is possible in any EtherCAT state.
  ▸ Determining the physical EtherCAT configuration (fieldbus scan) (📖 82)

AL Status Code

Possible errors during transitions between states are entered in the EtherCAT register "AL Status Code" (address 0x0134:0x0135).

AL Status Code [hex]	Description
0x0000	No error
0x0011	Invalid status change requested
0x0012	Unknown status requested
0x0013	"Bootstrap" status is not supported
0x0016	Invalid mailbox configuration "Pre-operational"
0x001A	Synchronisation error
0x001B	Sync manager watchdog
0x001D	Invalid output data configuration
0x001E	Invalid input data configuration
0x002B	Invalid input and output data
0x0030	Invalid configuration of DC synchronisation
0x9001	Firmware watchdog error
0x9002	Mapping error

4.1.2.2 Addressing of the slaves

The EtherCAT system uses two types of addressing for the slaves:

Auto-increment addressing

The auto-increment addressing is used by the master during the initialisation phase of the fieldbus. When the "Pre-Operational" state has been reached, the master uses the Fixed-Address addressing.

Fixed-address addressing

With the fixed-address addressing, the slaves are addressed via the station address distributed by the master during the start-up phase.

In the EtherCAT bus topology in »PLC Designer«, the first slave is given the address ’1001’, the second the address ’1002’ and so on. The EtherCAT addresses cannot be changed.

The EtherCAT address of the master is ’0’. Access to master objects with the address ’0’ is possible.

Example

The first slave of a configuration is given the following addresses ...

- ’0’ due to the automatic incrementation procedure;
- ’1001’ due to the fixed addressing procedure.

4.1.2.3 Working counter

Each EtherCAT datagram contains a working counter (WKC) which is incremented by each slave after the data have been processed successfully.

The working counter (WKC) can be used as a diagnostics option to check the processing of the EtherCAT telegrams by the slaves.

In each cycle, the Lenze Controller compares the expected value of the working counter with the value read back via the fieldbus. If the read-back value is smaller than the expected value, not all addressed slaves have been reached. The controller detects this and reports an error.

▸ Messages: WKC Error / Not all slaves "Operational" / SyncManager Watchdog (📖 198)

Example

- 10 slaves read/write process data in the "Operational" state
  Expected value of the WKC: 10
- A cable break between the 8th and 9th slave causes the master to be unable to access slave 9 and slave 10:
  - Value of the restored WKC: 8
  - An error response is initiated in the Lenze Controller.
  - The EtherCAT bus changes to the state "Pre-Operational".

4.2 Required hardware components

4.2.1 Field devices

The Lenze automation system supports the following EtherCAT-capable logic and motion components:

Field devices		EtherCAT	
		Logic	Motion
Controllers	Controller 32xx C	●	●
	Controller c300	●	●
	Controller p300	●	●
	Controller p500	●	●
Servo-Inverter i700	Single axis		●
	Double axis		●
Servo Drives 9400 *1)*	HighLine	●	
	HighLine with CiA402	●	●
	PLC	●	
	regenerative power supply module commissioning guidelines	●	
Inverter Drives 8400 *2)*	StateLine	●	
	HighLine	●	
	TopLine	●	
I/O system 1000	EPM-Sxxx	●	

1) With EtherCAT E94AYCET communication module

2) With EtherCAT E84AYCET communication module

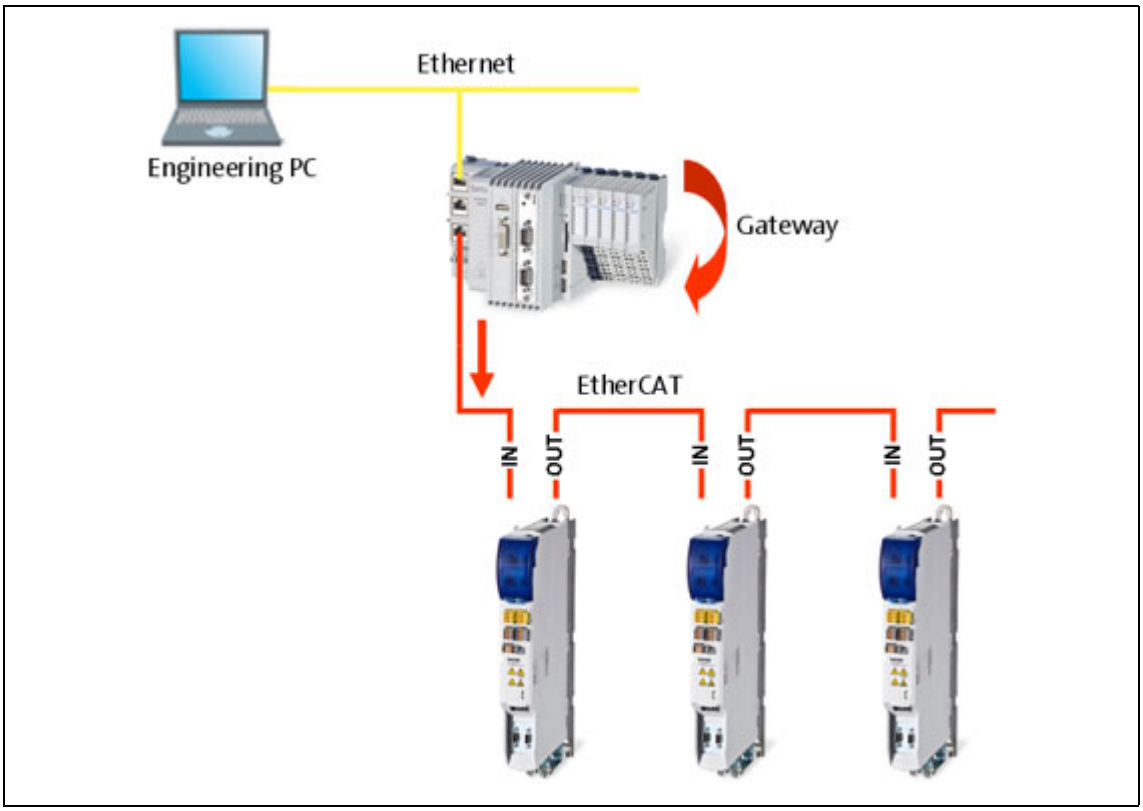
Note!

EPM-S130 (EtherCAT) **bus coupler module**

Only **EPM-Sxxx I/O compound modules from hardware version 1B onwards** are supported.

Field devices of other manufacturers can be integrated as Logic nodes if they provide a standard-compliant EtherCAT device description.

4.2.2 The Lenze Controller - the central component



[4-3] **Example:** EtherCAT bus system with 3231 C controller as gateway and i700 servo inverter

The Lenze Controller is the central component in the EtherCAT bus system:

- The controller is the EtherCAT master.
- The Lenze Controllers have an EtherCAT interface "on-board".
- The controller acts as an EtherCAT gateway in order to enable access to the field devices from the Engineering PC via Ethernet and EtherCAT.
- The devices are interconnected successively in line. For correct operation, it is necessary that the physical sequence of the EtherCAT field devices matches the bus topology created in »PLC Designer«.

  Otherwise the system will not become "Operational". (An error message indicates which slave (product code) is expected at what position.)
- Each EtherCAT slave has two EtherCAT ports.

  In contrast to Ethernet, one port is assigned as input, the other one as output.

  The inputs (IN) and outputs (OUT) must be correctly wired to each other
- A bus termination at the last slave is not required since the bus system at the last slave is terminated automatically.

4.2.3 EtherCAT product codes

Device descriptions can be assigned to the corresponding devices with the help of the product codes. In »PLC Designer«, you can install device descriptions with the menu command **Tools → Device repository...**.

▸ Importing missing devices / device description files (📖 85)

Structure of the device ID: <Manufacturer-ID>_<Product code><Revision number>

Identification	Meaning
Manufacturer ID	Unique identification for the manufacturer, for Lenze devices: 0x3B (59)
Product code	Product code of the product range/the device
Revision number	Revision number

If, for instance, a device available at the fieldbus without an installed device description is detected during a Determining the physical EtherCAT configuration (fieldbus scan) (📖 82) , a message with the device ID as hexadecimal value is displayed:

In this example, the device description for a Lenze Servo Drive 9400 HighLine, actuating drive - speed, is not installed (0x38079CD9 = 940023001).

Product codes for Servo Drives 9400

Product code [dec]									Meaning
9	4	0	0	2	1	x	x	x	Servo Drive 9400 in general
9	4	0	0	2	2	x	x	x	Servo Drive 9400 StateLine
9	4	0	0	2	3	x	x	x	Servo Drive 9400 HighLine
9	4	0	0	2	4	x	x	x	Servo Drive 9400 TopLine
9	4	0	0	2	5	x	x	x	Servo Drive 9400 PLC
9	4	0	0	2	6	x	x	x	Servo Drive 9400 V/R (regenerative power supply module)
									Applications:
						0	0	0	Empty application
						0	0	1	Actuating drive speed
						0	0	2	Actuating drive - torque
						0	0	3	Electronic gearbox
						0	0	4	Synchronism with mark synchronisation
						0	0	5	Table positioning
						0	0	6	Positioning sequence control
						0	0	7	PLC application
						0 ... 0	0 ... 9	8 ... 9	Reserved ... Reserved
						1	x	x	Reserved for device profiles
						1	0	1	CiA402
						2	x	x	Reserved for Lenze applications
						2	0	1	Regenerative power supply module application

Product codes for Inverter Drives 8400

Product code [dec]						Meaning
8	4	0	0	2	2	Inverter Drive 8400 StateLine
8	4	0	0	2	3	Inverter Drive 8400 HighLine
8	4	0	0	2	4	Inverter Drive 8400 TopLine

Product codes for i700 servo inverter

Product code [hex]								Meaning
6	9	0	7	0	0	0	1	Single axis
6	9	0	7	0	0	0	2	Double axis

Product codes for the I/O system 1000

Product code [dec]				Meaning
1	3	0	0	I/O system EPM-S130

4.2.4 The EtherCAT interface of the Lenze Controller

The EtherCAT interface links the controller to an EtherCAT network.

Note!

In the case of a correct connection to the field devices, the LEDs of the EtherCAT interface are lit.

▸ EtherCAT interface of the Lenze Controller (📖 35)

Example

ETC1: EtherCAT network connection

[4-4] EtherCAT interface at the 3231 C controller

4.3 Lenze Engineering tools

The Lenze Engineering tools enable the configuration and operation of controller-based Lenze automation systems according to individual requirements.

Use the corresponding Engineering tool applicable to the field device.

»EASY Navigator«: Starting the suitable Engineering tool

The Lenze Engineering software consists of the Engineering tools optimised for the respective Engineering stage.

The »EASY Navigator« represents the Lenze Engineering tools installed on the Engineering PC. Start the desired Engineering tool via the corresponding button:

The »EASY Navigator« ...

- simplifies orientation for selecting the suitable Engineering tool;
- allows for the simple start of the required Engineering tool (depending on the application):

What would you like to do?	Button	Engineering tool
Programming • Program the controller • Parameterise/commission the Servo-Inverter i700 • Parameterise the I/O system 1000		»PLC Designer«
Parameterising/configuring the inverter • Parameterising and configuring the automation/drive system • Parameterising Inverter Drives 8400/Servo Drives 9400		»Engineer«
Visualisation • Visualising the applications of the automation system • Creating the visualisation/user interfaces		»VisiWinNET«
Online diagnostics • Easy online diagnostics of the controllers (from »EASY Starter« V1.2) and other Lenze devices		»EASY Starter« (reading parameters)
Online parameterisation • Online parameterisation/commissioning of Lenze devices • Direct online parameterisation when the online connection to the Lenze devices is active.		»EASY Starter« (reading/writing parameters)

Further Engineering tools that are not called via the »EASY Navigator« are:

- »WebConfig« (web-based parameterisation, configuration, and online diagnostics)
- »Backup & Restore« (data backup/recovery, software update).

4.4 Interaction of the components

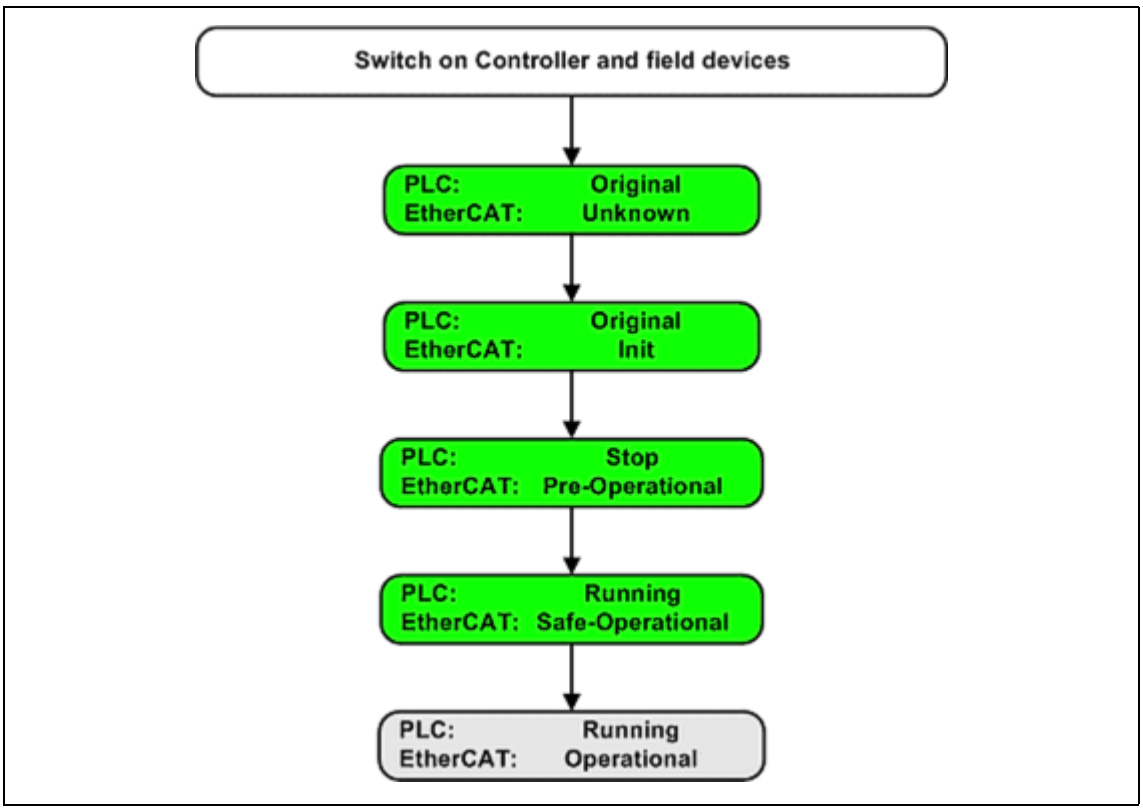
4.4.1 The state machine of the Lenze control technology

In the Lenze control technology, the states of the PLC and the EtherCAT bus are coupled. The PLC controls the fieldbus.

After switch-on, the system automatically powers up if the following conditions are fulfilled:

- There is an executable PLC boot project on the controller (»PLC Designer« project).
- The slaves that are on the fieldbus and have been configured in »PLC Designer« are accessible.

The following illustration shows the linkage of the states in the state machine of the Lenze control technology when the conditions for the automatic acceleration of the system are fulfilled (boot project with EtherCAT configuration):



[4-5] States in the Lenze control technology

Legend	
	Transitional state, automatic change to next state
	Stationary state, change to next state by external actions
PLC	State of the PLC
EtherCAT	State of the EtherCAT bus

Explanation of the transitions during system start

While a state is passed through, different tests are carried out (e.g. it is tested whether the physical topology complies with the configuration). If the tests are successful, the system automatically changes to the next state.

Status		What happens?	What is tested?
PLC	EtherCAT		
Source	Unknown	The system starts.	EtherCAT configuration available?
Source	Init	EtherCAT is initialised (a fieldbus scan is carried out).	Does the EtherCAT configuration match the results of the fieldbus scan?
Stop,	Pre-Operational	• EtherCAT is active. • SDO communication is possible.	
Running	Safe-operational	• The PLC program is being loaded. • The PLC is running. • The input data in the process image are updated. • The output data from the process image are not transferred to the slaves.	
Running	Operational	The system is running.	

If the tests are not successful, the fieldbus changes to the next state. Corresponding remedial measures are described in the chapter entitled "Error scenarios" (📖 195).

4.4.2 Communication between the Engineering PC and the field devices

For commissioning of the field devices, an online connection is required between the Engineering PC and the corresponding field device. Depending on the state of the EtherCAT bus, there are two options:

- EtherCAT bus not in operation (📖 33)
- EtherCAT bus in operation (gateway function) (📖 34)

4.4.2.1 EtherCAT bus not in operation

You can communicate serially or via CANopen.

Note!

For the **i700 servo inverter**, there is no possibility of communication in this case.

Prerequisite:

Serial communication:

- You require the E94AZCUS diagnostic adapter.
- The field device and Engineering PC (USB interface) must be connected via the diagnostic adapter.

Communication via CANopen

- You required the EMF2177IB USB system bus adapter .
- The field devices and the Engineering PC are connected via the system bus adapter - either via a point-to-point connection or via the bus system.

Advantage:

Quick option of communication without commissioning of the EtherCAT bus.

Disadvantage:

You require additional hardware.

Tip!

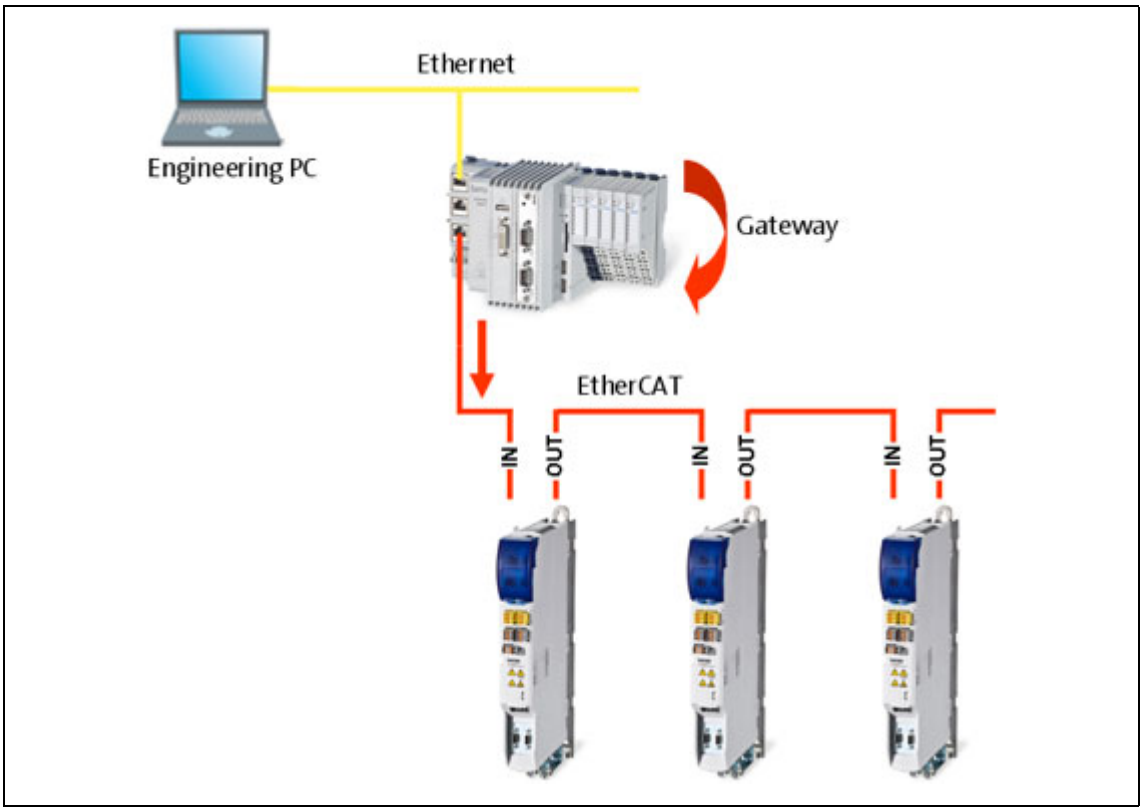
As soon as the fieldbus has been commissioned and is at least in the "Pre-Operational" state, this communication channel is of secondary importance. We recommend that the EtherCAT bus be commissioned as early as possible in order to enable use of the gateway function.

4.4.2.2 EtherCAT bus in operation (gateway function)

You communicate directly via EtherCAT and use the controller as a gateway.

Note!

A PLC program does not need to run to be able to use the gateway function.



[4-6] **Example:** EtherCAT bus system with 3231 C controller as gateway and i700 servo inverter

Prerequisite:

- The EtherCAT configuration has been created with »PLC Designer« to match the hardware configuration.
- The EtherCAT configuration has been loaded and activated on the controller with »PLC Designer«.
- The EtherCAT bus is at least in the "Pre-Operational" state.

Advantage:

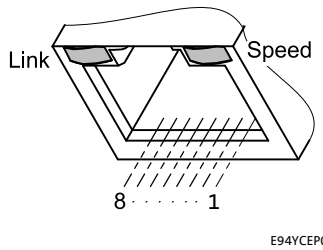
- You do not require any additional hardware.
- The process data, parameter data and diagnostic data are transferred via a single bus connection at the same time.

5 Technical data

5.1 General data

Range	Values
Higher-level network protocol	EtherCAT device protocol
Communication medium / cable type	S/FTP (Screened Foiled Twisted Pair, ISO/IEC 11801 or EN 50173), CAT5e Standard Ethernet (in accordance with IEEE 802.3), 100Base-TX (Fast Ethernet)
Network topology	Line
Type within the network	master
Number of nodes	Max. 65535 (in the entire network)
Max. cable length	100 m between two stations
Baud rate	100 Mbps
Supported communication profiles	CoE (CANopen over EtherCAT) FoE (File Access over EtherCAT)
Synchronisation	Distributed clocks

5.2 EtherCAT interface of the Lenze Controller

RJ45 socket	Pin	Assignment	LEDs
	1	Tx +	"Link" LED (green): On: connection ok
	2	Tx -	
	3	Rx +	"Speed" LED (yellow): Blinking: active data exchange
	4	-	
	5	-	
	6	Rx -	
	7	-	
	8	-	

5.3 Communication times and drive-specific data

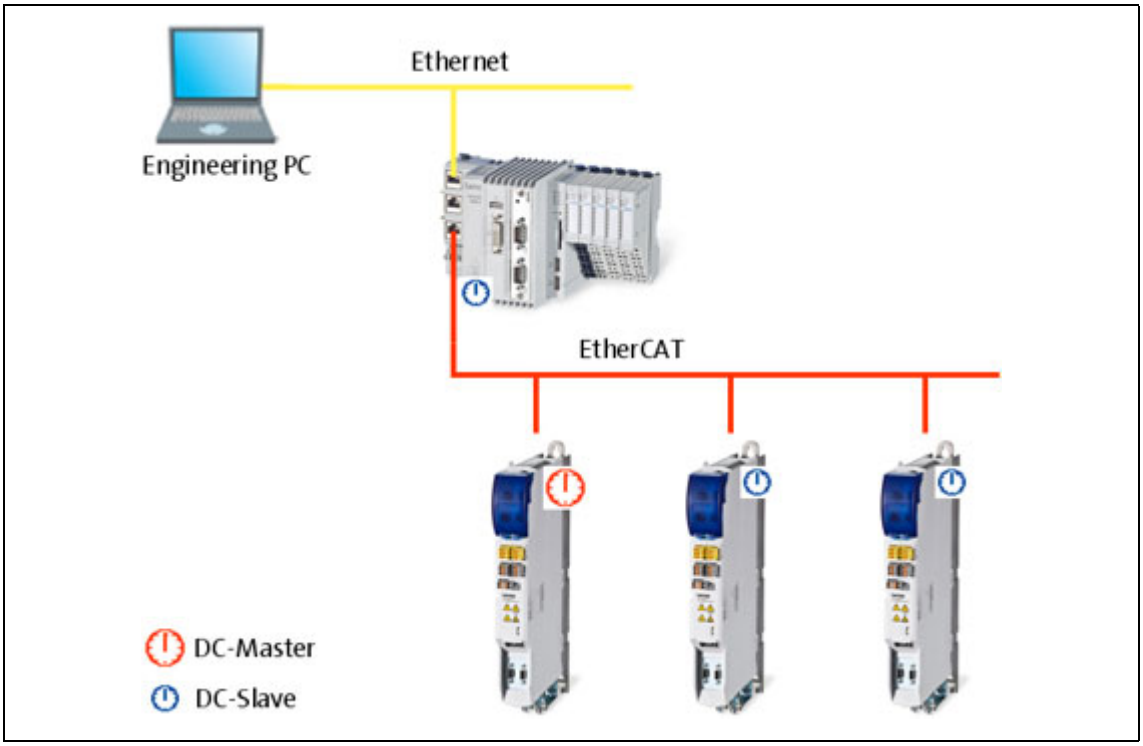
Range	Values	
User data per frame	1344 bytes	
Process data words (PCD)	Depending on the inverter used (see documentation of the inverter)	
Parameter data (SDO) transfer	Max. 128 bytes	
Permissible EtherCAT cycle times	1 ... 10 ms	
Max. number of drives per frame	User data of the frame (1344 bytes) divided by the process data length resulting from setpoints and actual values of the drives: • for 32 Tx/Rx bytes: 1344 bytes / 64 bytes = 21 drives • for 16 Tx/Rx bytes: 1344 bytes / 32 bytes = 42 drives	
Cross communication	Not possible	
Cycle synchronisation with locked PLL (Jitter)	+/-1 µs	
Instant of transmission for the EtherCAT frame (for setting, see Optimising the task utilisation (📖 112))	The EtherCAT frame is sent at the beginning of the bus cycle task.	The EtherCAT frame is sent at the end of the bus cycle task.
Total signal runtime for a cycle time of 1 ms Drive → controller → drive	4 ms	3 ms
Runtime of the setpoints Controller → drive	2 ms	1 ms
Runtime of the actual values Drive → controller	1 ms	1 ms

6 Synchronisation with "Distributed clocks" (DC)

The "Distributed clocks" (DC) functionality enables exact time synchronisation for applications in which several axes perform a coordinated movement simultaneously. Data are incorporated synchronously with the PLC program. During DC synchronisation, all slaves are synchronised with a reference clock, the so-called "DC master".

Note!

- DC synchronisation only occurs in the "Operational" state, see
  ▸ The EtherCAT state machine (📖 21)
- DC synchronisation is absolutely required for Motion applications.
- DC synchronisation can also be used for Logic applications.
- Not all slaves support the DC functionality.
- In order to be able to use the DC functionality, the first slave connected to the EtherCAT master (Lenze Controller) must have **DC master capability**. The other slaves can be connected in a mixed arrangement, either DC capable or non-DC-capable.
- The first EtherCAT slave after the Lenze Controller must be the **DC master** that supplies the other EtherCAT nodes (incl. controller) with the exact time.



[6-1] **Example:** EtherCAT bus system with 3231 C controller and i700 servo inverter

The settings for the DC synchronisation are made with the »PLC Designer«.

▸ Setting a DC synchronisation (📖 94)

6.1 Synchronous communication

DC synchronisation provides that the BuscycleTask of the master and the slaves run synchronously in phase:
Within one bus cycle the setpoints are accepted and the actual values are detected in the fieldbus at exactly the same time.

When the next DC synchronisation event occurs, the data are accepted.

Note!

After an "out-of-sync" the master must be restarted by the user.

If the synchronisation of the slaves is successful, the message "In-Sync" is displayed after the restart, see Diagnostic tabs of the EtherCAT master (📖 185)).

6.2 Test of DC synchronicity

DC synchronicity is only available in the "Operational" state.

Test of DC synchronicity in »PLC Designer«

- EtherCAT master: **Diagnostic Master** tab
  "DC In-Sync" is set (TRUE) if the DC master and all DC slaves have been synchronised.
  ▸ Diagnostic tabs of the EtherCAT master (📖 185)
- Function block L_ETC_GetMasterDiagnostic (FB) (📖 155) / Visualisation of the function block L_ETC_GetMasterDiagnostic (📖 187)
  "DC In-Sync" is set (TRUE) at the *oDiagnostic.xDC_InSync* output if the DC master **AND** all DC slaves are synchronised.
- Function block L_IODrvEtherCAT (FB) (📖 149)
  "DC In-Sync" is set (TRUE) at the *xDistributedClockInSync* output if all DC slaves are synchronised.

Test of DC synchronicity in »WebConfig«:

In the code C281/5 , "DC In-Sync" is set (TRUE) if all DC slaves have been synchronised.

▸ Diagnostic codes in the »WebConfig« (📖 190)

7 Commissioning of the system

This chapter provides information on how to commission the Lenze automation system with EtherCAT.

Depending on the field devices used, the following Lenze Engineering tools (📖 30) are required:

- »EASY Starter«
- »Engineer«
- »PLC Designer«

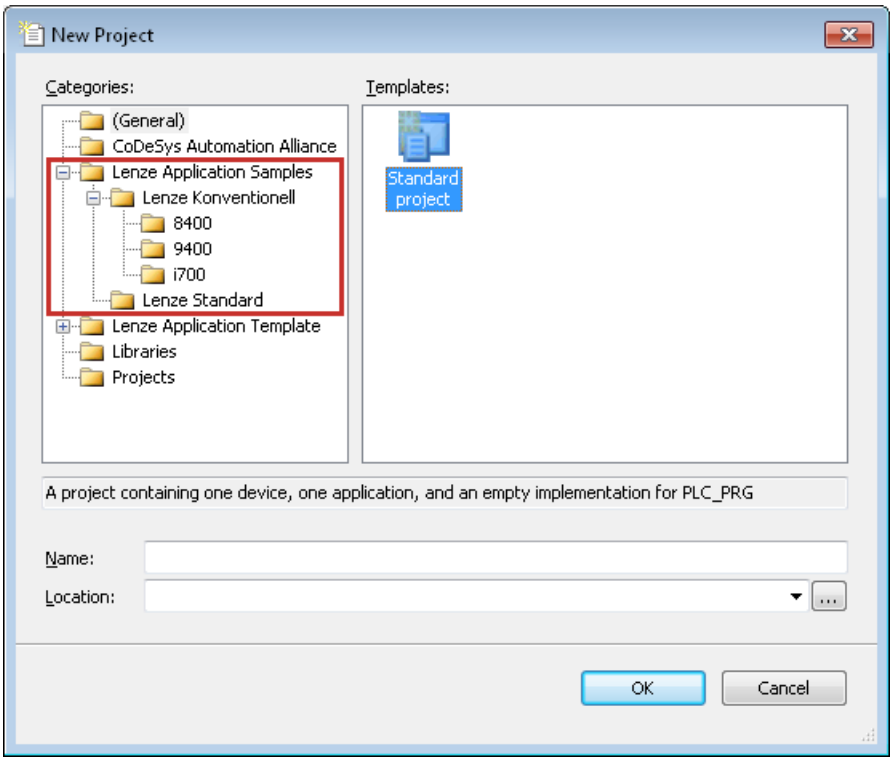
7.1 Sample projects (Application Samples)

There already exist sample projects (device application + PLC program) for commissioning of Lenze inverters.

The Lenze sample projects can be found in the **MS Windows start menu** under:

Start → All programs → Lenze → AppSamples → ...

The Lenze sample projects can also be opened in the »PLC Designer« via the menu command **File → New project...**, or using **<Ctrl>+<N>**.

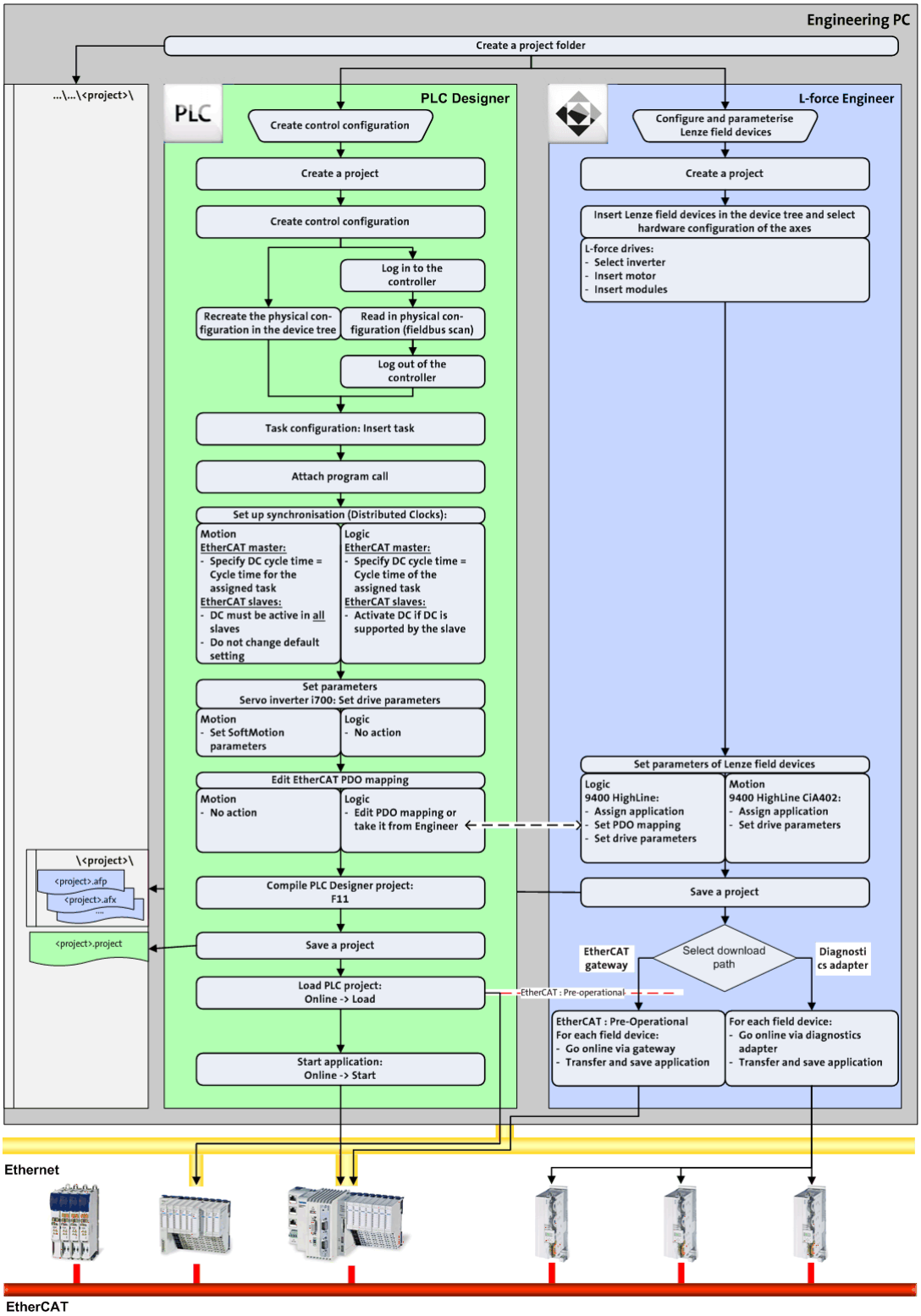


Detailed information on the sample projects can be found in the following documentation:

- SW_ApplicationSample_i700_(»PLC Designer« V3)_Vx-y_DE/EN.pdf
- SW_ApplicationSamples_(Controller-based)_Vx-y_DE/EN.pdf

7.2 Overview of the commissioning steps

In the following illustration, the individual commissioning steps and their processing order are summarised. Detailed information on the individual processing steps can be found in the chapter Detailed description of the commissioning steps (📖 44).



The main commissioning steps are listed in the following table:

Step	Activity	Lenze software to be used
1.	Installing field devices (📖 45)	
2.	Create a project folder (📖 45)	
3.	Commissioning the i700 servo inverter (📖 46)	»PLC Designer«
	Commissioning other Lenze field devices (📖 77)	»Engineer« / »EASY Starter«
4.	Creating a PLC program with a target system (Logic/Motion) (📖 78)	»PLC Designer«
5th	Configuring the communication parameters (📖 80)	
6.	Determining the physical EtherCAT configuration (fieldbus scan) (📖 82)	
	If required, Importing missing devices / device description files (📖 85)	
6.	Creating a control configuration (adding field devices) (📖 86)	
7.	Creating a task (📖 89)	
8.	Setting a DC synchronisation (📖 94)	
10.	Setting SoftMotion parameters (📖 99) Only required for drives with Motion functionality.	
11.	Processing EtherCAT I/O mapping (📖 102) Only required for drives that solely have the master functionality (logic bus).	
12.	Compiling the PLC program code (📖 110)	
13.	Logging in on the controller with the »PLC Designer« (📖 110) With the log-in, the fieldbus configuration and the PLC program are loaded to the controller.	
14.	Starting the PLC program (📖 110)	

7.3 Detailed description of the commissioning steps

In the following sections, the individual commissioning steps are described.

Follow the instructions of these sections step by step in order to commission your system.

For more details on how to use the Lenze engineering tools, please refer to the corresponding software manuals and the online help systems.

7.3.1 Planning the bus topology

Before you set up an EtherCAT network, draw up a plan of the network.

How to plan the bus topology for your configuration

1. Create an overview screen of the planned EtherCAT network with all the field devices to be integrated into the network.
2. Start with the Lenze Controller (master).
3. Add the other field devices (slaves) below.

   The following cases are distinguished:
   - Operation without synchronisation via distributed clocks:
     DC synchronisation is mostly not required if exclusively Logic field devices are to be operated on the network. The sequence of the field devices on the fieldbus can be freely selected.
   - Operation with synchronisation via distributed clocks:
     DC synchronisation is required if Motion and Logic field devices are to be operated on the network. The first node connected to the Lenze Controller must be capable of being a DC master. The sequence of the other Logic and Motion field device interface connections at the fieldbus can be freely selected.

7.3.2 Installing field devices

For the installation of a field device, follow the mounting instructions for the respective device.

Note!

- In the case of all field devices, the EtherCAT interfaces must be wired in accordance with the preceding topology planning. Make sure that the inputs (IN) and outputs (OUT) are not mixed up with each other; otherwise, the topology changes.
  ▸ communication (📖 20)
- The physical sequence of the field devices in the EtherCAT network must match the EtherCAT configuration created in »PLC Designer«.
- The master automatically assigns the node addresses to the slaves. Therefore, a manual address assignment is not required.

7.3.3 Create a project folder

Create a project folder on the Engineering PC.

Use this project folder to store the data generated in the following different project configuration steps:

- Project data created in the »Engineer« or »EASY Starter«
- The project file created in the »PLC Designer«

Tip!

Create a separate project folder for every EtherCAT configuration and store the project files.

7.3.4 Commissioning the i700 servo inverter

This chapter tells you how to commission the Servo-Inverter i700 in the Lenze automation system with the help of »PLC Designer«.

For speed commissioning, the Servo-Inverter i700 provides diverse functions for automatic calculation and setting of parameters.

Danger!

Severe personal injury and damage to the machine/installation

Activate the controller inhibit before you set the parameters for the Servo-Inverter i700 as uncontrolled movements can lead to severe personal injury and damage to the machine/installation.

Note!

You can also set the parameters for the Servo-Inverter i700 online. To do this, you must first configure EtherCAT communication.

Setting parameters online is possible from the EtherCAT state "Pre-Operational" onwards.

Reference manual/online help for the Servo-Inverter i700

Here, you can find detailed information on all parameters/Objects (object dictionary), functions and error messages of the Servo-Inverter i700.

Tip!

There already exist sample projects (device application + PLC program) for commissioning of the Servo-Inverter i700.

▸ Sample projects (Application Samples) (📖 40)

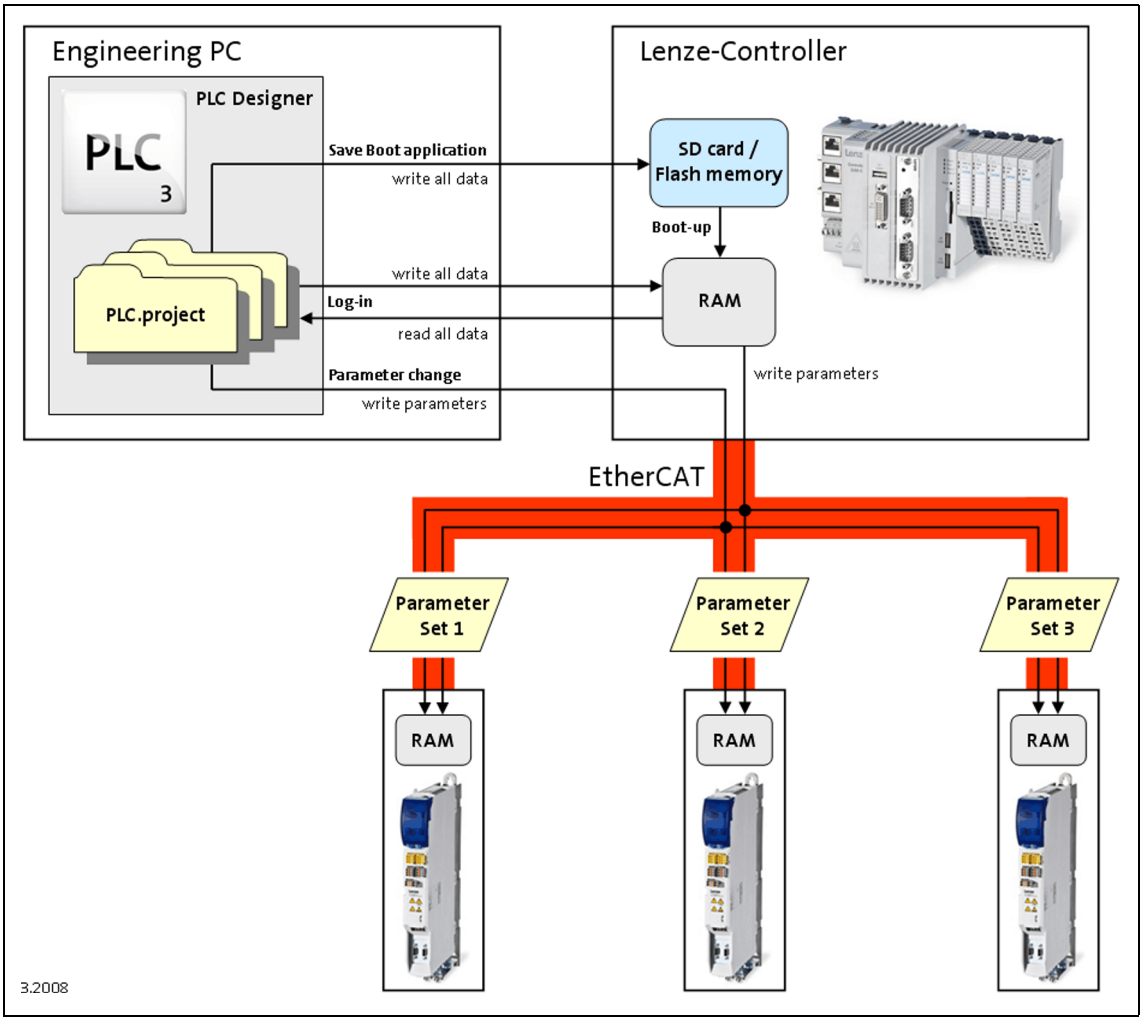
7.3.4.1 i700 parameter management in the Controller-based Automation system

Parameter download

The Servo-Inverter i700 itself does not store parameter settings safe against mains failure. All servo inverter settings deviating from the Lenze standard setting are maintained centrally in the Lenze Controller and stored there permanently (persistently). During the initialisation at run-up, only these deviations are transferred to the servo inverter by the controller. Like this it is ensured that the servo inverter works with the parameter settings provided for it.

Firmware download (optional)

The firmware of the i700 servo inverter can – if so desired – be stored together with the »PLC Designer« project. When the system is being powered up, the Lenze controller then checks whether the firmware version in the servo inverter matches the firmware version stored in the project for this device. If this is not the case, the controller loads the firmware version stored in the project into the servo inverter. In the event of service whereby a device has to be replaced, it can thus be ensured that the replacement device works with the same firmware version stored in the project for the original device.



[7-1] Parameter set transfer from »PLC Designer« to the Servo-Inverter i700 via the 3231 C controller

The parameters of the Servo-Inverter i700 are managed within the »PLC Designer« project. With the help of the storage function of »PLC Designer«, the »PLC Designer« project, including the i700 parameters, are stored on the Engineering PC.

During login to the Lenze Controller,»PLC Designer« transfers the i700 parameters to the controller. The latter, in turn, transfers the parameter sets to the lower-level Servo-Inverter i700.

Whenever the system is booted, the parameter sets are written into the Servo-Inverter i700 again by the Lenze Controller.

There are three cases of application for the management and alteration of parameters:

A. Changing parameters of an Servo-Inverter i700 online:

   If a parameter is changed online, »PLC Designer« writes the parameter directly into the corresponding servo inverter and, at the same time, changes the parameter in the»PLC Designer« project.

   N.B.:

   The parameter change is not recorded in the Lenze Controller. In the event of a "cold reset" of the controller, the parameter change in the servo inverter is lost.

   In order to avoid this, log out with the »PLC Designer« and then log in again. (Menu commands: **Online → Logout / Online → Login**)

   Like this, the complete parameter set is written to the controller and is transferred to the servo inverter.

   After the transfer by log-in, the parameter set is only available in the servo inverter and in the controller until the next switch-off of the system (it is not saved with mains failure protection).

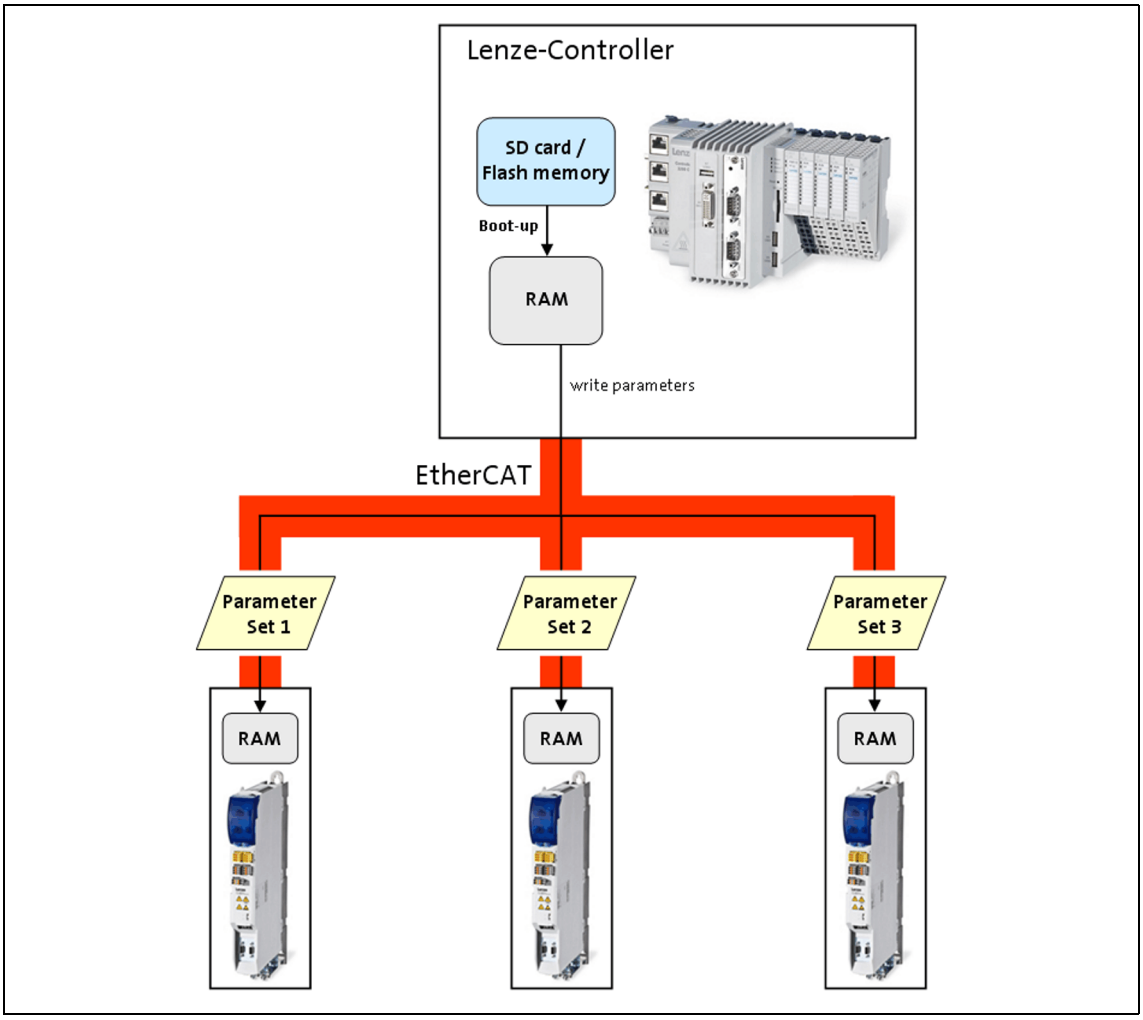
B. Changing parameters of a Servo-Inverter i700 offline:

   If a parameter is changed offline, the »PLC Designer« changes the value directly in the »PLC Designer« project. When logging in on the Lenze Controller, the complete parameter set is written to the controller, which, in turn, then transfers the parameter set to the servo inverter.

   After the transfer by log-in, the parameter set is only available in the servo inverter and in the controller until the next switch-off of the system (it is not saved with mains failure protection).

C. Store the parameters of the Servo-Inverter i700 in a non-volatile memory so that they are retained in the device when the power is switched off:

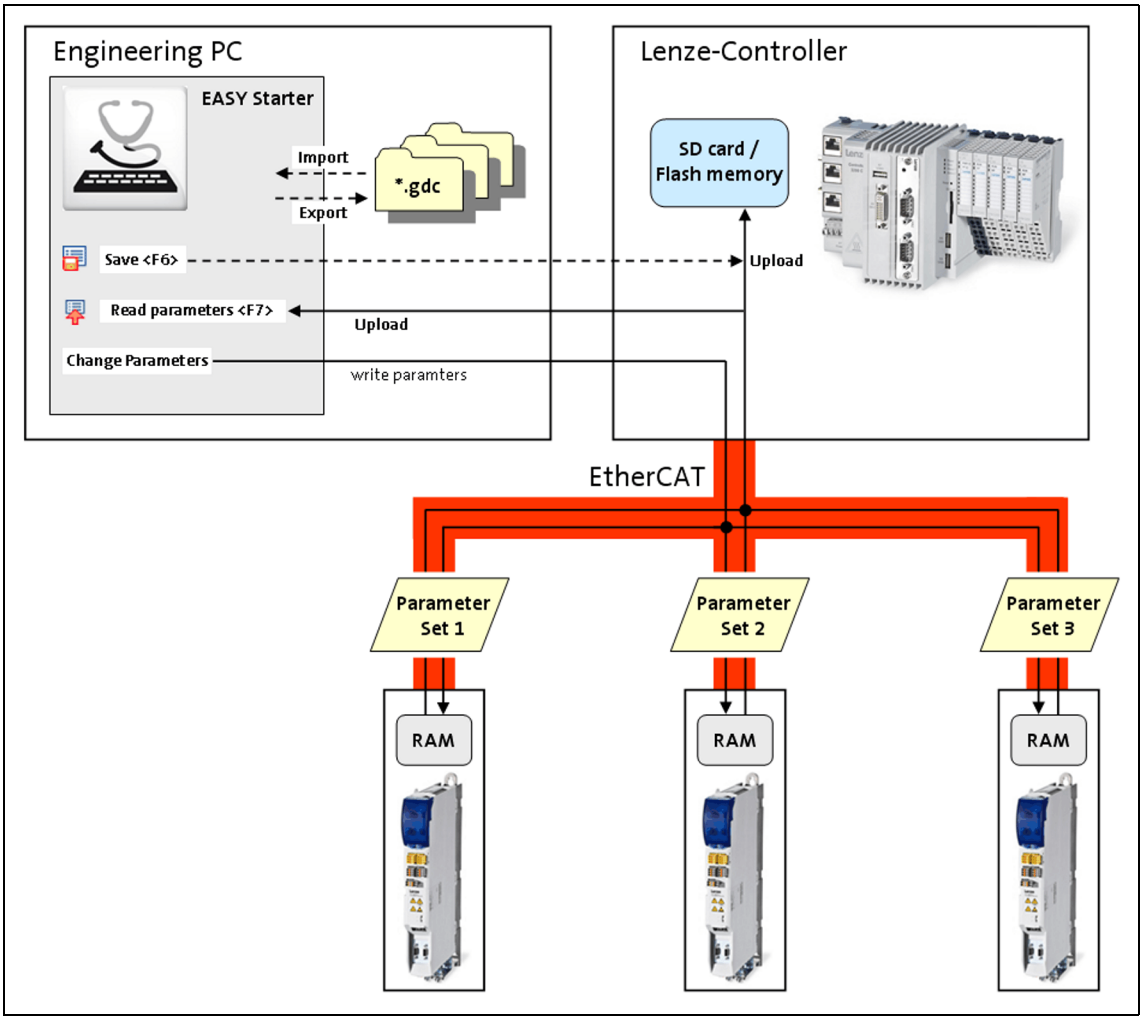
In the boot project of the Lenze Controller, there is a separate parameter set for the lower-level Servo-Inverters i700. The latter is then only updated when the "boot project" function is executed via the »PLC Designer«. The current parameterisation is then saved and is available after mains switching. During boot-up after mains connection, the saved parameter set is automatically written from the controller into the servo inverter.



[7-2] Parameter set transfer from the 3231 C controller to the Servo-Inverter i700 during a boot-up

7.3.4.2 i700 parameter management in »EASY Starter«

For parameter management in »EASY Starter«, you must insert the Lenze Controller and the Servo-Inverter i700 into the device list with the help of the button.



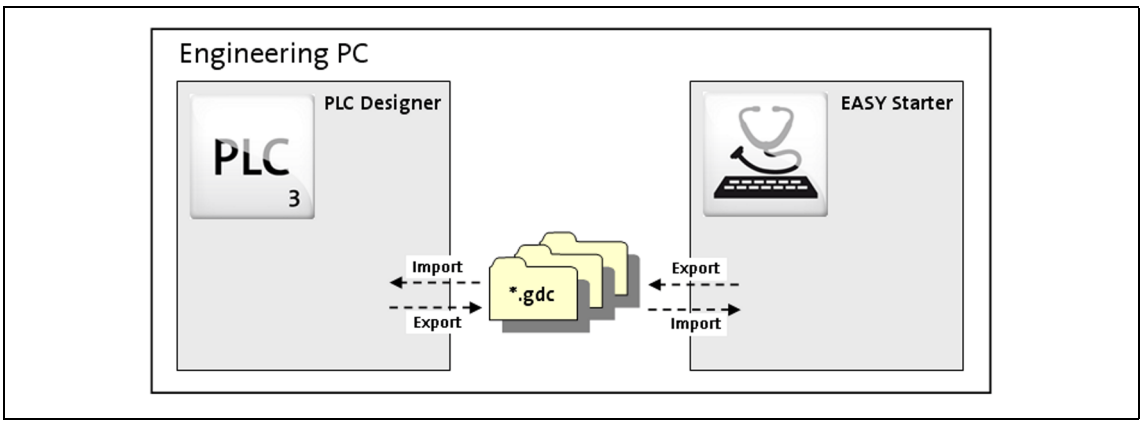
[7-3] Parameter set transfer with »EASY Starter«

With »EASY Starter«, there are two application cases for parameter management:

D. Storing the current Servo-Inverter i700 parameter setting in the Lenze Controller with mains failure protection:
   - Select the Lenze Controller from the device list.
   - With the button or the **<F6>** function key, store the parameter set in the controller so that it is saved there even in the event of a power failure. (To this end, the controller uploads all the parameter sets of the lower-level i700 servo inverter.)
   - After the system has been switched off and then switched on again. the controller transfers the parameter sets to the i700 inverter again.

E. Upload the parameters currently set in a Servo-Inverter i700 to the Engineering PC and store them there:
   - Select a Servo-Inverter i700 in the device list
   - With the button or the **<F7>** function key, upload the parameter set of the servo inverter. (The parameter set of the servo inverter is transferred directly to »EASY Starter« without the parameter set being stored on the controller.)
   - With the **Save** button, store the parameter as a **GDC file** on the Engineering PC.

7.3.4.3 Exchanging i700 parameter sets between »PLC Designer« and »EASY Starter«



[7-4] Parameter set transfer between »PLC Designer« and »EASY Starter«

The **GDC files** that have been stored on the Engineering PC with the help of »EASY Starter« can be imported in »PLC Designer«.

It is also possible to export GDC files with »PLC Designer«; they can then be imported in »EASY Starter«.

In the framework of this import/export functionality, the parameter settings of a single axis can be imported into a double axis, for example.

How to import parameters in »PLC Designer«

1. Log out of the Lenze Controller with the menu command **Online → Logout** or **<Ctrl>+<F8>**.
2. Select the corresponding Servo-Inverter i700 in the project tree.
3. Select and import the corresponding GDC file with the menu command **Project → Device parameters → Import device parameters**.
4. Use the menu command **Online → Login** or **<Alt>+<F8>** to log in on the Lenze Controller.
   - For this, the PLC program must be error-free.
   - The complete parameter set is written into the controller. The latter than transfers the parameter set to the i700 servo inverter.

Parameters are exported in »PLC Designer« in a similar way with the help of the menu command **Project → Device parameters → Export device parameters**.

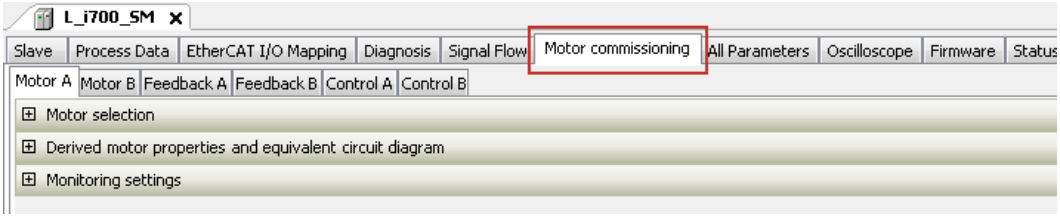
7.3.4.4 Overview of the commissioning steps

The main commissioning steps are listed in the following table:

Step	Activity
1.	Create a project folder (📖 45)
2.	Creating a PLC program with a target system (Logic/Motion) (📖 78)
3.	Configuring the communication parameters (📖 80)
4.	Determining the physical EtherCAT configuration (fieldbus scan) (📖 82) or Creating a control configuration (adding field devices) (📖 86)
5th	Creating a task (📖 89)
6.	Setting a DC synchronisation (📖 94)
6.	Entering motor and controller settings (📖 55)
7.	Setting the feedback system for servo control (📖 58)
8.	Checking the wiring by means of the "Cable check" function (📖 60)
10.	Integrating the L_SMC_AxisBasicControl function block (📖 66)
11.	Setting SoftMotion parameters (📖 99) Only required for drives with Motion functionality.
12.	Processing EtherCAT I/O mapping (📖 102) Only required for drives that solely have the master functionality (logic bus).
13.	Compiling the PLC program code (📖 110)
14.	Logging in on the controller with the »PLC Designer« (📖 110) With the log-in, the fieldbus configuration and the PLC program are loaded to the controller.
15.	Starting the PLC program (📖 110)
16.	Executing manual control (📖 72)
17.	Optimising control (📖 74)

7.3.4.5 Entering motor and controller settings

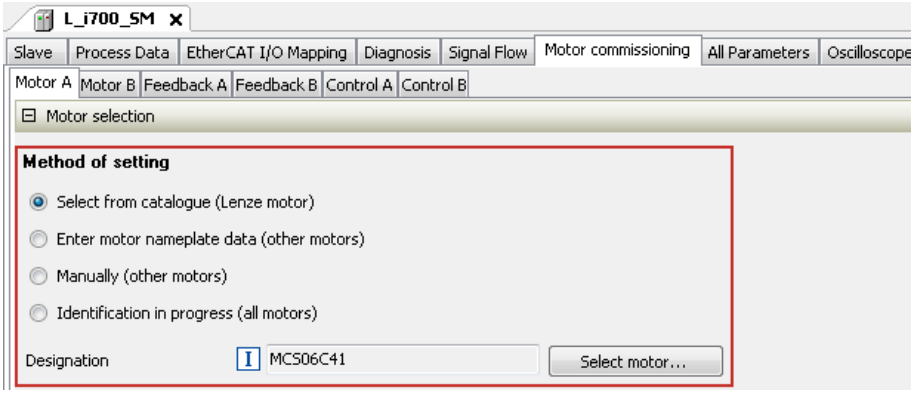
You can enter the motor and controller settings on the **Motor commissioning** tab of the i700 servo inverter.



[7-5] Example: i700 servo inverter, double axis

How to enter the motor and controller settings

1. Go to the **Motor selection** section and specify the data of the motor to be operated in conjunction with the i700 servo inverter.



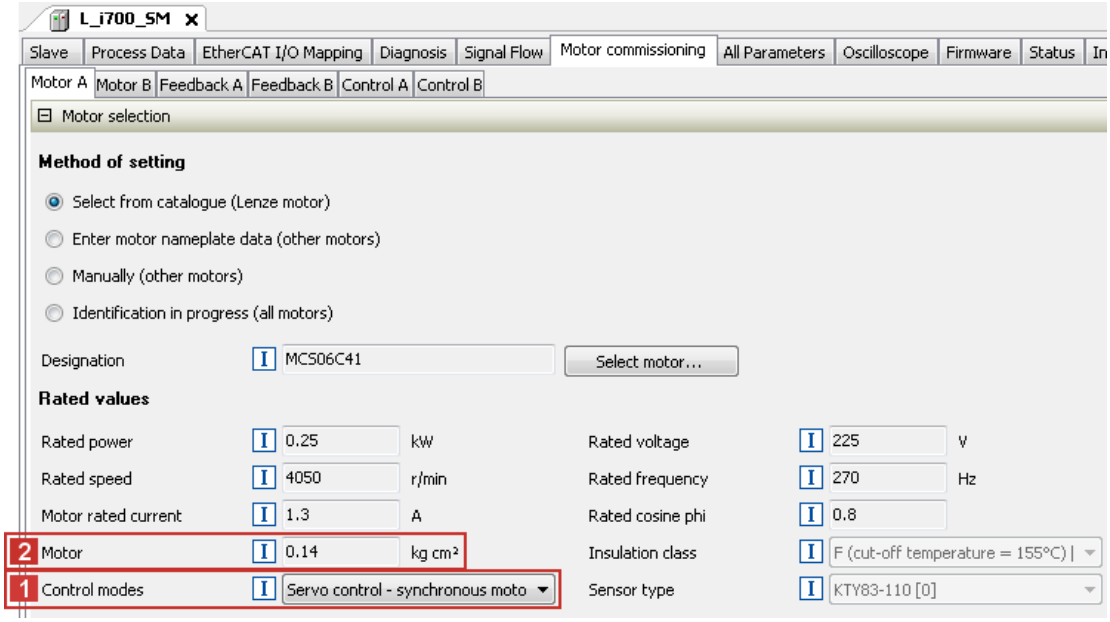
A. Select the Lenze motor from the motor catalogue by clicking the **Select motor...** button.
- The corresponding motor data will then be incorporated in the »PLC Designer« project.
- The controller parameters are calculated automatically.

<u>or</u>

B. Enter the motor data manually or specify them by means of an identification run (e.g. in the case of motors from other manufacturers).

The controller parameters are calculated automatically.

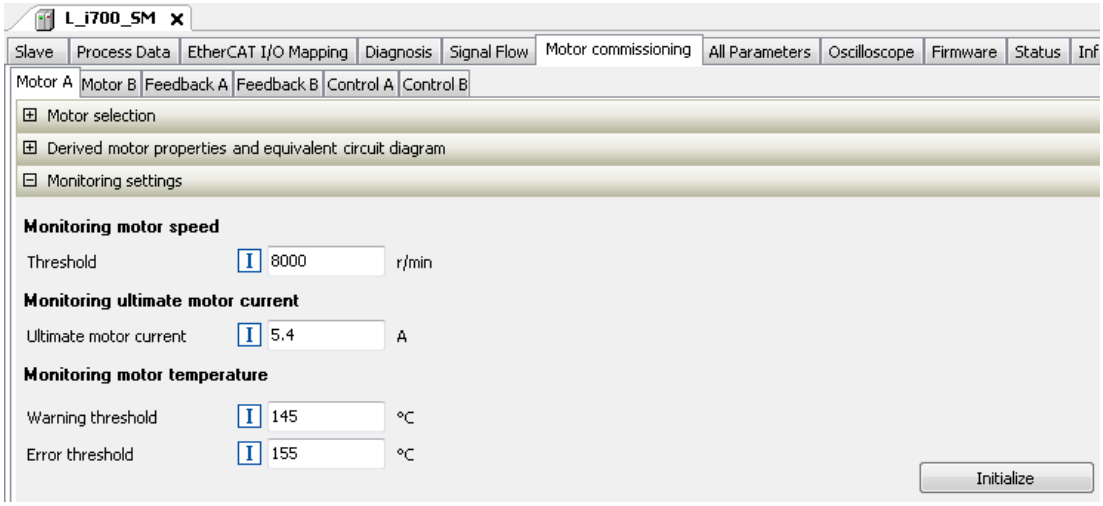
2. 1 Select **Control method (0x2C00 / 0x3400)**:
   - Servo control for synchronous motor (SM)
   - Servo control for asynchronous motor (ASM)
   - VFC: V/f characteristic control



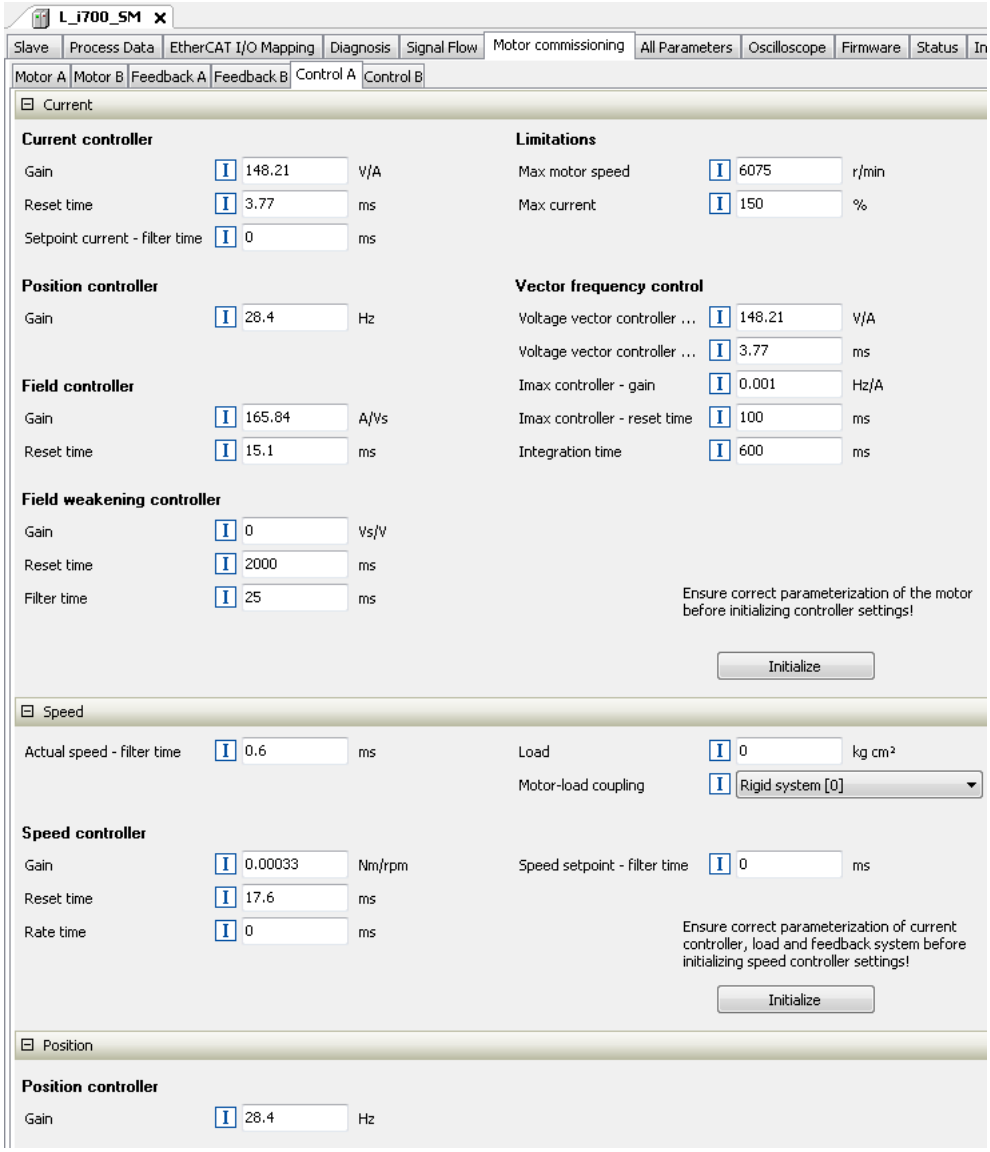
3. If you do not obtain the motor data from the catalog, you have to set the 2 **moment of inertia (0x2910/1 / 0x3110/1)** to a non-zero value.

   The controller parameters are calculated automatically.

4. You can select the parameters for the monitoring functions in the **Monitoring settings** section.



5. Retain or alter the automatically calculated controller parameters under **Control**.



7.3.4.6 Setting the feedback system for servo control

Danger!

Using the encoder/resolver as a motor encoder

In the Lenze setting, the resolver/encoder cable is monitored for open circuit.

In the event of a fault, safe operation of the motor is no longer guaranteed!

- For safety reasons, always select "Fault" (Lenze setting) as a response for the (open-circuit) monitoring of the encoder/resolver.
- To avoid interference injections when an encoder is used, only use shielded motor and encoder cables.

Determining the pole position of the synchronous motor

Pole position identification (angle between the motor phase U and the field axis of the rotor) is necessary ...

- for servo control with a synchronous motor of another manufacturer;
- for servo control with a synchronous motor and use of incremental encoders (TTL or sin/cos encoder and resolver);
- after changes to the feedback system (e.g. replacement of the encoder).

For Lenze motors with an absolute encoder or resolver, the pole position has already been set correctly.

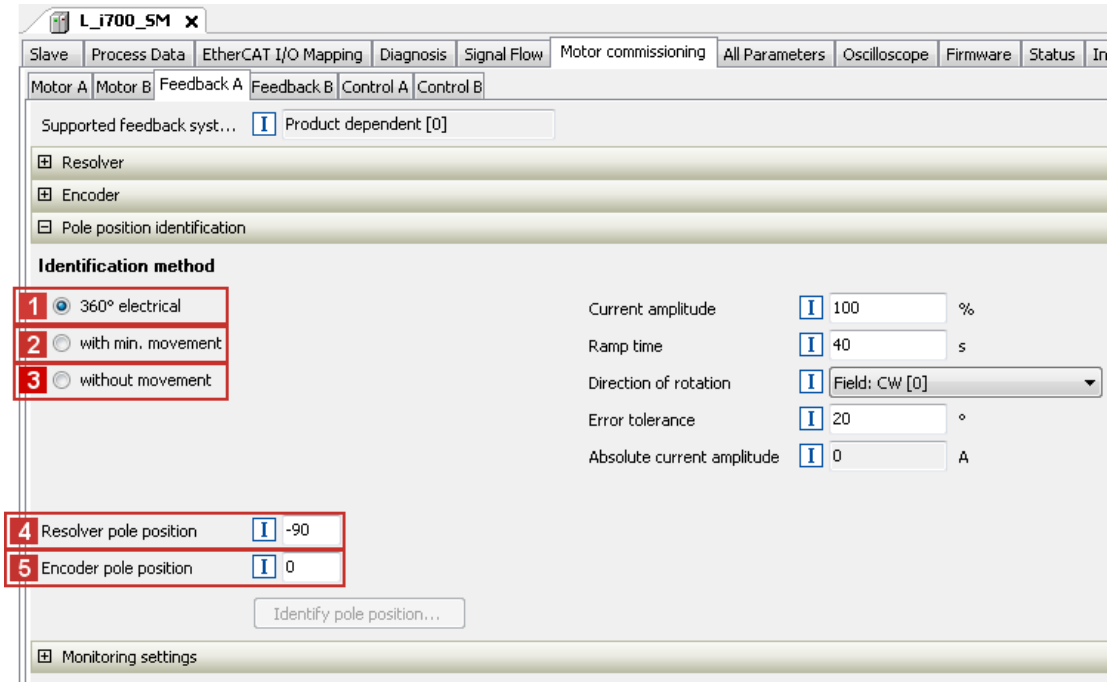
Danger!

- During pole position identification, it must be ensured that the motor is not braked or blocked. Pole position identification is therefore not permissible in the case of hanging loads.
- The rotor will align itself during pole position identification. The motor shaft will make a maximum of one electrical rotation, as a result of which the connected machine will move accordingly.
- For firmly braked motors, the function "Pole position identification PPI (without movement)" must be used.

Stop!

Before carrying out pole position identification, check that the parameters for maximum current monitoring have been set correctly so that the motor is not permanently damaged in the event of a fault.

Either of the functions 1 **Pole position identification for 360°**, 2 **Pole position identification with min. movement** and 3 **Pole position identification without movement** can be selected in order to determine the pole position for the currently activated motor encoder:



The functions should deliver approximately the same result. Due to e.g. friction, bearing forces, and a trapezoidal field curve, however, the results can differ from each other. Here the method with one full revolution (360°) will provide the most precise results and the method without any movement will provide the most inaccurate results. The precision of the results can be increased by increasing the percentage of the current amplitude.

After successful completion of pole position identification ...

... the controller inhibit is set automatically and the 4 **Resolver pole position (0x2C03/2 / 0x3403/2)** or the 5 **Encoder pole position (0x2C03/4 / 0x3403/4)** determined for the activated feedback system is set.

- For permanent storage, the changed settings must be uploaded from the Servo-Inverter i700 to the Lenze Controller.
  The »EASY Starter« can be used to upload the parameters of the servo inverter and to save them as a file. This file can then be imported to the »PLC Designer«.
- The controller inhibit automatically set by the procedure can be deactivated via the **CiA402 control word (0x6040 / 0x6840)**.

Fault

If an error occurs during pole position identification or if pulse inhibit becomes active (e.g. due to short-term undervoltage), the procedure is ended with controller inhibit without the settings being changed.

If the motor was braked or blocked during pole position identification, this is detected at the end of a measurement and no changes are made.

If pole position identification is aborted, the reaction set in the object, namely **0x2C60/0x3460 (monitoring of pole position identification: reaction)** is triggered (Lenze setting: Fault).

7.3.4.7 Checking the wiring by means of the "Cable check" function

Before you start to set the parameters of the drive control system, check the wiring of the motor (power and encoder connections) for faults.

This function detects wiring errors or cable damage which may cause uncontrolled machine movements during the commissioning phase or during operation. The function therefore helps preventing machine damage and serves to eliminate these errors as quickly as possible.

Use of the function in the life cycle of the machine

During the commissioning phase, the machine can be checked for installation errors on the motor and the motor encoder.

▸ Executing the "Cable Check" function during the commissioning phase (📖 61)

We furthermore recommend always executing this function immediately after the initial switch-on of the mains voltage after having completed the commissioning phase, or after servicing events.

Note!

Load encoders are not supported by the "Cable Check" function.

Parameter

0x2C64 | 0x3464 - Cable check

From software version V02.11.xx onwards

Sub.	Name	Lenze setting	Data type
▸ 1	Cable check: Response after switch-on	0: No action	UNSIGNED_8
▸ 2	Cable check: status word		UNSIGNED_16

Subindex 1: **Cable check: Response after switch-on**			
Selection list(Lenze setting printed in bold)			
0	**No Operation**		
1	Check only at initial switch-on and following an encoder error		
2	Check after every switch-on		
☑ Write access ☑ CINH ☐ OSC ☑ P ☐ RX ☐ TX			UNSIGNED_8

Subindex 2: **Cable check: Status word**			
Display range (min. value | unit | max. value)			Initialisation
0		65535	
Value is bit-coded:			Info
Bit 0	Enable cable check		
Bit 1	Cable check running		
Bit 2	Cable check completed		
Bit 3	Cable check failed		
Bits 4-15	Reserved		
☐ Write access ☐ CINH ☐ OSC ☐ P ☐ RX ☐ TX			UNSIGNED_16

Executing the "Cable Check" function during the commissioning phase

The **Cable Check** function should be executed manually by the user during machine commissioning, in order to identify typical errors that may occur when the machine is installed.

Typical errors:

1. The motor encoders are connected to the wrong device before initial switch-on, whereas the motor is connected to the correct device.
   If the motor encoders that are incorrectly connected are of the same type, the motor encoder monitoring function does not detect any error.
2. The motor encoders are connected to the correct device, however, the motor is connected to the wrong device.
3. The motor phases are connected on the device in an inverted fashion.
4. Individual wires of the rotary transducer are incorrectly connected.

Before executing the **Cable Check** function manually, observe the notes in

▸ Preconditions for the manual execution of the Cable Check function (📖 62)

Then perform the check

▸ How to execute the "Cable Check" function manually (📖 63)

Preconditions for the manual execution of the Cable Check function

- The motor can remain coupled to the kinematics.
  - In the case of drives without a motor holding brake, the rotor must be able to rotate by 20 ° (electrically). This requires setting the operating mode of the motor holding brake to "No brake connected" (0x282XX0:1 = 2).
  - In the case of drives with a motor holding brake, the test is performed against the closed motor holding brake. With 0x2820:1, the operating mode must be set to the value **0** or **1**.
- The device must not be in the **Fault** status.
- 24V supply voltage must be provided.
  The front display of the blue LED is **ON** or blinking.
- The i700 servo inverter must be supplied with mains voltage.
  Status: 0x6041, bit 4 = TRUE
- The i700 servo inverter must not be in the **STO** state. The safety functions must be parameterised.
  Status: 0x6041, bit 15 = FALSE
- The motor data must be set correctly.
- The data of the motor encoder must be set correctly.
- No error message must be active.
  Status: 0x6041, bit 3 = FALSE
  If an error message is active, eliminate these errors first and reset the error message.
- The motor control (0x2C00 or 0x3400) must be set to
  - Servo control - synchronous motor (SM) or
  - Servo control - asynchronous motor (ASM).

  The function only supports one of these two motor controls.
- The response after switch-on must be set: 0x2C64:1= **0** ("No action").

How to execute the "Cable Check" function manually

1. Open the »EASY Starter« engineering tool.
2. Establish an online connection to the i700 servo inverter.
3. Call the **Motor commissioning** tab in the workspace of the »EASY Starter«.
4. Call the **Feedback** tab.
5. Go to the **Cable check: Motor and motor encoder** jalousie and execute the **Cable Check** function
   - via the 0x2825 parameter, selection **15**, or
   - by clicking on the **Execute Cable Check** button.

   The **Cable Check** function is now activated, i. e.
   - 0x2C64:2, bit 0 = TRUE
   - 0x2832, bit 0 = TRUE
6. Activate the i700 servo inverter via the CiA402 control word (0x6040 or 0x6840 for axis B).

   After approx. one second, execution of the function in the »EASY Starter« is indicated as completed.

   Additionally, the »EASY Starter« indicates whether an error has occurred during the check:

   ▸ Result of the Cable Check

Tip!

The **Cable Check** function can always be executed via the **Execute Cable Check** button if the conditions for this are met.

Result of the Cable Check

Status bit	Result of the Cable Check		INFO
	No error	Error	
0x2C64, Cable Check			
Bit 00	FALSE		Cable check enabled
Bit 01	FALSE		Cable check running
Bit 02	TRUE	TRUE	Cable check completed The bit remains TRUE until the 24V supply is switched off (blue and red LED = OFF) or the cable check function is executed again.
Bit 03	FALSE	TRUE	Cable check failed In the event of an error, the bit remains TRUE until the function is executed again or the 24V supply is switched off (blue and red LED = OFF).
0x2832, identification			
Bit 00	FALSE		Identification enabled
Bit 01	FALSE		Identification running
Bit 02	TRUE	TRUE	Identification completed
Bit 03	FALSE		Identification failed

No errors detected by the "Cable Check" function

The i700 servo inverter then goes to the **Operation enabled** state.

- Deactivate the i700 servo inverter, e.g. via the control word. This can be carried out via the EASY Starter with keys F8 / F9.
- Deactivate the **Cable Check** function via the 0x2825 parameter and set a different selection than **15** .

Errors detected by the "Cable Check" function

If an error is detected by the **Cable Check** function, the i700 servo inverter automatically changes to the **Fault** state.

Error correction

1. Consult the logbook to identify the potential causes of the error.
2. Switch off the power supply and the 24V supply of the device.
3. Check the wiring and correct it, if necessary.
4. Perform the test again.

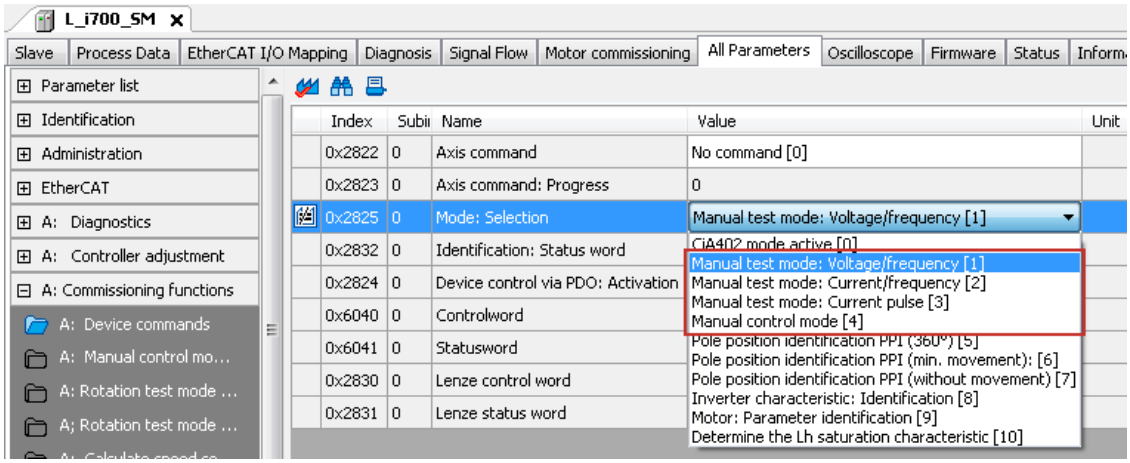
Note!

The "Identification" status word (0x2832) is also used by other functions. The display is therefore only valid as long as no other function utilising this status word is used.

Further options for carrying out a wiring test

For this purpose, you can activate the following test modes with the controller command **Operating mode (0x2825 / 0x3025)**:

- Test mode: Voltage/frequency
- Test mode: Current/frequency
- Test mode: Current pulse
- Executing manual control (📖 72)



[7-6] Example: operating modes for i700 servo inverter, double axis

7.3.4.8 Integrating the L_SMC_AxisBasicControl function block

For operating the Servo-Inverter i700, the **L_SMC_AxisBasicControl** function block has to be integrated into the »PLC Designer« project.

The function block ...

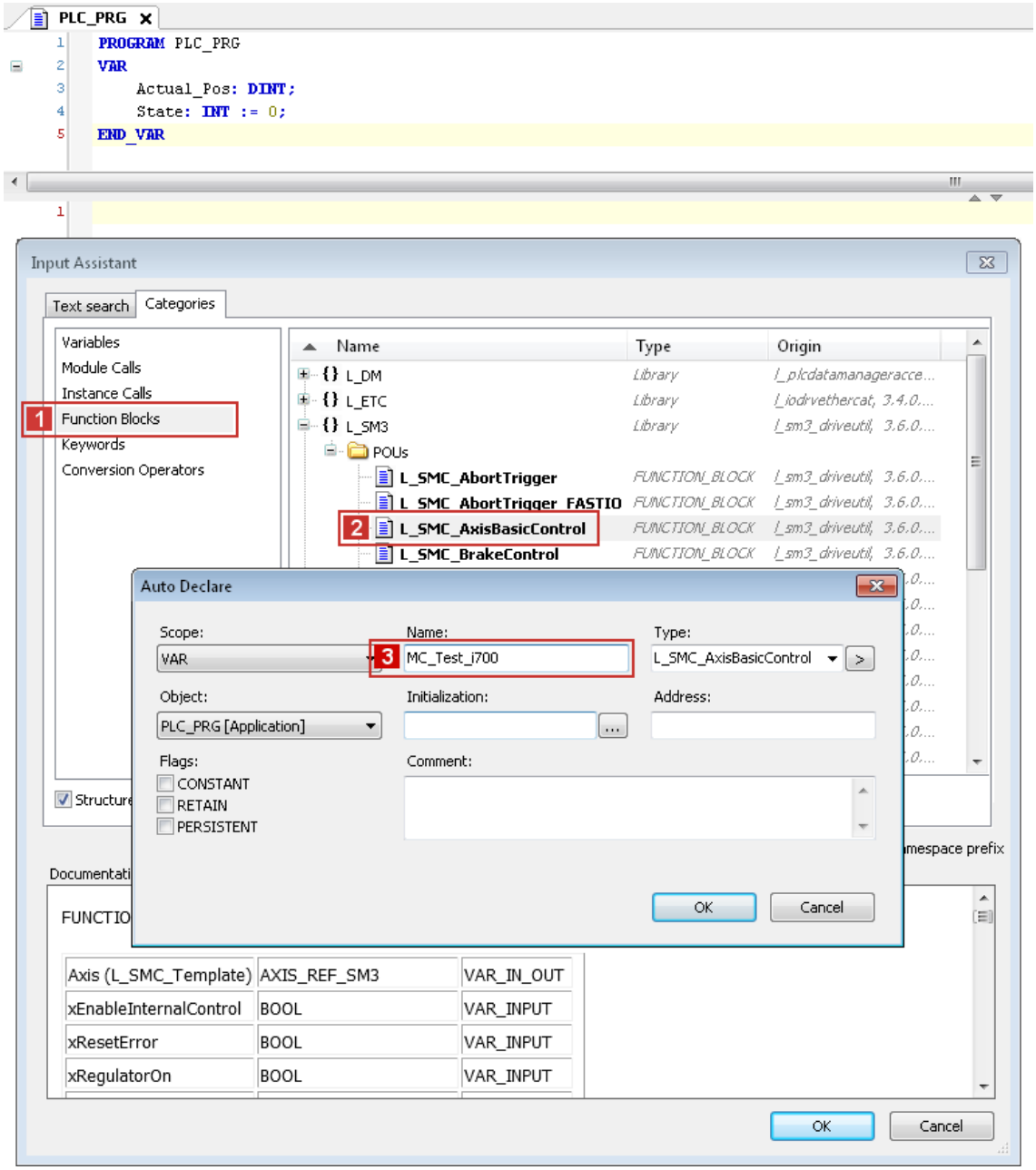
- contains various variables for drive control (e.g. for quick stop function (QSP), following error monitoring, etc.);
- is part of the **L_SM3_DriveUtil** function library.

»PLC Designer« Online help

Here you can find detailed information on the function block.

How to integrate the L_SMC_AxisBasicControl function block into the »PLC Designer« project:

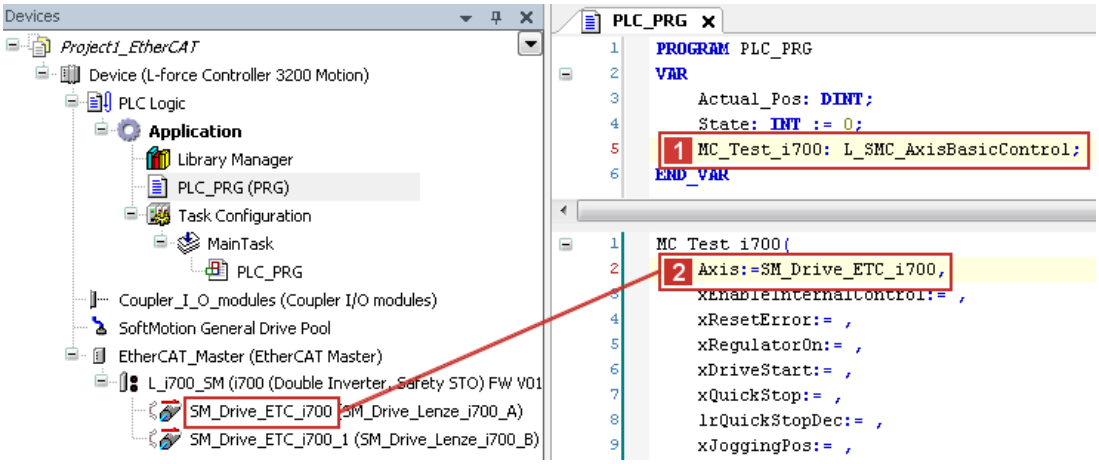
1. Open the PLC program code (PLC_PRG).
2. Open **Input assistance** in the lower input area with a right mouse click via the context menu.



3. 1 Open the category **Function blocks**.
4. 2 In the element **L_SM3**, select **POUs** and then the **L_SMC_AxisBasicControl** function block.
5. 3 In the "Declare variables" dialog box, enter a variable name (in the example above: "MC_Test_i700").

6. Close the variable declaration by clicking the **OK** button.

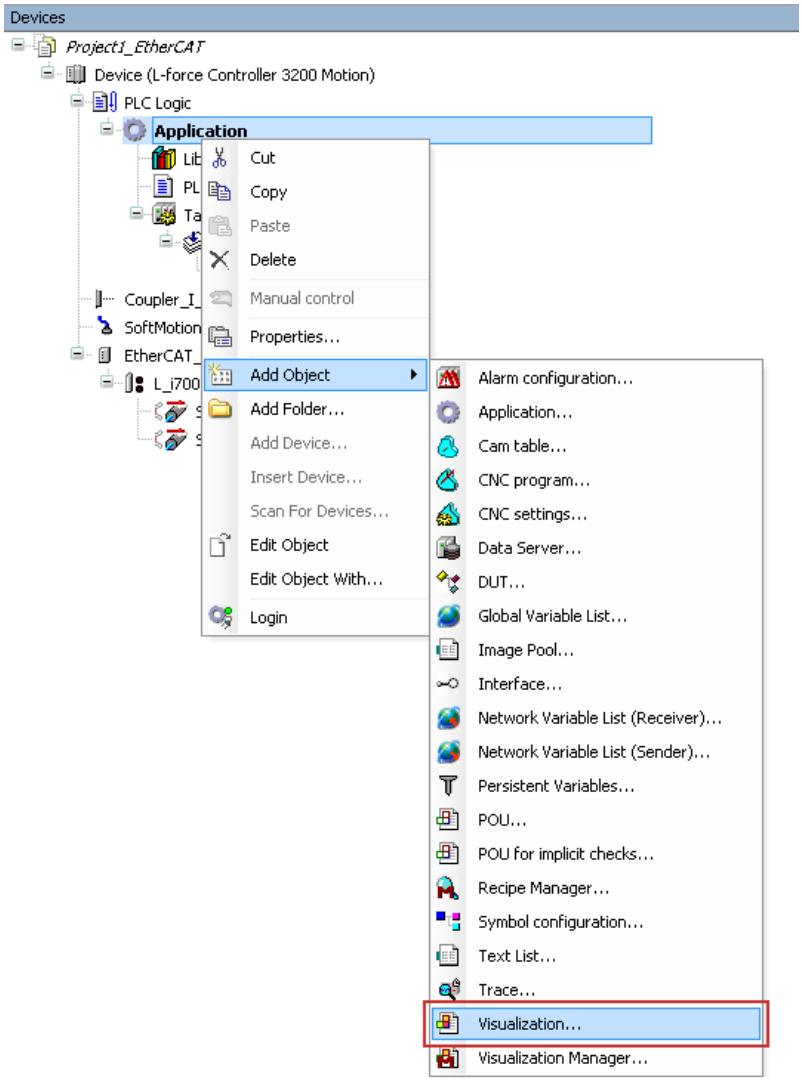
   1 The **L_SMC_AxisBasicControl** function block, together with its data structure, is integrated into the PLC program code.



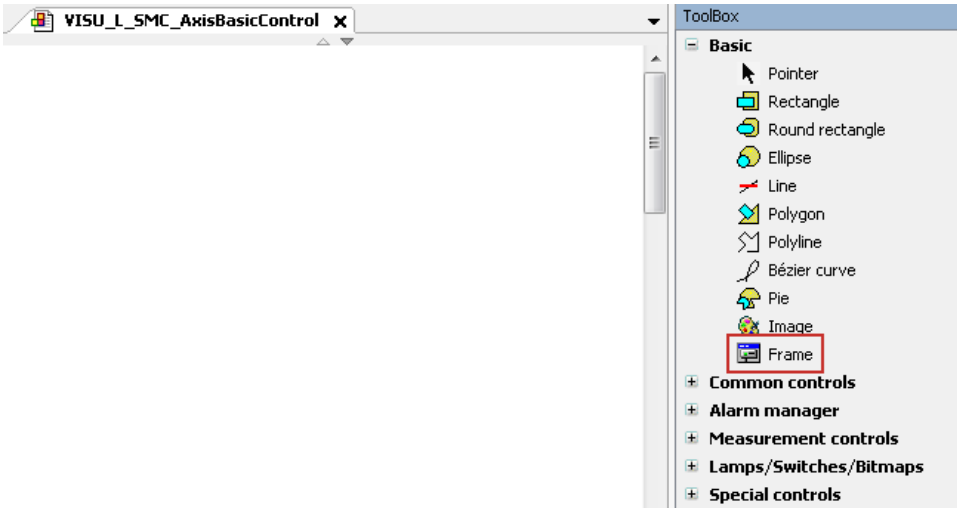
7. 2 Open the **L_SMC_AxisBasicControl** function block and set the reference to the axis data structure (in the example "Axis:= SM_Drive_ETC_i700").

8. Open the context menu for **Application**, select the command **Add object → Visualisation...** and then insert the visualisation of the function block.

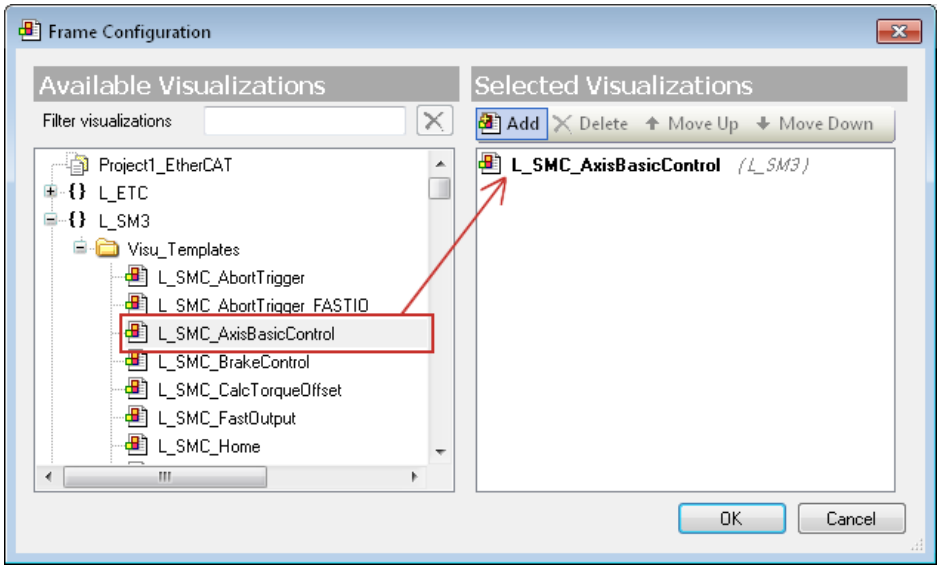
   Enter an appropriate name (e.g. "VISU_L_SMC_AxisBasicControl").



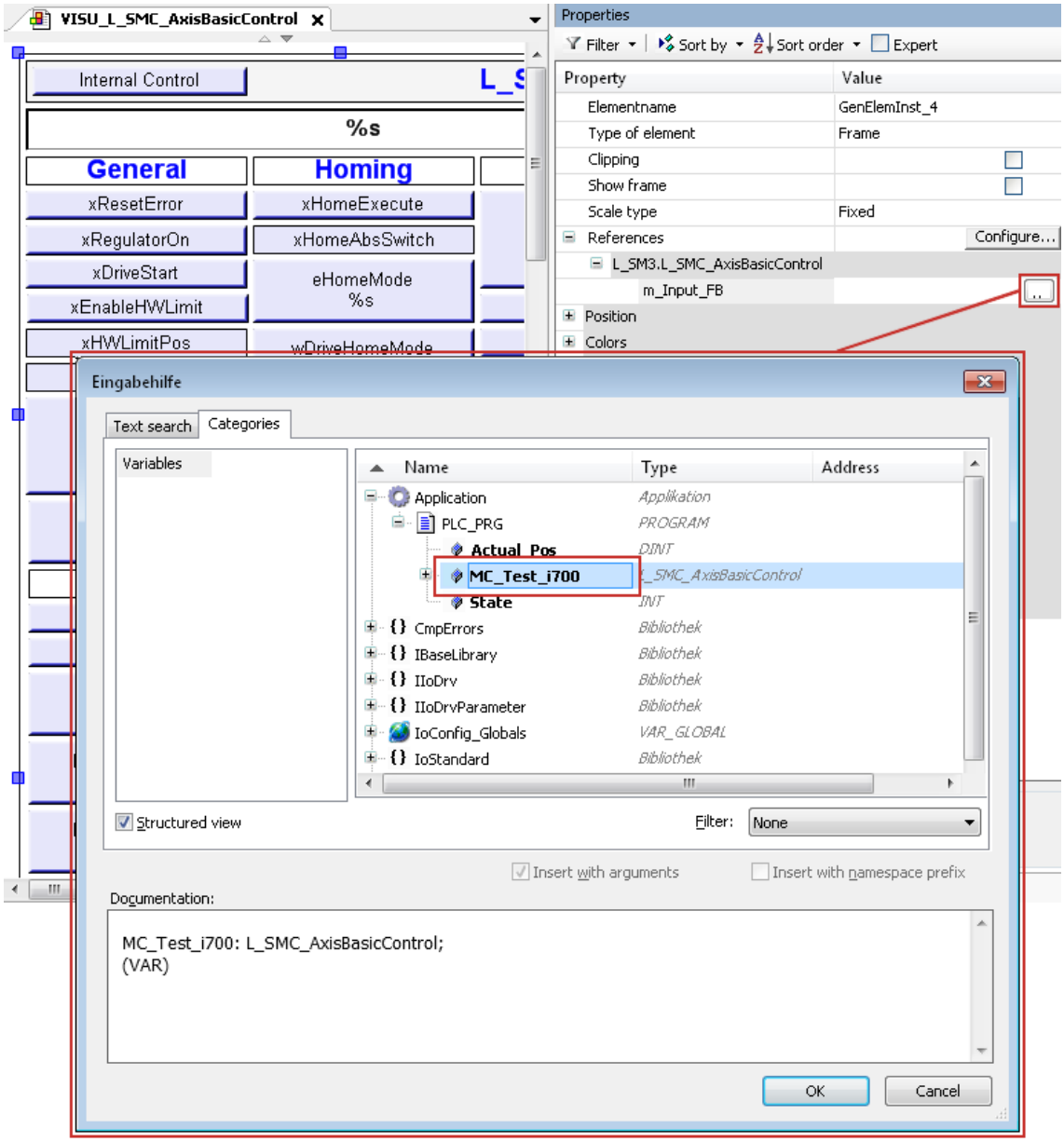
9. Insert a frame in the visualisation with the help of the **frame tool**.



10. Add the frame visualisation of the function block **L_SMC_AxisBasicControl** and close the dialog box by clicking the **OK** button.



11. Under "Properties", select the reference of the function block with which the visualisation is to be connected (in the example "MC_Test_i700").



12. Confirm your selection by clicking the **OK** button.
13. Translate the PLC program code.

    Menu command **Build→ Build** or **<F11>** function key
14. Store the »PLC Designer« project in the project folder.

    Menu command **File → Save project / Save project as ...**

7.3.4.9 Executing manual control

The purpose of manual control is to check the wiring (test mode) and to carry out a traversing movement.

Note!

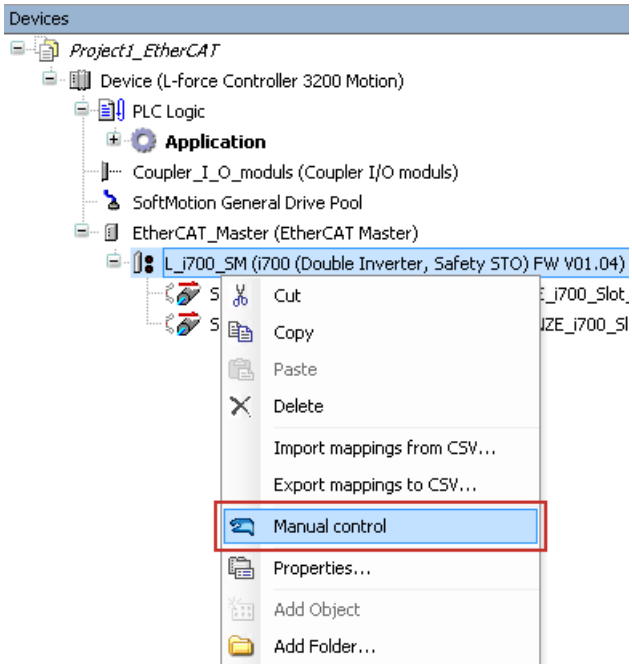
For problem-free manual control, the machine parameters – at least the gearbox factor and feed constant – must be set correctly.

Preconditions for manual control

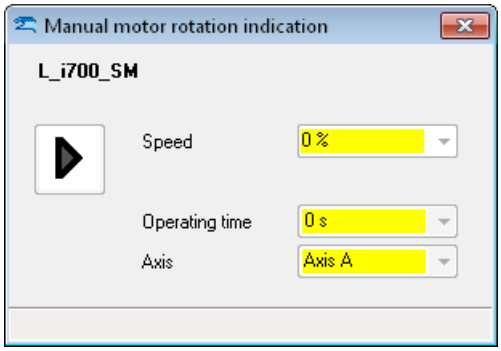
- No trouble is active.
- The mains voltage is switched on.
- The STO function (Safe Torque Off) is not active.
- The inverter drive has been inhibited by means of the software.

How to activate manual control

1. Use the menu command **Online → Login**, or log in on the Lenze Controller with **<Alt>+<F8>**.
   - For this, the PLC program must be error-free.
   - By means of the log-in, the PLC program is loaded into the controller. In doing so, any existing program is overwritten.
2. If the inverter has been enabled, inhibit it via »PLC Designer«.
3. Open the context menu of the Servo-Inverter i700 to be traversed and execute the command **Manual control**.

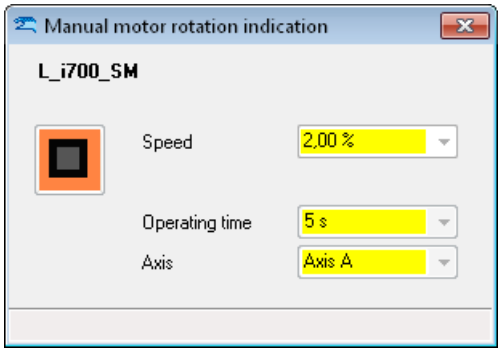


The dialog box for manual control appears:



4. Enable the inverter drive via the »PLC Designer«.
5. Enter the traversing speed (speed) in the dialog box.

   Value as a percentage of maximum motor speed (0x6080):
   - Positive % value: clockwise movement
   - Negative % value: counterclockwise movement
6. Click the button to start manual mode.



Manual mode can be stopped by clicking the button.

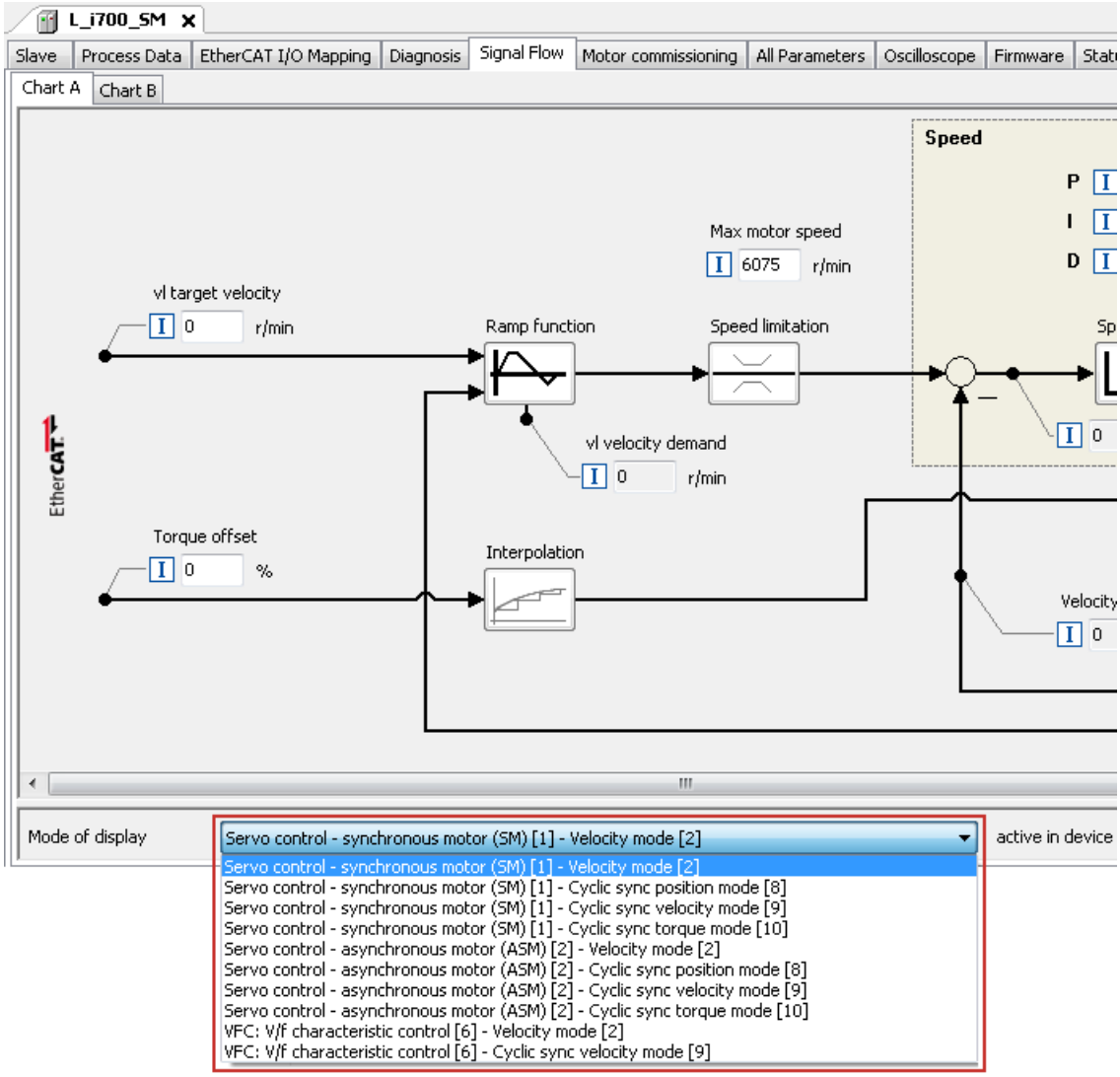
7.3.4.10 Optimising control

The final controller settings are carried out online during commissioning with load on the real machine.

How to optimise the control system

1. Use the menu command **Online → Login**, or log in on the Lenze Controller with **<Alt>+<F8>**.
   - For this, the PLC program must be error-free.
   - By means of the log-in, the PLC program is loaded into the controller. In doing so, any existing program is overwritten.
2. Press the **<F5>** function key to start the PLC program.
3. Open the tab **Signal flow → Chart A** and select the display mode (operating mode).

   The corresponding signal flow diagram is shown with the current parameter values.



4. Adapt parameter values in order to optimise the control system.
   - In the signal flow diagrams, you can adapt some parameter values directly in the corresponding input fields.
   - In addition, you can adapt parameter values for particular functions by means of function buttons, depending on the operating mode being used (see table below).

Functions and buttons in the signal flow diagrams

Function	Button	Operating mode					
		Servo control SM/ASM				V/f control	
		Velocity mode	Cyclical synchr. position mode	Cyclical synchr. velocity mode	Cyclical synchr. torque mode	Velocity mode	Cyclical synchr. velocity mode
Ramp function		●				●	
Speed limitation		●	●	●		●	●
Speed controller		●	●	●			
Speed limitation					●		
Torque limitation		●	●	●	●		
Interpolation		●	●	●	●		●
Angle/position controller			●				
Field-oriented control		●	●	●	●		
Slip compensation						●	●
Oscillation damping						●	●
Load adjustment						●	●
V/f characteristic						●	●
PWM control DC braking Flying restart function						●	●

7.3.5 Commissioning other Lenze field devices

Parameterise the Servo Drives 9400, Inverter Drives 8400 and the I/O system 1000 (EPM-Sxxx) connected to the EtherCAT network) using the »Engineer« or »EASY Starter«.

EtherCAT is exclusively configured by means of the »PLC Designer«.

EtherCAT settings of the field devices which have possibly been carried out with the »Engineer«/ »EASY Starter« are overwritten.

Documentation of the Lenze field devices

Here you are provided with some detailed information relating to the commissioning of the Lenze field devices.

Tip!

We recommend to commission each field device individually and then integrate them into the PLC program.

There already exist sample projects (device application + PLC program) for commissioning of Lenze inverters.

▸ Sample projects (Application Samples) (📖 40)

7.3.6 Creating a PLC program with a target system (Logic/Motion)

The »PLC Designer« serves to model the network topology in the control configuration.

Tip!

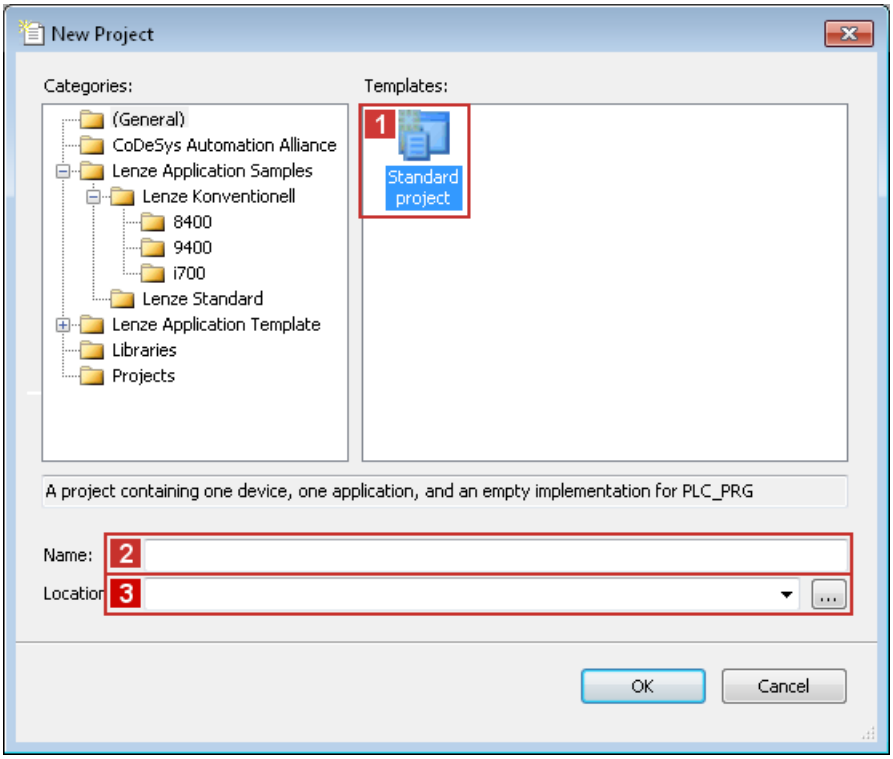
In »PLC Designer«, EtherCAT nodes as well as nodes of other fieldbus systems can be configured.

▸ Mixed operation - EtherCAT with other bus systems (📖 122)

How to create a PLC program in the »PLC Designer«:

1. Use the menu command **File → New project** to create a new »PLC Designer« project.
2. Select "Standard project" in the New project 1 dialog box.

   A "Standard project" simplifies the structure of a project in the »PLC Designer«; for instance, a device tree structure with a target system, PLC logic, etc. is provided.

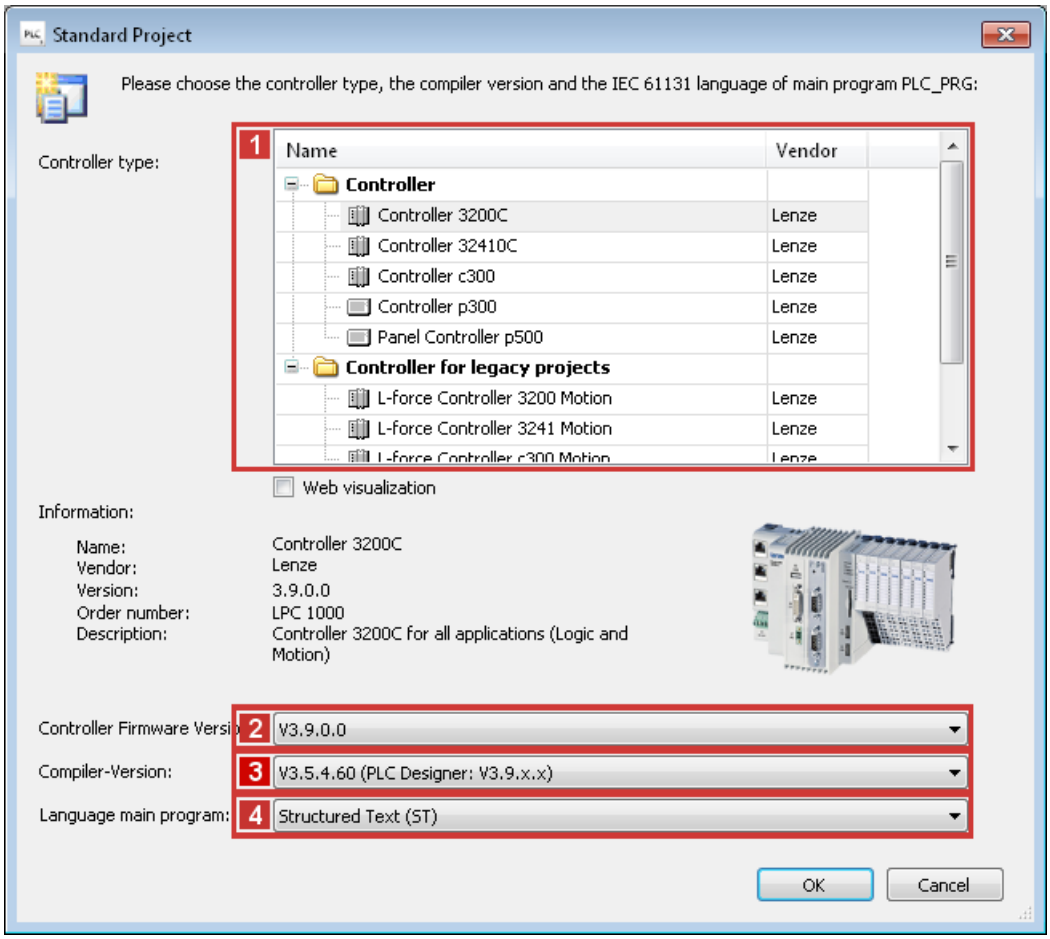


   - Enter a name for your project in the 2 **Name** »PLC Designer«input field.
   - Select the previously created project folder as storage location in the 3 **Location** selection field.

   ▸ Create a project folder (📖 45)

3. Confirm the entries with **OK**.

4. Go to the "Standard Project" dialog box and select the target system in the 1 **Controller type** selection field:



Further optional project settings

2 Selection of the Controller firmware version

3 Selection of the compiler version

4 Selection of the programming language:
- Sequential function chart (AS)
- Instruction list (AWL)
- Continuous Function Chart (CFC)
- Function block diagram (FUP)
- Ladder diagram (KOP)
- Structured text (ST)

5. Confirm the selection by clicking **OK**.

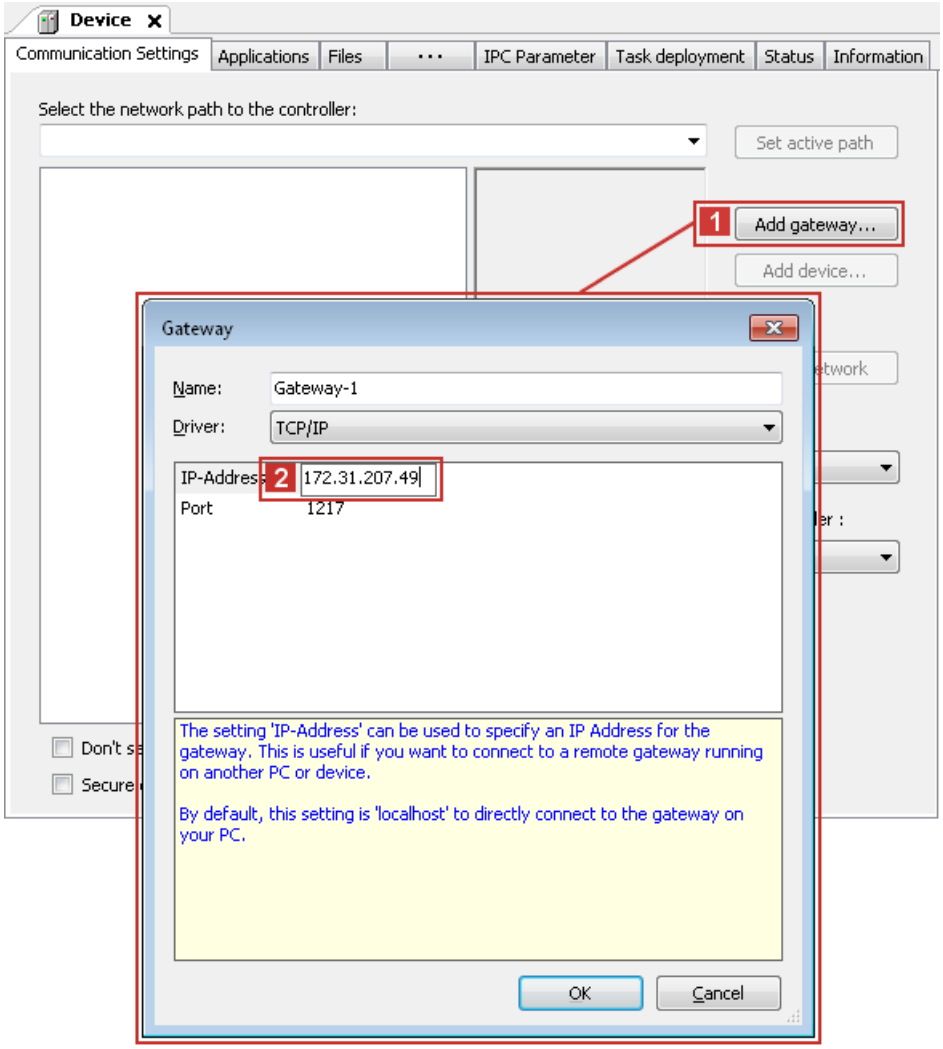
7.3.7 Configuring the communication parameters

Set the communication parameters in order to be able to carry out a fieldbus scan at a later time or in order to be able to establish an online connection to the Lenze Controller .

How to configure the communication parameters

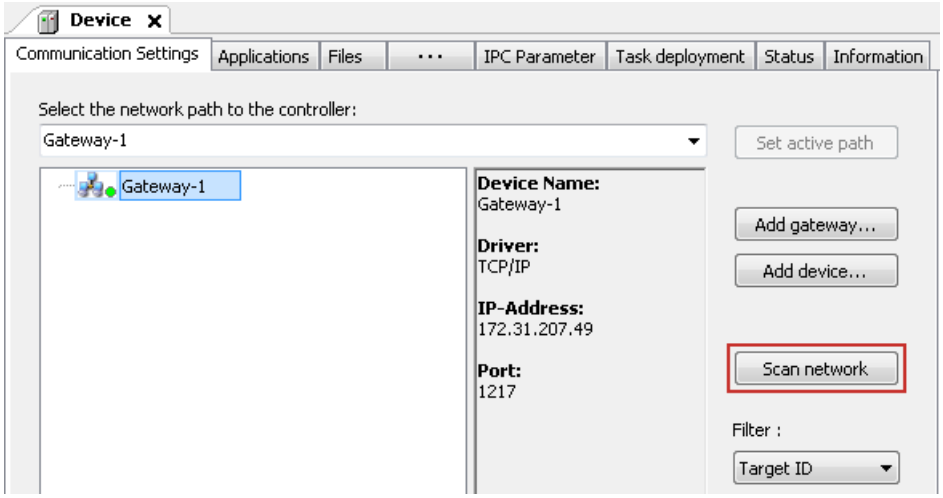
1. Go to the **Communication settings** tab of the target system (device) and click the 1 **Add gateway** button.

   Then go to the Gateway dialog box and enter the 2 **IP address** of the controller. (By double-clicking the predefined value it can be overwritten.)

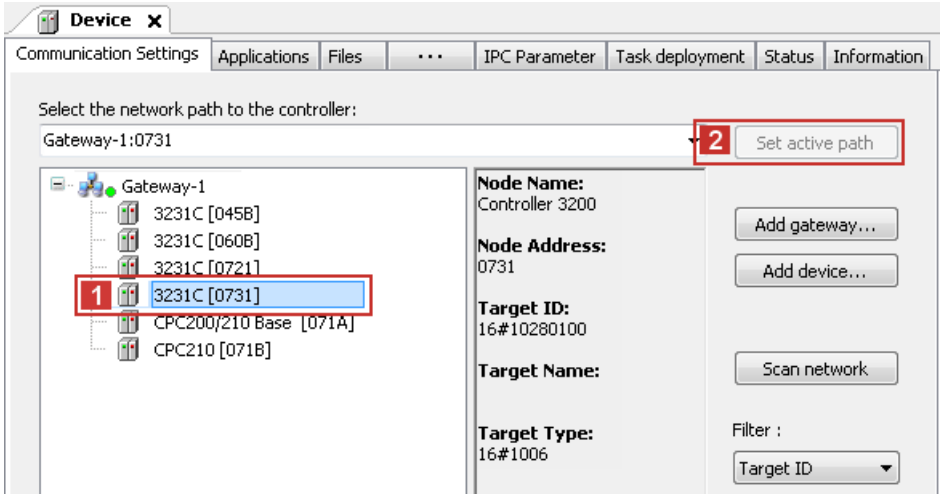


2. Confirm the entry by clicking **OK**.

3. Click the **Scan network** button.



4. Select the suitable 1 controller for the IP address entered under 2. and activate it by means of the 2 **Set active path** button (or by double-click).



5. Now you can carry out the following actions using the »PLC Designer«:
   - Determining the physical EtherCAT configuration (fieldbus scan) (📖 82)
   - Logging in on the controller with the »PLC Designer« (📖 110)

7.3.8 Determining the physical EtherCAT configuration (fieldbus scan)

In order to check the physical EtherCAT configuration, you can use »PLC Designer« to carry out a fieldbus scan on the Lenze Controller online.

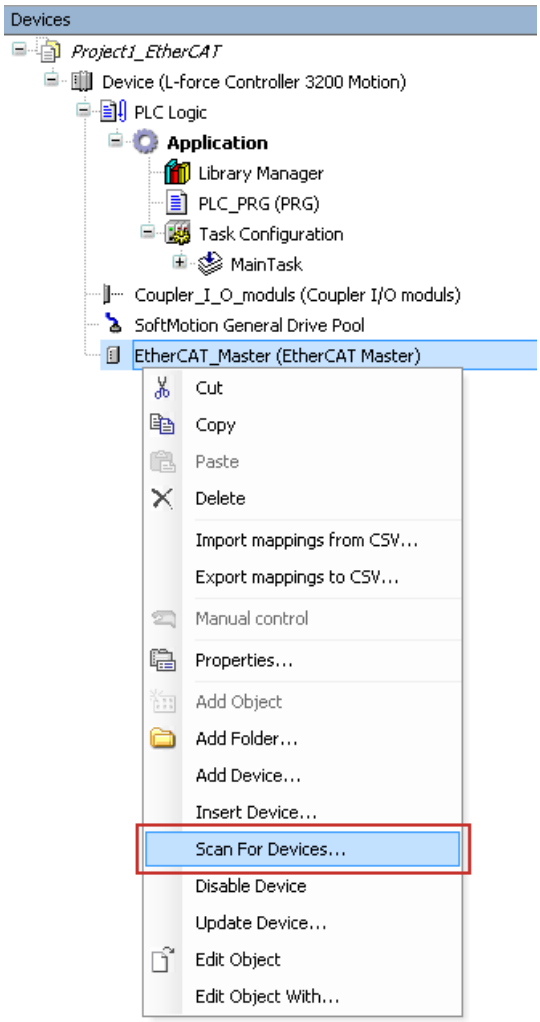
Preconditions

In order to execute a fieldbus scan, you must first configure the ...

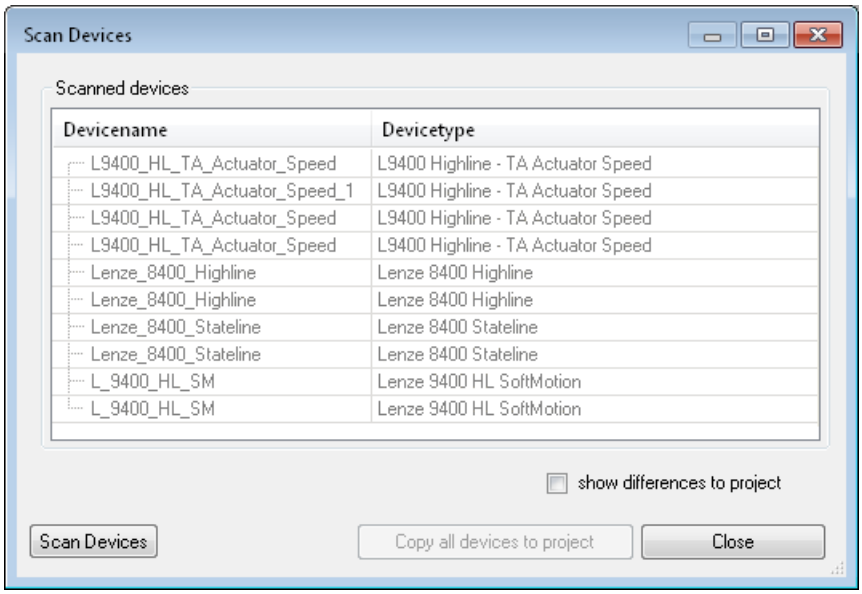
- Configuring the communication parameters (📖 80) and ...
- then use the menu command **Online → Login** or **<Alt>+<F8>** to log in on the Lenze Controller.
  - For this, the PLC program must be error-free.
  - By means of the log-in, the PLC program is loaded into the controller. In doing so, any existing program is overwritten.

How to carry out a fieldbus scan with »PLC Designer«

1. Execute the **Scan For Devices** command in the context menu of the EtherCAT master.



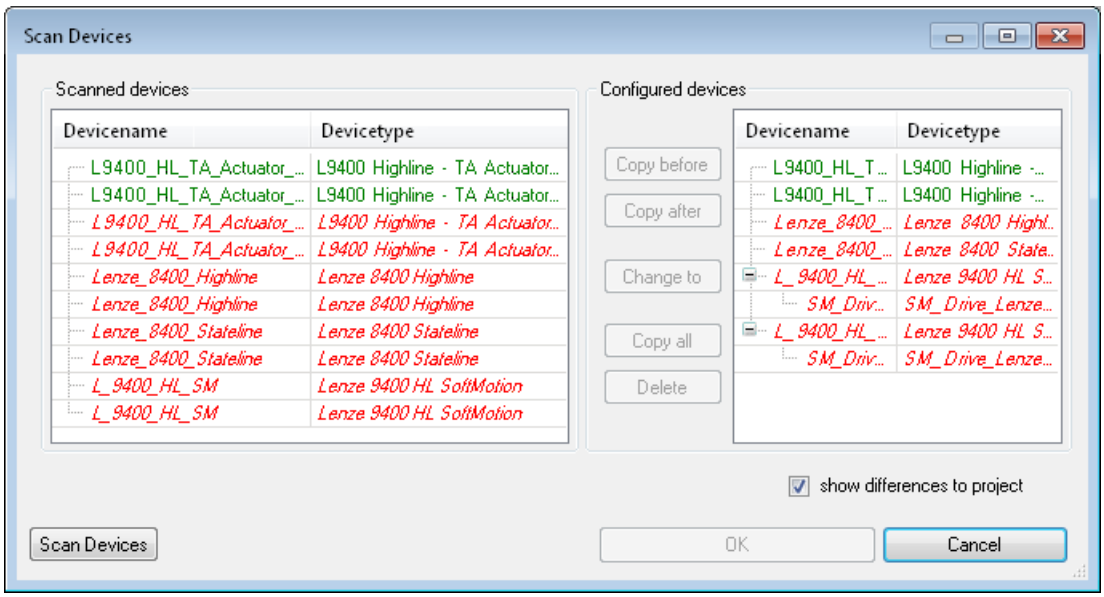
2. »PLC Designer« determines the EtherCAT nodes available on the fieldbus. In the "Scan Devices" dialog box, the devices are listed according to their physical sequence on the fieldbus.



Now you can ...

- Click the **Copy all devices to project** button to copy all available devices into your »PLC Designer« project.
- Go to the "Device name" column and select individual devices, then use the **Copy to project** button to copy them into your »PLC Designer« project.
  (The text of the **Copy all devices to project** button changes into **"Copy to project"** if one or more devices have been selected.)

By setting the checkbox "Show differences to project", the dialog is extended. Here, the physical bus structure and the configured bus structure can be adjusted.



Note!

Set up the sequence of devices in the »PLC Designer« project so that it is identical to the physical sequence in the network. Otherwise, a "bus mismatch" occurs during downloading.

In the case of field devices shown in **green**, their position in the physical network matches the position within the »PLC Designer« configuration. If field devices are shown in **red**, this is not the case.

You now have two ways of adapting the »PLC Designer« configuration:

- If you click the **Copy all** button and then confirm by clicking the **OK** button, all devices will be incorporated into the »PLC Designer« configuration.
- You can also incorporate individual devices into the »PLC Designer« configuration or replace ones already there:
  1. Select field device under "Searched devices".
  2. Select a device under "Configured devices".
  3. Click one of the buttons that are now active: **Copy (before)**, **Copy (after)** or **Replace with**.

Note!

The change in the device configuration only comes into effect after renewed translation of the »PLC Designer«-project:

1. Log out: menu command **Online → Logout** or **<Ctrl>+<F8>**
2. Translate: menu command **Build→ Build** or **<F11>**
3. Log in: menu command **Online → Login** or **<Alt>+<F8>**

Only then will all EtherCAT slaves be initialised.

Missing device descriptions

If a device available at the fieldbus is not present, an error message within the "Scan Devices" dialog box will inform you about it:

- The device cannot be interpolated into the project as the corresponding device description has not been installed.
- In order to install the device in the »PLC Designer«, the suitable device description file is required. The device identification (Vendor ID, Product Code, Revision) can be helpful to identify the device (see also Lenze EtherCAT product codes (📖 27)).

▸ Importing missing devices / device description files (📖 85)

7.3.9 Importing missing devices / device description files

The device description file contains the data of the fieldbus peripherals required for the master control. This file is required to program the control system.

With the »PLC Designer«, device descriptions for the following Lenze device series are installed as well:

- Servo-Inverter i700
- Servo Drives 9400
- Inverter Drives 8400
- I/O system 1000 (EPM-Sxxx)
- Fieldbus communication cards for Lenze Controllers (EtherCAT, CANopen, PROFIBUS, PROFINET)

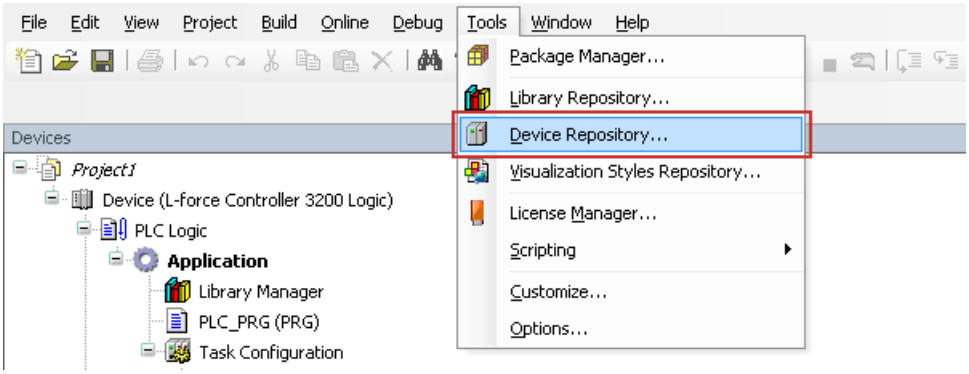
Note!

We recommend that the additionally installed EtherCAT device descriptions be retained and not replaced with the XML device descriptions for the download section at www.Lenze.com.

The installed device descriptions contain additional information on how to improve usability (pictograms etc.); this information is not contained in the XML files.

In order to furthermore integrate missing devices or devices of other manufacturers, the corresponding device description files of the manufacturer are required.

In the »PLC Designer« you can import device description files of the *.XML, *.devdesc.XML, *.EDS, *.DCF, and *.GSx type via the menu command **Tools → Device Repository...**.



For EtherCAT slaves, select the file type "EtherCAT XML Device Description Configuration".

7.3.10 Creating a control configuration (adding field devices)

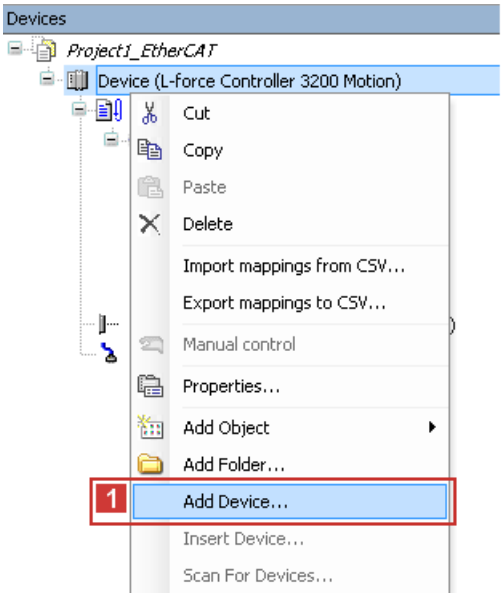
Note!

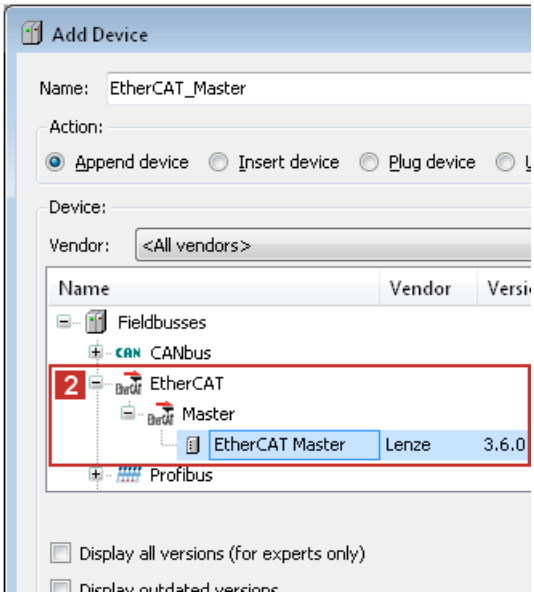
Before creating an EtherCAT configuration in »PLC Designer«, ensure that the following conditions have been met:

- The sequence of EtherCAT slaves in the device tree must correspond to the physical arrangement of the EtherCAT topology.
- SoftMotion operation is only possible with EtherCAT slaves that use the CiA402 application (SM_Drives, e.g. Servo Drives 9400 Highline CiA402).
- Select the cycle times according to the technical data, from 1 ... 10 ms.

How to create a control configuration in »PLC Designer«

1. Open the context menu of the target system and execute the command 1 **Append device** in order to extend the control configuration with 2 "EtherCAT Master".

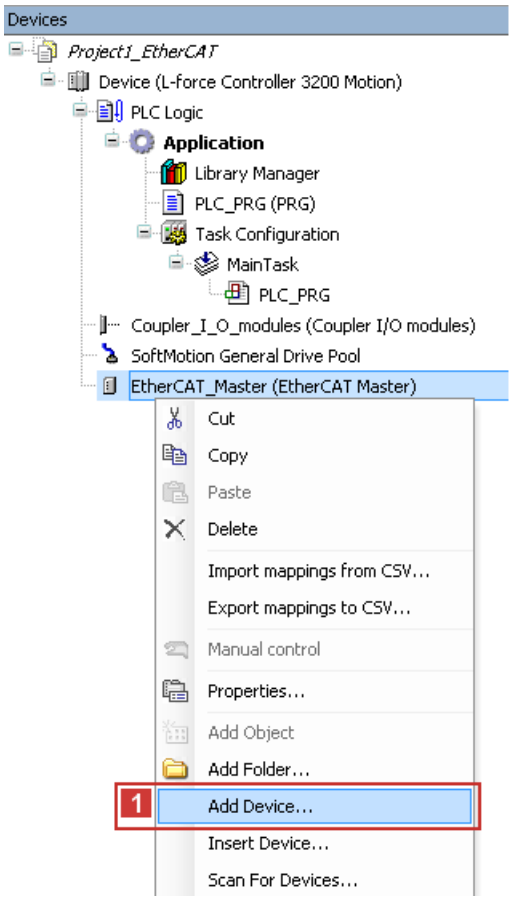


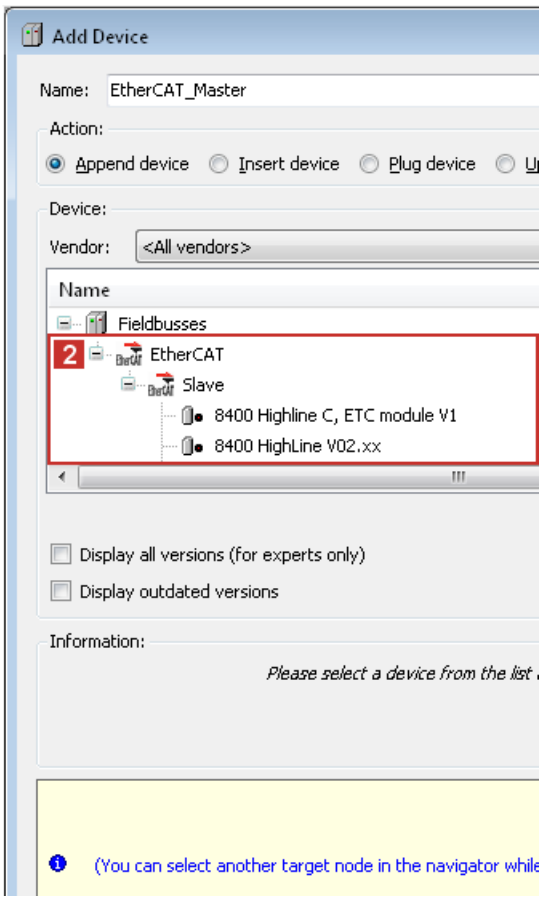


2. Append the EtherCAT slaves under the EtherCAT master.

   You have two options:

   - **Automatically** Determining the physical EtherCAT configuration (fieldbus scan) (📖 82) (before: Configuring the communication parameters (📖 80).)
   - **Manually** with the command 1 **Append device** in the context menu of the EtherCAT master





Select a field device from the 2 selection list. Only those devices can be selected whose EtherCAT device descriptions have been imported in »PLC Designer«.

▸ Importing missing devices / device description files (📖 85)

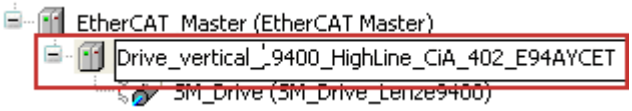
Repeat the command 1 **Append device** until all participating slaves on the field bus have been incorporated in the EtherCAT configuration.

3. Give the inserted slaves suitable names (e.g. "Drive_vertical").

   The names must ...
   - only contain the characters "A ... Z", "a ... z", "0 ... 9" or "_";
   - must not begin with a digit.

   With a mouse-click on the element or by pressing the space key, the name is enabled for entry.

   Example :



Tip!

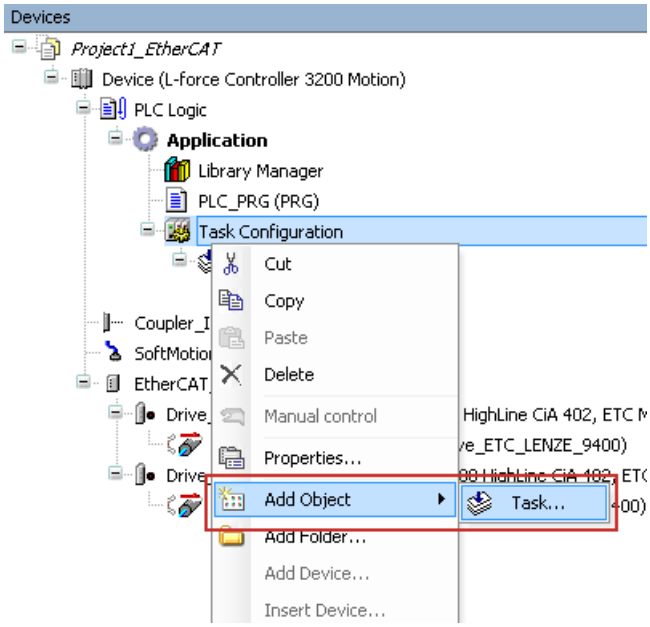
We recommend that 9400 servo drives and 8400 inverter drives be given the same designation that has been entered under the device code **C00199**.

7.3.11 Creating a task

How to create a task in »PLC Designer«

1. Open the context menu of the **Task configuration** and execute the command **Add object** in order to create a new task.

   Assign a reasonable task name, e.g. "MotionTask".

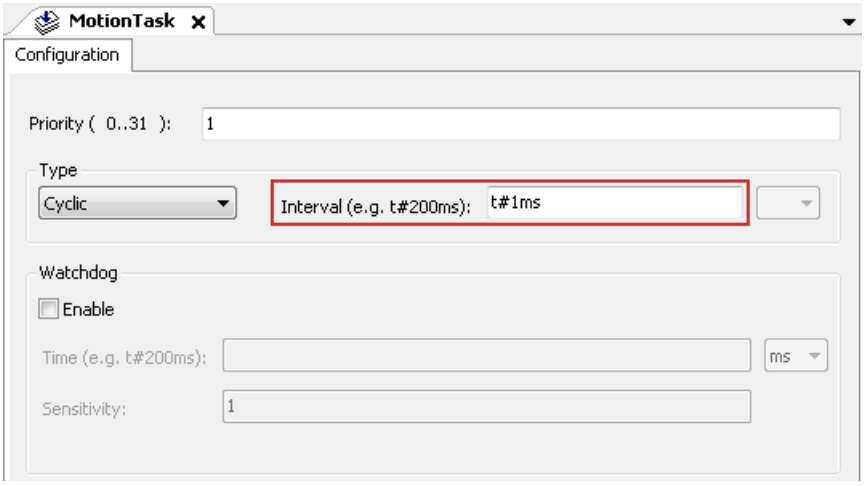


2. Enter a reasonable cycle time in milliseconds for the created task in the **Interval** input field.

Note!

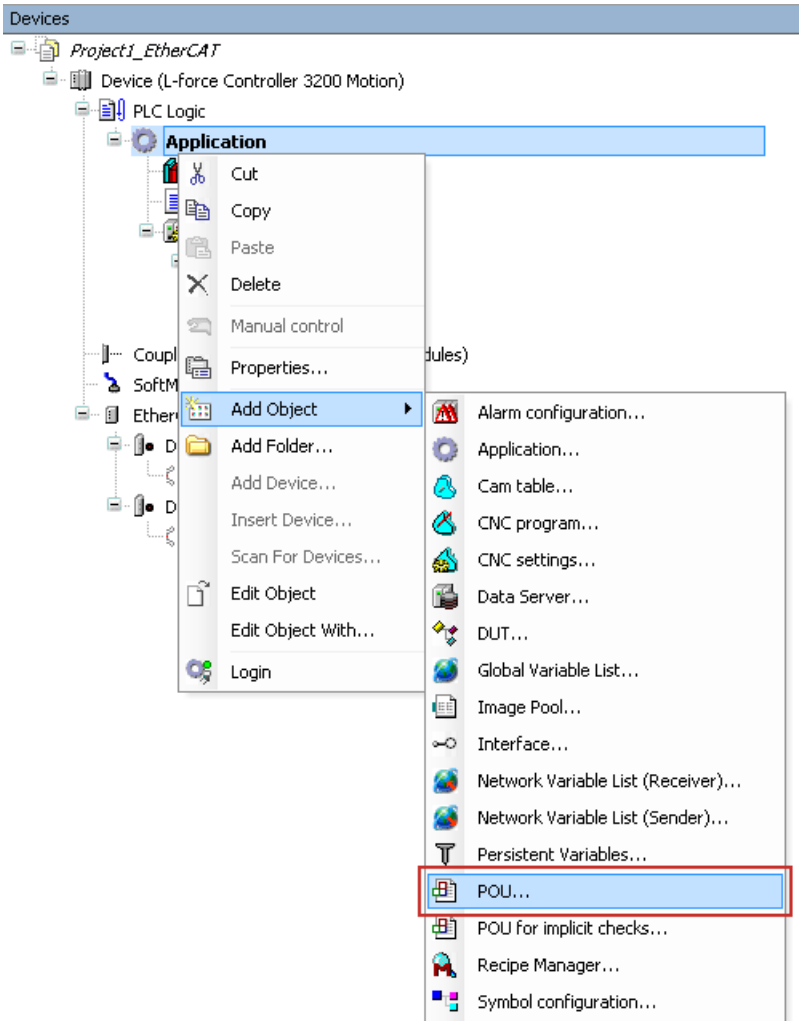
- Select a cycle time, according to the technical data, from 1 ... 10 ms.
- If "distributed clocks" (DC) are used, the task cycle time to be set must comply with the set DC cycle time.

▸ Setting a DC synchronisation (📖 94)

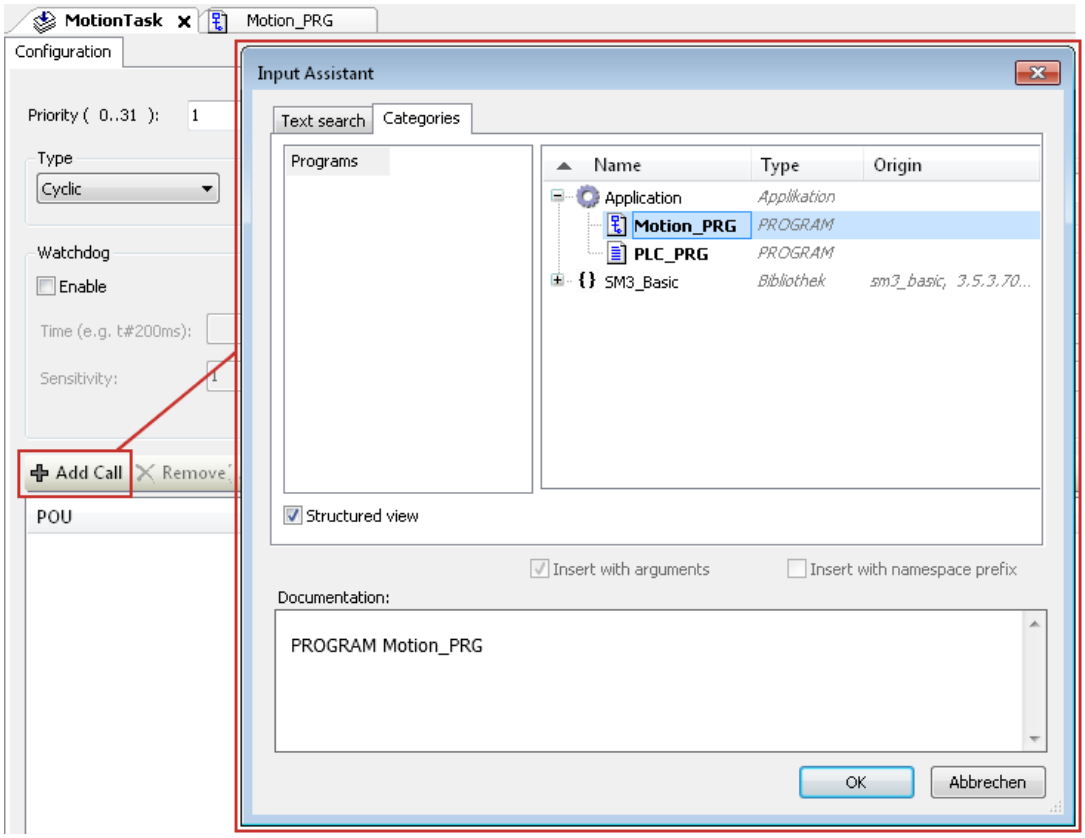


3. Open the context menu for **Application** and execute the command **Add object → POU...** in order to create a new program block (POU) in the application.

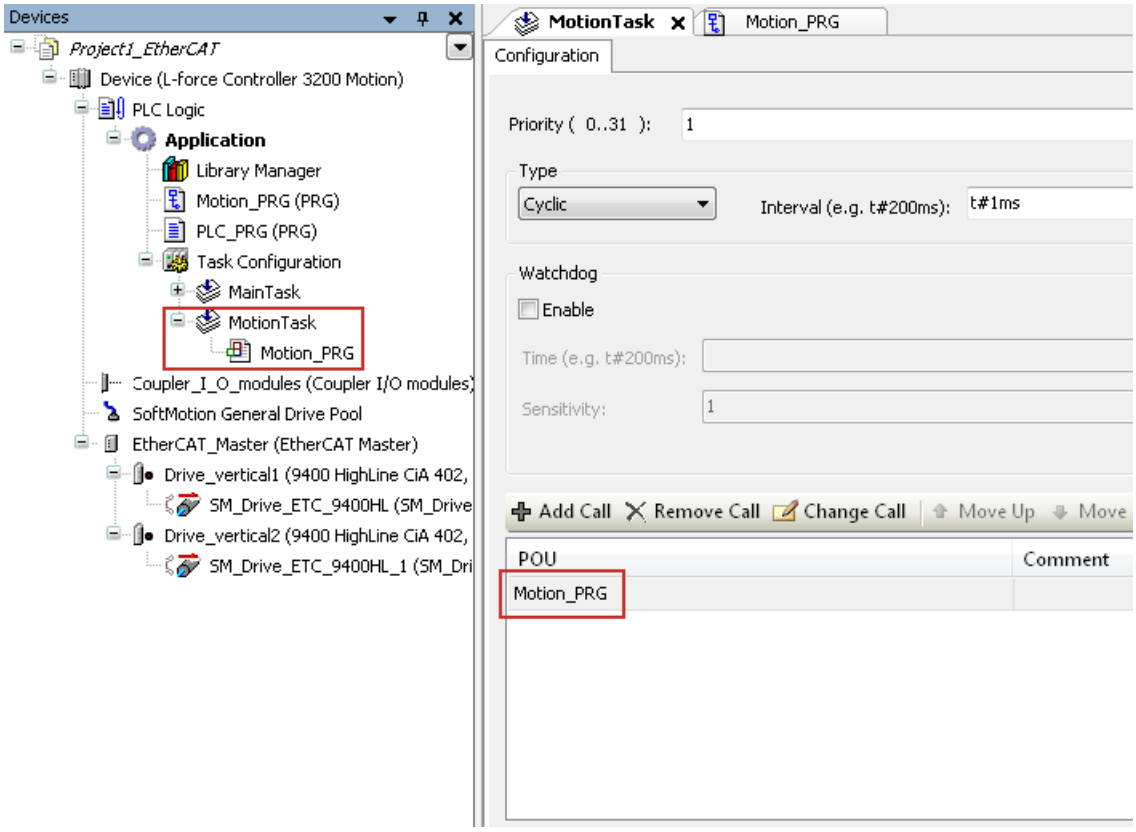
   Assign a reasonable POU name (e.g. "Motion_PRG").



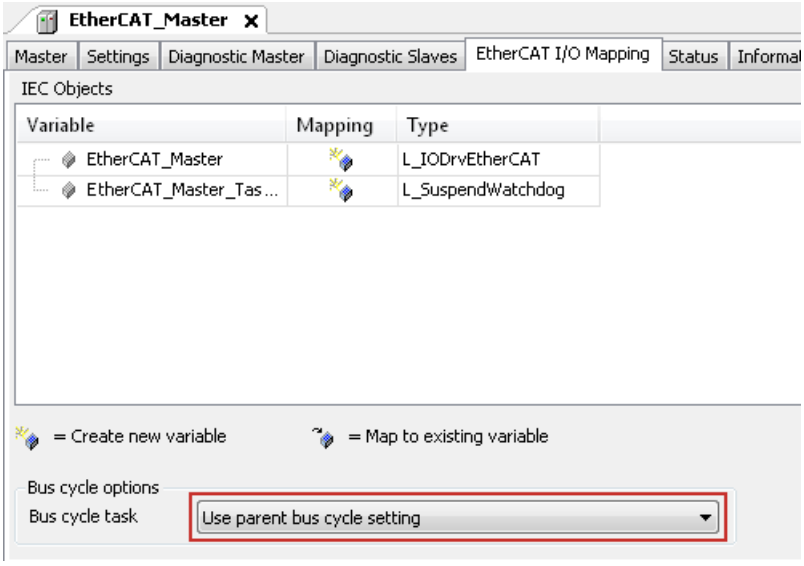
4. Click the **Add call** button to open the input assistant.
   Select the program call under "Application" and add it to the task by clicking the **OK** button.



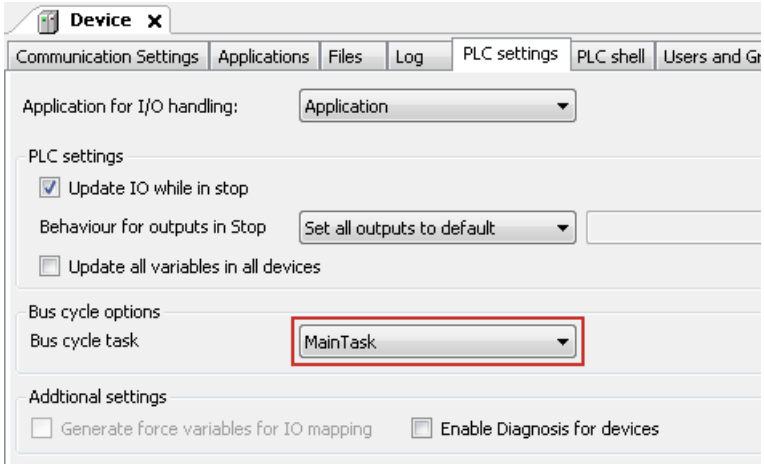
The following task configuration is caused:



5. Open the **EtherCAT I/O image** tab of the EtherCAT master and select the bus cycle task for the master (Lenze Controller).



The "Cycle settings of the higher-level bus" serve to use the bus cycle task set via the **PLC settings** tab of the Lenze Controller (device):



7.3.12 Setting a DC synchronisation

Note!

The manual configuration of the slave DC features requires detailed knowledge regarding EtherCAT and the field device. Thus, DC settings should only be made by experts.

We recommend that the basic DC settings be retained in the case of Lenze field devices in order to ensure correct DC synchronisation.

- DC synchronisation is absolutely required for Motion applications.
- DC synchronisation can also be used for Logic applications.
- Not all slaves support the DC functionality.
- In order to be able to use the DC functionality, the first slave connected to the EtherCAT master (Lenze Controller) must have **DC master capability**.
  When additional slaves are added, devices with and without DC capability can be mixed.
- The first EtherCAT slave after the Lenze Controller must be the **DC master** that supplies the other EtherCAT nodes (incl. controller) with the exact time.

Adjusting the task cycle time and DC cycle time

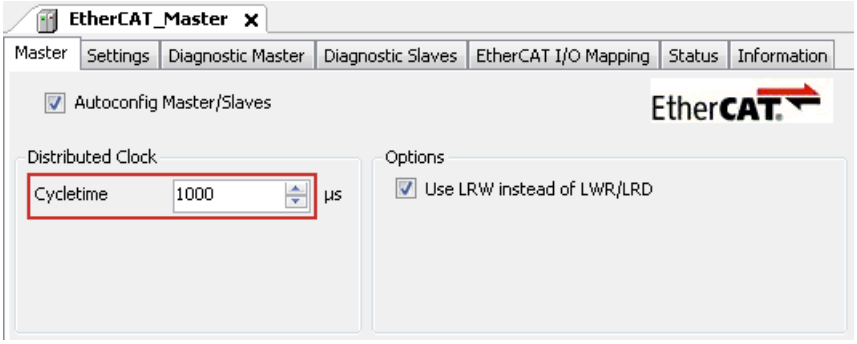
The Lenze Controller is the EtherCAT master. The clock pulse of the bus system is determined by the cycle time of the task that is assigned to the drives (slaves) integrated in the »PLC Designer«.

The task settings in the »PLC Designer« only support integer millisecond cycles and the smallest possible bus cycle is 1 millisecond. This cycle time can be defined via the »PLC Designer« in the **Task Configuration**.

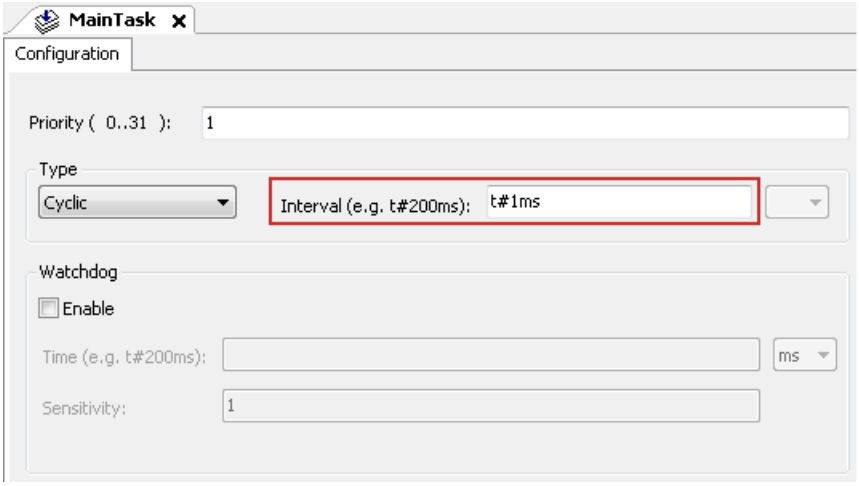
Note!

- The DC cycle time to be set must match the set task cycle time.
- Select the cycle times according to the technical data, from 1 ... 10 ms.

The DC cycle time on the tab of the EtherCAT **master**:

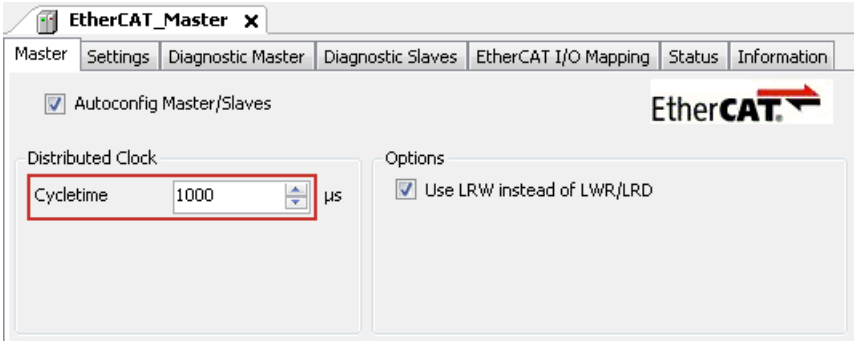


The task cycle time on the **Configuration** tab of the "MainTask":



How to set DC synchronisation:

1. Set the DC cycle time at the master (Lenze Controller) in the **Master** tab of the EtherCAT master.

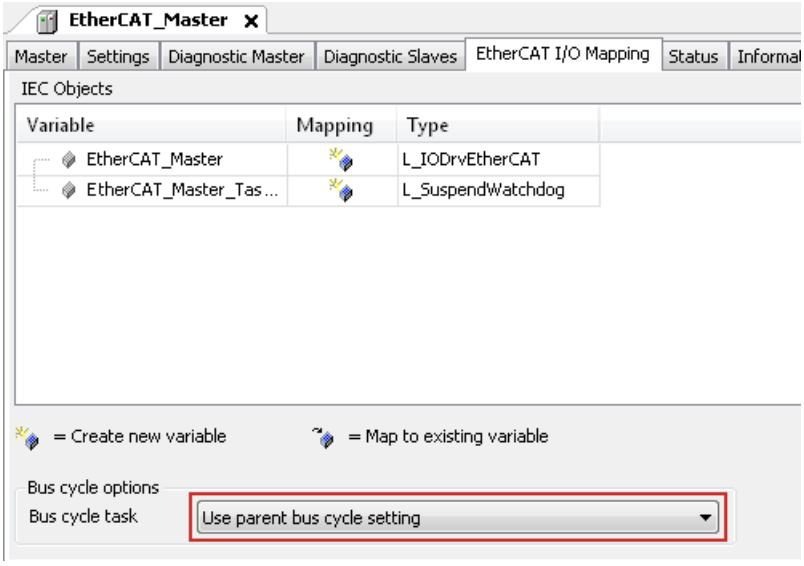


- Select the cycle times according to the technical data, from 1 ... 10 ms.
- The (basic) cycle time set here is valid for all Logic and Motion nodes synchronised by distributed clocks.
- For the Lenze field devices listed in the following table, the indicated settings must be made by means of the **»Engineer«**. The Lenze Controller does not write the values into the slave field devices.

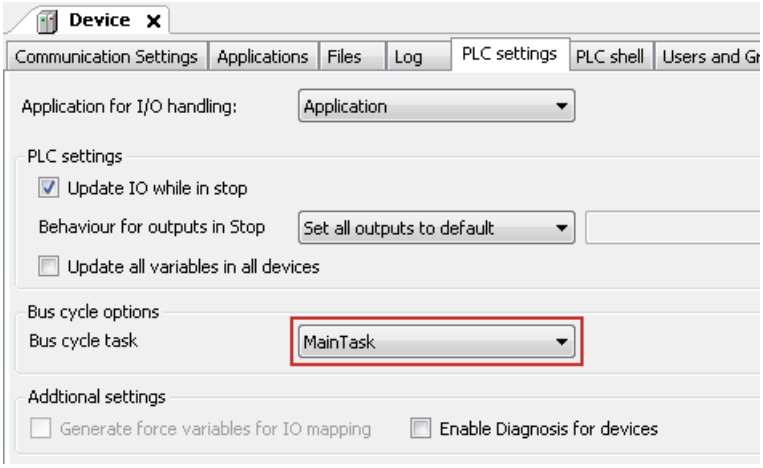
Field devices	Settings in »Engineer«
Inverter Drives 8400	• C01120 = 4 (sync source: EtherCAT module in MCI)
Servo Drives 9400	• C01120 = 4 or 5 (sync source: EtherCAT module in MXI1 or MXI2) • C013892/C14892 = 1 (process data mode = "deterministic mode")
	Note! For **Servo Drives 9400 CiA402**, C013892/C14892 (process data mode) is defined via the »PLC Designer«. See Start parameters of the Servo Drives 9400 HighLine CiA 402 (📖 110).

- In the case of the **Servo-Inverter i700**, all the parameters needed for operation are specified via the Lenze Controller (settings in the »PLC Designer«).
  ▸ Commissioning the i700 servo inverter (📖 46)
- If the DC setting and the selection of the sync source are inconsistent with each other (e.g. Servo Drives 9400: C01120 = MXI1 and "DC unused"), the devices cannot be set to the "Operational" state.
- The settings of the parameters sync cycle time (C01121), sync phase position (C01122), sync tolerance (C01123) and sync PLL increment (C01124) common for the Lenze system bus (CAN) cannot be made for EtherCAT. These values are automatically calculated by the EtherCAT communication module and set internally in the inverter.

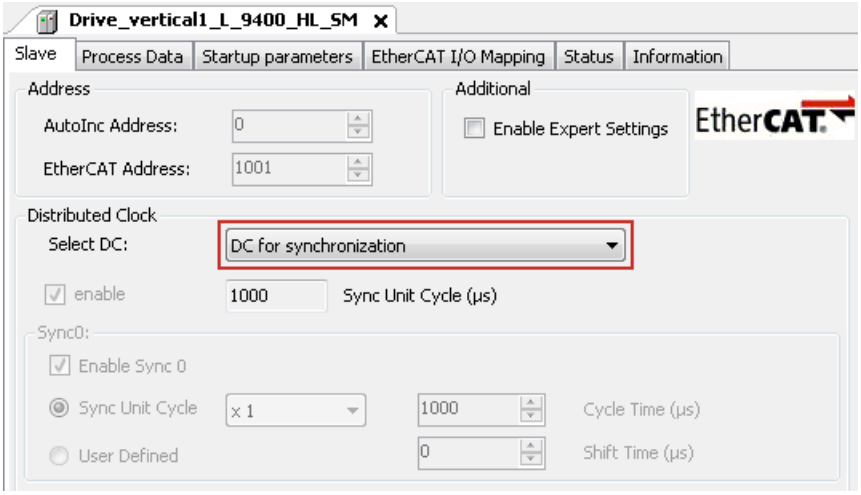
2. Open the **EtherCAT I/O image** tab and select the bus cycle task for the master (in so far as this has not yet happened in the task configuration).



The "Cycle settings of the higher-level bus" serve to use the bus cycle task set via the **PLC settings** tab of the Lenze Controller (device):



3. Select the DC functionality "DC for synchronization" in the device tree for the **first slave** (DC master) under the master (Lenze Controller).
   - The first slave connected to the master must have DC capability.
   - If a slave does not support any distributed clocks, only "DC unused" can be selected here.



4. Also select the DC functionality "DC for synchronization" for all other slave devices which are to use the DC synchronisation.

7.3.13 Setting SoftMotion parameters

Note!

In »PLC Designer«, the SoftMotion tabs are only available in the case of field devices that use a Motion application.

- i700 servo inverter
- Servo Drive 9400 Highline CiA402

The SoftMotion parameters must be set application-dependent.

In the case of the **Servo-Inverter i700**, all the parameters needed for operation are stipulated by means of the Lenze controller (settings via the »PLC Designer«).

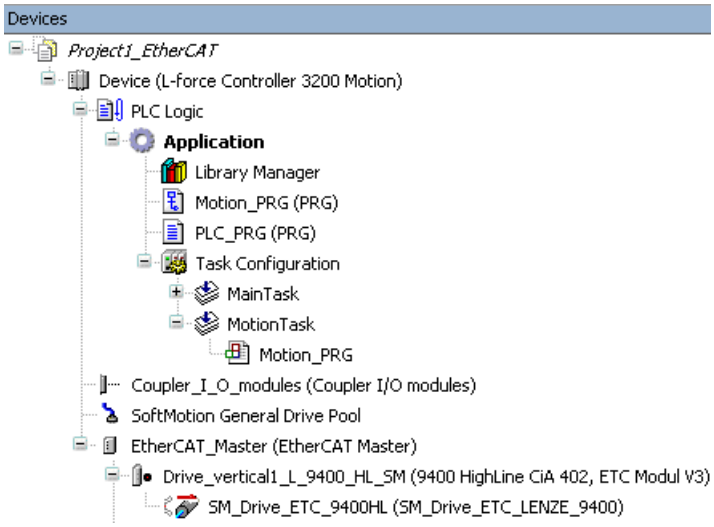
▸ Commissioning the i700 servo inverter (📖 46)

In the case of the **Servo Drive 9400 Highline CiA402** in contrast, the following parameters must be set manually via »Engineer«:

- Homing mode (C02640, set in application-dependent manner)
- Touch-probe interface (set in application-dependent manner)
- Control of the holding brake (0x60FB/2 | Brake control)
  Depending on the setting of this parameter, the holding brake is applied for a short time after the conclusion of the home position path. In order to avoid this, set bit 2 in this parameter ("disable stop": does not apply the brake at standstill).

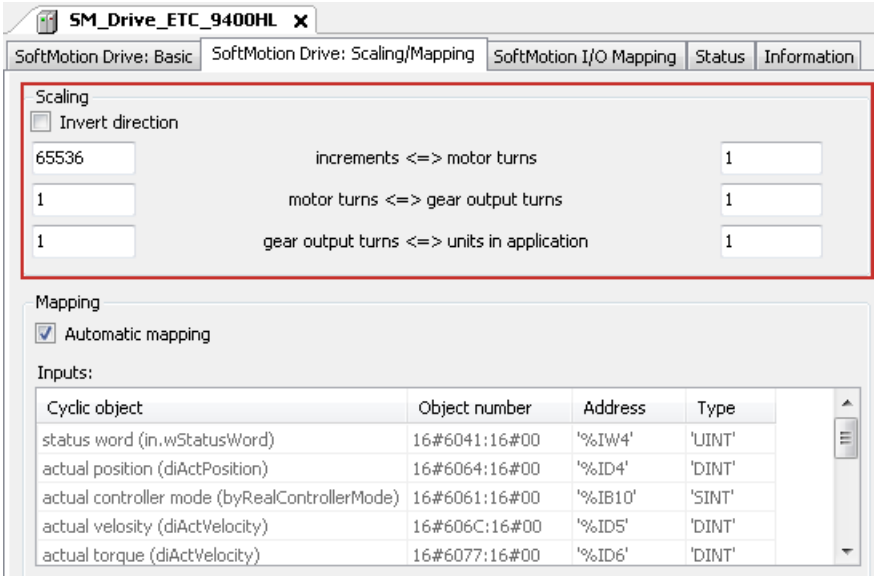
These parameters are not set via the Lenze Controller.

Example of a minimum configuration with a Motion device (Servo Drive 9400 HighLine CiA 402)



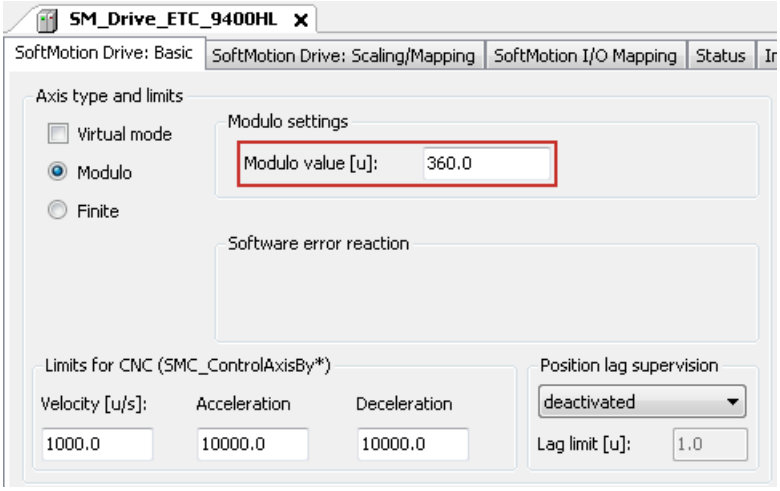
How to set the SoftMotion parameters

1. Open the tab **SoftMotion drive: Scaling/Mapping** and adapt the conversion factors in the "Scaling" area.

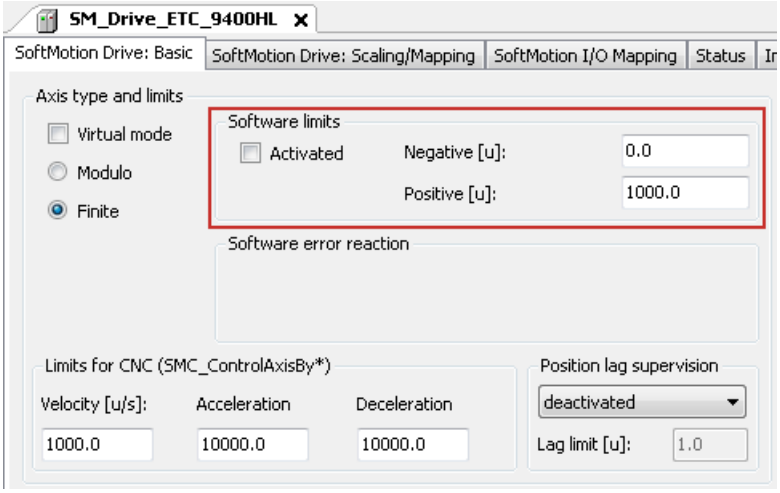


2. Open the tab **SoftMotion drive: basic parameters** and set the axis types and limitations.
   - Do not use the "virtual mode" setting.
   - Virtual axes are located in the "SoftMotion General Drive Pool".

   Configuration of a Motion device **Rotary axis**
   (type: Modulo, 360°/revolution, ratio 1:1):



   For configuring a **linear axis** Motion device (type: Finite), you can activate and determine the software limit switches:



3. Repeat steps 1 and 2 for all Motion devices connected to the field bus.

Online help of »PLC Designer«

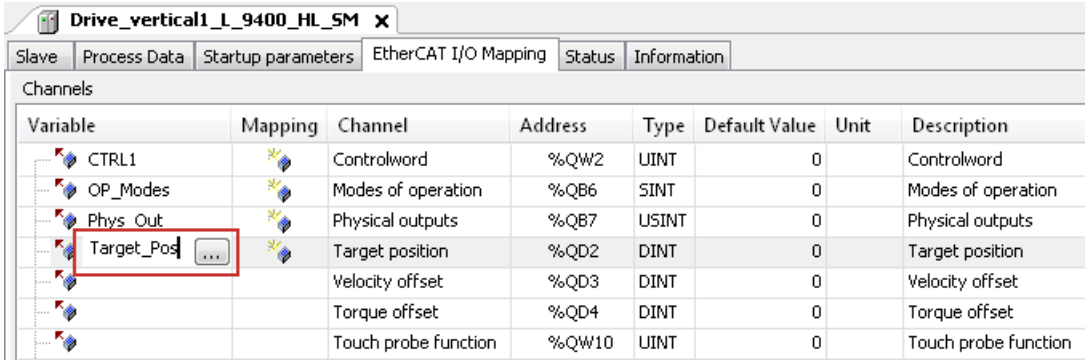
Here, you can find detailed descriptions of the **SoftMotion tabs**.

7.3.14 Processing EtherCAT I/O mapping

Note!

- If you insert more field devices in the control configuration or change the PDO mapping, the object addresses change as well (%Qxx, %Ixx). Hence, the input and output objects must be accessed via individual, **unambiguous** variables. The variable names must comply with the IEC 61131 syntax (no space and leading digits in the variable name).
- The manual assignment of object addresses in the "Address" column is not supported. Hence, only use the automatically assigned addresses of the process image. A manual assignment causes malfunctions.

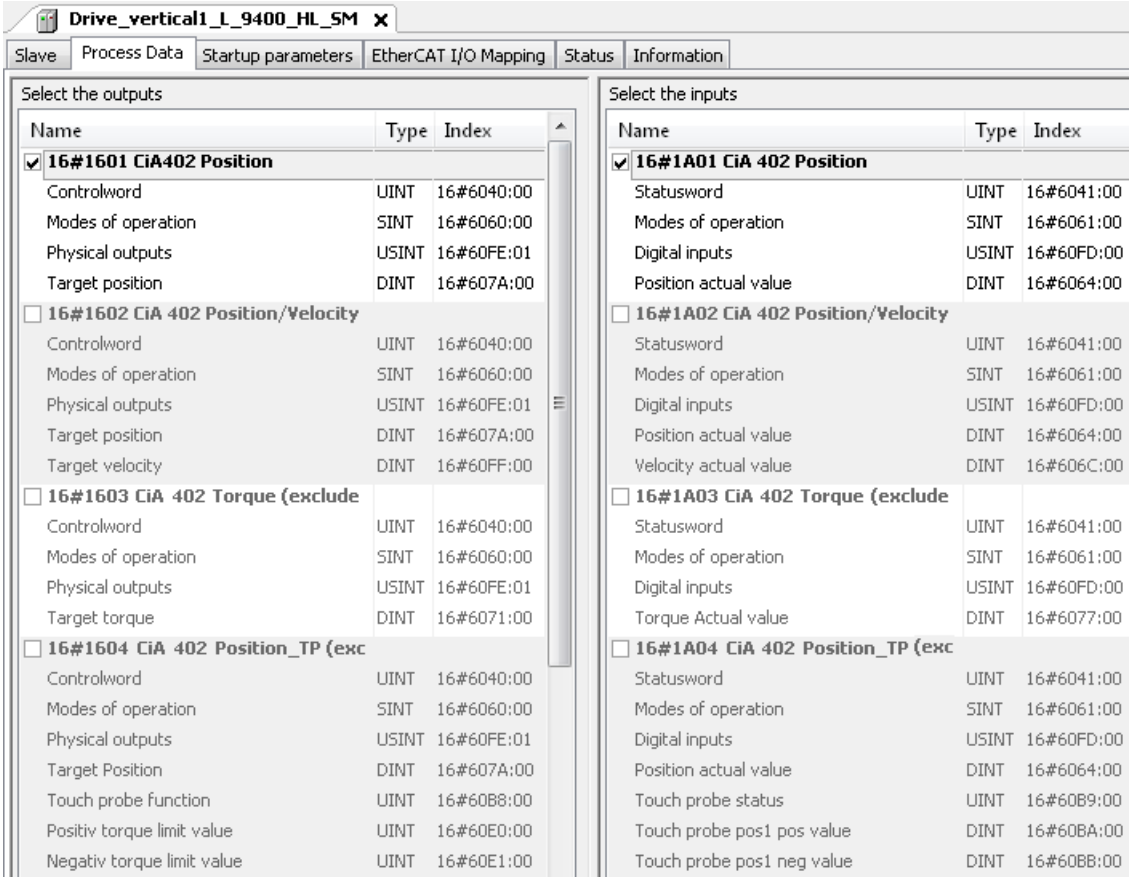
On the **EtherCAT I/O image** tab, you can enter variable names by double-clicking the variable fields or pressing the space key:



By clicking the [...] button, you can reference already existing variables (e.g. global variables from function libraries) or you can enter variable names directly in the input field and thus create system variables. The corresponding system variables for the PLC program are available.

7.3.14.1 Entering the settings for PDO mapping

You set PDO mapping by means of the **Process data** tab:



PDO mapping for Lenze inverters can be composed of three parts:

- The static part is permanent (cannot be changed) and cannot be deactivated either.
- The dynamic part contains PDOs that have been preconfigured for the different CiA402 operating modes. Depending on the CiA402 operating modes, one of these PDOs can be activated in »PLC Designer«.
- The freely configurable part can be activated in »PLC Designer« if necessary and enables individual mapping.
  ▸ Configuring individual PDO mapping (📖 104)

Some inverters only allow for one PDO mapping per direction.

The input and output PDO mappings selected must be identical (see illustration).

In order to change the setting, you must first deselect the current setting (remove checkmark in the checkbox). After that you can freely select the desired setting (set checkmark).

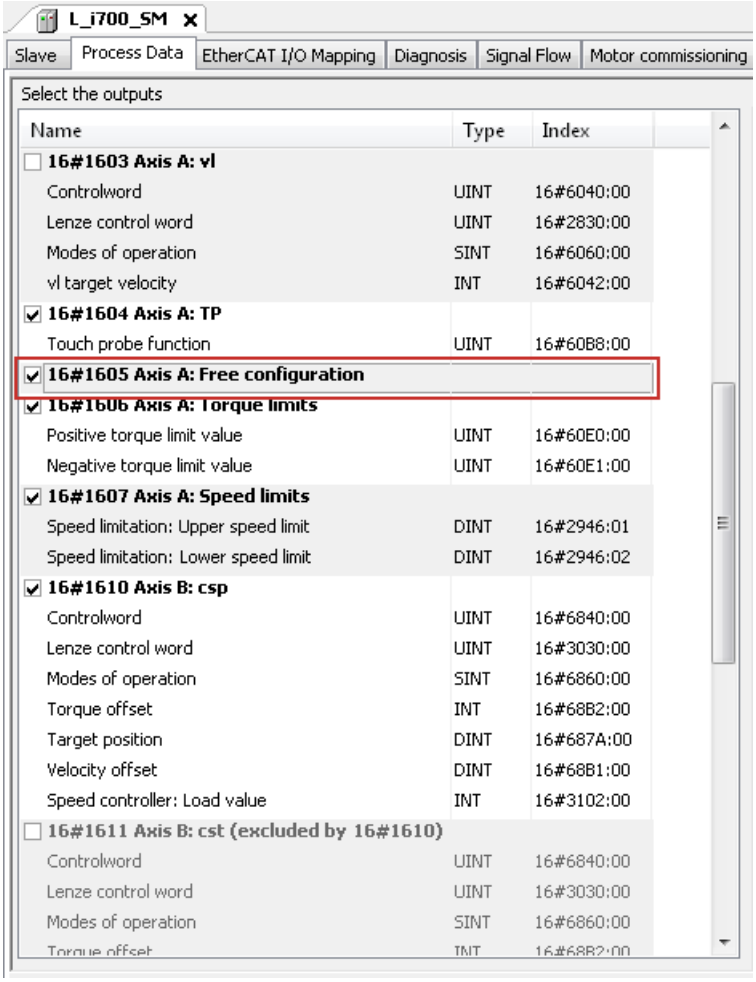
7.3.14.2 Configuring individual PDO mapping

In addition to the static and preconfigured PDO mappings, you can also configure an individual PDO mapping.

How to configure an individual PDO mapping in »PLC Designer«

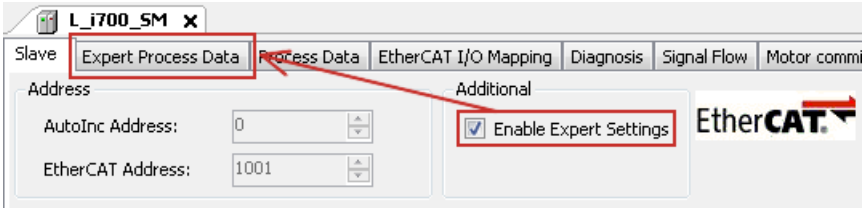
(example of an individual output PDO for the i700 servo inverter)

1. Open the **Process data** tab and set a checkmark in the checkbox of the free PDO mapping (16#1605).

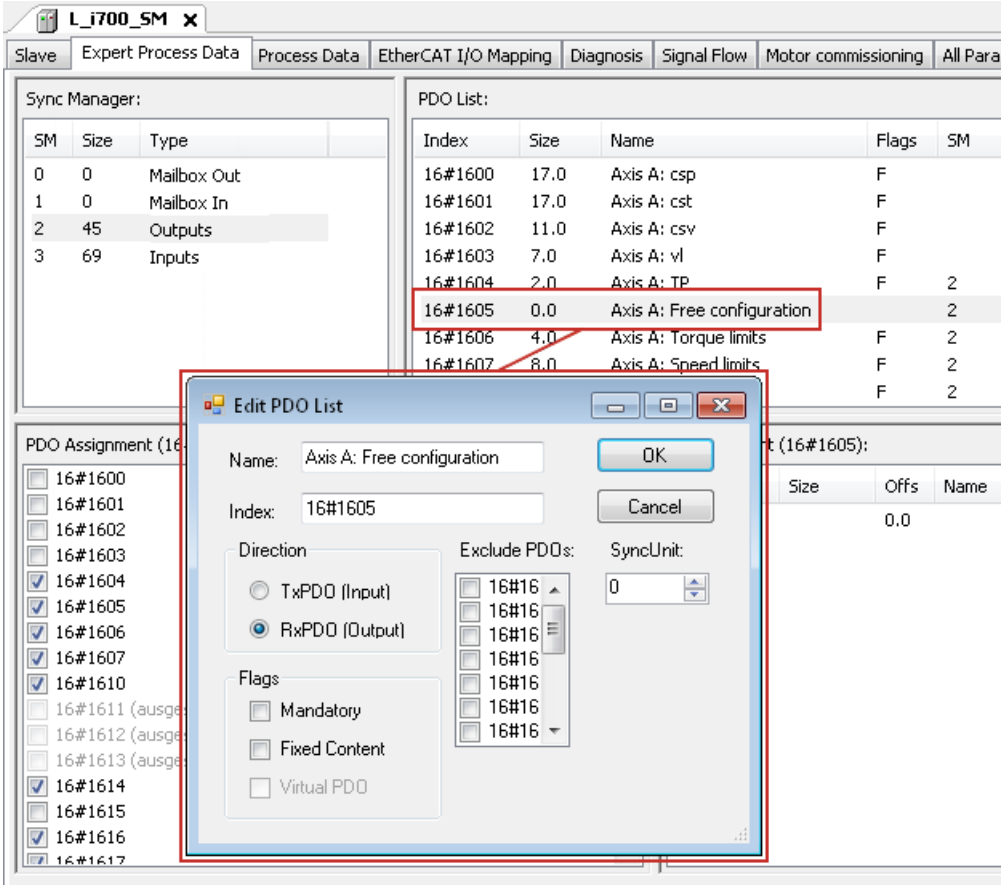


2. Activate the expert settings on the **Slave** tab of the slave drive.

   The **Expert Process Data** tab appears next to the **Slave** tab:

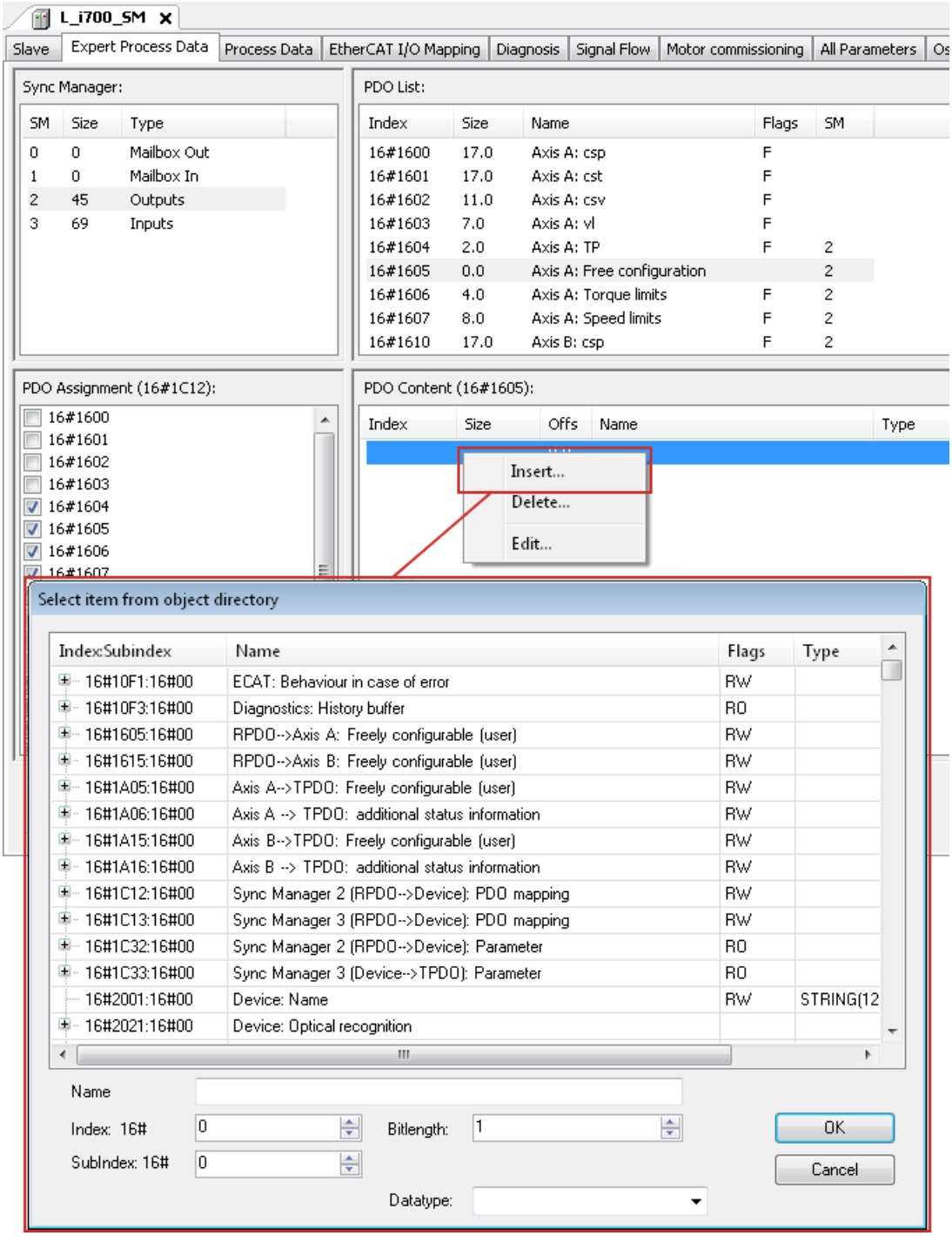


3. Open the **Expert Process Date** tab.
4. Double-click the free PDO mapping (16#1605) in the **PDO list**.



5. In the dialog box that appears, process the free PDO mapping and close the dialog box by clicking the **OK** button.

6. Open the context menu by right-clicking the **PDO Content (16#1605)** and execute the **Insert** menu command.

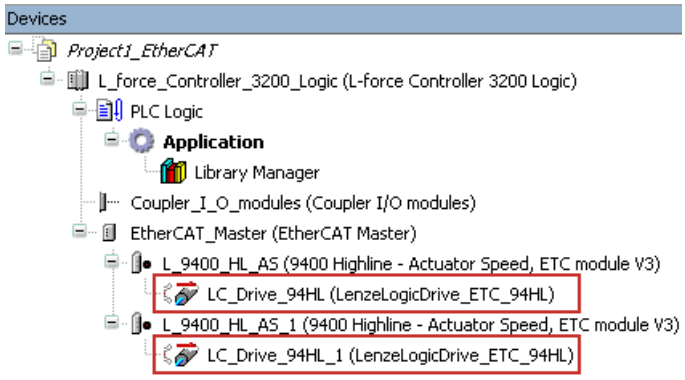


7. In the dialog box that appears, you can select a single index (parameter) or a group of indices and insert them into the **PDO Content (16#1605)** via the **OK** button.

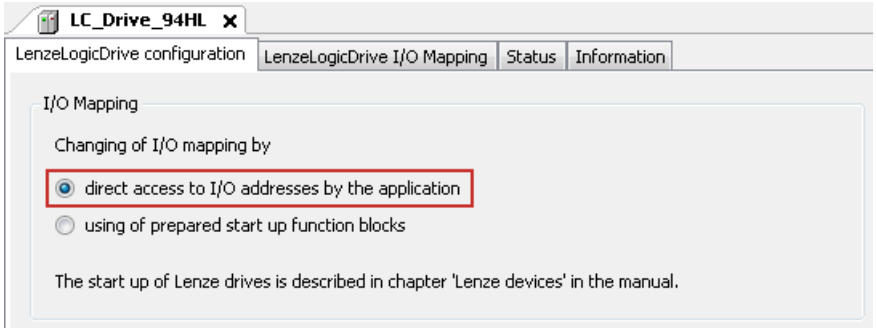
   The inserted indices (parameters) are also displayed in the free PDO mapping (16#1605) under the **process data** tab.

7.3.14.3 PDO mapping for logic devices

If the device descriptions for logic devices that are supplied with the »PLC Designer« are used, the process data are copied to the subordinate logic drive node automatically.



If the process data are still to be linked manually, activate the "direct access to I/O addresses by the application" option.



In this setting, the prepared function blocks are not usable.

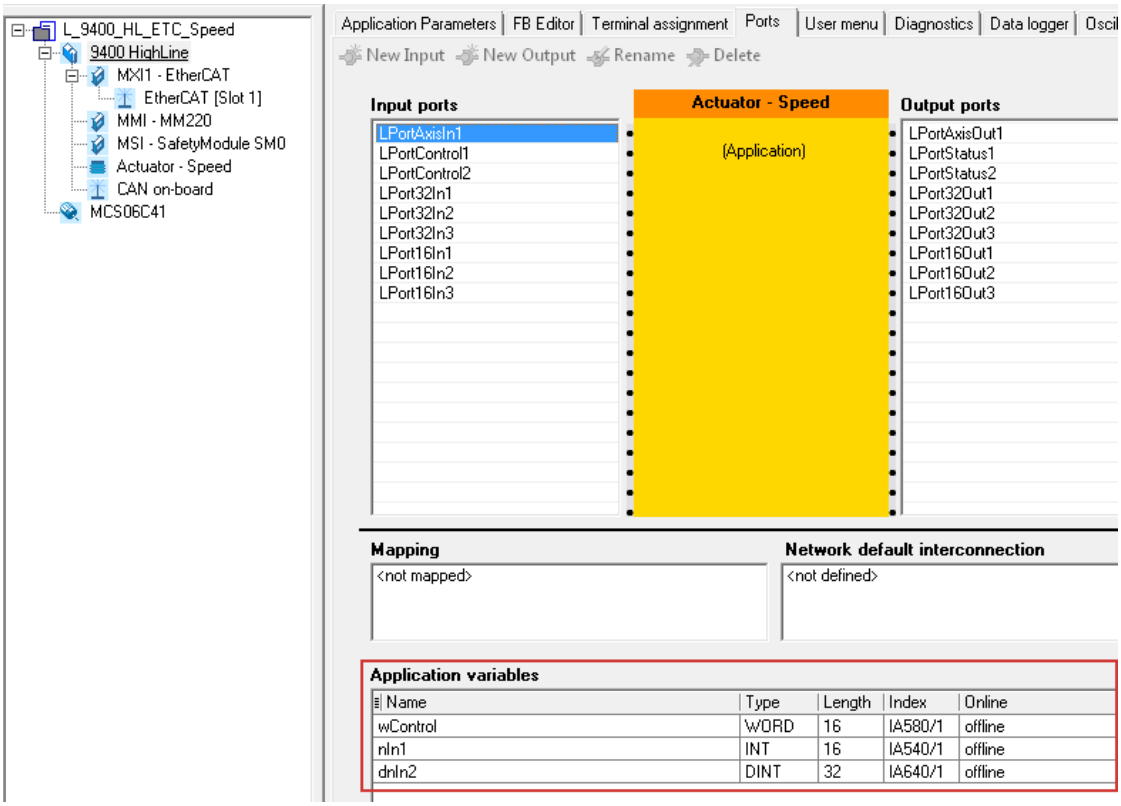
The process data (I/O addresses) must be linked manually.

7.3.14.4 Using PDO mapping settings from »Engineer«

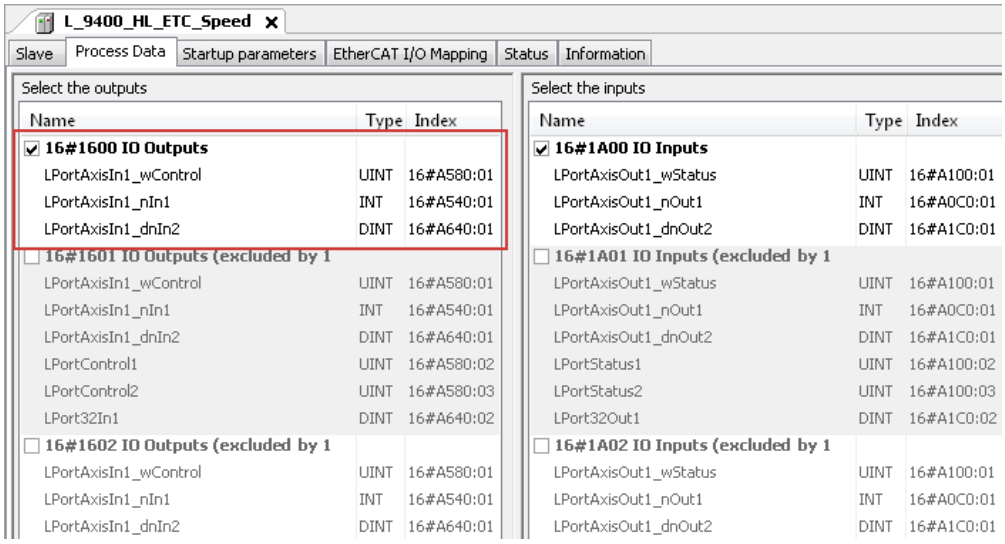
If the PDO mapping was set via the »Engineer«, the same settings must be made in the »PLC Designer« project. During the boot-up of the network, the mapping is written back to the inverter (slave). That way, it is ensured that the mapping indices in the EtherCAT master and in the slave are identical.

The illustrations below show examples of the PDO mappings of ports **LPortAxisIn1** and **LPortAxisOut1** (application: "Speed actuating drive") in the »Engineer« and in the »PLC Designer«, respectively.

PDO mapping in the »Engineer«:



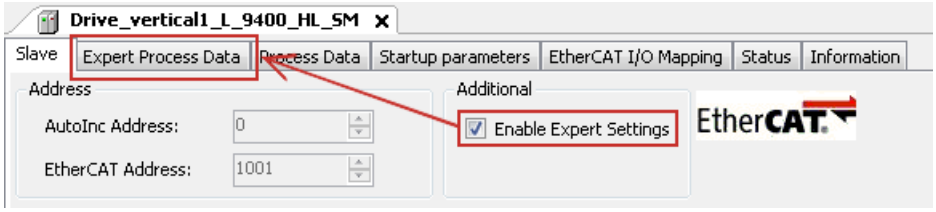
PDO mapping in the »PLC Designer«:



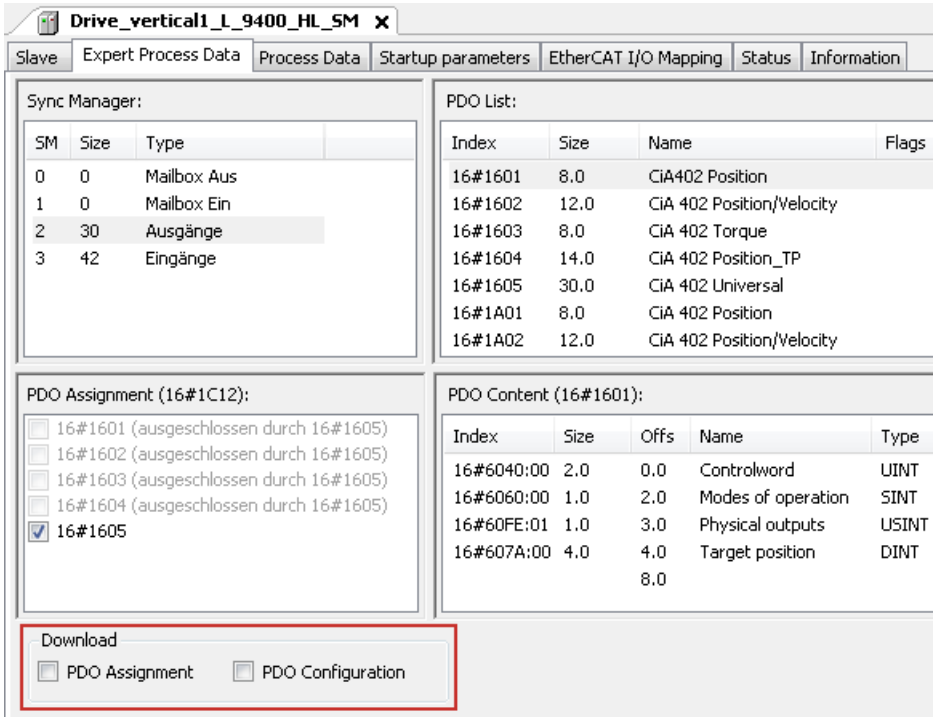
In »PLC Designer«, how to stipulate that the PDO mapping settings from »Engineer« are used for a slave drive

1. Activate the expert settings on the **Slave** tab of the slave drive.

   The **Expert Process Data** tab appears next to the **Slave** tab:



2. Remove the checkmarks in the checkboxes on the **Expert Process Data** tab in the **Download** area.



   If the checkboxes are empty, the complete PDO mapping from »Engineer« is applicable to the slave drive.

   The mapping settings carried out in »PLC Designer« are not written into the slave drive.

7.3.15 Compiling the PLC program code

In order to compile the PLC program code, select the menu command **Build → Build**, or press the function key **<F11>**.

- If errors occur during translation, they can be located and corrected on the basis of the »PLC Designer« error messages.
  Then re-translate the program code.
- If no errors occur during translation, save the »PLC Designer« project in the project folder with the menu command **File → Save project / Save project as ...**

7.3.16 Logging in on the controller with the »PLC Designer«

Use the menu command **Online → Login** or **<Alt>+<F8>** to log in on the Lenze Controller.

- For this, the PLC program must be error-free.
- With the log-in, the fieldbus configuration and the PLC program are loaded to the controller. Any configuration or a PLC program that is possibly available is overwritten.

7.3.17 Starting the PLC program

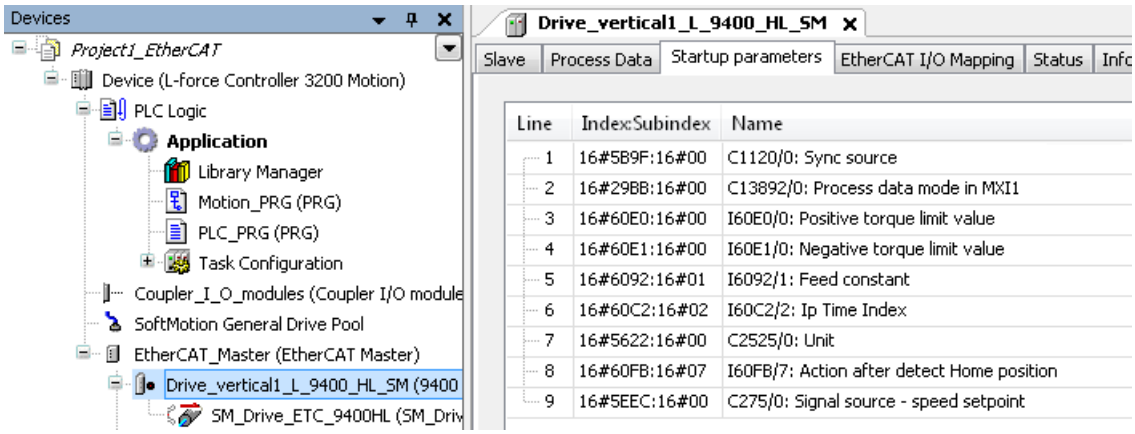
Before the start, the PLC program must be loaded to the Lenze Controller using the menu command **Online → Login**.

Use the menu command **Debug → Start** or the function key **<F5>** to start the PLC program.

7.3.18 Start parameters of the Servo Drives 9400 HighLine CiA 402

When the Lenze Controller is being powered up, some "start parameters" are automatically loaded into the servo drives 9400 HighLine CiA 402.

These parameters are shown on the **Start parameters** tab.



7.3.19 Start parameters of the Inverter Drives 8400 motec

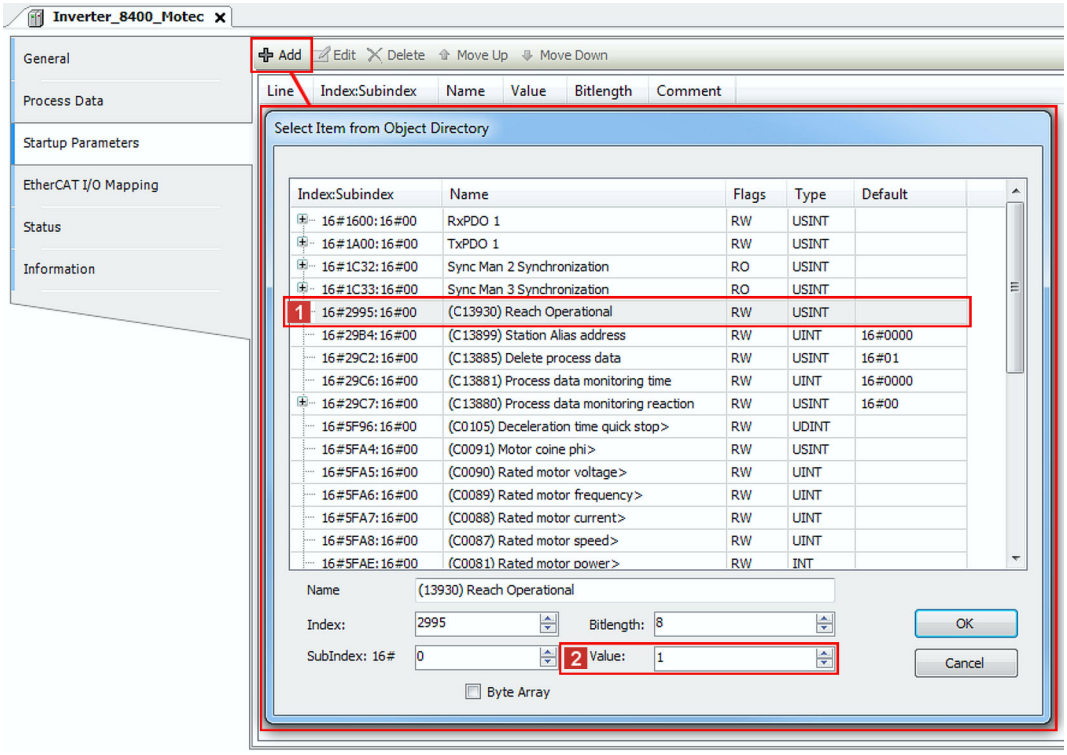
Communication manual EtherCAT – Inverter Drives 8400 motec

Please note the detailed information on the state change after "Operational" in the stay-alive operation.

Configuring the state change after "Operational" in the stay-alive operation:

To be used from Communication Unit SW version V01.02!

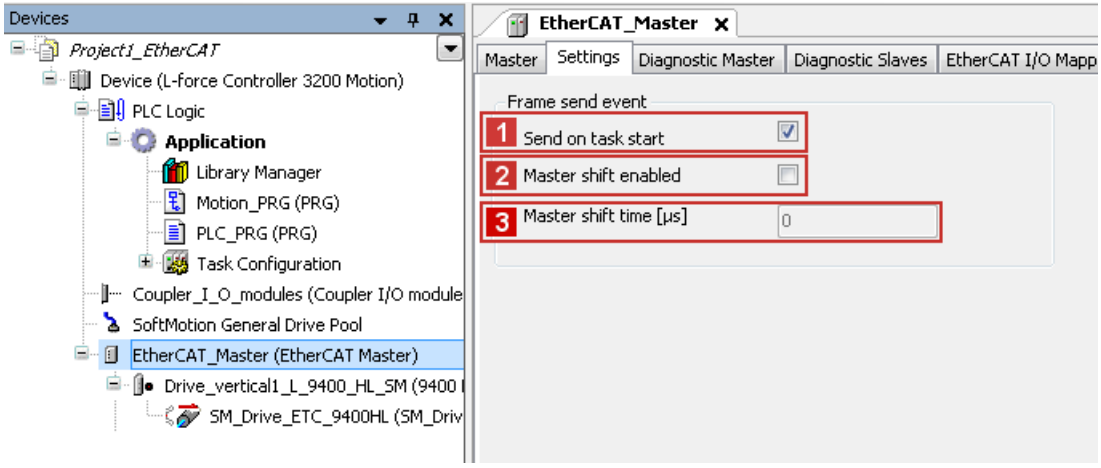
1. Add the 1 **0x2995** parameter (C13930) to the list of the start parameters of the inverter.
2. Set the 2 parameter value to **'1'**.
3. Complete the entry with **OK**.



7.3.20 Optimising the task utilisation

Optimise the task utilisation to obtain a lower jitter of the process data frames.

For this, you can enter the following settings on the **Settings** tab of the EtherCAT master:



Pos.	Setting
1	Instant of transmission for the EtherCAT bus cycle frame
	☑: The EtherCAT frame is sent at the beginning of the bus cycle task.
	☐: The EtherCAT frame is sent at the beginning of the bus cycle task.
2	Activation of "Master shift" The input field is only active if the checkbox 1 "Send at task start" is set.
	☑: "Master shift" active
	☐: No "Master shift"
3	"Master shift time" in µs Time by which the PLC system clock is placed before the SYNC0 event. Since the PLC system clock is 1 ms, only a value of 0 ... 1000 µs will be useful. The input field is only active if the checkbox 2 "Enable master shift" is set.

Note!

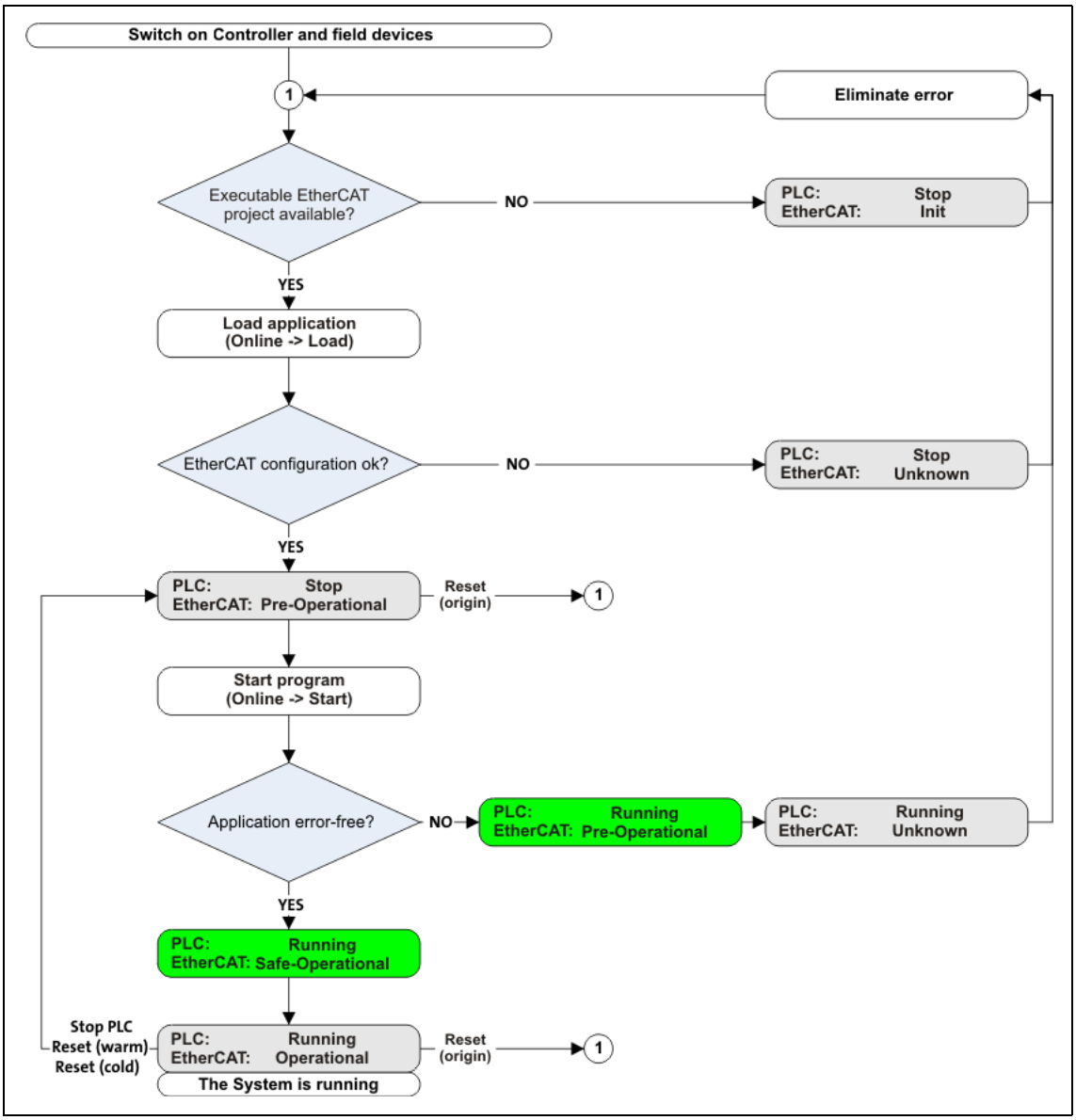
For projects up to and including release 3.5 and for updates to release 3.6, in the EtherCAT master, the 1 "Send at task start" option is not set. The EtherCAT frame is sent at the end of the bus cycle task.

For new projects of release 3.6, the "Send at task start" option is set in the EtherCAT master.

7.4 State diagram for commissioning

The state diagram displays the system behaviour. On the basis of the state diagram, you can locate errors.

You can find further information in the chapter entitled Error scenarios (📖 195).



[7-7] Status diagram for system behaviour

8 Modular machine configuration

The modular machine configuration can be used from release 3.10 onwards!

The modular machine configuration enables only one project to be used for all machine variants (maximum configuration).

Main features of the modular machine configuration are:

- Machine parts can be selected per configuration – without changing the code.
- The order of the EtherCAT node in the »PLC Designer« configuration does not have to comply with the physical order at the fieldbus.
- Changes (add/remove node) are reported to the application.
- EtherCAT nodes (change-over of the configuration) can be activated/deactivated without ...
  - stop/(re-)start of the PLC application;
  - Reset of the control.
- The application can detect whether an EtherCAT node is activated or not.
- In the event of an error in one or several nodes, the machine can continue to run in a defined constellation.

8.1 Behaviour of the EtherCAT master

When the modular machine configuration is used, the EtherCAT master behaviour derives from the behaviour known so far.

As soon as one of the function blocks L_ETC_MMCController (📖 163) or L_ETC_MMCControllerBus (📖 165) is instanced in the PLC application, the EtherCAT master waits with booting the bus.

Via the function blocks **L_ETC_MMCController** and **L_ETC_MMCControllerBus**, a service has to be defined that determines the operating mode of the master. Based on a configuration, the EtherCAT bus can be set to the "Operational" state. A Second Station Address (alias address) can be assigned to the EtherCAT slaves.

Note!

- The function blocks **L_ETC_MMCController** and **L_ETC_MMCControllerBus** may only be instanced once within the PLC application.
- The configuration is only checked while the EtherCAT master is booting. If slaves are removed or added during operation, respective checks have to be carried out by the PLC application.

8.2 Mandatory slaves / Optional slaves

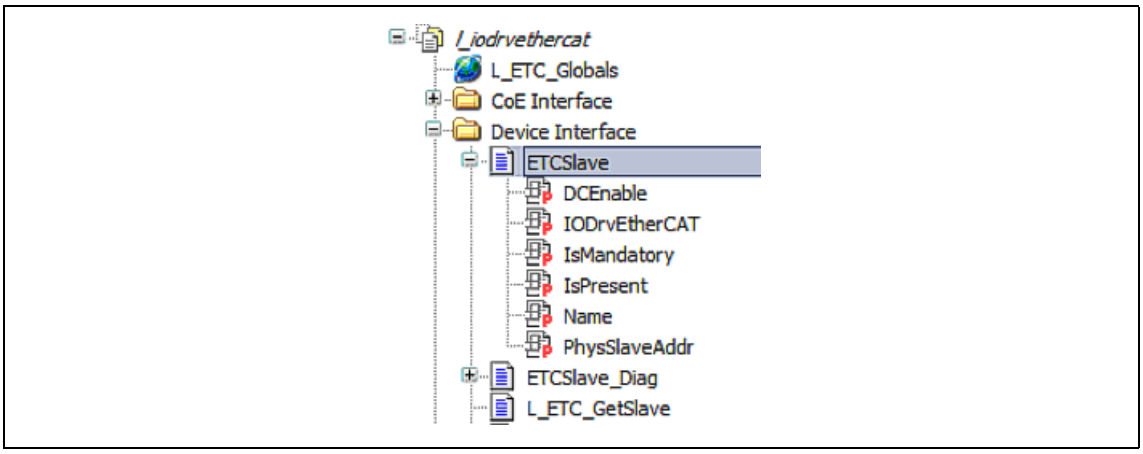
The concept of the modular machine configuration is based on the fact that, depending on the selected configuration within a project, certain EtherCAT slaves have to exist physically at the bus (Mandatory Slaves) or do not have to exist (Optional Slaves). The problem is the "clear" identification of identically constructed devices with the EtherCAT standard mechanisms. For this purpose, the "Second Station Address" (alias address) described the ETG is used which is saved in the EEPROM of the EtherCAT slave.

All EtherCAT slaves have to be defined in the PLC application. The order of the definition determines the EtherCAT address by assigning the addresses continuously from ’1001’. The address is the biunique identifying feature of a device in the network.

If, for instance, a slave shall contain the application A or the application B, a device has to be created in the project for each application. This way, the applications are identified by the different EtherCAT addresses. The same applies to the process data mapping, terminal configuration etc.)

Configurations (Configuration files (📖 117)) serve to indicate whether an EtherCAT slave is mandatory or optional. The configurations are summarised in the **mmc-0-conf.csv** text file on the Lenze Controller. More identifying features of the slaves are included in the **mmc-0-ident.csv** text file.

If the device is a "Mandatory Slave" or an "Optional Slave" and whether the slave is available at the EtherCAT bus, is displayed via the properties *IsMandatory* and *IsPresent* of the ETCSlave (FB) (📖 144) function block:



[8-1] Properties of the **ETCSlave** function block

Identifier/data type	Meaning/possible settings
IsMandatory (GET) BOOL	Depending on the currently selected modular machine configuration, the feature is set or not set for the EtherCAT slave. If the modular machine configuration is not used, the value 'TRUE' is automatically returned. • TRUE: Slave is "mandatory". • FALSE: Slave is "optional".
IsPresent (GET) BOOL	• TRUE: The slave is currently available at the bus. • TRUE: The slave is currently not available at the bus.

Sample calls in the program:

```
// Mandatory:
xMandatory_94 := L_9400_HL_SM.IsMandatory;
xMandatory_i700_S := L_i700_SM.IsMandatory;
xMandatory_i700_D := L_i700_SM_1.IsMandatory;

// Present:
xPresent_94 := L_9400_HL_SM.IsPresent;
xPresent_i700_S := L_i700_SM.IsPresent;
xPresent_i700_D := L_i700_SM_1.IsPresent;
```

8.3 Configuration files

The configurations if certain EtherCAT slaves are mandatory or optional, are summarised in the **mmc-0-conf.csv** text file on the Lenze Controller.

More identifying features of the slaves are included in the **mmc-0-ident.csv** text file.

Storage directory: `/SDCard/IPC/PLC` or `/USBStorage/IPC/PLC`

Note!

- The machine configuration **mmc-0-conf.csv** and the identifying features **mmc-0-ident.csv** are only loaded while a project (application download boot project) is loaded.
- If an error occurs during the initialisation phase, a corresponding error message is output stating the line number in the Logbook of the Lenze Controller in the »WebConfig« (📖 191).
  See:
  - L_ETC_MMCSERVICEERROR (📖 176)
  - Error messages for modular machine configuration (📖 212)

8.3.1 Machine configuration

Basically, the machine configuration file **mmc-0-conf.csv** consists of a matrix.

All EtherCAT slaves are given horizontally in ascending order, the single configurations are given vertically.

The EtherCAT slaves are referenced based on the EtherCAT address (Fixed Address), the configuration is based on the designator (STRING).

If a slave has to be contained in a configuration (Mandatory Slave), it is marked by an 'X' in the following sample table.

Address	1001	1002	1003
Inverter	i700	9400 CiA	9400 AS
Configuration 1 (all inverters)	X	X	X
Configuration 2 (1st and 2nd inverter)	X	X	
Configuration 3 (1st and 3rd inverter)	X		X

Contents of the configuration file as text:

```
version;1;0;;
address;1001;1002;1003;comment
name;i700;9400cia;9400as;
conf-1;x;x;x;all drives
conf-2;x;x;;first and second drive
conf-3;x;;x;first and last drive
```

The configuration file does not necessarily have to be available. Preconditions for this are:

- Manual address assignment via the L_ETC_MMCAssignAddress (FB) (📖 162) function block and the ADDR_ASSIGNMENT_EXTERNALLY service
- Operation of the EtherCAT master in the RUN_WITHOUT_CHECK mode

8.3.2 Further identification features of the EtherCAT slaves

An EtherCAT slave is identified by the following features:

- VendorID/ProductCode/Revision
- "Second Station Address" (alias address)
- ID selector (is not supported)

Moreover, the modular machine configuration makes it possible to use CoE objects for further identification of the EtherCAT slaves. These additional identification features are contained in the **mmc-0-ident.csv** text file.

If the identification file is not available on the Lenze Controller, the EtherCAT master assumes that no additional identification features are to be used.

The structure of the file is strictly defined. All columns have to be arranged in the given order or can contain an empty string. In the following sample table, the EtherCAT slaves are arranged in ascending order according to their addresses.

Name	Address	Index	Subindex	Type	Data
L_i700_SM	1001	16#1018	1	DWORD	03 B0 00 00
	1001	16#1019	2	DWORD	02 00 07 69
L_9400_HL_S	1002	16#1020	1	DWORD	3B 00 00 00
	1002	16#1021	2	DWORD	3D 9D 07 38
L_9400_HL_S	1003	16#1022	1	DWORD	3B 00 00 00
	1003	16#1023	2	DWORD	3D 9D 07 38

Column	Description	Notation
Name	Device name of the EtherCAT slave (optional)	STRING
Address	Address of the EtherCAT slave (required) • If no additional identification features are required for a slave, the entire line must be omitted. • If more than one identification feature is to be requested for a slave, a completely new line has to be created (see example below).	INT (decimal)
Index	Index of the CoE object to be requested (required)	INT (hexadecimal with prefix " 16# ")
Subindex	Subindex of the CoE object to be requested (required)	SINT (decimal)
Type	Type of the CoE object to be requested (required) • Only BYTE, WORD and DWORD are supported.	IEC basic types (STRING)
Data	Data the given CoE object has to contain (required) • The number of bytes has to comply with the type.	Octet stream (hexadecimal with spaces between the bytes, Little Endian) Example : The hexadecimal value '0x00001234' ('4660' decimal) is given here with '34 12 00 00'.
Comment	Comment field for the user (optional)	STRING

8.3.3 Dependencies between configuration files and services

Depending on the selected service (see L_ETC_MMCSERVICE (📖 175)), information from the machine configuration file (mmc-0-conf.csv) and the file for further identification features of the slaves (mmc-0-ident.csv) is required. If the files are not available or if a "Parsing Error" exists, the error message CONFIG_FILE_ERROR or IDENT_FILE_ERROR is caused as soon as a service is activated (see L_ETC_MMCSERVICEERROR (📖 176)).

Service	Machine configuration file mmc-0-conf.csv		Identification file mmc-0-ident.csv	
	Nonexistent	Parsing Error	Nonexistent	Parsing Error
RUN_WITHOUT_CHECK	OK	OK	OK	OK
RUN_OPTIONAL_SLAVES_ALLOWED	rejected [1)]	rejected [1)]	OK [2)]	rejected [3)]
RUN_OPTIONAL_SLAVES_PROHIBITED	rejected [1)]	rejected [1)]	OK [2)]	rejected [3)]
ADDR_ASSIGNMENT_EXTERNALLY	OK	OK	OK	OK
ADDR_ASSIGNMENT_CONFIG_SLAVEORDER	rejected [1)]	rejected [1)]	OK	OK
ADDR_ASSIGNMENT_CONFIG_PARAMETER	OK	OK	rejected [3)]	rejected [3)]

OK File is not required.

1) CONFIG_FILE_ERROR is set (see L_ETC_MMCSERVICEERROR (📖 176)).

2) File is not available. It is assumed that no additional identification features are required.

3) IDENT_FILE_ERROR is set (see L_ETC_MMCSERVICEERROR (📖 176)).

8.4 Address assignment

Before the EtherCAT master in the modular machine configuration can switch to the operating mode (RUN_[...]), each EtherCAT slave has to be assigned to a "Second Station Address" (alias address).

For this purpose, the L_ETC_MMCController (📖 163) function block provides the services **ADDR_ASSIGNMENT_EXTERNALLY** and **ADDR_ASSIGNMENT_CONFIG_SLAVEORDER**.

In case of all services for address assignment, the EtherCAT master reads out the relevant information of the slaves, creates a temporary configuration and sets the slaves into the "Pre-Operational" state. The master assigns the addresses for the slaves in ascending order starting with '1'. Thus, the address is identical to the position of the slave in the network (logical ring: Master – 1 – 2 – 3 – 4 – 5 – ...).

Hinweis!

The "Second Station Address" only becomes active when the slaves are switched on.

Service ADDR_ASSIGNMENT_EXTERNALLY

In case of this service, the "Second Station Address" (alias address) is assigned manually.

When manually assigning alias addresses, e.g. via a visualisation from the PLC application, the rules for address assignment must be observed:

1. The EtherCAT address is assigned automatically by the »PLC Designer«. The alias address is assigned in ascending order, starting with the first slave with address 1001.
2. The alias address and the EtherCAT address configured in the »PLC Designer« project must be identical.

This service provides the CoE function blocks. Parameters such as vendor-ID, product code, revision, serial number can be read out of the slave or parameters can be described for "Optical Tracking".

If a slave has been clearly identified and assigned, the L_ETC_MMCAssignAddress (FB) (📖 162) function block can be used for writing the address to the slave.

Service ADDR_ASSIGNMENT_CONFIG_SLAVEORDER

In case of this service, the "Second Station Address" (alias address) is automatically assigned.

For this purpose, a configuration has to be specified, e.g. via the **mmc-0-conf.csv** machine configuration file. Based on the configuration, the EtherCAT master recieves information on the type of slaves to be expected on the bus. If the slaves are actually available, the "Second Station Address" is written to the slaves. Here, the current (ascending) order of the slaves at the bus is important (besides vendor ID and product code).

Example :

Address	1001	1002	1003
Inverter	i700	9400 CiA	9400 AS
Configuration 1 (all inverters)	X	X	X
Configuration 2 (1st and 2nd inverter)	X	X	
Configuration 3 (1st and 3rd inverter)	X		X

For the **configuration 3**, the following slaves at the bus have to be switched on in the following order:

Master – **i700** (address ’1001’) – **9400as** (address ’1003’)

If a node is missing, the ADDR_LESS_CONNECTED error message is output.

If more nodes are available, the ADDR_MORE_CONNECTED error message is output.

(See L_ETC_MMCSERVICEERROR (📖 176).)

9 Mixed operation - EtherCAT with other bus systems

Within the Lenze Controller-based Automation arrangement, the EtherCAT bus system can be combined with CANopen, PROFIBUS or PROFINET. This makes senses if only some of the field devices are available for the same bus system or if a Motion bus is needed parallel to the logic bus (CANopen, PROFIBUS, PROFINET).

Note!

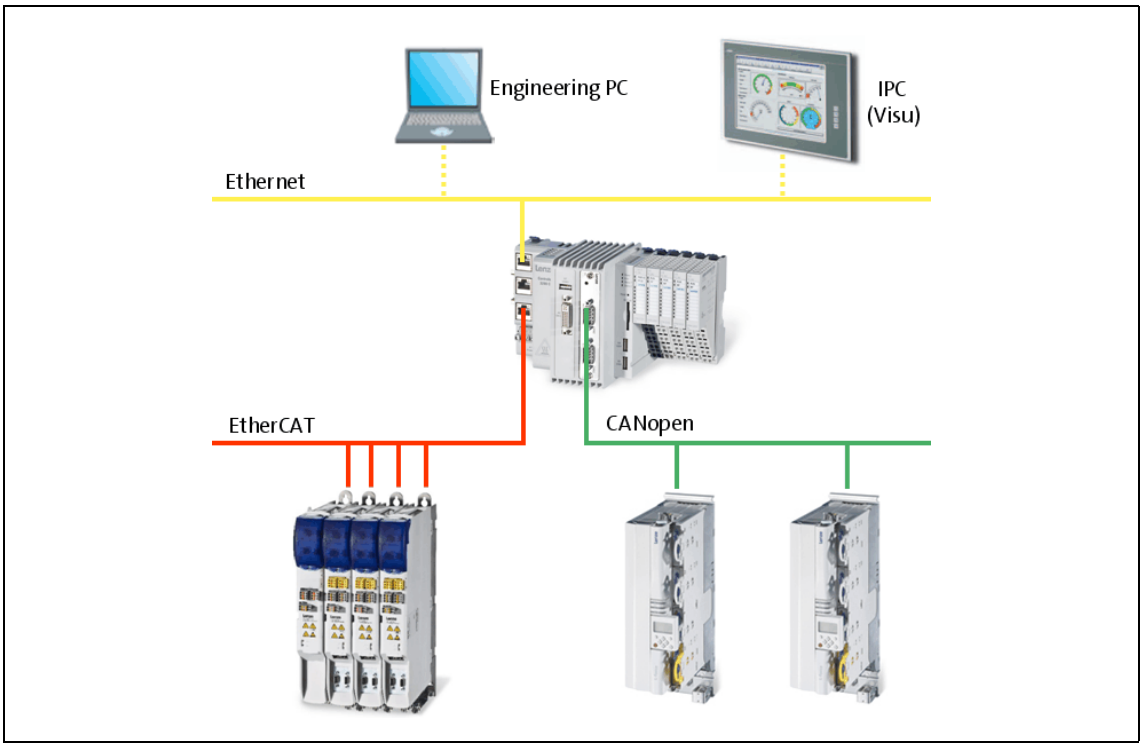
Lenze i700 servo inverter

In the case of the i700 servo inverter, fieldbus communication only takes place via EtherCAT. The servo inverter does not have any CANopen, PROFIBUS or PROFINET interfaces.

Mixed operation - EtherCAT with CANopen

- Mixed operation of EtherCAT with CANopen is not possible with the c300/p300 controllers. Either only EtherCAT or only CANopen can be used.
- Due to the demands on the real-time behaviour of the fieldbus system and the limited transfer capacity, for CANopen it is useful to operate Logic and Motion devices on separate fieldbus lines – on a Logic bus and a Motion bus.
- In mixed operation, ensure that the CAN Motion task has the highest priority. The task assigned to the EtherCAT bus should have the second-highest priority. The tasks assigned to the Logic bus systems should be configured with a lower priority.

9.1 EtherCAT and CANopen



[9-1] **Example:** Mixed operation of EtherCAT with CANopen connected to a 3231 C controller with Servo-Inverter i700 and Servo Drives 9400

Communication manual for Controller-based Automation with CANopen

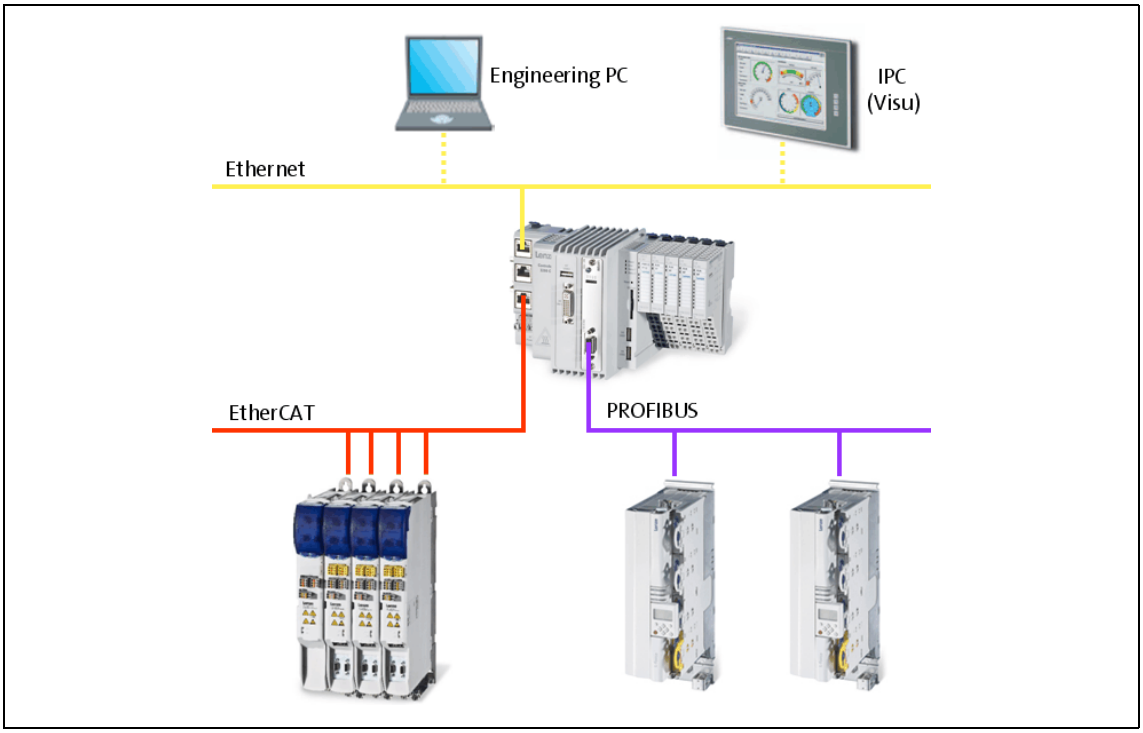
Here you can find detailed information on how to commission CANopen components.

9.2 PROFIBUS as the logic bus and EtherCAT as a logic bus or motion bus

Note!

In the Lenze automation system, only the PROFIBUS master functionality (Logic Bus) is supported.

The Motion functionality is not supported when PROFIBUS is used. Always use EtherCAT to connect inverters to be controlled via the central motion functionality.



[9-2] **Example:** Mixed operation of EtherCAT with PROFIBUS connected to a 3231 C controller with Servo-Inverter i700 and Servo Drives 9400

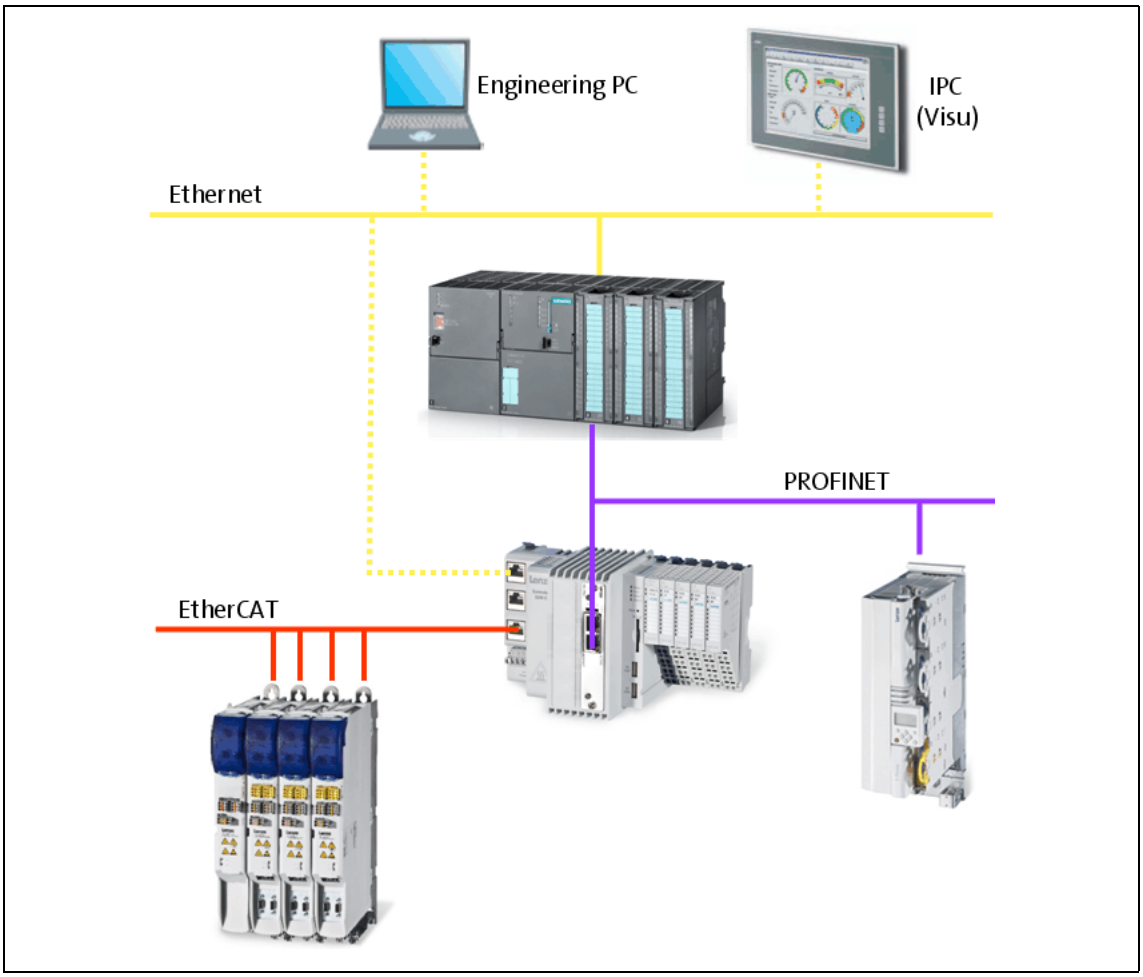
Communication manual for Controller-based Automation with PROFIBUS

Here you can find information on how to commission PROFIBUS components.

9.3 EtherCAT and PROFINET

Note!

- In the Lenze automation system, no PROFINET master functionality is supported. In a PROFINET network, a Lenze Controller can only be driven as I/O device (slave), e.g. by a Siemens SIMATIC S7 PLC.
- In the Lenze automation system, Logic field devices can be exclusively operated via PROFINET. Thus, as an I/O device, the Lenze Controller is a Logic field device.
- The Motion functionality is not supported when PROFINET is used. Always use EtherCAT to connect inverters to be controlled via the central motion functionality.



[9-3] **Example:** Mixed operation of PROFINET with EtherCAT on the Lenze Controller 3221 C

Communication manual Controller-based Automation PROFINET

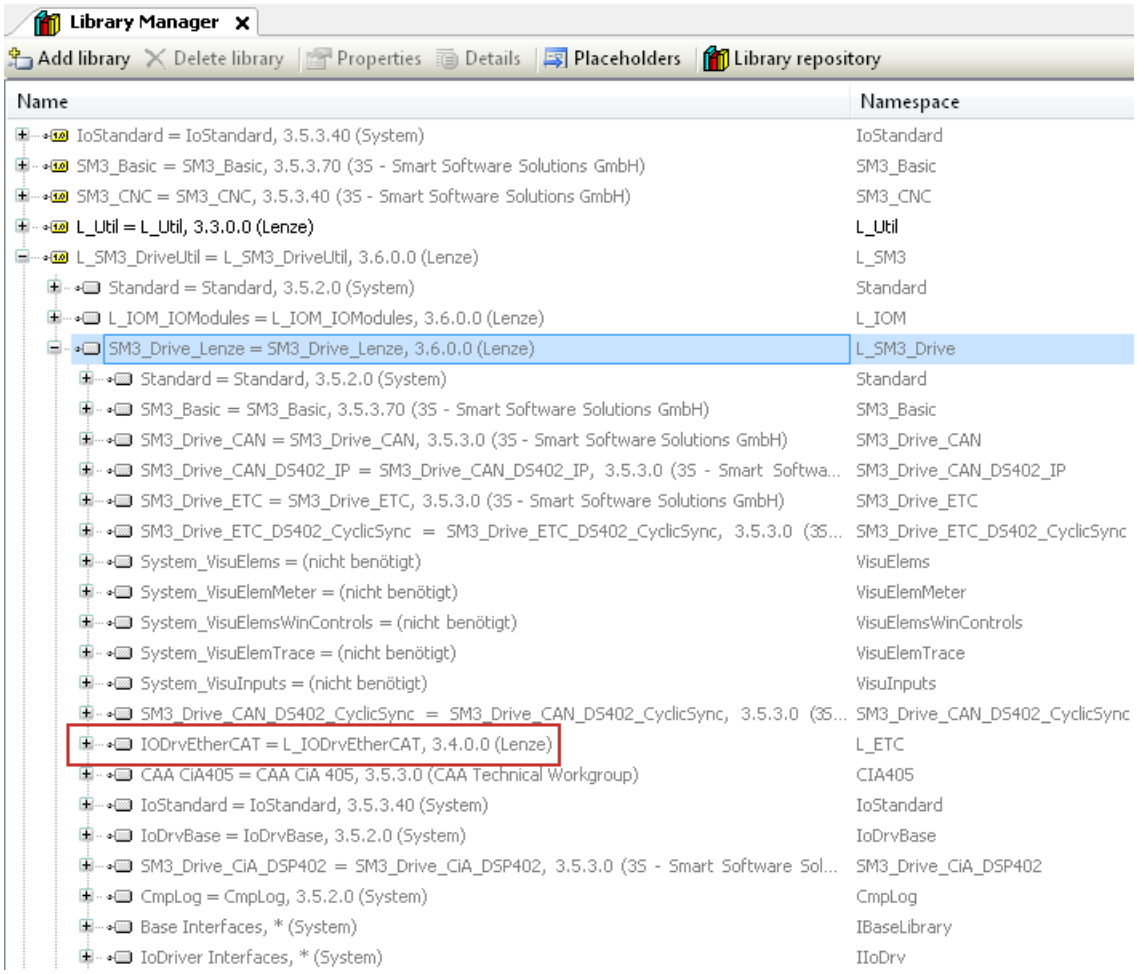
Here you can find information on how to commission PROFINET components.

10 L_IODrvEtherCAT function library

The **L_IODrvEtherCAT** function library contains all the functions and function blocks for controlling the master and slave status, for diagnostics and for sending and receiving service data.

The interface and its behaviour is in compliance with the "CoDeSys Automation Alliance" (CAA).

The **L_IODrvEtherCAT** function library is part of the **SM3_Drive_Lenze** function library.



Tip!

A sample test project in which parameters of an EtherCAT node is read and written via SDOs and how diagnostic functions of the **L_IODrvEtherCAT** function library are used, can be found in the download area at www.Lenze.com:

Application Knowledge Base: All articles → Application Ideas Pool → Controller 3200 C

The function blocks of the function library **L_IODrvEtherCAT** have inputs and outputs for ...

- activation of the POU;
- display of the current POU state;
- output of error messages.

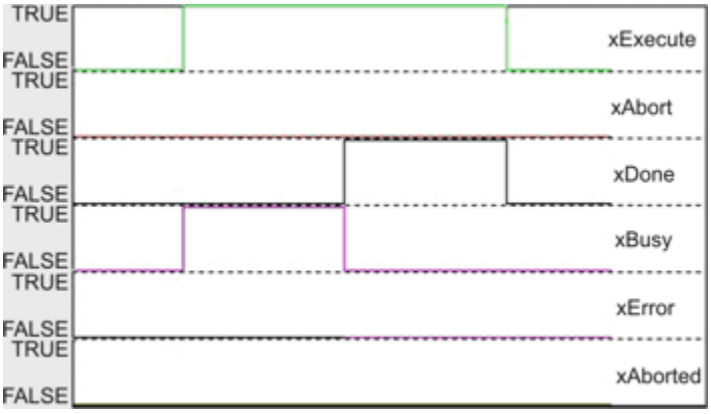
Input/output	Data type	Action
xExecute	BOOL	In the case of a positive edge (TRUE), the function block is executed.
xAbort	BOOL	With *xAbort* = TRUE, the requested service can be cancelled. **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
xDone	BOOL	If a function block has been executed, *xDone* is set to TRUE and *xBusy* is set to FALSE. • If *xExecute* has been reset, *xDone* is only active during the function block call. • If *xExecute* = TRUE, then *xDone* = TRUE as long as *xExecute* is reset.
xBusy	BOOL	As long as a function block is executed, *xBusy* is TRUE and *xDone* is FALSE.
xError	BOOL	If an error has occurred, *bError* is set to TRUE. • The *eErrorCode* output displays the error code. • The error code is an enumeration of the L_ETC_ERRORCODE (📖 173) type. • An error in online mode is shown as string. A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
xAborted	BOOL	When *xAborted* = TRUE, a requested service is aborted. **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).

Behaviour of the function blocks

The function blocks of the **L_IODrvEtherCAT** function library behave in accordance with PLCopen (IEC 61131-3).

The following diagrams show the behaviour when a function block is executed free of errors and when it is executed with errors:

- Error-free execution:



- Error-free execution with falling edge at *xExecute* while *xBusy* = TRUE:

- Error case:

- Error case with falling edge at *xExecute* while *xBusy* = TRUE:

10.1 Overview of the functions and function blocks

The functions and function blocks of the EtherCAT interface are divided into different groups.

10.2 CoE Interface

The function blocks of the "CoE interface" (CAN over EtherCAT) allow objects on the EtherCAT master and the EtherCAT slaves to be read and written.

The SDO read and write services are performed serially in the case of EtherCAT. In the Lenze R3.x control technology (Controller-based Automation), approx. 5 services per slave can be temporarily stored for processing. If no more services can be accepted because the temporary storage buffer is full, the value '7' is sent back as the error code.

10.2.1 Reading and writing parameters

Parameters ...

- for instance are set for one-time system settings or if materials are changed within a machine;
- are transmitted with a low priority.

In the case of Lenze inverters, the parameters to be changed are contained in codes or in the case of the CANopen device profile "CiA402" as device profile objects.

Indexing of the Lenze codes

When they are accessed, the codes of the Lenze Controllers are addressed by the index.

The index for Lenze code numbers is in the manufacturer-specific area of the object directory between 8192 (0x2000) and 24575 (0x5FFF).

Conversion formula	
Index [dec]	**Index [hex]**
24575 - Lenze code	0x5FFF - Lenze code [hex]

Example for C00002 (device commands)	
Index [dec]	**Index [hex]**
24575 - 2 = 24573	0x5FFF - 2 = 0x5FFD

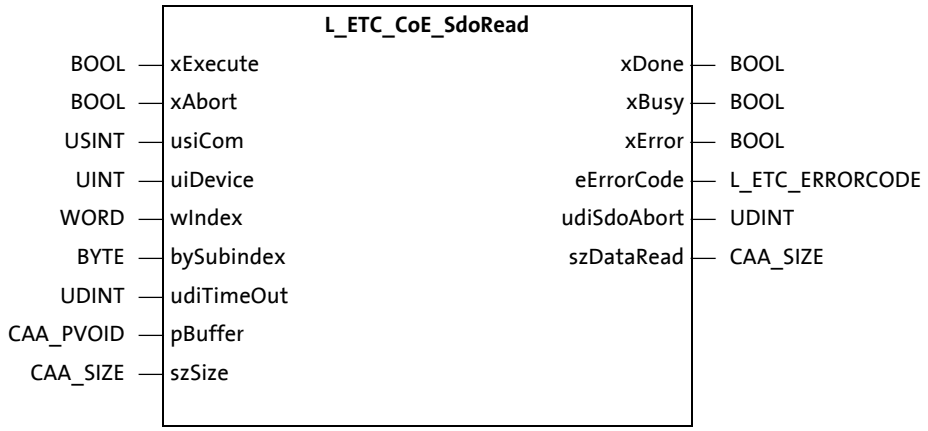
10.2.2 L_ETC_CoE_SdoRead (FB)

This function block triggers uploading of a CoE object (SDO) from the slave or from the master.

Visualisation: VISU_L_ETC_CoE_SdoRead

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) triggers a read request (uploading) of a CoE object.
xAbort BOOL	A positive edge (TRUE) aborts a running read request (upload). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address (station address) of the master/slave • The station addresses are assigned by the master during the start-up phase (fixed-address assignment, see also Addressing of the slaves (📖 23)). • The address '0' directly accesses the object directory of the EtherCAT master. • The first EtherCAT slave is given the address '1001', the second the address '1002' and so on. • The EtherCAT addresses cannot be altered.
wIndex WORD	CANopen index to be read in the object directory of the master or slave. • Formula for converting a Lenze code number into a CANopen index: **Index = 0x5FFF - code number**
bySubindex BYTE	Subindex in the object directory of the master or slave.
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).

Identifier/data type	Meaning/possible settings
pBuffer CAA_PVOID	Reference to memory buffer to which the values to be read are to be copied.
szSize CAA_SIZE	Size of the memory buffer transmitted to *pBuffer*. • The memory buffer must be big enough to accept the read object.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
udiSdoAbort UDINT	If a read request (upload) from the slave or master is rejected with an error, here the CANopen abort code is returned.
szDataRead CAA_SIZE	Number of bytes for the read request (upload) that is actually read

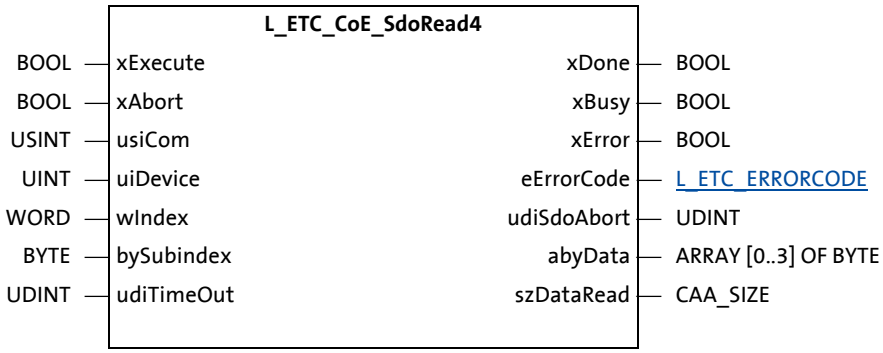
10.2.3 L_ETC_CoE_SdoRead4 (FB)

This function block triggers uploading of a CoE object (SDO) from the slave or from the master.

Visualisation: VISU_L_ETC_CoE_SdoRead4

Note!

- The function of this function block is identical with the function of L_ETC_CoE_SdoRead (FB) (📖 132), except that only up to 4 bytes can be read with **L_ETC_CoE_SdoRead4**.
- For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) triggers a read request (uploading) of a CoE object.
xAbort BOOL	A positive edge (TRUE) aborts a running read request (upload). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address 0 directly accesses the object directory of the EtherCAT master.
wIndex WORD	CANopen index to be read in the object directory of the master or slave. • Formula for converting a Lenze code number into a CANopen index: **Index = 0x5FFF - code number**
bySubindex BYTE	Subindex in the object directory of the master or slave.
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
udiSdoAbort UDINT	If a read request (upload) from the slave or master is rejected with an error, here the CANopen abort code is returned.
abyData ARRAY [0..3] OF BYTE	Memory buffer with the value to be written.
szDataRead CAA_SIZE	Number of bytes for the read request (upload) that is actually read

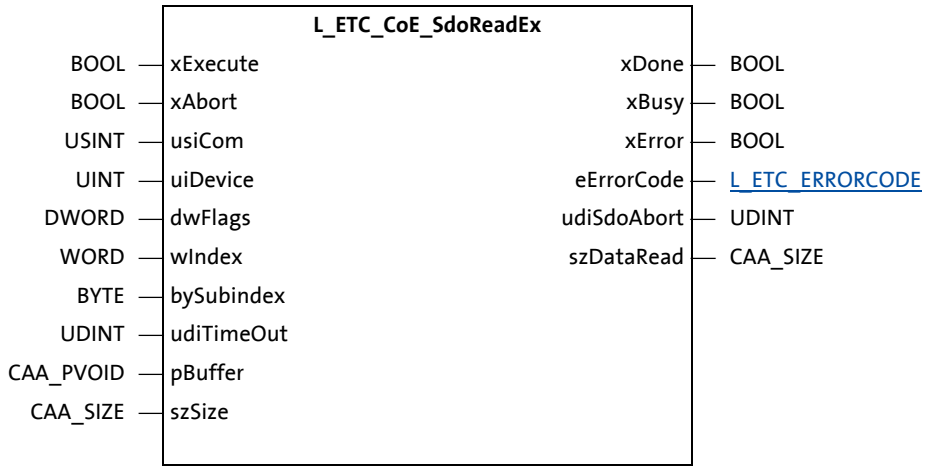
10.2.4 L_ETC_CoE_SdoReadEx (FB)

This function block triggers uploading of a CoE object (SDO) from the slave or from the master.

Visualisation: VISU_L_ETC_CoE_SdoReadEx

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Complete access

By setting bit 0 at *dwFlags*, all subindices of an object are read by the slave using a single SDO service. The following rules apply for transmission via "complete access":

- Subindex 1 always begins at an even byte address.
- Boolean/bit variables are packed in bytes. Following non-bit objects begin at the next byte address.
- Non-existent subindices do not require any memory space.
- "Complete access" can start with subindex 0 or subindex 1. Other subindices are not permitted.

Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) triggers a read request (uploading) of a CoE object.
xAbort BOOL	A positive edge (TRUE) aborts a running read request (upload). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address 0 directly accesses the object directory of the EtherCAT master.
dwFlags DWORD	*dwFlags* controls the function block behaviour: • 16#00000000: Function block behaviour L_ETC_CoE_SdoRead (FB) (📖 132) • 16#00000001: "Complete access" Type definition see L_ETC_COE_FLAGS (📖 172).
wIndex WORD	CANopen index to be read in the object directory of the master or slave. • Formula for converting a Lenze code number into a CANopen index: **Index = 0x5FFF - code number**
bySubindex BYTE	Subindex in the object directory of the master or slave.
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).
pBuffer CAA_PVOID	Reference to memory buffer to which the values to be read are to be copied.
szSize CAA_SIZE	Size of the memory buffer transmitted to *pBuffer*. • The memory buffer must be big enough to accept the read object.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
udiSdoAbort UDINT	If a read request (upload) from the slave or master is rejected with an error, here the CANopen abort code is returned.
szDataRead CAA_SIZE	Number of bytes for the read request (upload) that is actually read

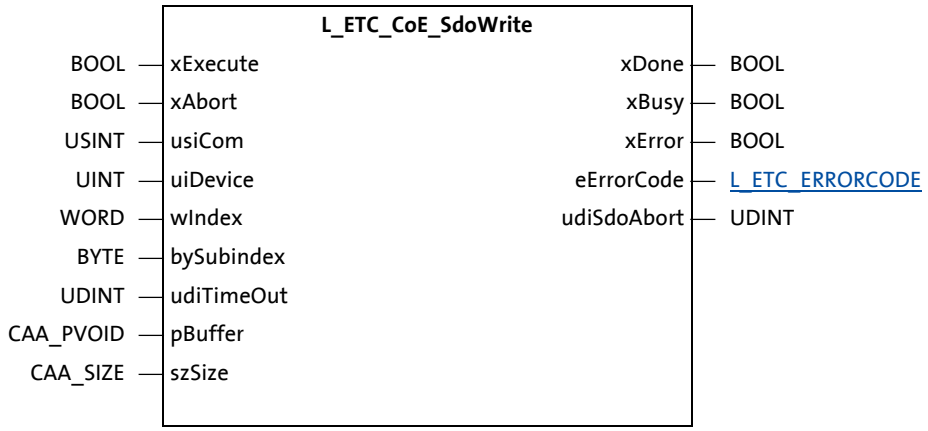
10.2.5 L_ETC_CoE_SdoWrite (FB)

This function block triggers downloading of a CoE object (SDO) to the slave or to the master.

Visualisation: VISU_L_ETC_CoE_SdoWrite

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) triggers a write request (downloading) of a CoE object.
xAbort BOOL	A positive edge (TRUE) aborts a running write request (download). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address 0 directly accesses the object directory of the EtherCAT master.
wIndex WORD	CANopen index to be read in the object directory of the master or slave. • Formula for converting a Lenze code number into a CANopen index: **Index = 0x5FFF - code number**
bySubindex BYTE	Subindex in the object directory of the master or slave.
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).
pBuffer CAA_PVOID	Reference to the memory buffer from which the values to be written are to be taken.
szSize CAA_SIZE	Number of the bytes to be written

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
udiSdoAbort UDINT	If a write request (download) to the slave or master is rejected with an error, here the CANopen abort code is returned.

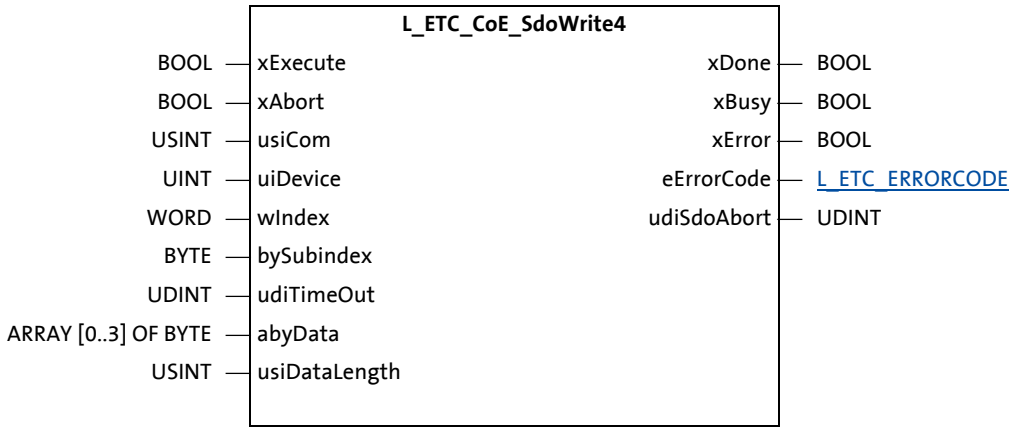
10.2.6 L_ETC_CoE_SdoWrite4 (FB)

This function block triggers downloading of a CoE object (SDO) to the slave or to the master.

Visualisation: VISU_L_ETC_CoE_SdoWrite4

Note!

- The function of this function block is identical with the function of L_ETC_CoE_SdoWrite (FB) (📖 138), except that only up to 4 bytes can be written with **L_ETC_CoE_SdoWrite4**.
- For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) triggers a write request (downloading) of a CoE object.
xAbort BOOL	A positive edge (TRUE) aborts a running write request (download). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number ’1’ is currently supported. *usiCom* has the number ’1’ pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address 0 directly accesses the object directory of the EtherCAT master.
wIndex WORD	CANopen index to be read in the object directory of the master or slave. • Formula for converting a Lenze code number into a CANopen index: **Index = 0x5FFF - code number**
bySubindex BYTE	Subindex in the object directory of the master or slave.
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).

Identifier/data type	Meaning/possible settings
abyData ARRAY [0..3] OF BYTE	Memory buffer with the value to be written.
usiDataLength USINT	Number of the bytes to be written

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
udiSdoAbort UDINT	If a write request (download) to the slave or master is rejected with an error, here the CANopen abort code is returned.

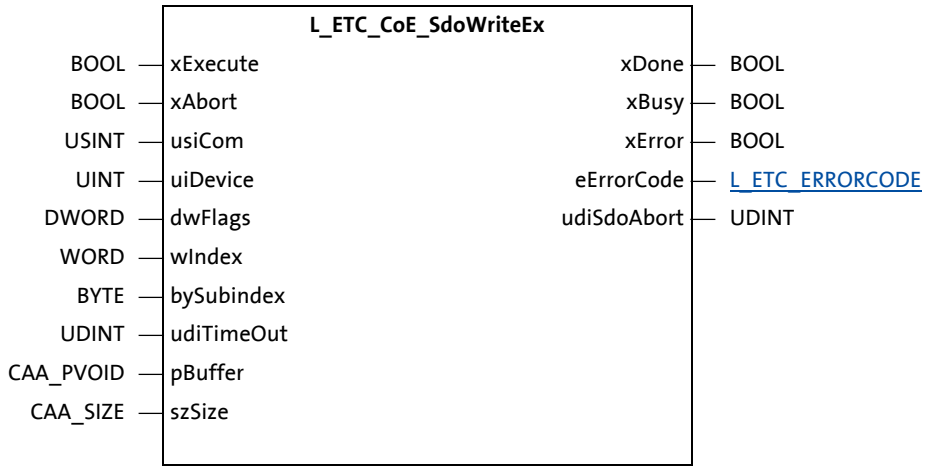
10.2.7 L_ETC_CoE_SdoWriteEx (FB)

This function block triggers downloading of a CoE object (SDO) to the slave or to the master.

Visualisation: VISU_L_ETC_CoE_SdoWriteEx

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Complete access

By setting bit 0 at *dwFlags*, all subindices of an object are read by the slave using a single SDO service. The following rules apply for transmission via "complete access":

- Subindex 1 always begins at an even byte address.
- Boolean/bit variables are packed in bytes. Following non-bit objects begin at the next byte address.
- Non-existent subindices do not require any memory space.
- "Complete access" can start with subindex 0 or subindex 1. Other subindices are not permitted.

Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) triggers a write request (downloading) of a CoE object.
xAbort BOOL	A positive edge (TRUE) aborts a running write request (download). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address 0 directly accesses the object directory of the EtherCAT master.
dwFlags DWORD	*dwFlags* controls the function block behaviour: • 16#00000000: Function block behaviour L_ETC_CoE_SdoWrite (FB) (📖 138) • 16#00000001: "Complete access" Type definition see L_ETC_COE_FLAGS (📖 172).
wIndex WORD	CANopen index to be read in the object directory of the master or slave. • Formula for converting a Lenze code number into a CANopen index: **Index = 0x5FFF - code number**
bySubindex BYTE	Subindex in the object directory of the master or slave.
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).
pBuffer CAA_PVOID	Reference to the memory buffer from which the values to be written are to be taken.
szSize CAA_SIZE	Number of the bytes to be written

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
udiSdoAbort UDINT	If a write request (download) to the slave or master is rejected with an error, here the CANopen abort code is returned.

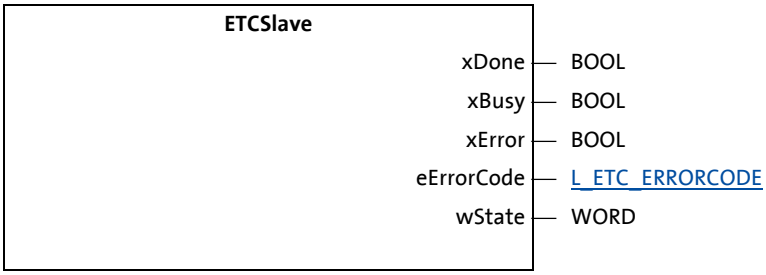
10.3 Device Interface

In addition to the EtherCAT master and slave types for access to slaves, this group also provides a generic function for accessing the EtherCAT master.

10.3.1 ETCSlave (FB)

For every slave in the control configuration, the system creates an object of the type **ETCSlave**. The name of the slave and the name of the EtherCAT slave instance are identical. In the case of a SoftMotion slave, the instance is called cyclically in the context of the SoftMotion application. If the slave is not a SoftMotion slave, the instance must be called when the *wState* output is used.

Visualisation: VISU_ETCSlave



Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
wState WORD	Current slave state of the L_ETC_STATE (📖 178) type

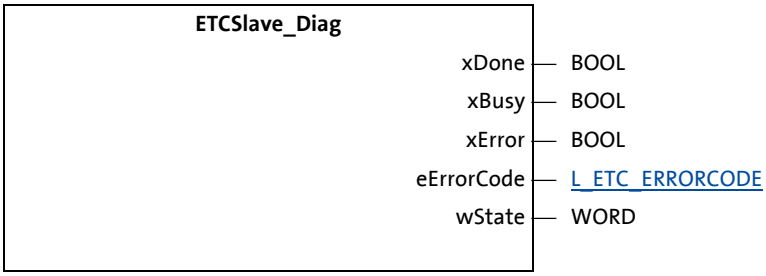
Product features

Identifier/data type	Meaning/possible settings
DCEnable BOOL	• TRUE: The "Distributed clocks" (DC) function is activated for the slave. • FALSE: The "Distributed clocks" (DC) function is not activated. ▸ Synchronisation with "Distributed clocks" (DC) (📖 37)
IODrvEtherCAT POINTER TO L_IODrvEtherCAT	Reference to the EtherCAT master function block L_IODrvEtherCAT (FB) (📖 149).
Name STRING	Name of the EtherCAT slave
PhysSlaveAddr UINT	EtherCAT address of the EtherCAT slave
IsMandatory (GET) BOOL	Depending on the currently selected modular machine configuration, the feature is set or not set for the EtherCAT slave. If the modular machine configuration is not used, the value 'TRUE' is automatically returned. • TRUE: Slave is "mandatory". • FALSE: Slave is "optional". ▸ Modular Machine Configuration Interface (📖 162)
IsPresent (GET) BOOL	• TRUE: The slave is currently available at the bus. • TRUE: The slave is currently not available at the bus.

10.3.2 ETCSlave_Diag (FB)

The **ETCSlave_Diag** function block is an extension for the diagnostics of the EtherCAT slave and contains the same inputs and outputs as ETCSlave (FB) (📖 144).

Visualisation: VISU_ETCSlave



Outputs (VAR_OUTPUT)

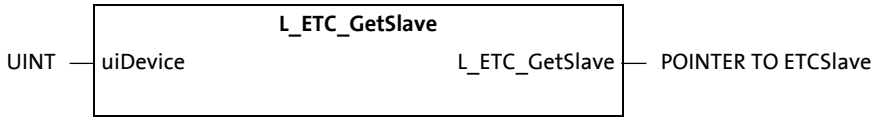
Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
wState WORD	Current slave state of the L_ETC_STATE (📖 178) type

Product features

Identifier/data type	Meaning/possible settings
DCEnable BOOL	• TRUE: The "Distributed clocks" (DC) function is activated for the slave. • FALSE: The "Distributed clocks" (DC) function is not activated. ▸ Synchronisation with "Distributed clocks" (DC) (📖 37)
IODrvEtherCAT POINTER TO L_IODrvEtherCAT	Reference to the EtherCAT master function block L_IODrvEtherCAT (FB) (📖 149).
Name STRING	Name of the EtherCAT slave
PhysSlaveAddr UINT	EtherCAT address of the EtherCAT slave
IsMandatory (GET) BOOL	Depending on the currently selected modular machine configuration, the feature is set or not set for the EtherCAT slave. If the modular machine configuration is not used, the value 'TRUE' is automatically returned. • TRUE: Slave is "mandatory". • FALSE: Slave is "optional". ▸ Modular Machine Configuration Interface (📖 162)
IsPresent (GET) BOOL	• TRUE: The slave is currently available at the bus. • TRUE: The slave is currently not available at the bus.

10.3.3 L_ETC_GetSlave (FUN)

On the basis of the EtherCAT address, this function provides the reference to the slave instance of the ETCSlave (FB) (📖 144) type.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
uiDevice UINT	The EtherCAT address of the slave for which the reference is to be returned.

Return value

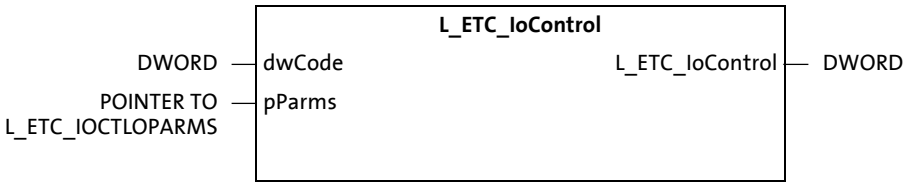
Identifier/data type	Meaning/possible settings
L_ETC_GetSlave POINTER TO ETCSlave	Reference to the EtherCAT slave object by the EtherCAT address provided by *uiDevice*.

10.3.4 L_ETC_IoControl (FUN)

This function transmits a "IoControl" to the EtherCAT master stack.

Note!

This function is only to be used Lenze-internally!



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
dwCode DWORD	The EtherCAT address of the slave for which the reference is to be returned.
pParms POINTER TO L_ETC_IOCTLOPARMS	Reference to input and output buffers of the L_ETC_IOCTLOPARMS (📖 174) type

Return value

Identifier/data type	Meaning/possible settings
L_ETC_IoControl DWORD	The return value of this function corresponds to the error code (L_ETC_ERRORCODE (📖 173)).

10.3.5 L_IODrvEtherCAT (FB)

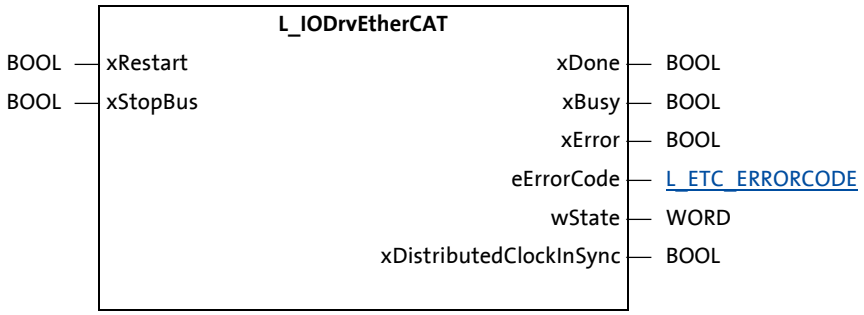
If an EtherCAT master is created in the configuration tree of a »PLC Designer« project, the system automatically creates an object of the **L_IODrvEtherCAT** type.

The name of the function block for the EtherCAT master ist "EtherCAT_Master".

Visualisation: VISU_L_IODrvEtherCAT

Note!

The **L_IODrvEtherCAT** function block is called in the context of the bus cycle task and must not be called again in the PLC program.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xRestart BOOL	A positive edge (TRUE) executes a **bus restart**. • The EtherCAT master is first set to the Init state and then to "Operational". • When *xStopBus* = TRUE, the positive edge is ignored. ▸ Restarting the EtherCAT fieldbus (📖 179)
xStopBus BOOL	A positive edge (TRUE) executes a bus stop. • The fieldbus is set to the "Init" state.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

Identifier/data type	Meaning/possible settings
wState WORD	Current master state of the L_ETC_STATE (📖 178) type
xDistributedClockInSync BOOL	• TRUE: All DC slaves have been synchronised. • FALSE: Not all DC slaves have been synchronised. **Notes** • The flag may only be evaluated in the "Operational" state. • It may happen that the master and the slaves are in the "Operational" state but the flag is FALSE. The cause is either a high DC deviation of a slave or a missing engagement of the bus cycle task with the Sync0. The master and the slaves remain in the "Operational" state if the EtherCAT master flag *xDistributedClockInSync* is set to FALSE due to a drift of the distributed clocks. ▸ Synchronous communication (📖 38)

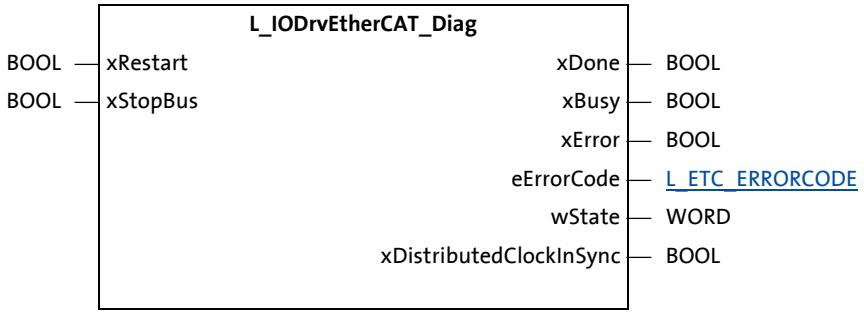
10.3.6 L_IODrvEtherCAT_Diag (FB)

The **L_IODrvEtherCAT_Diag** function block is an extension for the diagnostics of the EtherCAT network and contains the same inputs and outputs as L_IODrvEtherCAT (FB) (📖 149).

Visualisation: VISU_L_IODrvEtherCAT

Note!

The **L_IODrvEtherCAT** function block is called in the context of the bus cycle task and must not be called again in the PLC program.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xRestart BOOL	A positive edge (TRUE) executes a **bus restart**. • The EtherCAT master is first set to the Init state and then to "Operational". • When *xStopBus* = TRUE, the positive edge is ignored. ▸ Restarting the EtherCAT fieldbus (📖 179)
xStopBus BOOL	A positive edge (TRUE) executes a bus stop. • The fieldbus is set to the "Init" state.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
wState WORD	Current master state of the L_ETC_STATE (📖 178) type
xDistributedClockInSync BOOL	• TRUE: All DC slaves have been synchronised. • FALSE: Not all DC slaves have been synchronised.

10.4 Diagnostic Interface

The "Diagnostic Interface" provides diagnostic blocks for the master and the slaves. The **L_ETC_GetErrorString** function converts the internal error code into a readable string.

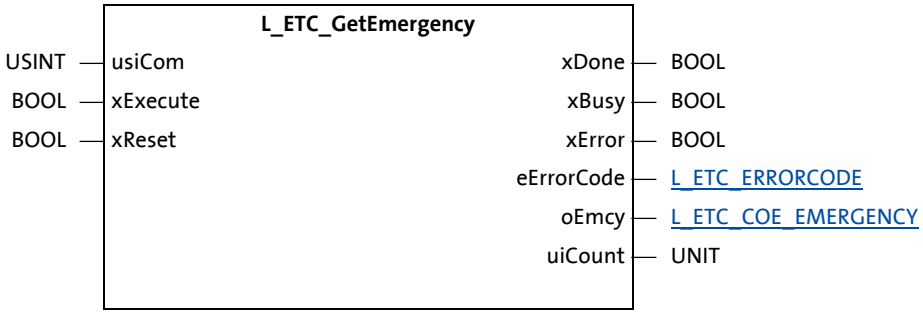
10.4.1 L_ETC_GetEmergency (FB)

This function block outputs CoE emergency frames which have been stored in a buffer by the I/O driver before.

Visualisation: VISU_L_ETC_GetEmergency

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

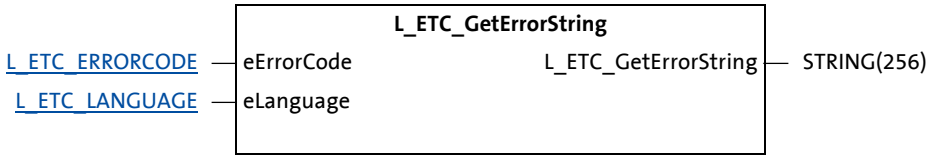
Identifier/data type	Meaning/possible settings
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
xExecute BOOL	A positive edge (TRUE) activates a read request of a CoE emergency frame from the buffer.
xReset BOOL	A positive edge (TRUE) resets the buffer and deletes all CoE emergency frames saved until then.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). If no emergency telegram was received, the error message "0x9811000C: ETC_E_NOTFOUND" is output. A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
oEmcy L_ETC_COE_EMERGENCY	If *xExecute* is TRUE, a CoE emergency frame read from the buffer is pending at the *oEmcy* output. Information in the CoE emergency frame: • *uiDevice* : UNIT (slave address) • *dwTimestamp* : DATE_AND_TIME (timestamp UTC, see FUNC SysTimeRtcGet) • *wErrorCode* : WORD (error code, device-specific) • *byErrorRegister* : BYTE (error register, device-specific) • *abyData* : ARRAY[0..4] OF BYTE (error data, device-specific)
uiCount UINT	If *xExecute* is TRUE, the number of CoE emergency frames remaining in the buffer is specified at the *uiCount* output.

10.4.2 L_ETC_GetErrorString (FUN)

This function returns a language-specific string for an **L_ETC_ErrorCode**.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
eLanguage L_ETC_LANGUAGE	Language selection • In the Lenze R3.x control technology, only English strings are supported at the moment.

Return value

Identifier/data type	Meaning/possible settings
L_ETC_GetErrorString STRING(256)	Language-specific error string for *eErrorCode*

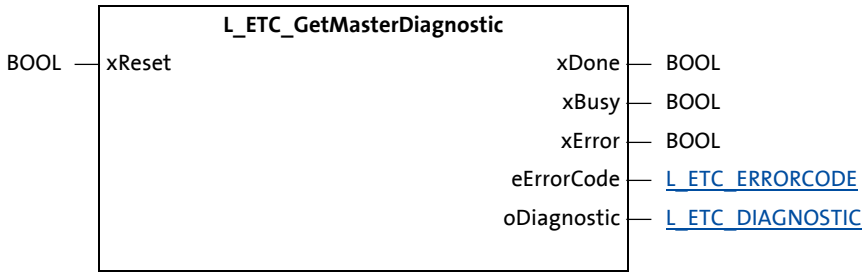
10.4.3 L_ETC_GetMasterDiagnostic (FB)

When called, this function block returns a structure with diagnostic information at the *oDiagnostic* output (type L_ETC_DIAGNOSTIC (📖 173)).

The processing of the POU requires some microseconds (µs).

Visualisation: VISU_L_ETC_GetMasterDiagnostic

▸ Visualisation of the function block L_ETC_GetMasterDiagnostic (📖 187)



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xReset BOOL	A positive edge (TRUE) resets the error counters and the output *xNotAllSlavesOperational* in the output structure *oDiagnostic*. • Exception: "Frame Lost Counter"

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
oDiagnostic L_ETC_DIAGNOSTIC	Diagnostic information that is returned when the function block is called. • The output *oDiagnostic.xDC_InSync* is set (TRUE) if the DC master and all DC slaves have been synchronised. **Setting the flag** *xNotAllSlavesInOperational:* The flag is set in the **Operational state** of the EtherCAT master if one or more slaves have left the **Operational state**. Possible causes for leaving the Operational state of slaves are request by L_ETC_SetSlaveState(), switching off the slave(s), disconnecting the slave(s) from the EtherCAT-Bus or absence of the slave(s). The flag is not set, if • the EtherCAT master is just starting up or • a bus restart was caused by a command at input *xRestart* of the EtherCAT master. **Resetting the Flag** *xNotAllSlavesInOperational:* The flag is not automatically reset, if all stations are back in the **Operational state**. The flag is reset • by restarting the EtherCAT master or • by setting the input *xReset* at the block **L_ETC_GetMasterDiagnostic()**.

10.4.4 L_ETC_ReadErrCnt (FB)

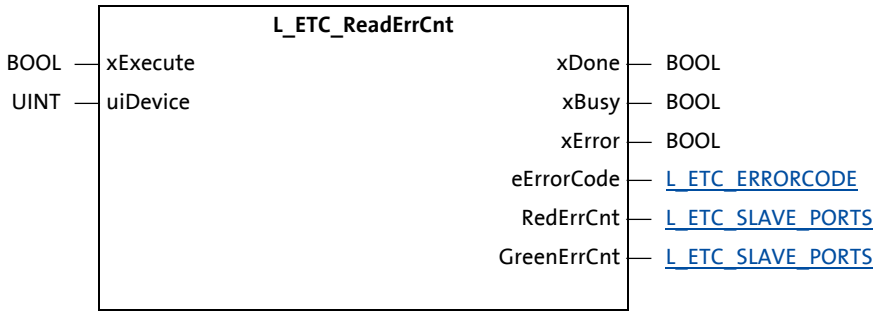
This function block reads the frame error counters of the connected slaves.

On the basis of the values in the *RedErrCnt* array, the wiring quality (EMC sensitivity) of the EtherCAT bus can be evaluated.

Visualisation: VISU_L_ETC_ReadErrCnt

Note!

- We recommend executing the **L_ETC_ReadErrCnt** function block cyclically in greater distances (e.g. every 10 minutes) in order to evaluate the bus state.
- For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) reads the error counter of the slave given by *uiDevice*.
uiDevice UINT	The EtherCAT address of the slave

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
RedErrCnt L_ETC_SLAVE_PORTS	Error counter for errors detected in the slave for the first time ▸ Error counters of the EtherCAT slaves (📖 193)
GreenErrCnt L_ETC_SLAVE_PORTS	Error counter for forwarded errors, i.e. errors that have been detected in a previous slave

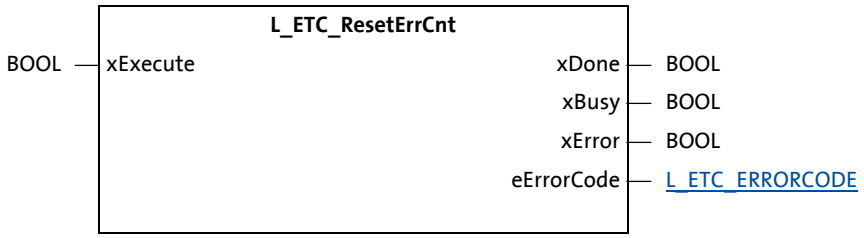
10.4.5 L_ETC_ResetErrCnt (FB)

This function block resets the frame error counters of all connected slaves.

Visualisation: VISU_L_ETC_ResetErrCnt

Note!

We recommend executing the **L_ETC_ResetErrCnt** function block before one of the frame error counters of the slaves has reached the maximum value of '255'.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) resets the frame error counters of all connected slaves.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

10.5 FoE interface

The function blocks of the "FoE interface" (File over EtherCAT) allow you to transmit files between the EtherCAT master and the EtherCAT slaves.

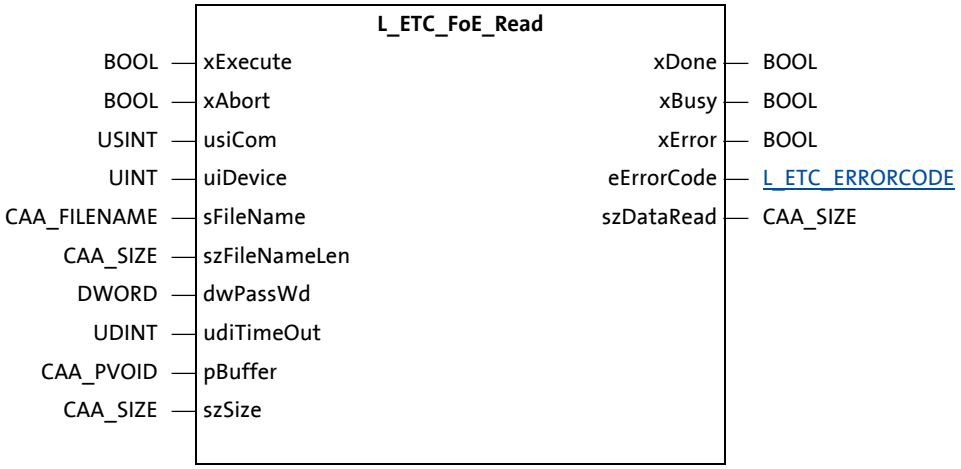
10.5.1 L_ETC_FoE_Read (FB)

This function block activates a file upload from the slave or from the master.

Visualisation: VISU_L_ETC_FoE_Read

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) activates a read request (upload) of a file.
xAbort BOOL	A positive edge (TRUE) aborts a running read request (upload). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address '0' directly accesses the object directory of the EtherCAT master.
sFileName CAA_FILENAME	File name of file to be read • Max. 32 characters (see *szFileNameLen*) • Directory structures are not supported.

Identifier/data type	Meaning/possible settings
szFileNameLen CAA_SIZE	File name length • The file name must not be longer than 32 characters. • In the case of more than 32 characters, the ETC_E_INVALIDPARM (0x9811000B) error is returned. Example: "firmware.efw" corresponds to 12 characters.
dwPassWd DWORD	Password
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).
pBuffer CAA_PVOID	Reference to memory buffer to which the values to be read are to be copied.
szSize CAA_SIZE	Size of the memory buffer transmitted to *pBuffer*. • The memory buffer must be big enough to accept the read object.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
szDataRead CAA_SIZE	Number of bytes for the read request (upload) that is actually read

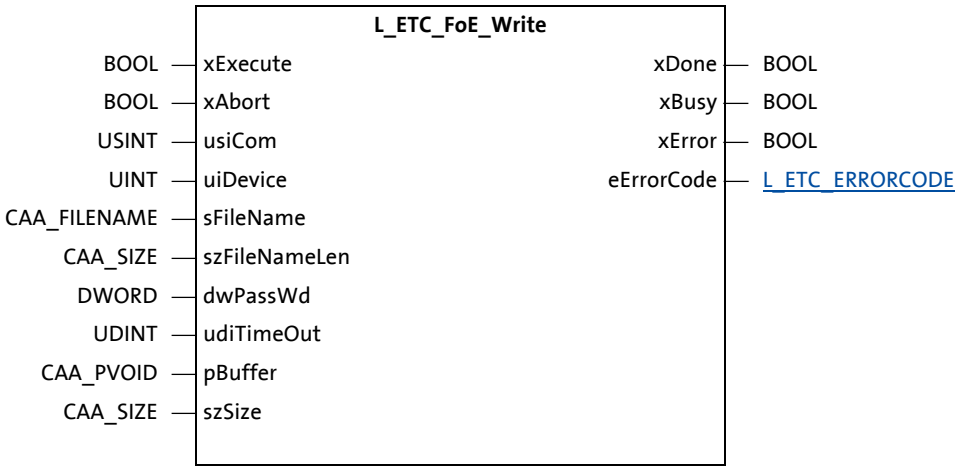
10.5.2 L_ETC_FoE_Write (FB)

This function block activates a file download to the slave or to the master.

Visualisation: VISU_L_ETC_FoE_Write

Note!

For executing the function block, the EtherCAT master and the slave must be at least in the "Pre-Operational" state.



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) activates a write request (download) of a file.
xAbort BOOL	A positive edge (TRUE) aborts a running write request (download). **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
usiCom USINT	EtherCAT master instance number • In the Lenze R3.x control technology, only the instance number '1' is currently supported. *usiCom* has the number '1' pre-assigned to it so that the input can be left open. • A value that does not equal '1' causes the error ETC_E_INVALIDPARM (0x9811000B).
uiDevice UINT	EtherCAT address of the master/slave. • The address 0 directly accesses the object directory of the EtherCAT master.
sFileName CAA_FILENAME	File name of file to be written • Max. 32 characters (see *szFileNameLen*) • Directory structures are not supported.
szFileNameLen CAA_SIZE	File name length • The file name must not be longer than 32 characters. • In the case of more than 32 characters, the ETC_E_INVALIDPARM (0x9811000B) error is returned. Example: "firmware.efw" corresponds to 12 characters.
dwPassWd DWORD	Password

Identifier/data type	Meaning/possible settings
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).
pBuffer CAA_PVOID	Reference to the memory buffer from which the values to be written are to be taken.
szSize CAA_SIZE	Number of the bytes to be written

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

10.6 Modular Machine Configuration Interface

The function blocks of the "Modular Machine Configuration Interface" can be used from release 3.10 and provide the following options:

- Assigning "Second Station Addresses" (alias addresses) to the EtherCAT slaves
  ▸ Address assignment (📖 120)
- Controlling the EtherCAT master and the behaviour
- Controlling the EtherCAT bus

▸ Modular machine configuration (📖 114)

10.6.1 L_ETC_MMCAssignAddress (FB)

This function block assigns a "Second Station Address" (alias address) to an EtherCAT slave.

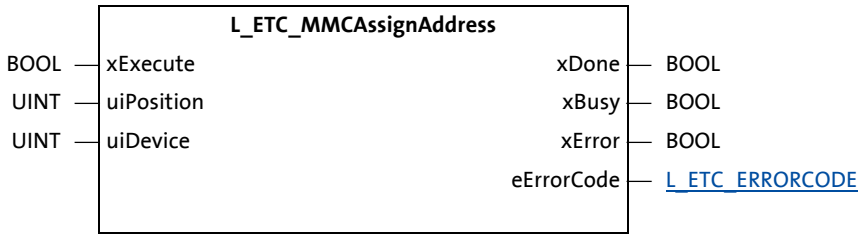
This function block only works if the function block L_ETC_MMCController (📖 163) or L_ETC_MMCControllerBus (📖 165) is operated with the ADDR_ASSIGNMENT_EXTERNALLY service. At the *szConfiguration* input, the current machine configuration that is really available at the EtherCAT bus needs to be selected.

Note!

Transfer of the "Second Station Address" (alias address)

- For transferring the alias address, the slave must be restarted by mains switching.
- In case of the **I/O module EPM-S130** from firmware version >1.28 and from release 3.12, the transfer of the alias address is executed by setting the index '0x2359' (subindex '0') = 1. An access is only possible in the "Pre-Operational" state.

Visualisation: VISU_L_ETC_MMCAssignAddress



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) assigns the slave at the (*uiPosition*) position in the logical ring of the EtherCAT network to the *uiDevice* address.
uiPosition UINT	Position in the logical ring of the EtherCAT network The first slave behind the EtherCAT master has the position '1'.
uiDevice UINT	"Second Station Address" (alias address) to be assigned The value is written into the EEPROM of the slave.

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No active action.
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

If additional parameters have to be written or initialised for a slave, this is the responsibility of the user.

10.6.2 L_ETC_MMCController

Note!

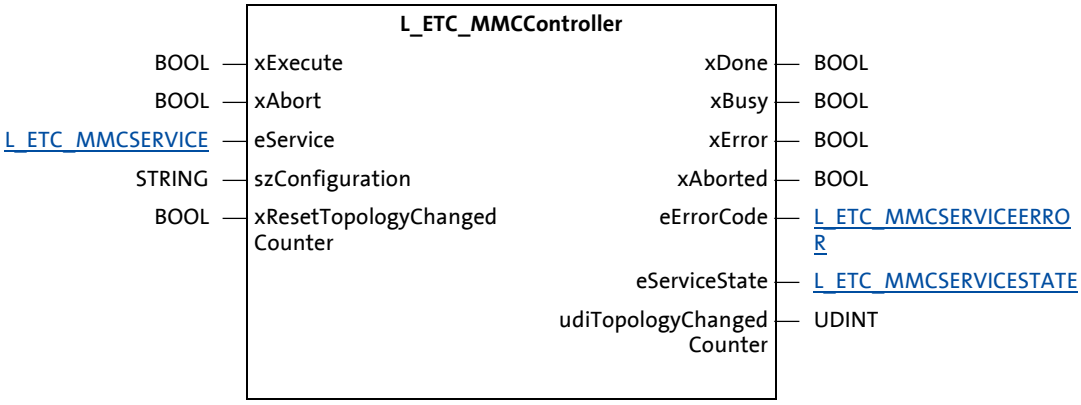
This function block ...
- may only be instanced once in the PLC program.
- must be called in the context of the EtherCAT bus cycle task.

This function block control the EtherCAT master and its behaviour.

If an instance of the **L_ETC_MMCController** function block is created in the PLC program, the EtherCAT master waits before starting the EtherCAT bus. Only after a service/mode has been defined (*eService* input) and the configuration (*szConfiguration* input), the EtherCAT master can be started.

The EtherCAT master is started with *xRestart* = TRUE (see L_IODrvEtherCAT (FB) (📖 149)).

Visualisation: VISU_L_ETC_MMCController



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	In case of a positive edge (TRUE), the data at the inputs *szConfiguration* and *eService* is accepted and the EtherCAT master is set to the mode given under *eService*.
xAbort BOOL	A positive edge (TRUE) stops the running service. **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
eService L_ETC_MMCSERVICE	Service to be executed
szConfiguration STRING	Name/designator of the current machine configuration (file mmc-0-conf.csv). The designator may only contain these characters: [a..z], [A..Z], [0..9], und [.-].
xResetTopologyChanged Counter BOOL	A positive edge (TRUE) resets the counter for topology changes to 'zero' (*udiTopologyChangedCounter* output).

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
xAborted BOOL	When *xAborted* = TRUE, the requested service is aborted. **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
eErrorCode L_ETC_MMCSERVICEERROR	Error code of the incorrectly executed service (xError = TRUE)
eServiceState L_ETC_MMCSERVICESTATE	Current status of the active service: • STOPPED: No active service. • RUNNING: Service is started and executed.
udiTopologyChanged Counter UDINT	Counter for topology changes • If the physical EtherCAT bus is changed and with an active service, the counter is incremented. • Between the change and the incrementation of the counter, a certain time elapses because if new slaves are added, they can be detected and initialised. • The counter is reset by the start of a service and by the xResetTopologyChangedCounter input.

Product features

Identifier/data type	Meaning/possible settings
GetActualConfiguration STRING	Current configuration under *szConfiguration* input If no configuration is active, an empty string is displayed.
GetRunningService L_ETC_MMCSERVICE	Currently running service

10.6.3 L_ETC_MMCControllerBus

Note!

This function block ...
- may only be instanced once in the PLC program.
- must be called in the context of the EtherCAT bus cycle task.

This function block is an extension for controlling the EtherCAT bus and contains the same inputs, outputs and features as L_ETC_MMCController (📖 163).

The following table shows the bus control as a function of the service:

Service	Final state when xExecute = TRUE	Final state when xAbort = TRUE
RUN_WITHOUT_CHECK RUN_OPTIONAL_SLAVES_ALLOWED RUN_OPTIONAL_SLAVES_PROHIBITED	Operational	Init
ADDR_ASSIGNMENT_EXTERNALLY	Pre-Operational [1)]	Init
ADDR_ASSIGNMENT_CONFIG_SLAVEORDER ADDR_ASSIGNMENT_CONFIG_PARAMETER	Pre-Operational [1)]	Init [2)]

1) Due to a temporarily generated configuration, the master sets the slaves to the "Pre-Operational" state.

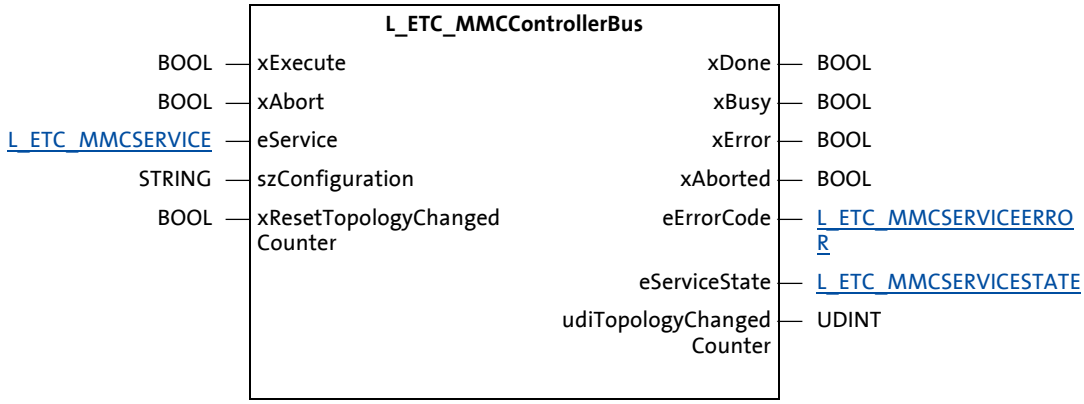
2) After the addresses have been automatically assigned, the master is set to the "Init" state again.

As soon as a RUN service is started, the EtherCAT bus is automatically set internally to the "Operational" state.

When a RUN service is started, the EtherCAT master is set to the "Operational" state.

When a RUN service is terminated, the EtherCAT master is set to the "Init" state.

Visualisation: VISU_L_ETC_MMCControllerBus



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	In case of a positive edge (TRUE), the data at the inputs *szConfiguration* and *eService* is accepted and the EtherCAT master is set to the mode given under *eService*.
xAbort BOOL	A positive edge (TRUE) stops the running service. **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
eService L_ETC_MMCSERVICE	Service to be executed
szConfiguration STRING	Name/designator of the current machine configuration (file mmc-0-conf.csv). The designator may only contain these characters: [a..z], [A..Z], [0..9], und [.-].
xResetTopologyChanged Counter BOOL	A positive edge (TRUE) resets the counter for topology changes to 'zero' (*udiTopologyChangedCounter* output).

Outputs (VAR_OUTPUT)

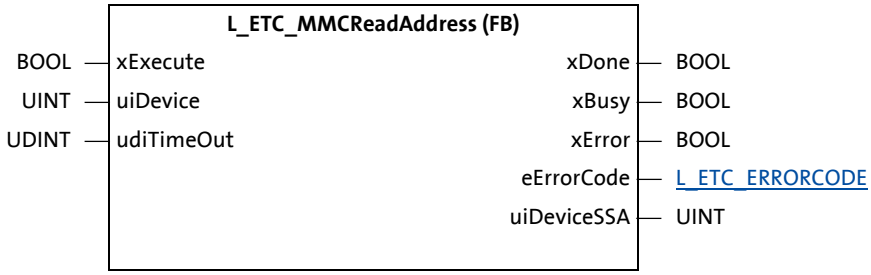
Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
xAborted BOOL	When *xAborted* = TRUE, the requested service is aborted. **Note:** An abort can only be executed in the EtherCAT state "Init" (before, set *xStopBus* = TRUE, see L_IODrvEtherCAT (FB) (📖 149)).
eErrorCode L_ETC_MMCSERVICEERROR	Error code of the incorrectly executed service (xError = TRUE)
eServiceState L_ETC_MMCSERVICESTATE	Current status of the active service: • STOPPED: No active service. • RUNNING: Service is started and executed.
udiTopologyChanged Counter UDINT	Counter for topology changes • If the physical EtherCAT bus is changed and with an active service, the counter is incremented. • Between the change and the incrementation of the counter, a certain time elapses because if new slaves are added, they can be detected and initialised. • The counter is reset by the start of a service and by the xResetTopologyChangedCounter input.

Product features

Identifier/data type	Meaning/possible settings
GetActualConfiguration STRING	Current configuration under *szConfiguration* input If no configuration is active, an empty string is displayed.
GetRunningService L_ETC_MMCSERVICE	Currently running service

10.6.4 L_ETC_MMCReadAddress (FB)

This function block reads a previously assigned "Second Station Address" (alias address) from an EtherCAT slave.



Inputs (VAR_INPUT)

Bezeichner/Datentyp	Bedeutung/Einstellmöglichkeiten
xExecute BOOL	On a positive edge (TRUE), the "Second Station Address" (alias address) is read from the address specified in *uiDevice*.
uiDevice UINT	Address of the EtherCAT slave to be read from.
udiTimeOut UDINT	Timeout value for the read operation in [ms].

Outputs (VAR_OUTPUT)

Bezeichner/Datentyp	Bedeutung/Einstellmöglichkeiten
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
uiDeviceSSA UINT	EEPROM value for the "Second Station Address" (alias address) of the EtherCAT slave whose addresses are read via input *iuDevice*.

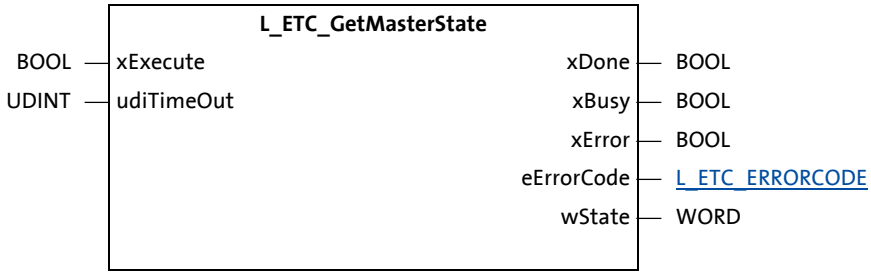
10.7 State Machine Interface

The "State Machine Interface" provides function blocks for setting and maintaining the state of master and slave.

10.7.1 L_ETC_GetMasterState

This function block detects the current state of the EtherCAT master.

Visualisation: VISU_L_ETC_GetMasterState



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) detects the current EtherCAT master state.
udiTimeout UDINT	Timeout in milliseconds (ms)

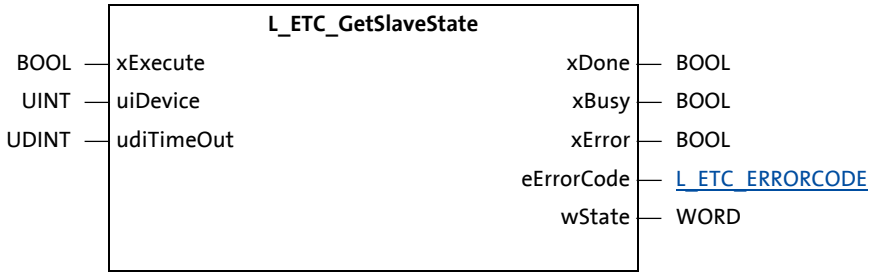
Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
wState WORD	Current master state of the L_ETC_STATE (📖 178) type

10.7.2 L_ETC_GetSlaveState

This function block detects the current state of the EtherCAT slave.

Visualisation: VISU_L_ETC_GetSlaveState



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) detects the current EtherCAT slave status of the slave given by *uiDevice*.
uiDevice UINT	EtherCAT address of the slave.
udiTimeout UDINT	Timeout in milliseconds (ms)

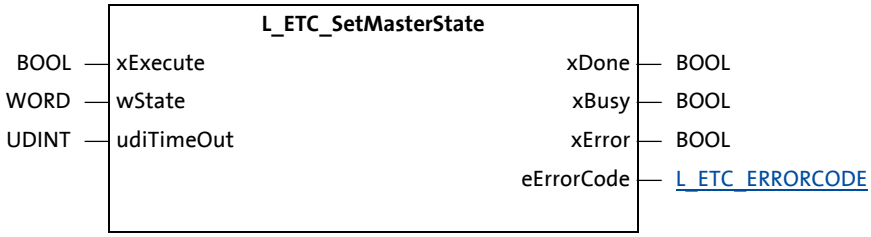
Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).
wState WORD	Current slave state of the L_ETC_STATE (📖 178) type

10.7.3 L_ETC_SetMasterState

This function block sets the state of the EtherCAT master.

Visualisation: VISU_L_ETC_SetMasterState



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) sets the EtherCAT master state given by *wState*.
wState WORD	EtherCAT master state to be set of the L_ETC_STATE (📖 178) type
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

10.7.4 L_ETC_SetSlaveState

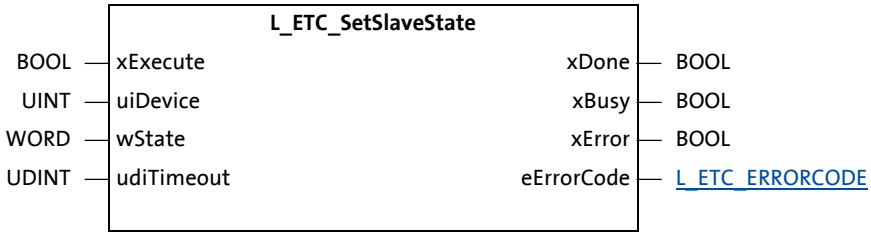
This function block sets the state of the EtherCAT slave.

Visualisation: VISU_L_ETC_SetSlaveState

Note!

The status of the slaves cannot be higher than the status of the EtherCAT master.

Example: If the master is in the Pre-Operational state, the state of a slave cannot be "Safe-Operational" or "Operational".



Inputs (VAR_INPUT)

Identifier/data type	Meaning/possible settings
xExecute BOOL	A positive edge (TRUE) sets the EtherCAT slave state by the slave given by *uiDevice*.
uiDevice UINT	EtherCAT address of the slave.
wState WORD	EtherCAT slave state to be set of the L_ETC_STATE (📖 178) type
udiTimeout UDINT	Timeout in milliseconds (ms) • The value '0' is not permissible and causes the error ETC_E_INVALIDPARM (0x9811000B).

Outputs (VAR_OUTPUT)

Identifier/data type	Meaning/possible settings
xDone BOOL	• TRUE: An action has been executed successfully. • FALSE: No action active / action is still being executed.
xBusy BOOL	• TRUE: An action is currently being executed. • FALSE: No action active
xError BOOL	• TRUE: An error has occurred. • FALSE: No error
eErrorCode L_ETC_ERRORCODE	Error code of the incorrectly executed action (*xError* = TRUE). A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

10.8 Data types

The Lenze-specific data types described in the following are used in some functions and function blocks of the L_IODrvEtherCAT function library (📖 126).

10.8.1 L_ETC_COE_EMERGENCY

The type **L_ETC_COE_EMERGENCY** describes errors that occur at a particular EtherCAT slave.

```
TYPE L_ETC_COE_EMERGENCY :
STRUCT
   uiDevice : UNIT;                // Slave address
   dwTimestamp : DATE_AND_TIME;    // Timestamp UTC (see FUNC SysTimeRtcGet())
   wErrorCode : WORD;                // Error code, see specification for details
   byErrorRegister : BYTE;           // Error register
   abyData : ARRAY[0..4] OF BYTE;    // Error data

END_STRUCT
END_TYPE
```

10.8.2 L_ETC_COE_EMERGENCY_BUFFER_DATA

The **L_ETC_COE_EMERGENCY_BUFFER_DATA** type describes a flag for displaying an emergency buffer overflow.

```
TYPE L_ETC_COE_EMERGENCY_BUFFER_DATA :
STRUCT
   structEmergency : L_ETC_COE_EMERGENCY; // Struct Emergency
   bBufferOverflow : BOOL;                // Flag for emergency buffer overflow

END_STRUCT
END_TYPE
```

10.8.3 L_ETC_COE_FLAGS

The **L_ETC_COE_FLAGS** type controls the behaviour of the function blocks L_ETC_CoE_SdoReadEx (FB) (📖 136) and L_ETC_CoE_SdoWriteEx (FB) (📖 142).

```
TYPE L_ETC_COE_FLAGS :
(
   ETC_E_COMPLETEACCESS := 16#00000001
)  DWORD;
END_TYPE
```

10.8.4 L_ETC_DIAGNOSTIC

The **L_ETC_DIAGNOSTIC** type describes diagnostic information which is returned e.g. via the L_ETC_GetMasterDiagnostic (FB) (📖 155) function block.

Observe the instructions for setting and resetting the flag "xNotAllSlavesInOperational" at output *oDiagnostic* of the FB ▸ L_ETC_GetMasterDiagnostic (FB) (📖 155) .

```
TYPE L_ETC_DIAGNOSTIC :
STRUCT
   wState : L_ETC_STATE;                      // Master state
   stState : STRING;                          // Master state as text

   xEthernetLinkup : BOOL;                    // Link up
   xNotAllSlavesOperational : BOOL;          // Indicates that not all slaves set in
                                                operational

   udiFrameCounter : UDINT;                   // Ethernet frame counter
   udiFrameResponseErrorCounter : UDINT;      // Frame loss counter
   udiCycWorkingCounterErrorCounter : UDINT; // WKC error counter

   xBusMismatch : BOOL;                      // Configuration and topology mismatch
   uiNumberOfSlavesFound : UINT;              // Number of found slaves
   uiNumberOfSlavesConfigured : UINT;         // Number of slaves configured

   dwLastErrorCode : DWORD;                   // Last error
   stLastError : STRING;                      // Last error as string

   xDC_Enabled : BOOL;                        // DC in use
   xDC_InSync : BOOL;                         // DC all slaves are in-sync
   xDC_Busy : BOOL;                           // DC re-sync running
   diDC_CurrentDeviation : DINT;              // DC current deviation

   udiEmergencyNo : UDINT;                    // COE Emergency Counter

END_STRUCT
END_TYPE
```

A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

10.8.5 L_ETC_ERRORCODE

The type **L_ETC_ERRORCODE** describes all possible EtherCAT error codes that can be output at the *eErrorCode* output of the EtherCAT function blocks.

A detailed description of the EtherCAT error codes is given in the chapter "System error messages" (📖 205).

10.8.6 L_ETC_EVTPARAM_PARAMETERTRANSFER

The type **L_ETC_EVTPARAM_PARAMETERTRANSFER** describes parameter transfer to a particular EtherCAT node or to all EtherCAT nodes.

(See also L_ETC_PARAMETERTRANSFERSERVICE_CODE (📖 177))

```
TYPE L_ETC_EVTPARAM_PARAMETERTRANSFER :
   (wService,    // Parameter transfer service
                 // (start: 16#0001, state: 16#0002, done: 16#0004, abort: 16#8000)
    uiDevice     // EtherCAT address (All devices: 16#FFFF)
   ) WORD;

   (dwResult     // Local parameter transfer in progress (wired OR)
   )  DWORD;
END_TYPE
```

10.8.7 L_ETC_IOCTLOPARMS

The **L_ETC_IOCTLOPARMS** type describes the reference to the I/O buffer e.g. for the L_ETC_IoControl (FUN) (📖 148) function.

```
TYPE L_ETC_IOCTLOPARMS :
STRUCT
   pbyInBuf : POINTER TO BYTE;        // Input data buffer
   dwInBufSize : DWORD;               // Size of input data buffer in bytes
   pbyOutBuf : POINTER TO BYTE;       // Output data buffer
   dwOutBufSize : DWORD;              // Size of output data buffer in bytes
   pdwNumOutData : POINTER TO DWORD; // Number of output data bytes stored in
                                      // output data buffer
END_STRUCT
END_TYPE
```

10.8.8 L_ETC_LANGUAGE

The **L_ETC_LANGUAGE** type describes the languages in which e.g. an error text can be output via the L_ETC_GetErrorString (FUN) (📖 154) function.

```
TYPE L_ETC_LANGUAGE :
(
   eng,    // english
   de      // german
);
END_TYPE
```

10.8.9 L_ETC_MMCSERVICE

The **L_ETC_MMCSERVICE** type contains the possible services for the function blocks L_ETC_MMCController (📖 163) and L_ETC_MMCControllerBus (📖 165).

Principally, there are two types of services:

- RUN services for the operating mode
- ADDR_ASSIGNMENT services for the automatic and manual address allocation

Please note: Dependencies between configuration files and services (📖 119)

No. [hex]	Identifier/data type	Meaning/possible settings
0x0000	NONE	No operation (NOP)
0x0001	RUN_WITHOUT_CHECK	When the EtherCAT bus is started, no check for mandatory or optional slaves and the additional slave identification is carried out. The prerequisite for this is the address assignment to the slaves. Once started successfully, this service runs until a positive edge is applied to the xAbort input.
0x0002	RUN_OPTIONAL_SLAVES_ALLOWED	When the bus is started, a check for mandatory or optional slaves is carried out. The EtherCAT bus changes to the "Pre-Operational" state and more if ... • all mandatory slaves are available; • no or several optional slaves are available; • the additional identification for all available slaves is correct. The prerequisite for this is the address assignment to the slaves. Once started successfully, this service runs until a positive edge is applied to the xAbort input.
0x0003	RUN_OPTIONAL_SLAVES_PROHIBITED	When the bus is started, a check for mandatory slaves is carried out. The EtherCAT bus changes to the "Pre-Operational" state and more if ... • all mandatory slaves are available; • no optional slave is available; • the additional identification for the mandatory slaves is correct. The prerequisite for this is the address assignment to the slaves. Once started successfully, this service runs until a positive edge is applied to the xAbort input.
0x0100	ADDR_ASSIGNMENT_EXTERNALLY	Address allocation via an external tool or from the PLC application: • The EtherCAT master is set to the "Pre-Operational" state based on the slave EEPROM information. • An access via an external tool or from the PLC application can be carried out. • The addresses can be assigned via the L_ETC_MMCAssignAddress (FB) (📖 162) function block. • CoE objects can be accessed in read-only mode. Once started successfully, this service runs until a positive edge is applied to the xAbort input.

No. [hex]	Identifier/data type	Meaning/possible settings
0x0200	ADDR_ASSIGNMENT_CONFIG_SLAVEORDER	Automatic address allocation via the configuration: • The EtherCAT master is set to the "Pre-Operational" state based on the slave EEPROM information. • If the order of the slaves at the EtherCAT complies with the selected configuration (vendor-ID and product code), the persistent addresses are written to the EEPROM. Once started successfully, this service runs until the service has been completed successfully (xDone = TRUE) or an error has occurred (xError = TRUE).
0x0300	ADDR_ASSIGNMENT_CONFIG_PARAMETER	Is currently not supported.

10.8.10 L_ETC_MMCSERVICEERROR

The **L_ETC_MMCSERVICEERROR** describes the errors that can be provided at the *eErrorCode* output of the function blocks L_ETC_MMCController (📖 163) and L_ETC_MMCControllerBus (📖 165).

More error descriptions regarding modular machine configuration can be found in the "Diagnostics" chapter:

▸ Error messages for modular machine configuration (📖 212)

No. [hex]	Identifier/data type	Meaning/possible settings
0x00000000	NO_ERROR	No error
0x00000001	CONFIG_INVALID	The selected configuration is not valid.
0x00000002	CONFIG_FILE_ERROR	• The **mmc-0-conf.csv** configuration file is not available. • An error has occurred during data import (Parsing Error).
0x00000003	IDENT_FILE_ERROR	No configuration file (mmc-0-conf-csv) is available or an error has occurred during data import (Parsing Error)
0x00000100	SERVICE_INVALID	The selected service (L_ETC_MMCSERVICE (📖 175)) is not valid.
0x00000101	SERVICE_INVALID_STATE	In the current status of the EtherCAT master, the selected service cannot be executed. The EtherCAT master has to be set to the "Init" state.
0x00010000	ADDR_INTERNAL_ERROR	When the service has been started, an internal error has occurred. Please contact Lenze!
0x00010001	ADDR_NO_MEMORY	When the service has been started, an internal memory error has occurred. Please contact Lenze!
0x00010002	ADDR_LESS_CONNECTED	In the address assignment, less slaves are connected to the physical EtherCAT bus than given in the active configuration.
0x00010003	ADDR_MORE_CONNECTED	In the address assignment, more slaves are connected to the physical etherCAT bus than given in the active configuration.
0x00010004	ADDR_UNEXPECTED_DEVICE	In the address assignment, the physical EtherCAT bus and the configuration provide the same number of slaves but at least one slave comes with a different device type. (For more information, see the logbook).
0x00010005	ADDR_ASSIGN_ERROR	An error has occurred during the address assignment. (For more information, see the logbook).

10.8.11 L_ETC_MMCSERVICESTATE

The **L_ETC_MMCSERVICESTATE** type displays the status of a service (L_ETC_MMCSERVICE (📖 175)) during processing.

```
TYPE L_ETC_MMCSERVICESTATE :
(
    STOPPED,    // Service was stopped
    RUNNING,    // Service was started and is running
    NOT_READY   // Function block is not ready
) WORD;
END_TYPE
```

No. [hex]	Identifier/data type	Meaning/possible settings
0x0000	STOPPED	Currently, no service is started.
0x0001	Running	Current service has been started and running.
0x0002	NOT_READY	The function block is not ready at the moment. Especially during the initialisation phase when the PLC program has been started but the MMC files have not been imported yet. If the NOT_READY state has been set, the inputs are ignored and no actions can be executed.

10.8.12 L_ETC_PARAMETERTRANSFERSERVICE_CODE

The type **L_ETC_PARAMETERTRANSFERSERVICE_CODE** describes all possible services or actions for parameter data transfer.

(See also L_ETC_EVTPARAM_PARAMETERTRANSFER (📖 174))

```
TYPE L_ETC_PARAMETERTRANSFERSERVICE_CODE :
   (START := 16#0001,   // Start of service
    STATE := 16#0002,   // State polling
    DONE  := 16#0004,   // End of service
    ABORT := 16#8000,   // Abort of service
   ) WORD;
END_TYPE
```

10.8.13 L_ETC_SLAVE_PORTS

The type **L_ETC_SLAVE_PORTS** describes ports 0 ... 3 of the EtherCAT slaves.

```
TYPE L_ETC_SLAVE_PORTS :
STRUCT
   Port0 : BYTE;    // Slave port 0
   Port1 : BYTE;    // Slave port 1
   Port2 : BYTE;    // Slave port 2
   Port3 : BYTE;    // Slave port 3

END_STRUCT
END_TYPE
```

10.8.14 L_ETC_STATE

The **L_ETC_STATE** describes all possible states of an EtherCAT master and the EtherCAT slaves.

```
TYPE L_ETC_STATE :                          // EtherCAT states
(
    ETC_STATE_NONE            := 16#0000,   // Unknown state
    ETC_STATE_INIT            := 16#0001,   // INIT state
    ETC_STATE_PREOPERATIONAL  := 16#0002,   // Pre-Operational state
    ETC_STATE_BOOT            := 16#0003,   // Bootstrap state
    ETC_STATE_SAVEOPERATIONAL := 16#0004,   // Safe-Operational state
    ETC_STATE_OPERATIONAL     := 16#0008,   // Operational state
    ETC_STATE_UNKNOWN         := 16#FFEF    // Unknown state
) WORD;
END_TYPE
```

ETC_STATE_NONE is available for the Modular machine configuration (📖 114).

ETC_STATE_UNKNOWN is currently not available for Lenze devices.

11 Restarting the EtherCAT fieldbus

During operation it may be required to restart the EtherCAT fieldbus. This can for instance be necessary after fatal faults like a cable break.

Restart the EtherCAT field bus (general description):

1. Activate the controller inhibit for the inverters.
2. Request restart of the EtherCAT master.

   In the L_IODrvEtherCAT (FB) (📖 149)/L_IODrvEtherCAT_Diag (FB) (📖 151) function block from the L_IODrvEtherCAT function library (📖 126), set the **xRestart input = TRUE**:

   ```
   Ethercat_Master.xRestart := TRUE;
   ```

Restart the EtherCAT fieldbus using FAST Motion FBs:

- The restart of the EtherCAT field bus including the basic initialisation of the motion axes can be carried out most easily via the FB **L_MC1P_ReinitNode**. The FB **L_MC1P_ReinitNode** can be found in the function library L_MC1P_MotionControlBasic.
- Optionally, by setting the input `xInitCommunication := TRUE;` the bus system can also be reinitialised. This eliminates the need for application-specific control of the EtherCAT master to restart the bus system.

Restart the EtherCAT fieldbus using L-force Motion FBs:

1. Run the FB **SMC3_ReInitDrive** from the SM3_Basic function library.
2. Execute the FB **MC_Reset** from the function library SM3_Basic.

Please note for Motion drives:

- When restarting EtherCAT the control mode is set to **SMC_position**.
- If another control mode was active before the restart, switch the control mode using the function block **SMC_SetControllerMode.**

Please note for the Servo Inverter i700:

- When restarting EtherCAT, the control mode is set to **CSP: Cyclic Synchronous Position**.
- If another control mode was active before the restart, switch the control mode using the function block **SMC_SetControllerMode**.

Note!

Use of "FAST Motion" function blocks

When the EtherCAT bus is restarted, the control mode of the Motion drives is set to the "PosCtrlDrive" mode.

If a different control mode was active before the restart, change the control mode by means of the **L_MC1P_SetControlMode** function block.

Use of "L-force Motion" function blocks

When the EtherCAT bus is restarted, the control mode of the Motion drive is set to the "SMC_position" mode or in case of the Servo Inverter i700 to "CSP: Cyclic Synchronous Position".

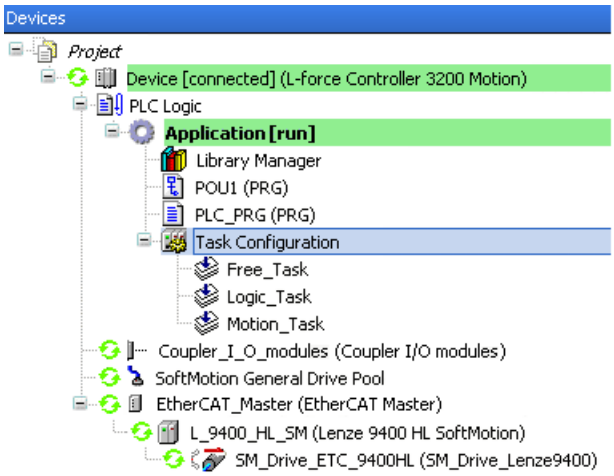
If a different control mode was active before the restart, change the control mode by means of the **SMC_SetControllerMode** function block.

12 Defining the cycle time of the PLC project

In this chapter you'll learn how to ...

12.1 Determine the task utilisation of the application

In the online mode, the **Monitor** tab of the **Task Configuration** shows current status details and measurements of the cycles, cycle times, and jitters of the tasks contained.



Task Configuration

Properties | Monitor

Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)
Free_Task	Valid	353587	365215	227	210	1488	25	2
Logik_Task	Valid	353589	365217	114	104	650	20	-3
Motion_Task	Valid	353589	365217	299	271	647	20	-2

The values are updated in the same time interval as that used for monitoring the values from the controller.

If the cursor is on a task name field, the values displayed can be reset to 0 by the **Reset** context menu command (right-click the task name field).

How to determine the task utilisation:

Initial situation: A complete project, e.g. with a EtherCAT task and 2 lower priority tasks has been created.

1. For a first measurement of the task utilisation, set the cycle times of all cyclic tasks available in the PLC system "high" (e.g. EtherCAT task = 10 ms, all other cyclic tasks = 20 ms).
2. Use the menu command **Online → Login**, or log in on the Lenze Controller with **<Alt>+<F8>**.
   - For this, the PLC program must be error-free.
   - With the log-in, the fieldbus configuration and the PLC program are loaded to the controller.
3. Reset the values displayed on the **Monitor** tab of the **Task Configuration** to 0 after the complete run-up of the system.

   Execute the **Reset** command from the context menu of the task name field.
4. Read the displayed maximum computing time of the task with the highest priority.

   In the example above, the max. cycle time of the EtherCAT task is 647 µs.

The minimum cycle time (T_{min}) for a system is the result of the formula:

$$T_{min} = \text{Task utilisation x safety factor}$$

Note!

A safety factor of 1.5 should be included in the calculation.

12.2 Optimising the system

How to optimise the system:

1. Use the menu command **Online → Login**, or log in on the Lenze Controller with **<Alt>+<F8>**.
   - For this, the PLC program must be error-free.
   - With the log-in, the fieldbus configuration and the PLC program are loaded to the controller.
2. Check the task processing times.
3. Optimising the cycle times:
   - If technologically required, the cycle times of the remaining tasks with lower priorities can be decreased.
   - Condition: No task with a low priority must assign more than 60 percent of the corresponding cycle time in its task utilisation.

13 Diagnostics

This chapter provides information on diagnostics using the »PLC Designer« and the »WebConfig«. Moreover, error scenarios for the most frequent user errors are shown and system error message are described.

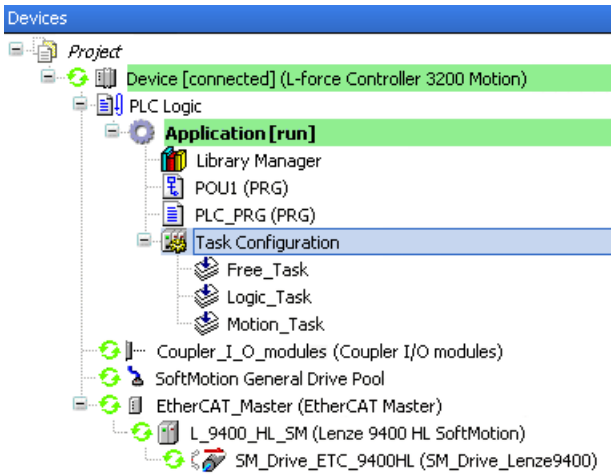
13.1 Diagnostics in the »PLC Designer«

A test project in which the diagnostic functions of the L_IODrvEtherCAT function library (📖 126) are used can be found in the Download area at www.Lenze.com:

Application Knowledge Base: All articles → Application Ideas Pool → Controller 3200 C

13.1.1 Representation in the online mode

When an online connection has been established to the Lenze Controller, the icon in front of the individual entries in the configuration tree provides information on the status of the respective EtherCAT node (in the example: two green arrows in each case):

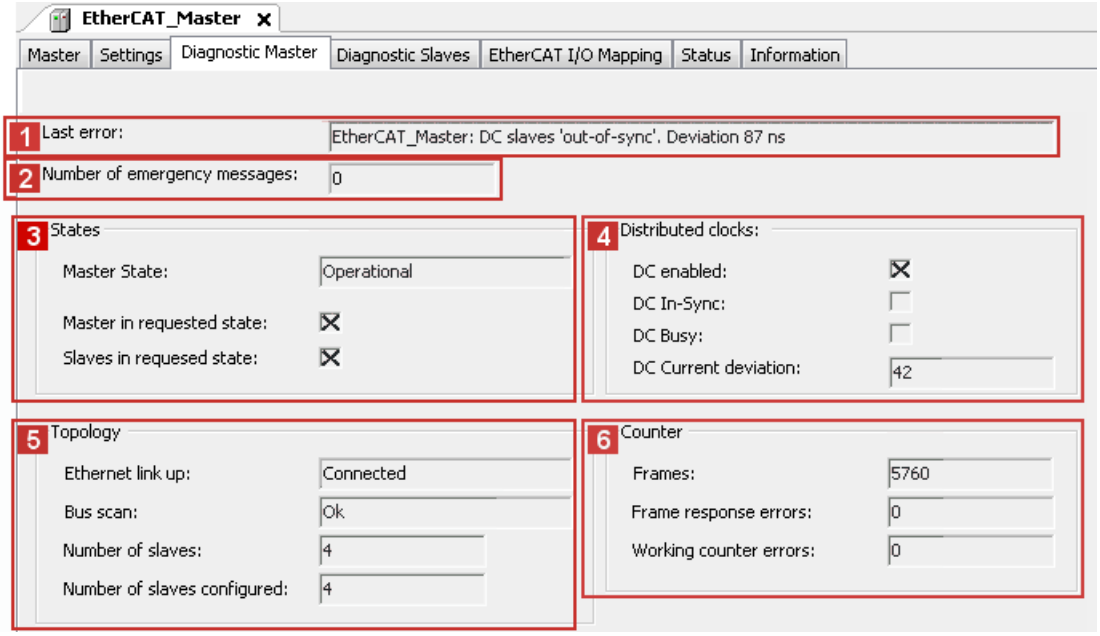


Symbol	Meaning
	Device is online • Successful online connection to the Lenze Controller • Successful online connection to the EtherCAT node. • Status of the node: Operational (OP)
	Device is online Possible states of the EtherCAT node: • INIT (initialisation) • PREOP (Pre-operational) • SAFEOP (Safe-operational) • No information on the node
no icon	Device is offline • No connection to the Lenze Controller

13.1.2 Diagnostic tabs of the EtherCAT master

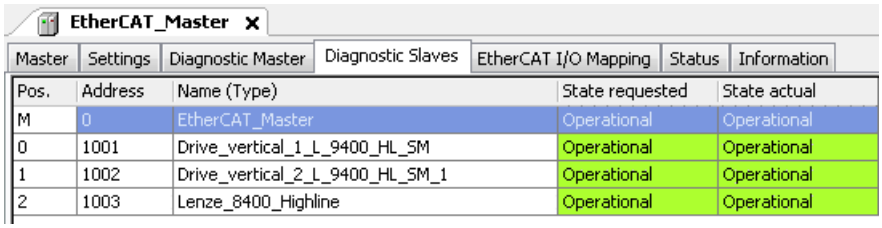
Only if an online connection to the Lenze Controller has been established, information is displayed in the diagnostic tabs.

The following information is displayed on the **Diagnostic Master** tab:



- 1 Error occurred last
- 2 Number of emergency frames
- 3 Status information
- 4 Information on "Distributed clocks"
- 5 Information on network topology
- 6 Frame and error counter

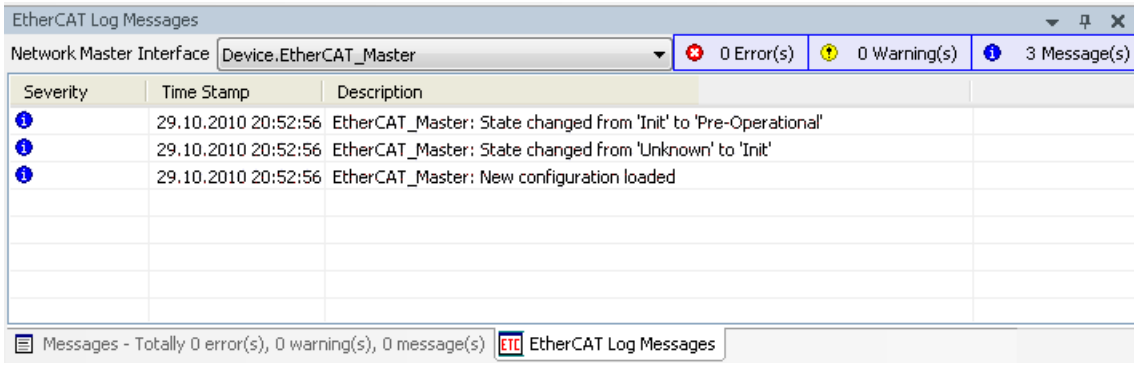
In addition to the EtherCAT states, the **Diagnostic Slaves** tab also displays the positions, addresses, and names of the EtherCAT nodes:



Pos.	Address	Name (Type)	State requested	State actual
M	0	EtherCAT_Master	Operational	Operational
0	1001	Drive_vertical_1_L_9400_HL_SM	Operational	Operational
1	1002	Drive_vertical_2_L_9400_HL_SM_1	Operational	Operational
2	1003	Lenze_8400_Highline	Operational	Operational

13.1.3 Display window for EtherCAT logbook messages

If you execute the menu command **View → EtherCAT log messages**, a window is opened in which Lenze Controller logbook messages (📖 214) are displayed.



Column	Description
Severity	Message type: error, warning, information
Time stamp	Date and time of occurrence of a message
Description	For message text, see Lenze Controller logbook messages (📖 214)

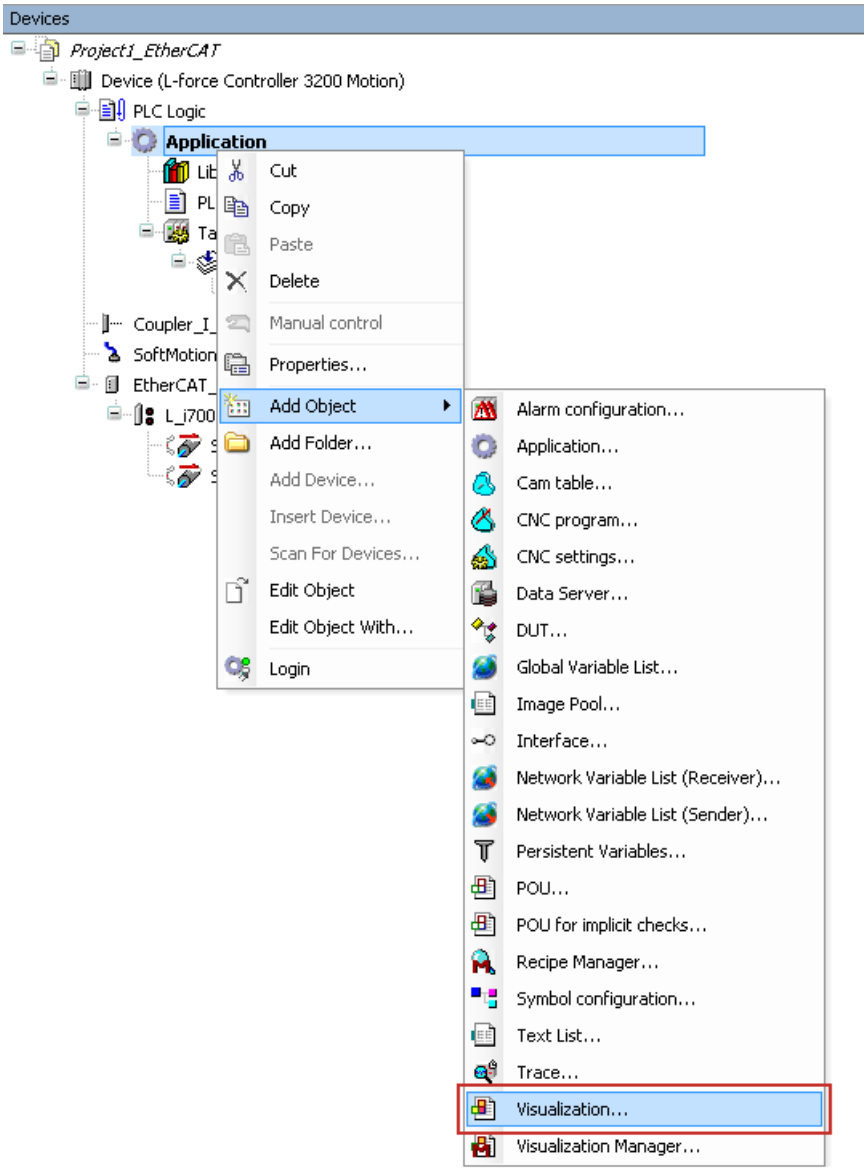
13.1.4 Visualisation of the function block L_ETC_GetMasterDiagnostic

In the visualisation of the L_ETC_GetMasterDiagnostic (FB) (📖 155) function block, variables for EtherCAT and DC states, (error) counters, error numbers etc. are displayed for diagnostic purposes.

How to create the visualisation of the L_ETC_GetMasterDiagnostic function block

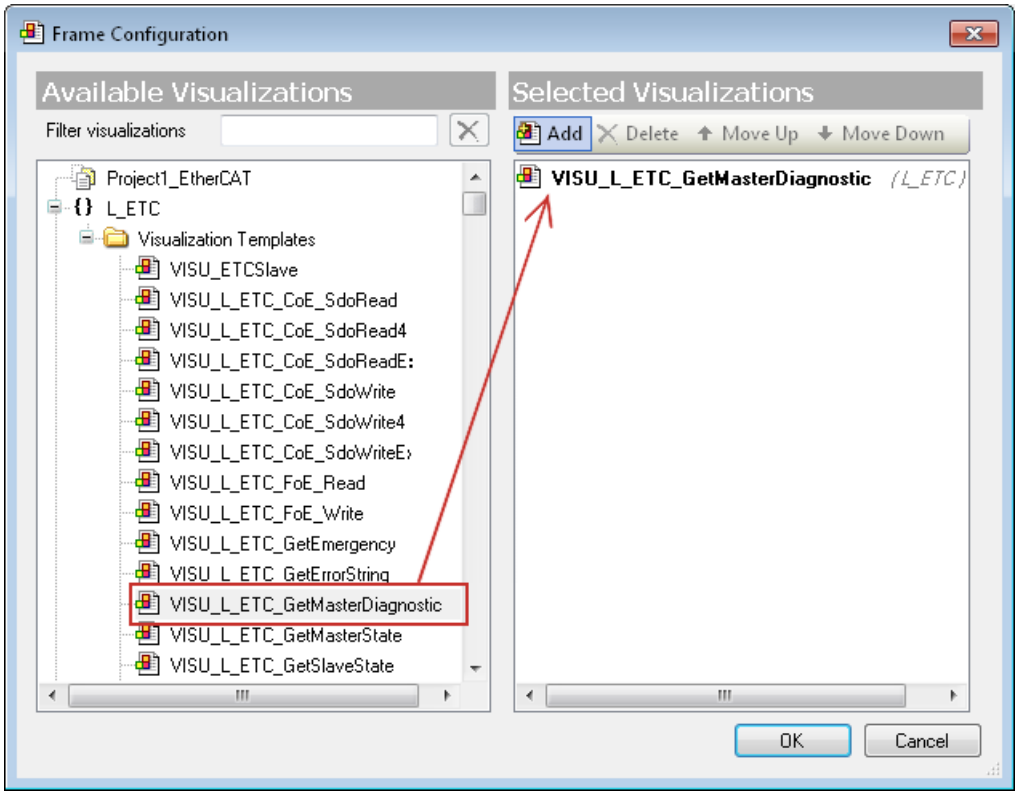
1. Open the context menu for **Application**, select the command **Add object → Visualisation...** and insert the visualisation of the function block **L_ETC_GetMasterDiagnostic**.

   Enter an appropriate name (e.g. "VISU_L_ETC_GetMasterDiagnostic").



2. Insert a frame in the visualisation with the help of the **frame tool**.

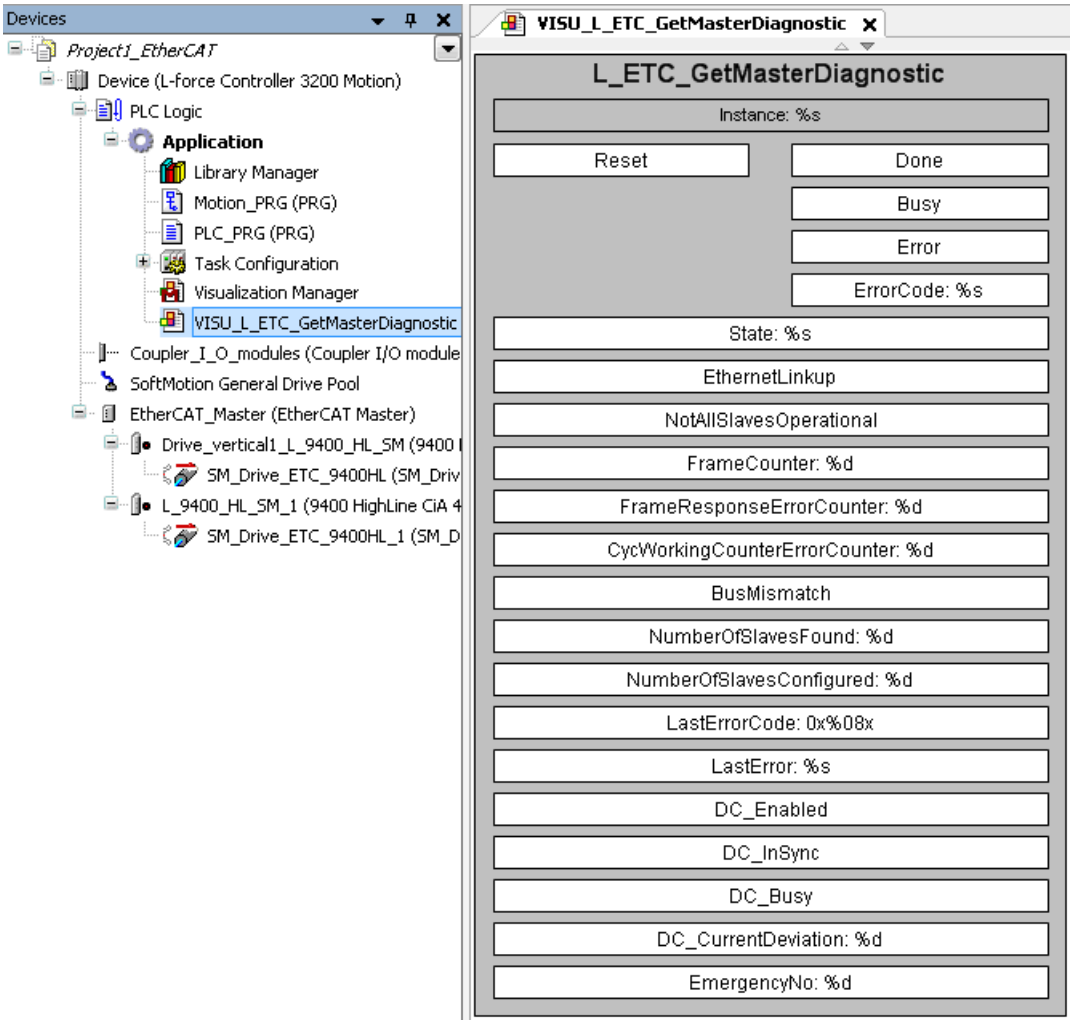
3. Add the frame visualisation of the function block **L_ETC_GetMasterDiagnostic** and close the dialog box by clicking the **OK** button.



The visualisation is added to the configuration tree of the »PLC Designer« project.

In general, all fields in the visualisation are initially white. If a status variable or a state is set or active (TRUE), the corresponding field is shown in green or red:

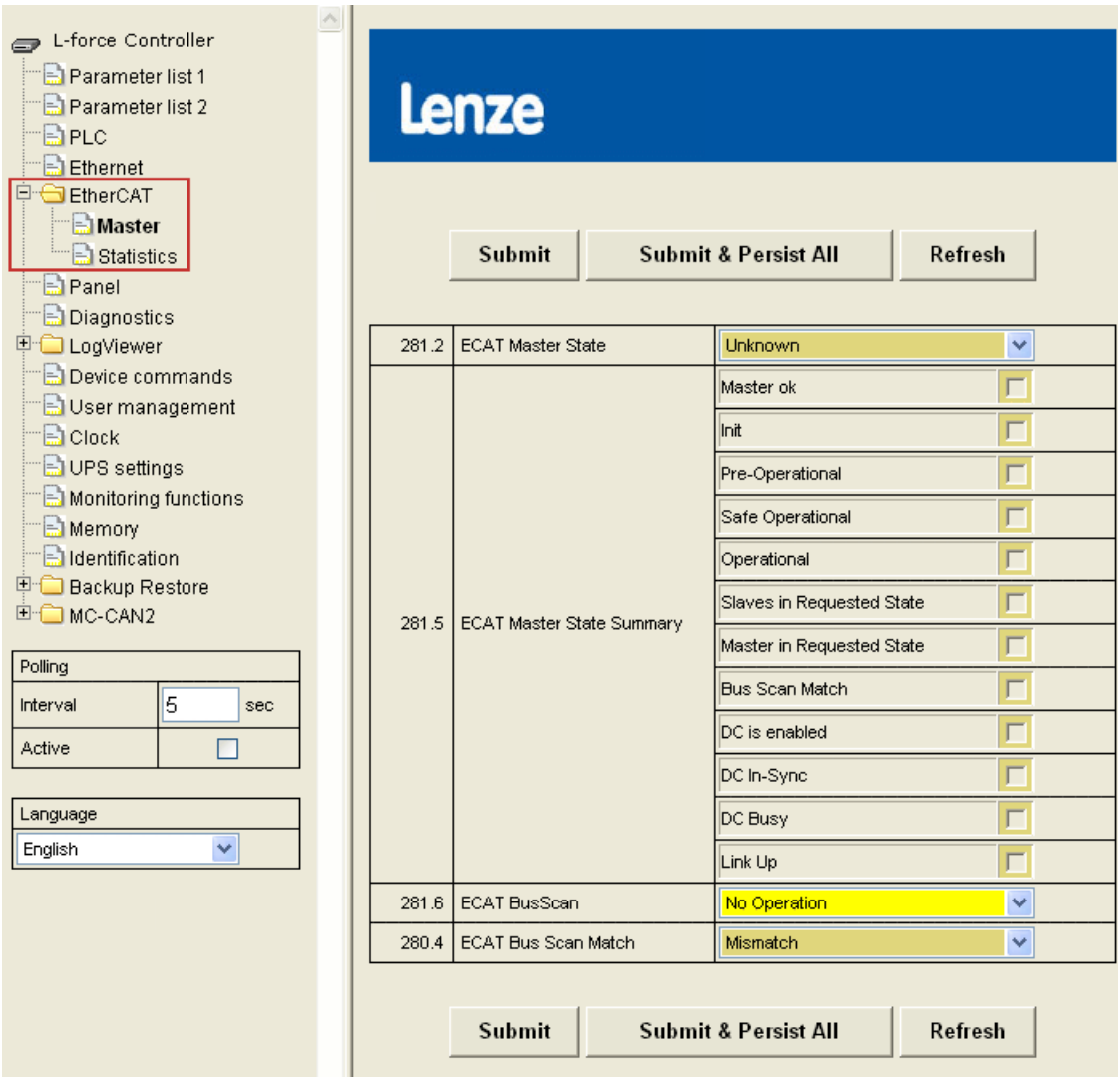
- Red fields represents an "error".
- Green fields display "information".



13.2 Diagnostic codes in the »WebConfig«

In »WebConfig«, you can view the EtherCAT diagnostic parameters under **EtherCAT → Master** and **EtherCAT → Statistics**.

▸ Parameter reference (📖 220)

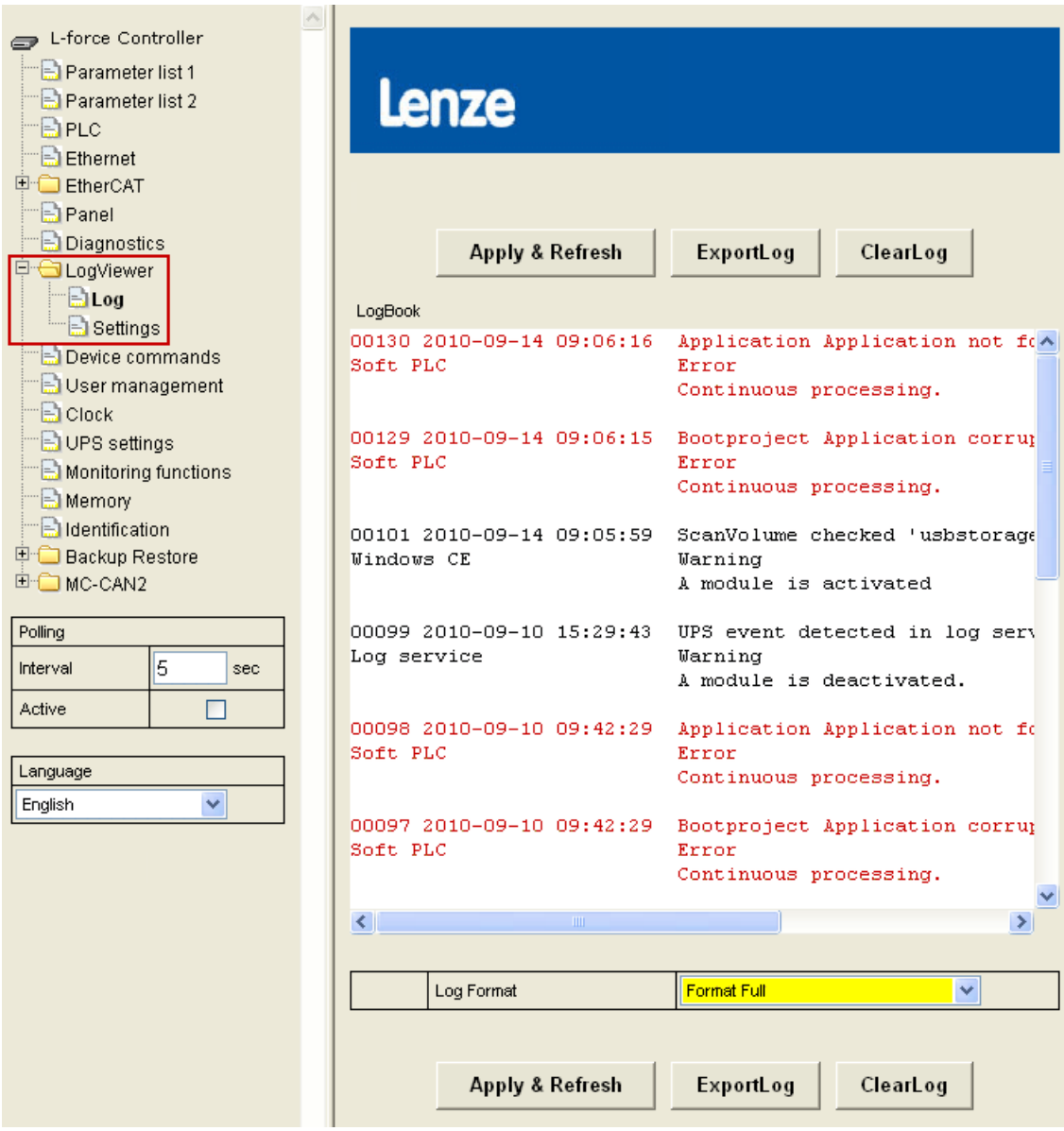


13.3 Logbook of the Lenze Controller in the »WebConfig«

The **logbook** of the »WebConfig« displays errors (highlighted in red), warnings, or information.

Read the messages in the logbook from bottom to top. The most recent message always appears at the top of the logbook.

▸ Lenze Controller logbook messages (📖 214)



Structure of the messages in the logbook:

• Running number • Date/time of occurrence • Location of occurrence	• Error description • Message type: error / warning / information • Action which caused the message

Settings for a compact representation of the messages:

1. Under **Logbook → Settings** in the **Application** area, set a checkmark in the checkbox for **EtherCAT Master Stack**.

2. Under **Logbook → Log** for **Log Format**, select "Format severity".

   Compact representation of the messages:

```
00130 F  Application Application not found to start
00129 F  Bootproject Application corrupt. CRC File is not valid
00101 W  ScanVolume überprüfte die 'usbstorage'-Partition und behob alle Fehler.
00099 W  Log-Service hat USV-Ereignis erkannt
00098 F  Application Application not found to start
00097 F  Bootproject Application corrupt. CRC File is not valid
00069 W  ScanVolume überprüfte die 'usbstorage'-Partition und behob alle Fehler.
00067 W  Log-Service hat USV-Ereignis erkannt
00066 F  Application Application not found to start
00065 F  Bootproject Application corrupt. CRC File is not valid
00035 W  Korrupte Restoredaten im Log-Service
00034 W  Log-Service hat USV-Ereignis erkannt
00033 F  Application Application not found to start
00032 F  Bootproject Application corrupt. CRC File is not valid
```

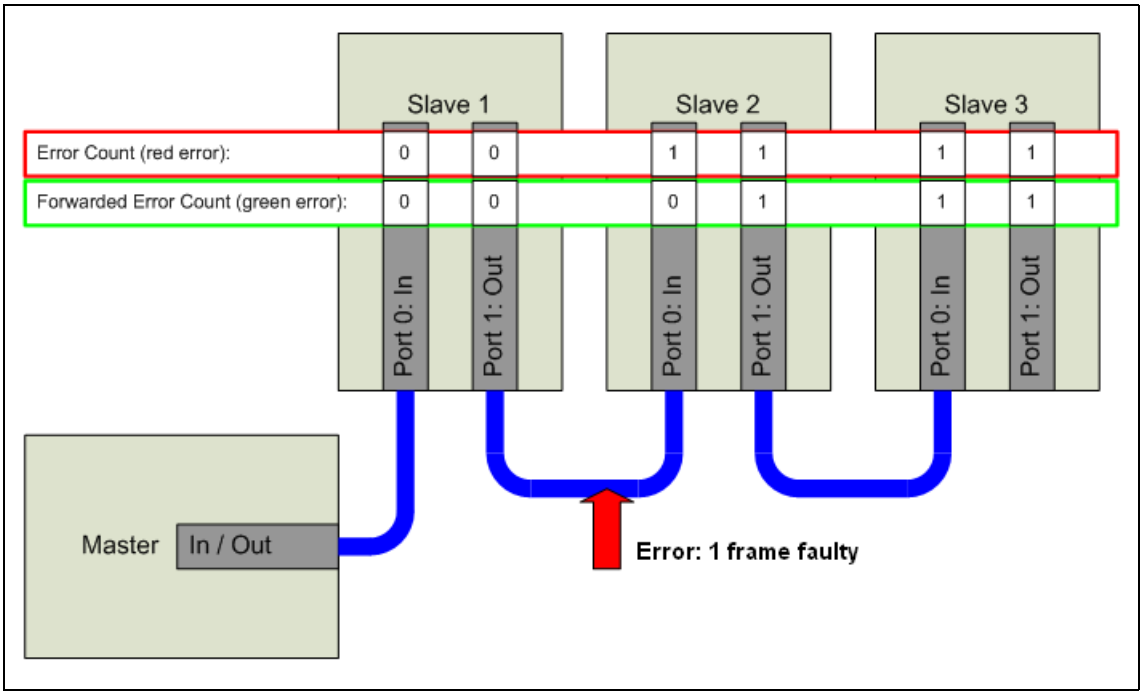
13.4 Error counters of the EtherCAT slaves

The EtherCAT slaves have numerical error counters for detecting and analysing error states. All error counters have a limited counting range of 0 ... 255. After the maximum value of 255 is reached, no "wrap-around" takes place. If the PLC application is to make a sensible evaluation of the error counters, these must be deleted by the application after the evaluation via a write access.

13.4.1 Error types "Errors" and "Forwarded Errors"

The EtherCAT slaves differ between errors detected in the slave for the first time (red error) and forwarded errors, i.e. errors that have already been detected in a previous slave (green error).

When the corresponding error counters are evaluated, an error in the EtherCAT network can be clearly assigned to a bus segment or a slave.



[13-1] Error types "Errors" and "Forwarded Errors"

13.4.2 Error counter reset from the application

The L_ETC_ReadErrCnt (FB) (📖 156) function block enables the PLC application to access the error counters by reading.

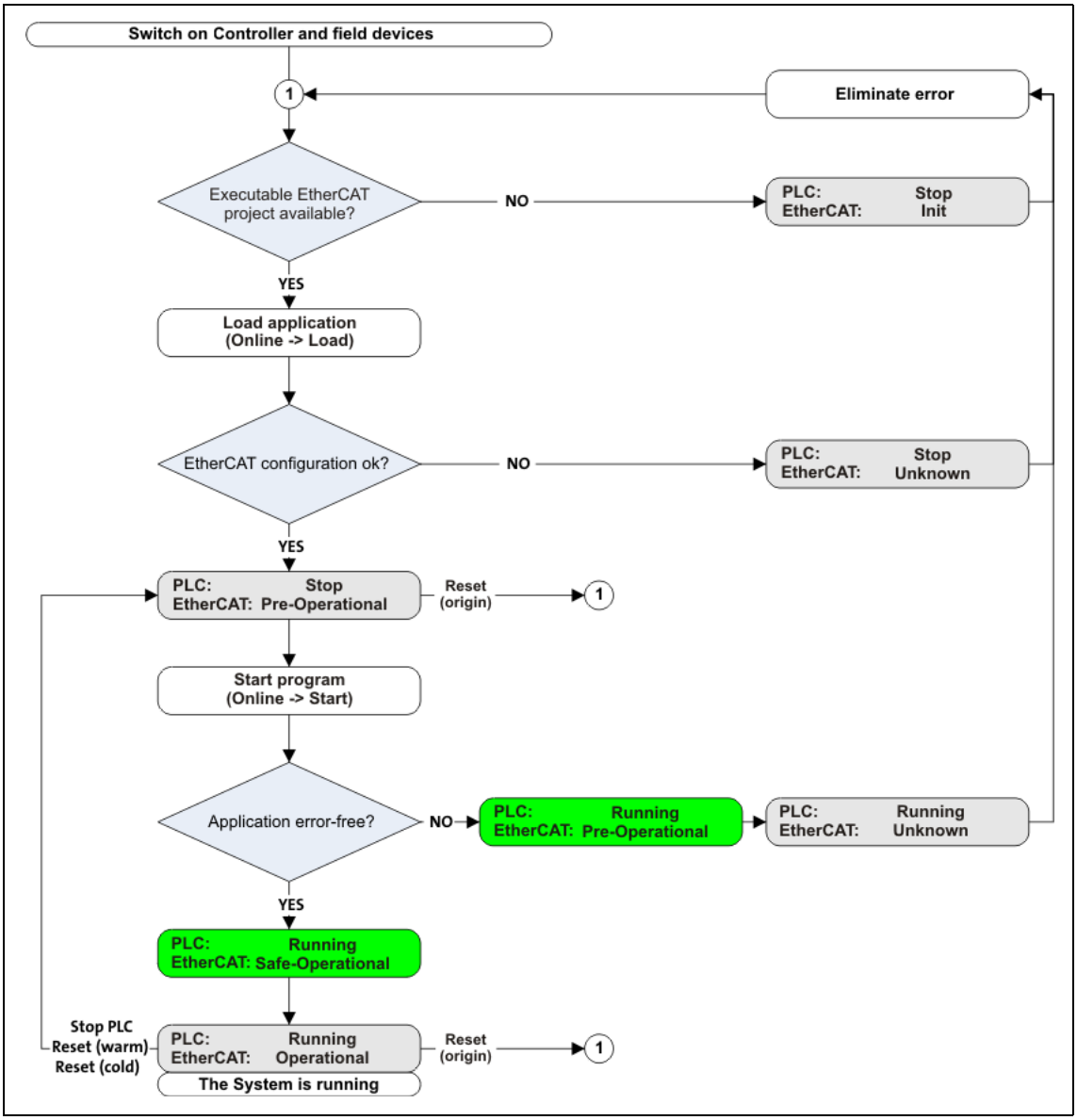
The L_ETC_ResetErrCnt (FB) (📖 157) function block resets the error counters to the value '0'.

Example

Once per minute the PLC reads the error counters and evaluates the contents. If it recognises an error counter value of '250', the application will reset the error counters. Depending on the EMC load of the network environment, approx. 1 to 2 frame errors per day in the fieldbus are normal. The error counters should not reach the value of 255.

13.5 Error scenarios

In the following sections, the causes and remedies for the most frequent user errors are described. The state diagram and descriptions on the next pages serve to localise and remove an error.



[13-2] Status diagram for system behaviour

13.5.1 The EtherCAT bus does not assume the "Pre-Operational" state.

During the start-up of the EtherCAT bus, a check is carried out at the transition from "Init" to "Pre-Operational" to determine whether the physical bus configuration corresponds to the bus configuration configured. If theses configurations are different, the master does not enter the "Pre-Operational" state.

Furthermore, the slaves are initialised during the transition from "Init" to "Pre-Operational". If this fails because, for instance, a slave denies the configuration, the master does not enter the "Pre-Operational" state.

Cause	**Faulty bus configuration and bus structure** • Slaves are missing. • Slaves have been inverted or their configuration is faulty. • A wrong slave type has been configured or is at the fieldbus. • The inputs and outputs of the EtherCAT communication module are reversed (IN/OUT connections). • Faulty wiring

Error message	`EtherCAT_Master: Start master failed. Bus mismatch`
	`...`
	`[DeviceName] (1002): Configuration mismatch. Check VendorID failed (0x3B / missing )`

Remedies	Correct the bus configuration or the physical bus structure. Afterwards, reload the »PLC Designer« application into the automation system.

Cause	**Wiring error: EtherCAT cable is not connected to the master.**
Error message	`EtherCAT_Master: Start master failed. EtherCAT cable disconnected`
Remedies	Correct wiring. Afterwards, reload the »PLC Designer« application into the automation system.

13.5.2 The EtherCAT bus does not assume the "Operational" state

The Lenze Controller causes the EtherCAT bus to assume the "Operational" state when the controller is set in RUN mode. The EtherCAT bus can only reach the "Operational" state if the fieldbus has previously allowed itself to be set to the "Pre-Operational" state.

Cause	**Start parameter could not be written.**

Error message	`EtherCAT_Master: Set master 'Operational' failed (0x4000005)`
	
	`[DeviceName] (1001): CoE 0x1234:5 - SDO Abort 'Object does not exist in the object dictionary (0x06020000)'`

Remedies	Correct the start-up parameters in »PLC Designer« on the **Configuration** tab of the EtherCAT master.

Cause	**Wiring error: The EtherCAT terminals (IN/OUT) of the slave were inverted.** A fieldbus scan does not indicate this error!
Error message	`EtherCAT_Master: Set master 'Operational' failed. DCM not in-sync`
Remedies	Correct wiring. Afterwards, reload the »PLC Designer« application into the automation system.

Cause	**The real structure at an EPM-S130 head end does not correspond to the control configuration configured.** This error can occur if I/O panels have been added manually to the control configuration.

Error message	`EtherCAT_Master: Set master 'Operational' failed. Timeout`
	
	`I_O_System_1000_EPM_S130 (1005): CoE emergency request. id=0x0, len=8, ErrCode=0xa000, ErrReg=0x2, data: 0xe 0x9 0x0 0x9 0x0`
	
	`I_O_System_1000_EPM_S130 (1005): Slaves signals Error. AL Status: 'PRE OPERATIONAL' (0x12), AL Status Code: 'Invalid Input Configuration'`

Remedies	Correct control configuration in the »PLC Designer«. **Note:** In "`data: ...`", a coded information says which error has occurred in which slave device/module. Detailed information on the coding of error messages can be found in the documentation of the corresponding slave device/module.

Cause	**Synchronisation settings in the standard device (slave) and the »PLC Designer« configuration are inconsistent.** Example : • »Engineer«: Sync source C01120 = MXI1 • »PLC Designer«: Distributed clocks deactivated ("DC = unused")

Error message	`EtherCAT_Master: Set master 'Operational' failed (0x4000005)`
	
	`[DeviceName] (1001): Slaves signals Error. AL Status: 'SAFE OPERATIONAL' (0x14), AL Status Code: 'Synchronization error' (0x1A)`

Remedies	Adapt the synchronisation settings in the slave.

Cause	**The first slave after the master has not been defined as DC master.**
Error message	`EtherCAT_Master: DC slaves 'out-of-sync'. Deviation xxxxxxxx ns`
Remedies	Declare the first slave after the master as DC master ("DC for Synchronization").

13.5.3 Messages: WKC Error / Not all slaves "Operational" / SyncManager Watchdog

In the "Operational" state, the process data are exchanged cyclically.

If a slave does not accept the cyclical frame (WKC is not increased), this error is caused.

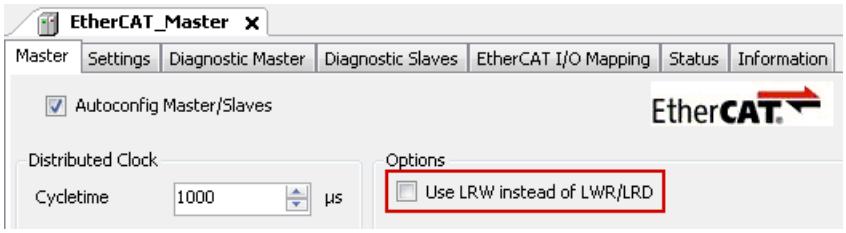
Cause	• The bus cable between two EtherCAT nodes has been unplugged. • The node at position X is deenergised. • A slave no longer receives cyclical frames so that the watchdog determined by the device description is triggered. This message will only be transmitted when the connection to the master has been re-established.

Error message	`EtherCAT_Master: Not all slaves 'Operational' (repeated 100 times)`
	`EtherCAT_Master: Cyclical command WKC error (repeated 100 times)`
	`EtherCAT_Master: Not all slaves 'Operational' (repeated 10 times)`
	`[DeviceName] (1002): Communication to device interrupted`
	`EtherCAT_Master: Cyclical command WKC error (repeated 10 times)`
	`EtherCAT_Master: Not all slaves 'Operational' (repeated 1 times)`
	`EtherCAT_Master: Cyclical command WKC error (repeated 1 times)`

Remedies	Correct bus topology and Restarting the EtherCAT fieldbus (📖 179).

13.5.4 Message: Invalid SyncManager Configuration

When the state is changed from "Pre-operational" to "Safe-operational", a slave reports "Invalid SyncManager Configuration".

Cause	• One of the slaves does not support an LRW command (Logical Read/Write). • A slave is not written to correctly.
Error message	`[DeviceName] (1001): Invalid SyncManager Configuration`
Remedies	In the EtherCAT master tab, do not select the "Use LRW instead of LWR/LRD" checkbox. 

13.5.5 Messages: Invalid Input Configuration / Invalid Output Configuration

When the status changes from "Pre-Operational" to "Safe-Operational", a slave reports "Invalid Input Configuration" or "Invalid Output Configuration".

Cause	The process data configuration of a slave is not correct. • In case of a modular device such as the I/O system 1000 (EPM-Sxxx), the configuration in the project does not comply with the real assembly. • More process data than permissible are mapped for the device.
Error message	• `[DeviceName] (1001): Slave signals Error. AL state: 'PRE OPERATIONAL' (0x12), AL state code: 'Invalid Input Configuration' (0x1E)` • `[DeviceName] (1001): Slave signals Error. AL state: 'PRE OPERATIONAL' (0x12), AL state code: 'Invalid Output Configuration' (0x1D)`
Remedies	• In case of modular devices such as the I/O system 1000 (EPM-Sxxx): Correct the control configuration in the »PLC Designer« (adjustment with the real setup). • Reduction of the process data: The maximum process data length must not be exceeded (see also the documentation of the device).

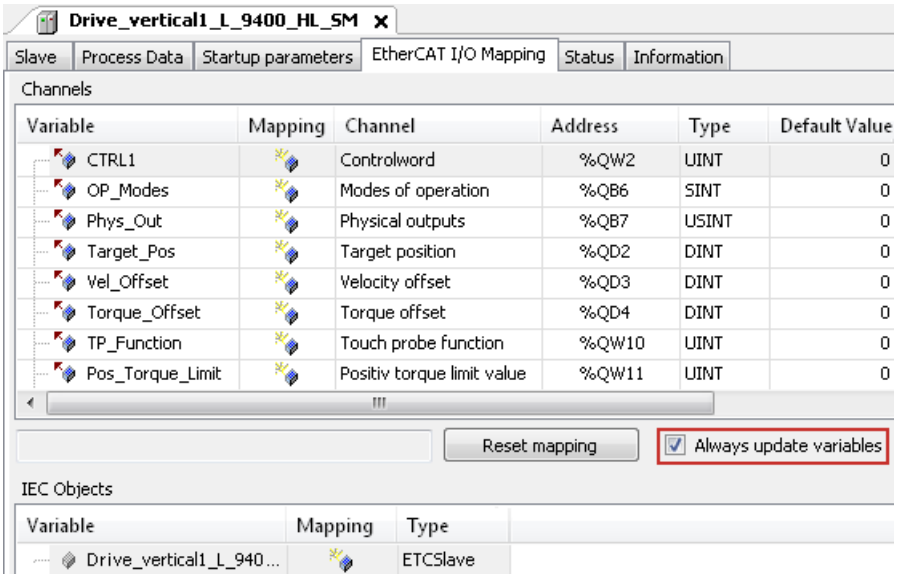
13.5.6 Error during process data transfer

A faulty EtherCAT I/O mapping causes errors during the process data transfer.

Cause	**Use of logic addresses** In the »PLC Designer« application, access does not take place symbolically but directly via the I/O addresses (%Ixx, %Qxx) of the EtherCAT input and output objects and the bus structure, the PDO selection, etc, have changed.
Error message	-
Remedies	In the »PLC Designer« application, the input and output objects must be accessed via individual, **non-ambiguous** variables. The variable names must comply with the IEC 61131 syntax (no space characters and leading digits in the variable names). Example : Drive_vertical1_L_9400_HL_SM Slave | Process Data | Startup parameters | EtherCAT I/O Mapping | Status | Information Channels Variable | Mapping | Channel | Address | Type | Default Value CTRL | Controlword | %QW34 | UINT | 0 Modes_OP | Modes of operation | %QB70 | SINT | 0 Phys_Out | Physical outputs | %QB71 | USINT | 0 Target_Pos | Target position | %QD18 | DINT | 0 Vel_offs | Velocity offset | %QD19 | DINT | 0 Torque_offs | Torque offset | %QD20 | DINT | 0 TP_func | Touch probe function | %QW42 | UINT | 0 Pos_Torque_Limit | Positiv torque limit value | %QW43 | UINT | 0 ▸ Processing EtherCAT I/O mapping (📖 102)

Cause	**Manual definition of the logic address in the EtherCAT I/O mapping** Example: The address %QB70 has been changed to %QB1000. Drive_vertical1_L_9400_HL_SM Slave | Process Data | Startup parameters | EtherCAT I/O Mapping | Status | Information Channels Variable | Mapping | Channel | Address | Type | Default Value CTRL | Controlword | %QW34 | UINT | 0 Modes_OP | Modes of operation | %QB1000 | SINT | 0 Phys_Out | Physical outputs | %QB1001 | USINT | 0 Target_Pos | Target position | %QD251 | DINT | 0 Vel_offs | Velocity offset | %QD252 | DINT | 0 Torque_offs | Torque offset | %QD253 | DINT | 0 TP_func | Touch probe function | %QW508 | UINT | 0 Pos_Torque_Limit | Positiv torque limit value | %QW509 | UINT | 0
Error message	-
Remedies	It is not permissible to manually manipulate the I/O addresses for the EtherCAT bus!

Cause	**Missing or incorrect I/O mapping** In the case of Servo Drives 9400 and Inverter Drives 8400, the ports in the »Engineer« are displayed incorrectly or not at all.
Error message	-
Remedies	Check and correct the mapping settings in the control configuration and in the inverter. When the Lenze Controller is started, the complete configuration/PDO mapping is written into the EtherCAT slaves. When this is done, mapping entries, e.g. from the »Engineer« are overwritten.

Cause	**Variables are not used in the »PLC Designer« application.** I/O variables that are not used in the »PLC Designer« application are not copied into the process image (and vice versa) so that they are not updated by the peripherals.
Error message	-
Remedies	With the slave, go to the **EtherCAT I/O Mapping** tab and set a checkmark at **Always update variables**: 

13.5.7 Messages: EtherCAT cable not connected / EtherCAT cable connected

Cause	**The bus cable between the Lenze Controller and the first node has been unplugged.** If a previously removed bus cable has been plugged into the first EtherCAT node, the message "`EtherCAT_Master: EtherCAT cable connected`" is entered in the logbook of the controller. The EtherCAT connection is re-established. Since the EtherCAT slave sync managers do not receive any messages, a time-out expires and the slaves change to the "Safe-Operational" state.

Error message	`EtherCAT_Master: EtherCAT cable not connected`
	`...`
	`EtherCAT_Master: EtherCAT cable connected`

Remedies	After the bus cable has been plugged in again, Restarting the EtherCAT fieldbus (📖 179).

13.5.8 Message: Frame Response Error

Cause	**A frame sent by the master does not return to the master until the next cycle.** • The task utilisation is too high so that a sent frame takes longer than the time to the next start of the bus cycle task. • The EtherCAT bus cycle task does not have the highest IEC task priority or another task has the same IEC task priority so that the EtherCAT bus cycle task is suppressed. • Due to an error, the slave does not forward any frames. • Only a switch or an ET2000 is connected to the Lenze Controller, but no further slave.

Error message	`EtherCAT_Master: Frame response error (repeated 1 time)`
	`EtherCAT_Master: Frame response error (repeated 10 times)`
	`EtherCAT_Master: Frame response error (repeated 100 times)`

Remedies	• Reduce the program code or increase the bus task cycle time. • Assign the sole and highest IEC task priority to the EtherCAT bus cycle task. • Correct the slave error. • Correct the bus structure.

13.5.9 Shafts make clicking noises

If the shafts make clicking noises, this is often caused by faulty synchronisation or a shift of data in the process image.

Cause	**The task and DC cycle times set in the logic/motion system differ.**
Error message	-
Remedies	Adjust the task cycle time and DC cycle time. ▸ Adjusting the task cycle time and DC cycle time (📖 95)

Cause	**Wiring error: The EtherCAT terminals (IN/OUT) of the slave were inverted.** A fieldbus scan does not indicate this error!
Error message	`EtherCAT_Master: Set master 'Operational' failed. DCM not in-sync`
Remedies	Correct wiring. Afterwards, reload the »PLC Designer« application into the automation system.

Cause	**Clicking noise of the shafts after "out-of-sync"** If due to a fault, the preset DC deviation limit is exceeded, a re-synchronisation of the DC slaves is carried out until the slaves are synchronised again ("In-Sync") and the DC deviation is under the preset limit value again. Currently, the Lenze Controller is not re-synchronised to the distributed clocks so that the sync pulses of the master and the ones of the slaves are different.
Error message	`EtherCAT_Master: DC slaves 'out-of-sync'. Deviation xxxxxxxx ns`
Remedies	Restarting the EtherCAT fieldbus (📖 179) so that the DC slaves and the DC master synchronise again.

Cause	**Wrong selection of the device sync source (9400 HighLine CiA402, C01120)** After the sync source has been changed by **C01120**, the subsequent download and the setting of the slave to the "Operational" state may fail
Error message	-
Remedies	• Manual setting of code C01120 to MXI1 or MXI2 • Repeated download with PLC start • Restarting the EtherCAT fieldbus (📖 179) with reset of the SoftMotion drive

13.5.10 Shafts do not rotate

Cause	**The EtherCAT bus could not be set to the "Operational" state** ▸ The EtherCAT bus does not assume the "Operational" state (📖 197)
Error message	–
Remedies	▸ The EtherCAT bus does not assume the "Operational" state (📖 197)

Cause	**Clicking noise of the shafts after "out-of-sync"** ▸ Error during process data transfer (📖 200)
Error message	–
Remedies	▸ Error during process data transfer (📖 200)

Cause	**Faulty SoftMotion scaling/mapping** With SoftMotion scaling/mapping, the increments per revolution are not set.
Error message	–
Remedies	Check the following settings and correct them if required: • Gearbox ratio in the »PLC Designer« application • Mapping settings in the master configuration When the Lenze Controller is started, the complete configuration/PDO mapping is written into the EtherCAT slaves. When this is done, mapping entries, e.g. from the »Engineer«, are overwritten. **Tip:** In the case of the Servo Drive 9400 HighLine Cia402, 65536 increments per revolution are correct.

13.6 System error messages

In the case of system error messages, the following types of error are distinguished:

- General error codes (L_ETC_ERRORCODE) (📖 205)
- Error messages for modular machine configuration (📖 212)
- Lenze Controller logbook messages (📖 214)
- SDO abort codes (📖 219)

13.6.1 General error codes (L_ETC_ERRORCODE)

General error codes are provided at the *eErrorCode* output of the EtherCAT function blocks (see L_IODrvEtherCAT function library (📖 126)).

If required, these error messages are also output as additional error information in the Logbook of the Lenze Controller in the »WebConfig« (📖 191).

Error no. [hex]	Name	Description
0x00000000	ETC_E_NOERROR	No error / function completed successfully.
0x02000000	ETC_TEXTBASE	Unknown (basis) text **Internal error. Please contact Lenze.**
0x03000000	ETC_ALSTATEBASE	AL status: No error **Internal error. Please contact Lenze.**
0x98110000	ETC_E_ERROR	Unspecified error
0x98110180	ETC_EMRAS_E_ERROR	Unspecified RAS error **Internal error. Please contact Lenze.**
0x981201C0	ETC_DCM_E_ERROR	Unspecified DCM error **Internal error. Please contact Lenze.**
0x98110001	ETC_E_NOTSUPPORTED	Function or feature not available.
0x98110002	ETC_E_INVALIDINDEX	CoE: invalid SDO index
0x98110003	ETC_E_INVALIDOFFSET	Invalid offset value during access to process data image **Internal error. Please contact Lenze.**
0x98110004	ETC_E_CANCEL	The master should cancel the current request, e.g. mailbox transmission (CoE, FoE, EoE etc.).
0x98110005	ETC_E_INVALIDSIZE	Invalid offset value ... during access to process data image during storage of data in a data area
0x98110006	ETC_E_INVALIDDATA	Invalid data
0x98110007	ETC_E_NOTREADY	Internal software error (numerous possible causes) **Internal error. Please contact Lenze.**
0x98110008	ETC_E_BUSY	The master is busy at the moment and cannot process the API function. The function should be repeated at a later time.
0x98110009	ETC_E_ACYC_FRM_FREEQ_EMPTY	The queue/the memory for acyclic commands is full. **Internal error. Please contact Lenze.**
0x9811000A	ETC_E_NOMEMORY	Not enough application memory available. **Internal error. Please contact Lenze.**
0x9811000B	ETC_E_INVALIDPARM	An API function has been called with incorrect parameters.
0x9811000C	ETC_E_NOTFOUND	An API function has been called with an invalid slave ID.
0x9811000E	ETC_E_INVALIDSTATE	Invalid state
0x9811000F	ETC_E_TIMER_LIST_FULL	Not enough application memory available. **Internal error. Please contact Lenze.**
0x98110010	ETC_E_TIMEOUT	A time-out is active.
0x98110011	ETC_E_OPENFAILED	Internal software error (numerous possible causes) **Internal error. Please contact Lenze.**
0x98110012	ETC_E_SENDFAILED	The transmission of the frame has failed. **Internal error. Please contact Lenze.**
0x98110013	ETC_E_INSERTMAILBOX	The mailbox command cannot be enqueued on the internal queue. **Internal error. Please contact Lenze.**

Error no. [hex]	Name	Description
0x98110014	ETC_E_INVALIDCMD	Unknown mailbox command code **Internal error. Please contact Lenze.**
0x98110015	ETC_E_UNKNOWN_MBX_PROTOCOL	Unknown mailbox protocol Mailbox command ID with unknown protocol assignment **Internal error. Please contact Lenze.**
0x98110016	ETC_E_ACCESSDENIED	Access denied (internal software error on the master) **Internal error. Please contact Lenze.**
0x9811001A	ETC_E_PRODKEY_INVALID	The evaluation version of the master is used. The master enters the "Stop" mode after 30 minutes. **Internal error. Please contact Lenze.**
0x9811001B	ETC_E_WRONG_FORMAT	The XML file contains no or faulty content. **Internal error. Please contact Lenze.**
0x9811001C	ETC_E_FEATURE_DISABLED	Attempt to execute a non-existing or deactivated function. **Internal error. Please contact Lenze.**
0x9811001D	ETC_E_SHADOW_MEMORY	The shadow memory has been requested in wrong mode. **Internal error. Please contact Lenze.**
0x9811001E	ETC_E_BUSCONFIG_MISMATCH	The EtherCAT configuration of the master and the connected slaves does not comply with the physical bus structure.
0x9811001F	ETC_E_CONFIGDATAREAD	The XML file cannot be read. **Internal error. Please contact Lenze.**
0x98110021	ETC_E_XML_CYCCMDS_MISSING	The XML file of the master does not contain any cyclical commands. **Internal error. Please contact Lenze.**
0x98110022	ETC_E_XML_ALSTATUS_READ_MISSING	The XML file of the master does not contain the command for reading the AL Status Register. **Internal error. Please contact Lenze.**
0x98110023	ETC_E_MCSM_FATAL_ERROR	The master state machine is in an invalid state. **Internal error. Please contact Lenze.**
0x98110024	ETC_E_SLAVE_ERROR	The slave cannot be addressed.
0x98110025	ETC_E_FRAME_LOST	An EtherCAT frame was lost on the fieldbus, i.e. it has not been received. If this error occurred frequently, this indicates the wiring may be faulty.
0x98110026	ETC_E_CMD_MISSING	The received EtherCAT frame is not complete. **Internal error. Please contact Lenze.**
0x98110028	ETC_E_INVALID_DCL_MODE	This function cannot be used when DC latching is in the "Auto Read" operating mode. **Internal error. Please contact Lenze.**
0x98110029	ETC_E_AI_ADDRESS	The connected slaves do not comply with the control configuration. This error only occurs if a previously existing slave disappears. **Internal error. Please contact Lenze.**
0x9811002A	ETC_E_INVALID_SLAVE_STATE	The mailbox commands are not permissible in the current slave state.
0x9811002B	ETC_E_SLAVE_NOT_ADDRESSABLE	The slave has been switched on/off.
0x9811002C	ETC_E_CYC_CMDS_OVERFLOW	Error during XML file creation by the configurator **Internal error. Please contact Lenze.**
0x9811002D	ETC_E_LINK_DISCONNECTED	The EtherCAT cable has not be connected/plugged into the Lenze controller.
0x9811002E	ETC_E_MASTERCORE_INACCESSIBLE	The connection to the master (server) is interrupted or the master has been stopped. **Internal error. Please contact Lenze.**
0x9811002F	ETC_E_COE_MBXSND_WKC_ERROR	It is not possible to write to the CoE mailbox in the slave. The slave has not yet read the mailbox. **Internal error. Please contact Lenze.**
0x98110030	ETC_E_COE_MBXRCV_WKC_ERROR	It is not possible to read the CoE mailbox in the slave. **Internal error. Please contact Lenze.**
0x98110031	ETC_E_NO_MBX_SUPPORT	The slave does not support mailbox transfer.
0x98110032	ETC_E_NO_COE_SUPPORT	Configurator error or the slave description file does not correspond with the slave firmware.
0x98110033	ETC_E_NO_EOE_SUPPORT	Configurator error or the slave description file does not correspond with the slave firmware.
0x98110034	ETC_E_NO_FOE_SUPPORT	Configurator error or the slave description file does not correspond with the slave firmware.
0x98110035	ETC_E_NO_SOE_SUPPORT	Configurator error or the slave description file does not correspond with the slave firmware. Is not supported.
0x98110036	ETC_E_NO_VOE_SUPPORT	Configurator error or the slave description file does not correspond with the slave firmware. Is not supported.

Error no. [hex]	Name	Description
0x98110037	ETC_E_EVAL_VIOLATION	The number of slaves indicated in the XML file is too large for the evaluation version of the master. **Internal error. Please contact Lenze.**
0x98110038	ETC_E_EVAL_EXPIRED	The evaluation time has expired. The fieldbus is stopped. **Internal error. Please contact Lenze.**
0x98110040	ETC_E_SDO_ABORTCODE_TOGGLE	The status of the toggle bit has not changed. Abort code 0x05030000
0x98110041	ETC_E_SDO_ABORTCODE_TIMEOUT	SDO protocol time-out Abort code 0x05040000
0x98110042	ETC_E_SDO_ABORTCODE_CCS_SCS	Invalid or unknown specification symbol for the client/server command Abort code 0x05040001
0x98110043	ETC_E_SDO_ABORTCODE_BLK_SIZE	Invalid block size (only in "Block mode") Abort code 0x05040002
0x98110044	ETC_E_SDO_ABORTCODE_SEQNO	Invalid sequence number (only in "Block mode") Abort code 0x05040003
0x98110045	ETC_E_SDO_ABORTCODE_CRC	CRC error (only in "Block mode") Abort code 0x05040004
0x98110046	ETC_E_SDO_ABORTCODE_MEMORY	The space in the main memory is not sufficient. Abort code 0x05040005
0x98110047	ETC_E_SDO_ABORTCODE_ACCESS	Access to object not supported Abort code 0x06010000
0x98110048	ETC_E_SDO_ABORTCODE_WRITEONLY	Read access to a write-protected object Abort code 0x06010001
0x98110049	ETC_E_SDO_ABORTCODE_READONLY	Write access to a write-protected object Abort code 0x06010002
0x9811004A	ETC_E_SDO_ABORTCODE_INDEX	An object does not exist in the object directory Abort code 0x06020000
0x9811004B	ETC_E_SDO_ABORTCODE_PDO_MAP	An object cannot be mapped into the PDO Abort code 0x06040041
0x9811004C	ETC_E_SDO_ABORTCODE_PDO_LEN	The number and/or length of the objects mapped would exceed the PDO length Abort code 0x06040042
0x9811004D	ETC_E_SDO_ABORTCODE_P_INCOMP	General parameter incompatibility Abort code 0x06040043
0x9811004E	ETC_E_SDO_ABORTCODE_I_INCOMP	General internal device incompatibility Abort code 0x06040047
0x9811004F	ETC_E_SDO_ABORTCODE_HARDWARE	Access has failed due to a fault in the hardware Abort code 0x06060000
0x98110050	ETC_E_SDO_ABORTCODE_DATA_SIZE	The data type or the parameter length does not correspond Abort code 0x06070010
0x98110051	ETC_E_SDO_ABORTCODE_DATA_SIZE1	Incorrect data type (The parameter length is too large) Abort code 0x06070012
0x98110052	ETC_E_SDO_ABORTCODE_DATA_SIZE2	Wrong data type (parameter length is too small). Abort code 0x06070013
0x98110053	ETC_E_SDO_ABORTCODE_OFFSET	A subindex is not available Abort code 0x06090011
0x98110054	ETC_E_SDO_ABORTCODE_DATA_RANGE	The value range for parameters is too great (only for write access) Abort code 0x06090030
0x98110055	ETC_E_SDO_ABORTCODE_DATA_RANGE1	The parameter value is too high Abort code 0x06090031
0x98110056	ETC_E_SDO_ABORTCODE_DATA_RANGE2	The parameter value is too low Abort code 0x06090032
0x98110057	ETC_E_SDO_ABORTCODE_MINMAX	The maximum value is lower than the minimum value Abort code 0x06090036
0x98110058	ETC_E_SDO_ABORTCODE_GENERAL	General error Abort code 0x08000000
0x98110059	ETC_E_SDO_ABORTCODE_TRANSFER	Data cannot be transferred/saved to the application. Abort code 0x08000020
0x9811005A	ETC_E_SDO_ABORTCODE_TRANSFER1	Data cannot be transferred/saved to the application because of local control. Abort code 0x08000021
0x9811005B	ETC_E_SDO_ABORTCODE_TRANSFER2	Due to the current device state, data cannot be transferred to the application or stored in the application Abort code 0x08000022

Error no. [hex]	Name	Description
0x9811005C	ETC_E_SDO_ABORTCODE_DICTIONARY	The dynamic generation of an object directory has failed, or no object directory is available. Abort code 0x08000023
0x9811005D	ETC_E_SDO_ABORTCODE_UNKNOWN	Unknown internal slave error
0x98110060	ETC_E_FOE_ERRCODE_NOTDEFINED	Manufacturer-specific FoE error
0x98110061	ETC_E_FOE_ERRCODE_NOTFOUND	Not found
0x98110062	ETC_E_FOE_ERRCODE_ACCESS	Access denied
0x98110063	ETC_E_FOE_ERRCODE_DISKFULL	Memory (floppy disk/hard disk) is full.
0x98110064	ETC_E_FOE_ERRCODE_ILLEGAL	Invalid/impermissible
0x98110065	ETC_E_FOE_ERRCODE_PACKENO	Wrong package number
0x98110066	ETC_E_FOE_ERRCODE_EXISTS	Already available
0x98110067	ETC_E_FOE_ERRCODE_NOUSER	User/consumer is missing.
0x98110068	ETC_E_FOE_ERRCODE_BOOTSTRAPONLY	Only bootstrap state
0x98110069	ETC_E_FOE_ERRCODE_NOTINBOOTSTRAP	No bootstrap state
0x9811006A	ETC_E_FOE_ERRCODE_INVALIDPASSWORD	No required access authorisation
0x9811006B	ETC_E_FOE_ERRCODE_PROGERROR	Program error
0x98110070	ETC_E_CFGFILENOTFOUND	The master configuration has not been found.
0x98110071	ETC_E_EEPROMREADERROR	Command error during EEPROM upload
0x98110072	ETC_E_EEPROMWRITEERROR	Command error during EEPROM download
0x98110073	ETC_E_XML_CYCCMDS_SIZEMISMATCH	The cyclical command has a wrong size or is too long. **Internal error. Please contact Lenze.**
0x98110074	ETC_E_XML_INVALID_INP_OFF	Invalid input offset in cyclical command **Internal error. Please contact Lenze.**
0x98110075	ETC_E_XML_INVALID_OUT_OFF	Invalid output offset in cyclical command **Internal error. Please contact Lenze.**
0x98110076	ETC_E_PORTCLOSE	Closing of the port failed. **Internal error. Please contact Lenze.**
0x98110077	ETC_E_PORTOPEN	Opening of the port failed. **Internal error. Please contact Lenze.**
0x98110078	ETC_E_SOE_ERRORCODE_INVALID_ACCESS	Impermissible access to element 0 Is not supported.
0x98110079	ETC_E_SOE_ERRORCODE_NOT_EXIST	Nonexistent Is not supported.
0x9811007a	ETC_E_SOE_ERRORCODE_INVL_ACC_ELEM1	Impermissible access to element 1 Is not supported.
0x9811007b	ETC_E_SOE_ERRORCODE_NAME_NOT_EXIST	The name is not available. Is not supported.
0x9811007c	ETC_E_SOE_ERRORCODE_NAME_UNDERSIZE	The name is too short for transmission. Is not supported.
0x9811007d	ETC_E_SOE_ERRORCODE_NAME_OVERSIZE	The name is too short for transmission. Is not supported.
0x9811007e	ETC_E_SOE_ERRORCODE_NAME_UNCHANGE	The name cannot be changed. Is not supported.
0x9811007f	ETC_E_SOE_ERRORCODE_NAME_WR_PROT	The name is currently write-protected. Is not supported.
0x98110080	ETC_E_SOE_ERRORCODE_UNDERS_TRANS	The attribute is too small for transmission. Is not supported.
0x98110081	ETC_E_SOE_ERRORCODE_OVERS_TRANS	The attribute is too big for transmission. Is not supported.
0x98110082	ETC_E_SOE_ERRORCODE_ATTR_UNCHANGE	The attribute cannot be changed. Is not supported.
0x98110083	ETC_E_SOE_ERRORCODE_ATTR_WR_PROT	The attribute is currently write-protected. Is not supported.
0x98110084	ETC_E_SOE_ERRORCODE_UNIT_NOT_EXIST	The unit is not available. Is not supported.
0x98110085	ETC_E_SOE_ERRORCODE_UNIT_UNDERSIZE	The unit is too small for transmission. Is not supported.
0x98110086	ETC_E_SOE_ERRORCODE_UNIT_OVERSIZE	The unit is too big for transmission. Is not supported.

Error no. [hex]	Name	Description
0x98110087	ETC_E_SOE_ERRORCODE_UNIT_UNCHANGE	The unit cannot be changed. Is not supported.
0x98110088	ETC_E_SOE_ERRORCODE_UNIT_WR_PROT	The unit is currently write-protected. Is not supported.
0x98110089	ETC_E_SOE_ERRORCODE_MIN_NOT_EXIST	The minimum input value is not available. Is not supported.
0x9811008a	ETC_E_SOE_ERRORCODE_MIN_UNDERSIZE	The minimum input value is too small for transmission. Is not supported.
0x9811008b	ETC_E_SOE_ERRORCODE_MIN_OVERSIZE	The minimum input value is too big for transmission. Is not supported.
0x9811008c	ETC_E_SOE_ERRORCODE_MIN_UNCHANGE	The minimum input value cannot be changed. Is not supported.
0x9811008d	ETC_E_SOE_ERRORCODE_MIN_WR_PROT	The minimum input value is currently write-protected. Is not supported.
0x9811008e	ETC_E_SOE_ERRORCODE_MAX_NOT_EXIST	The maximum input value is not available. Is not supported.
0x9811008f	ETC_E_SOE_ERRORCODE_MAX_UNDERSIZE	The maximum input value is too small for transmission. Is not supported.
0x98110090	ETC_E_SOE_ERRORCODE_MAX_OVERSIZE	The maximum input value is too big for transmission. Is not supported.
0x98110091	ETC_E_SOE_ERRORCODE_MAX_UNCHANGE	The maximum input value cannot be changed. Is not supported.
0x98110092	ETC_E_SOE_ERRORCODE_MAX_WR_PROT	The maximum input value is currently write-protected. Is not supported.
0x98110093	ETC_E_SOE_ERRORCODE_DATA_NOT_EXIST	The data element is not available. Is not supported.
0x98110094	ETC_E_SOE_ERRORCODE_DATA_UNDERSIZE	The data element is too small for transmission. Is not supported.
0x98110095	ETC_E_SOE_ERRORCODE_DATA_OVERSIZE	The data element is too big for transmission. Is not supported.
0x98110096	ETC_E_SOE_ERRORCODE_DATA_UNCHANGE	The data element cannot be changed. Is not supported.
0x98110097	ETC_E_SOE_ERRORCODE_DATA_WR_PROT	The data element is currently write-protected. Is not supported.
0x98110098	ETC_E_SOE_ERRORCODE_DATA_MIN_LIMIT	The data element is smaller than the minimum input value limit. Is not supported.
0x98110099	ETC_E_SOE_ERRORCODE_DATA_MAX_LIMIT	The data element exceeds the maximum input value limit. Is not supported.
0x9811009a	ETC_E_SOE_ERRORCODE_DATA_INCOR	The data element is not correct. Is not supported.
0x9811009b	ETC_E_SOE_ERRORCODE_PASWD_PROT	The data element is protected by a password. Is not supported.
0x9811009c	ETC_E_SOE_ERRORCODE_TEMP_UNCHANGE	The data element can currently not be changed (in AT or MDT). Is not supported.
0x9811009d	ETC_E_SOE_ERRORCODE_INVL_INDIRECT	Invalid/indirect Is not supported.
0x9811009e	ETC_E_SOE_ERRORCODE_TEMP_UNCHANGE1	The data element can currently not be changed (parameter or OP mode). Is not supported.
0x9811009f	ETC_E_SOE_ERRORCODE_ALREADY_ACTIVE	The command is already active. Is not supported.
0x98110100	ETC_E_SOE_ERRORCODE_NOT_INTERRUPT	The command can be interrupted. Is not supported.
0x98110101	ETC_E_SOE_ERRORCODE_CMD_NOT_AVAIL	The command is not available in this phase. Is not supported.
0x98110102	ETC_E_SOE_ERRORCODE_CMD_NOT_AVAIL1	The command is not available (invalid parameter) Is not supported.
0x98110103	ETC_E_SOE_ERRORCODE_DRIVE_NO	The received drive number does not comply with the requested drive number. Is not supported.
0x98110104	ETC_E_SOE_ERRORCODE_IDN	The received ID does not comply with the requested ID. Is not supported.
0x98110105	ETC_E_SOE_ERRORCODE_FRAGMENT_LOST	At least on fragment got lost. Is not supported.

Error no. [hex]	Name	Description
0x98110106	ETC_E_SOE_ERRORCODE_BUFFER_FULL	The Rx memory buffer is full (EtherCAT call with too little data buffer). Is not supported.
0x98110107	ETC_E_SOE_ERRORCODE_NO_DATA	No data state Is not supported.
0x98110108	ETC_E_SOE_ERRORCODE_NO_DEFAULT_VALUE	No standard value Is not supported.
0x98110109	ETC_E_SOE_ERRORCODE_DEFAULT_LONG	The standard value transmission is too long. Is not supported.
0x9811010a	ETC_E_SOE_ERRORCODE_DEFAULT_WP	The standard value cannot be changed (read only). Is not supported.
0x9811010b	ETC_E_SOE_ERRORCODE_INVL_DRIVE_NO	Invalid drive number Is not supported.
0x9811010c	ETC_E_SOE_ERRORCODE_GENERAL_ERROR	General error Is not supported.
0x9811010d	ETC_E_SOE_ERRCODE_NO_ELEM_ADR	No element has been addressed. Is not supported.
0x9811010e	ETC_E_SLAVE_NOT_PRESENT	The slave is not available at the fieldbus.
0x9811010f	ETC_E_NO_FOE_SUPPORT_BS	The FoE protocol is not supported in the bootstrap state.
0x98110110	ETC_E_EEPROMRELOADERROR	Command error during EEPROM reload
0x98110111	ETC_E_SLAVECTRLRESETERROR	Command error during slave controller reset
0x98110112	ETC_E_SYSDRIVERMISSING	The system driver ect.sys could not be opened. **Internal error. Please contact Lenze.**
0x9811011E	ETC_E_BUSCONFIG_TOPOCHANGE	The fieldbus configuration could not be detected. The bus topology has been changed.
0x9811011F	ETC_E_EOE_MBX_WKC_ERROR	Error at EoE mailbox reception: Working counter
0x98110120	ETC_E_FOE_MBX_WKC_ERROR	Error at FoE mailbox reception: Working counter
0x98110121	ETC_E_SOE_MBX_WKC_ERROR	Error at SoE mailbox reception: Working counter Is not supported.
0x98110122	ETC_E_AOE_MBX_WKC_ERROR	Error at AoE mailbox reception: Working counter Is not supported.
0x98110123	ETC_E_VOE_MBX_WKC_ERROR	Error at VoE mailbox reception: Working counter Is not supported.
0x98110124	ETC_E_EEPROMASSIGNERROR	The EEPROM assignment has failed. **Internal error. Please contact Lenze.**
0x98110125	ETC_E_MBX_ERROR_TYPE	Error at mailbox reception **Internal error. Please contact Lenze.**
0x981201C1	ETC_DCM_E_NOTINITIALIZED	The initialisation has not been successful. The initialisation function has not been called. **Internal error. Please contact Lenze.**
0x981201C2	ETC_DCM_E_MAX_CTL_ERROR_EXCEED	Controller error: Synchronisation beyond the limits **Internal error. Please contact Lenze.**
0x981201C3	ETC_DCM_E_NOMEMORY	Not enough memory location available. **Internal error. Please contact Lenze.**
0x981201C4	ETC_DCM_E_INVALID_HWLAYER	Hardware error: Invalid (BSP) **Internal error. Please contact Lenze.**
0x981201C5	ETC_DCM_E_TIMER_MODIFY_ERROR	Hardware error: Error at timer change **Internal error. Please contact Lenze.**
0x981201C6	ETC_DCM_E_TIMER_NOT_RUNNING	Hardware error: The timer does not run. **Internal error. Please contact Lenze.**
0x981201C7	ETC_DCM_E_WRONG_CPU	Hardware error: The function has been called on the wrong CPU. **Internal error. Please contact Lenze.**
0x981201C8	ETC_DCM_E_INVALID_SYNC_ PERIOD	Invalid DC-Sync. Period length (invalid DC-master?) **Internal error. Please contact Lenze.**
0x981201C9	ETC_DCM_E_INVALID_SETVAL	DCM controller error: The set value is too low. **Internal error. Please contact Lenze.**
0x981201CA	ETC_DCM_E_DRIFT_TO_HIGH	DCM controller error: The deviation between the local timer and the reference clock is too high. **Internal error. Please contact Lenze.**
0x98110181	ETC_EMRAS_E_INVALIDCOOKIE	Reconnecting using the old cookie has failed. A new reconnection attempt is made automatically. **Internal error. Please contact Lenze.**

Error no. [hex]	Name	Description
0x98110183	ETC_EMRAS_E_MULSRVDISMULCON	Attempt to connect to another remote server has been rejected because the multi-instance API has not been used for establishing an already existing connection. **Internal error. Please contact Lenze.**
0x98110184	ETC_EMRAS_E_LOGONCANCELLED	Server aborts connection during client logon. **Internal error. Please contact Lenze.**
0x98110186	ETC_EMRAS_E_INVALIDVERSION	Server and client version are not identical (different protocol versions). Therefore, connecting has been rejected. **Internal error. Please contact Lenze.**
0x98110187	ETC_EMRAS_E_INVALIDACCESSCONFIG	Access configuration is invalid **Internal error. Please contact Lenze.**
0x98110188	ETC_EMRAS_E_ACCESSLESS	No access to call on access level **Internal error. Please contact Lenze.**
0x98110191	ETC_EMRAS_EVT_SERVERSTOPPED	Detailed description for connection abort/termination if connection to server has been closed due to "API call (local)". **Internal error. Please contact Lenze.**
0x98110192	ETC_EMRAS_EVT_WDEXPIRED	Detailed description for connection abort/termination if connection has been closed due to missing keep-alive messages. **Internal error. Please contact Lenze.**
0x98110193	ETC_EMRAS_EVT_RECONEXPIRED	Client attempts to reopen an old connection (after the connection has been aborted), but the server has already cleared the session. A new connection must be established (register client and mailbox objects must be created again). **Internal error. Please contact Lenze.**
0x98110194	ETC_EMRAS_EVT_CLIENTLOGON	Server message when a new client has connected. **Internal error. Please contact Lenze.**
0x98110195	ETC_EMRAS_EVT_RECONNECT	Server message when a client has successfully reopened an old connection. **Internal error. Please contact Lenze.**
0x98110196	ETC_EMRAS_EVT_SOCKCHANGE	Detailed description (event) which marks the successful socket transfer of a new connection to an already existing session object (reconnect). **Internal error. Please contact Lenze.**
0x98110197	ETC_EMRAS_EVT_CLNTDISC	Client disconnected/switched off **Internal error. Please contact Lenze.**

13.6.2 Error messages for modular machine configuration

The error messages are arranged in alphabetical order (A - Z).

Error message	Description
MMC - address assignment - done	INFO: The address assignment has been completed successfully.
MMC - address assignment - invalid device on position ... (..._.../..._...)	ERROR: During the address assignment by means of the ADDR_ASSIGNMENT_EXTERNALLY or ADDR_ASSIGNMENT_CONFIG_SLAVEORDER service, a slave has been detected unexpectedly at the given position.
MMC - address assignment - less slaves connected (...) as configured (...)	ERROR: In the address assignment, less slaves are connected to the physical EtherCAT bus than given in the active configuration.
MMC - address assignment - more slaves connected as configured	ERROR: In the address assignment, more slaves are connected to the physical etherCAT bus than given in the active configuration.
MMC - address assignment - writing address ... at position ... by CoE ... (error ...)	ERROR: During the address assignment, an error has occurred for the slave at the given position. For further information see General error codes (L_ETC_ERRORCODE).
MMC - address assignment - writing address ... at position ... failed (error ...)	ERROR: During the address assignment, an error has occurred for the slave at the given position. For further information see General error codes (L_ETC_ERRORCODE).
MMC - address assignment - written address ... at position ... successfully	INFO: The address has been successfully assigned to the slave at the given position.
MMC - devices not ascending or device(s) missing at line ... (...)	ERROR: In the **mmc-0-conf.csv** configuration file, the slaves are not given in ascending order (starting with '1001') or are missing. Or the address space is incomplete.
MMC - duplicated Alias Address ... at positions ... and ...	ERROR: While the EtherCAT bus is booted and the slaves are checked, several slaves have been found with identical "Second Station Address" (alias address) at the given position. The first slave behind the EtherCAT master has the position '1'.
MMC - Error in configuration files	ERROR: During the analysis of the **mmc-0-conf.csv** configuration file or the **mmc-0-ident.csv** identification file, errors have been detected. Further information about the error is displayed in the logbook above.
MMC - file does not exist ...	INFO: The **mmc-0-conf.csv** configuration file or **mmc-0-ident.csv** identification file have not been found in the directory `.../USBStorage/IPC/PLC` or `.../SDCard/IPC/PLC`. As both files do not have to be available in the system (depending on the service), this is only an information.
MMC - Internal Error (...)	ERROR: An internal error has occurred. The internal error number is output in the error message. Please contact Lenze!
MMC - invalid Alias Address ... at position ... (.../.../.../...)	ERROR: While the EtherCAT bus is booted and the slaves are checked, a slave with invalid or unexpected "Second Station Address" (alias address) has been detected at the given position. The first slave behind the EtherCAT master has the position '1'. Parameters in bracket: Vendor-ID/Product code/Revision/Serial number.
MMC - invalid configuration	ERROR: An action has been aborted because no valid configuration is active.
MMC - mandatory slave ... is not present	ERROR: While the EtherCAT bus and the slaves are checked, a mandatory slave has not been found at the bus. In the error message, the EtherCAT address or "Second Station Address" (alias address) of the slave is given.
MMC - 'Modular Machine Configuration' is active - EtherCAT Master is controlled by L_ETC_MMCController	INFO: The L_ETC_MMCController function block is used in the PLC program. The behaviour of the EtherCAT master is controlled by the function block.
MMC - no configuration checks	INFO: While the EtherCAT bus is booted, the configuration is not checked because the RUN_WITHOUT_CHECK service is active.
MMC - no valid service active	ERROR: When the L_ETC_MMCController function block was used, it was tried to boot the EtherCAT bus (xRestart = TRUE). But no service is active.
MMC - number of device in device tree differs at line ... (...)	Warning: The number of slaves from the **mmc-0-conf.csv** configuration file is higher than defined in the »PLC Designer« project. This is a warning because the excessive devices from the configuration are simply ignored.

Error message	Description
MMC - optional slave ... is present, but not allowed	ERROR: While the EtherCAT bus is booted and the RUN_OPTIONAL_SLAVES_PROHIBITED service is active, a "non-mandatory" slave has been found.
MMC - parsing error at line ... (...)	ERROR: During the analysis of the **mmc-0-conf.csv** configuration file or **mmc-0-ident.csv** identification file, errors in the given line numbers have been detected. The line number starts with '1'.
MMC - parsing file succeeded ...	INFO: The given **mmc-0-conf.csv** configuration file or **mmc-0-ident.csv** identification file have been analysed without any errors.
MMC - service ... started, configuration '...'	INFO: A service has been started with the given configuration.
MMC - service stopped	INFO: A service has been stopped by the user via the L_ETC_MMCController function block or per reset.
MMC - slave ... ident data failed - CoE ... (set .../act ...)	ERROR: While the EtherCAT bus is booted and the additional identification parameters from the **mmc-0-ident.csv** configuration file are checked, an error has occurred in the given slave. The expected and current parameter contents do not match.
MMC - slave ... ident failed - CoE ... (error ...)	ERROR: While the EtherCAT bus is booted and the additional identification parameters from the **mmc-0-ident.csv** configuration file are checked, an error has occurred in the given slave. For further information see General error codes (L_ETC_ERRORCODE).
Modular Machine Configuration: Only one instance of L_ETC_MMCController allowed!	ERROR: When the EtherCAT master configuration is generated, this message is sent if more than one instance of the L_ETC_MMCController function block is used in the application. **Caution:** An instance might also have been created in a library!

13.6.3 Lenze Controller logbook messages

Lenze Controller logbook messages are displayed in the »WebConfig« as errors (highlighted in red), warnings, or information.

The same message text is displayed in the »PLC Designer« logbook and the EtherCAT logbook.

▸ Logbook of the Lenze Controller in the »WebConfig« (📖 191)

Note!

Cyclical messages as "WKC Error" or "Frame Lost" are displayed at the 1., 10., 100., 1000., 10000. etc. occurrence.

Error no.	Message text in the Lenze Controller logbook	Description
5063	... (...): CoE 0x... : ... - invalid slave state	CAN over EtherCAT: CoE is only provided from the "Pre-Operational" state onwards. An attempt has been made to access a CoE parameter in the "Bootstrap" or "Init" state.
5064	... (...): CoE 0x... : ... - SDO abort 'toggle bit unchanged (0x05030000)'	The status of the toggle bit has not changed.
5065	... (...): CoE 0x... : ... - SDO abort 'Protocol timeout (0x05040000)'	SDO protocol time-out
5066	... (...): CoE 0x... : ... - SDO abort 'Client/server command specifier invalid or unknown (0x05040001)'	Invalid or unknown specification symbol for the client/server command
5067	... (...): CoE 0x... : ... - SDO abort 'Invalid block size (only block mode) (0x05040002)'	Invalid block size (only in "Block mode")
5068	... (...): CoE 0x... : ... - SDO abort 'Invalid sequence number (only block mode) (0x05040003)'	Invalid sequence number (only in "Block mode")
5069	... (...): CoE 0x... : ... - SDO abort 'CRC error (only block mode) (0x05040004)'	CRC error (only in "Block mode")
5070	... (...): CoE 0x... : ... - SDO abort 'Memory overflow (0x05040005)'	The space in the main memory is not sufficient.
5071	... (...): CoE 0x... : ... - SDO abort 'Not supported access to object (0x06010000)'	Access to object not supported
5072	... (...): CoE 0x... : ... - SDO abort 'Attempt to read a write-only object (0x06010001)'	Read access to a write-only object
5073	... (...): CoE 0x... : ... - SDO abort 'Attempt to write to a read-only object (0x06010002)'	Write access to a read-only object
5074	... (...): CoE 0x... : ... - SDO abort 'Object does not exist in the object directory (0x06020000)'	An object does not exist in the object directory
5075	... (...): CoE 0x... : ... - SDO abort 'Object cannot be mapped to the PDO (0x06040041)'	An object cannot be mapped into the PDO
5076	... (...): CoE 0x... : ... - SDO abort 'Number and length of the objects to be mapped are greater than the PDO length (0x06040042)'	The number and/or length of the objects mapped would exceed the PDO length
5077	... (...): CoE 0x... : ... - SDO abort 'General incompatibility of the parameters (0x06040043)'	General parameter incompatibility
5078	... (...): CoE 0x... : ... - SDO abort 'General internal incompatibility within the device (0x06040047)'	General internal device incompatibility
5079	... (...): CoE 0x... : ... - SDO abort 'Access failed due to a hardware error (0x06060000)'	Access has failed due to a fault in the hardware
5080	... (...): CoE 0x... : ... - SDO abort 'Data format incompatible, length of the service parameter incompatible (0x06070010)'	The data type or the parameter length does not correspond
5081	... (...): CoE 0x... : ... - SDO abort 'Data format incompatible, service parameter too long (0x06070012)'	Incorrect data type (The parameter length is too large)
5082	... (...): CoE 0x... : ... - SDO abort 'Data format incompatible, service parameter too short (0x06070013)'	Incorrect data type (The parameter length is too small)
5083	... (...): CoE 0x... : ... - SDO abort 'Subindex does not exist (0x06090011)'	A subindex is not available
5084	... (...): CoE 0x... : ... - SDO abort 'Write access - parameter value beyond the permissible range (0x06090030)'	The value range for parameters is too great (only for write access)
5085	... (...): CoE 0x... : ... - SDO abort 'Write access - parameter value too high (0x06090031)'	The parameter value is too high
5086	... (...): CoE 0x... : ... - SDO abort 'Write access - parameter value too low (0x06090032)'	The parameter value is too low

Error no.	Message text in the Lenze Controller logbook	Description
5087	... (...): CoE 0x... : ... - SDO abort 'Maximum value is lower than minimum value (0x06090036)'	The maximum value is lower than the minimum value
5088	... (...): CoE 0x... : ... - SDO abort 'General error (0x08000000)'	General error
5089	... (...): CoE 0x... : ... - SDO abort 'Data cannot be transferred or saved in the application (0x08000020)'	Data cannot be transferred/saved to the application.
5090	... (...): CoE 0x... : ... - SDO abort 'Data cannot be transferred or saved in the application due to local control (0x08000021)'	Data cannot be transferred/saved to the application because of local control.
5091	... (...): CoE 0x... : ... - SDO abort 'Data cannot be transferred or saved in the application due to device status (0x08000022)'	Due to the current device state, data cannot be transferred to the application or stored in the application
5092	... (...): CoE 0x... : ... - SDO abort 'Dynamic generation of the object directory failed, or the object directory is missing (0x08000023)'	The dynamic generation of an object directory has failed, or no object directory is available.
5093	... (...): CoE 0x... : ... - SDO abort 'Unknown code'	Unknown code
5094	... (...): CoE 0x... : ... - invalid parameter	CAN over EtherCAT: An invalid parameter was transferred to a CoE function block (e.g: 'timeout = 0' or invalid slave address).
5095	... (...): CoE 0x... : ... - CoE protocol is not supported.	CAN over EtherCAT: An attempt was made to access a slave parameter, but the slave supports no CoE protocol.
5096	SLV: Undefined FoE error	File over EtherCAT: Manufacturer-specific error (see slave documentation)
5097	SLV: FoE error - not found	File over EtherCAT: Internal error
5098	SLV: FoE error - access denied	File over EtherCAT: No access to file
5099	SLV: FoE error - storage medium full	File over EtherCAT: No memory space to store file
5100	SLV: FoE error - illegal	File over EtherCAT: Internal error
5101	SLV: FoE error - wrong package number	File over EtherCAT: Internal error
5102	SLV: FoE error - already existing	File over EtherCAT: Internal error
5103	SLV: FoE error - user missing	File over EtherCAT: Internal error
5104	SLV: FoE error - bootstrap only	File over EtherCAT: Transfer only permitted in bootstrap.
5105	SLV: FoE error - no bootstrap	File over EtherCAT: Transfer only permitted in bootstrap.
5106	SLV: FoE error - no rights	File over EtherCAT: No access authorisation
5107	SLV: FoE - program error	File over EtherCAT: Internal error
5108	SLV: FoE - invalid parameter	File over EtherCAT: An invalid parameter was transferred to a FoE function block (e.g. 'timeout = 0' or invalid slave address).
5513	...: State change from '...' to '...'	Info: The EtherCAT master has successfully executed a state change.
5514	...: Bus scan successful. (...) slaves found.	Busscan was successfull with given slave count.
5515	...: Bus scan error ' ... '. (...) slaves found.	Busscan failed with given slave count.
5518	...: CoE - SDO download failed. statVal=..., errCode=0x... (...)	CAN over EtherCAT: Internal error - error during CoE download: parameter, mailbox status, error code
5519	...: CoE - SDO upload failed. statVal=..., errCode=0x... (...)	CAN over EtherCAT: Internal error - error during CoE upload: parameter, mailbox status, error code
5520	...: CoE - OD list upload failed. statVal=..., errCode=0x... (...)	CAN over EtherCAT: Internal error - error uploading the object directory: parameter, mailbox status, error code
5521	...: CoE - Object entry description upload failed. statVal=..., errCode=0x... (...)	CAN over EtherCAT: Internal error - error uploading the object directory/parameter description: parameter, mailbox status, error code
5522	...: CoE - Object entry description upload failed. statVal=..., errCode=0x... (...)	CAN over EtherCAT: Internal error - error uploading the object directory/parameter description: parameter, mailbox status, error code
5523	...: CoE - emergency transfer failed. statVal=..., errCode=0x... (...)	CAN over EtherCAT: Internal error during emergency message transfer
5524	... (...): CoE - emergency request. id=0x..., len=..., errCode=0x..., ErrReg=0x..., data: 0x... 0x... 0x... 0x... 0x...	CAN over EtherCAT: Internal error during emergency message transfer **Note:** "data: ..." indicates by means of codes which error has occurred in which slave device/module. Detailed information on the coding of error messages is provided in the documentation for the corresponding slave device/module.
5525	...:Cyclical command WKC error command: ... - logic/physical address: 0x... - WKC act/set=.../...	Error during cyclical command: One or more slaves have not processed the command.
5526	...: Master init command WKC error - command: ... - logic/physical address: 0x..., WKC act/set=.../...	Error during the initialisation command: One or more slaves have not processed the command.
5527	... (...): Slave init command WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	Error during the initialisation command: One or more slaves have not processed the command.

Error no.	Message text in the Lenze Controller logbook	Description
5528	... (...): EoE receive WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	Ethernet over EtherCAT: error during initialisation command for 'EoE Receive Request' One or more slaves have not processed the command.
5529	... (...): CoE receive WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	CAN over EtherCAT: error during initialisation command for 'CoE Receive Request' One or more slaves have not processed the command.
5530	... (...): FoE receive WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	File over EtherCAT: error during initialisation command for 'FoE Receive Request' One or more slaves have not processed the command.
5531	... (...): SoE receive WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	Sercos over EtherCAT: error during initialisation command for 'SoE Receive Request' One or more slaves have not processed the command.
5532	... (...): EoE send WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	Ethernet over EtherCAT: error during initialisation command for 'EoE Send Request' One or more slaves have not processed the command.
5533	... (...): CoE send WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	CAN over EtherCAT: error during initialisation command for 'CoE Send Request' One or more slaves have not processed the command.
5534	... (...): FoE send WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	File over EtherCAT: error during initialisation command for 'FoE Send Request' One or more slaves have not processed the command.
5535	... (...): SoE send WKC error. Command: ..., logic/physical address: 0x..., WKC act/set=.../...	Sercos over EtherCAT: error during initialisation command for 'SoE Send Request' One or more slaves have not processed the command.
5541	... (...): Error at response to Init command - no response. State change='...'	Slave does not respond to Init command
5542	... (...): Error at response to Init command - validation error. State change='...'	Slave does not respond correctly to Init command
5543	... (...): Error at response to Init command - failed. State change='...'	Init command cannot be written to slave
5544	...: Error at response to master init command - no response. State change='...'	Slaves do not respond to Init command (broadcast)
5545	...: Error at response to master init command - validation error. State change='...'	Slaves do not respond correctly to Init command (broadcast)
5546	... (...): EtherCAT command ... is missing in the Ethernet frame. Index of the missing command in the Ethernet frame=...	Internal error
5547	... (...): Mailbox init command timeout. Current state change of the slave='...'	Time-out during mailbox initialisation in the case of a state change
5549	...: Ethernet cable is connected	Ethernet cable reconnected (link-up available)
5550	...:Ethernet cable is not connected	Ethernet cable is not connected (no link-up available)
5551	...: Timeout for cyclical commands. Time between transmission processes too long	Internal error
5552	...: Redundant operation. Ethernet cable missing at the 2. EtherCAT interface	Lenze controller does not support redundancy.
5554	... (...): Slaves signals error. AL status: '...' (0x...), AL status code: '...' (0x...)	Slave indicates an error. AL status and AL status code are slave-specific.
5555	... (...): Communication to device interrupted	Connection to slave is interrupted. The slave does not respond or is no longer available.
5557	...: DC slaves 'in-sync'. Deviation ... ns	Information that the DC deviation is within the permissible limits (standard 8 µs).
5558	...: DC slaves 'out-of-sync'. Deviation ... ns	Information that the DC deviation is not within the permissible limits (standard 8 µs).
5562	...: Client registration lost	Internal error
5704	...: Remote API diagnosis port - connection established	Internal information
5705	...: Remote API diagnosis port - disconnected	Internal information
5706	...: Client was registered by cookie 0x... instance 0x... Id 0x... result ...	Internal information
5707	...: Client was deleted by cookie 0x... instance 0x... Id 0x... result ...	Internal information
5708	...: Unknown registration: 0x...	Internal error: Master outputs unknown message.
5719	... (...): Slave has incorrect status. Status set/act '...'/'...'	Slave is not in expected status: 'req' is the requested status and 'act' the current status
5729	...: RAP - invalid parameter size for ...	Internal error
5730	...: RAP - Marshaling error. Cookie: 0x..., Command: 0x..., Cause: ... (0x...), Protocol Header: 0x...	Internal error

Error no.	Message text in the Lenze Controller logbook	Description
5731	...: ... could not be included in queue (missing calls of the ProcessNotificationJobs)	Internal error
5733	...: ecatSetTargetState - EtherCAT master could not be set to "target state". The master is busy (timeout)	Time-out during status change 'Request'
5740	...: Error 0x... when reading out the bus scan status	Internal error: Internal bus scan failed.
5743	...: Error 0x... during restart of the bus scan	Internal error: Internal bus scan failed.
6200	... (...): Configuration mismatch. VendorID check failed (0x... / 0x...)	Configured and current bus structure do not match. At the indicated position, a slave with the following vendor ID is expected (expected/current).
6201	... (...): Configuration mismatch. ProductCode check failed (0x... / 0x...)	Configured and current bus structure do not match. At the indicated position, a slave with the following vendor product code is expected (expected/current).
6202	... (...): Configuration mismatch. Revision check failed (0x... / 0x...)	Configured and current bus structure do not match. At the indicated position, a slave with the following revision is expected (expected/current).
6203	... (...): Configuration mismatch. VendorID check failed (0x... / missing...)	Configured and current bus structure do not match. At the indicated position, a slave with the following vendor ID is expected (currently, there is no slave available here).
6204	...: Configuration mismatch. Odd device at bus end after '... (...)'. Identification 0x... / 0x... / 0x...)	Configured and current bus structure do not match. More slaves than configured are connected to the bus. For the first odd slave, vendor ID, product code and revision are indicated.
6212	...: All slaves 'Operational' again	Information: Slaves reset from "Operational" state to a lower state were set to "Operational" again (e.g. by means of L_ETC_SetSlaveState())
6213	...: Cyclical command WKC error (repeated ... times)	One or several slaves do not process the commands of the cyclic frames. Possible causes may e.g. be that slaves are no longer available or were reset from the "Operational" status to a lower status. Only the 1st, 10th, 100th, 1000th, 10000th, etc. error is logged.
6214	...: Frame response error (repeated ... times)	Master did not receive transmitted EtherCAT frame with the next bus cycle task call. Causes may be wiring errors, contact problems, and an excessive cycle time load of the EtherCAT task.
6215	...: Not all slaves 'Operational' (repeated ... times)	The master is in the "Operational" state and one or more slaves are falling back to a lower status. Only the 1st, 10th, 100th, 1000th, 10000th, etc. error is logged.
6216	... (...): Emergency message overflow. Further emergency messages are blocked	A slave repeatedly sends the same emergency message. After receipt of five messages, this error message appears and further emergency messages from the slave are no longer logged until the status of the slave changes from "Init" to "Pre-Operational".
6220	...: New configuration loaded	Information: New IEC application with EtherCAT master was loaded.
6221	...: New configuration loaded. No slaves defined	Information: New IEC application with EtherCAT master was loaded. There are no slaves defined.
6222	...: Master start failed. Configuration error	Internal error: Master was not configured correctly.
6230	...: Master start failed	The master cannot be started. General message (no 'Bus mismatch', DC/DCM, cable problem). Observe previous messages in the logbook!
6231	...: Master start failed. Bus mismatch	The master does not start due to a 'Bus mismatch': Configured and current bus structure do not match. Which slave is not correct is logged shortly before this message.
6232	...: Master start failed. EtherCAT cable not connected	The master cannot be started because the Ethernet cable is not connected (no link-up).
6233	...: Master start failed. DC/DCM configuration	Internal error: The master cannot be started because of a faulty DC/DCM configuration.
6234	...: Master start failed. Slaves cannot be set to Pre-Operational.	The master cannot be started because of a slave error. Observe previous messages (slave error) in the logbook!
6240	...: Status change 'Operational' failed (0x...)	The master cannot be set to "Operational". General message, none of the following errors. Observe previous messages in the logbook!
6241	...: Status change 'Operational' failed. Master is not initialised	Internal error: The master cannot be started because of a faulty DC/DCM configuration.
6242	...: Status change 'Operational' failed. EtherCAT cable not connected	The master cannot be set to "Operational" because the Ethernet cable is not connected (no link-up).
6243	...: Status change 'Operational' failed. DCM is not in-sync	The master cannot be set to "Operational" because the Ethernet cable is not connected (no link-up).
6244	...: Status change 'Operational' failed. Times for bus cycle task and DC are not identical	Internal error: The DC cycle time and the cycle time of the EtherCAT bus cycle task are not identical.

Error no.	Message text in the Lenze Controller logbook	Description
6245	...: Status change 'Operational' takes some time ...	Information: The transition from "Safe-operational" -> "Operational" takes longer. This message will be output after 10 s if one or several slaves did not change to the "Operational" state. E.g. for Servo Drives 9400 with long cycle times, because the standard device must synchronise with the communication module (SYNC0).
6246	...: Status change 'Operational' failed. Time-out.	The master cannot be set to "Operational" because of a time-out (standard 55 s).
6247	...: Status change 'Operational' failed. Slave error.	Setting the master to "Operational" failed due to a slave error. Observe previous messages (slave error) in the logbook!
6248	...: Status change 'Operational' cancelled by reset command.	The master cannot be set to "Operational" because the user cancelled the procedure.
6250	...: Master stop failed (0x...).	The master cannot be stopped or cannot be set to the "Init" state.
6251	...: Master stop failed (0x...). Slaves cannot be set to Pre-operational.	The master cannot be stopped and slaves cannot be set to the "Pre-operational" state.
6260	...: Master shutdown failed (0x...).	The master cannot be shut down and cannot be set to the "Init" state.
6270	...: Remote API server start failed	Internal error: The remote API server cannot be started. Communication of CoE parameters from the engineering tool (»EASY Starter«/»Engineer«) is not possible.
6280	...: Start download	Information: Firmware/parameter set download was started.
6281	...: Download complete.	Information: Firmware/parameter set download was completed.

13.6.4 SDO abort codes

The abort codes are relevant for ...

- the *eErrorCode* output in some function blocks of the L_IODrvEtherCAT function library (📖 126);
- SDO read/write error messages, cause by SDO queries from the system (e.g. initialisation code or SDO queries from the engineering tool).

Error number [hex]	Description
0x00000000	No error
0x05030000	The status of the toggle bit has not changed.
0x05040000	SDO protocol time-out
0x05040001	Invalid or unknown specification symbol for the client/server command
0x05040002	Invalid block size (only in "Block mode")
0x05040003	Invalid sequence number (only in "Block mode")
0x05040004	CRC error (only in "Block mode")
0x05040005	The space in the main memory is not sufficient.
0x06010000	Access to object not supported
0x06010001	Read access to a write-protected object
0x06010002	Write access to a write-protected object
0x06020000	An object does not exist in the object directory
0x06040041	An object cannot be mapped into the PDO
0x06040042	The number and/or length of the objects mapped would exceed the PDO length
0x06040043	General parameter incompatibility
0x06040047	General internal device incompatibility
0x06060000	Access has failed due to a fault in the hardware
0x06070010	The data type or the parameter length does not correspond
0x06070012	Incorrect data type (The parameter length is too large)
0x06070013	Incorrect data type (The parameter length is too small)
0x06090011	A subindex is not available
0x06090030	The value range for parameters is too great (only for write access)
0x06090031	The parameter value is too high
0x06090032	The parameter value is too low
0x06090036	The maximum value is lower than the minimum value
0x08000000	General error
0x08000020	Data cannot be transferred/saved to the application.
0x08000021	Data cannot be transferred/saved to the application because of local control.
0x08000022	Due to the current device state, data cannot be transferred to the application or stored in the application
0x08000023	The dynamic generation of an object directory has failed, or no object directory is available.

14 Parameter reference

This chapter complements the parameter list of the online help for the Lenze Controller by the parameters of the EtherCAT communication interface.

These parameters ...

- are for instance shown in the Lenze »WebConfig« (Engineering tool for web-based parameterisation);
- are listed in numerically ascending order.

C280/4

Parameter | Name: **C280/4 | ECAT bus scan compliance**		Data type: UNSIGNED_8 Index: 24295.4 = 0x5EE7.0x04
Brief information on whether the master configuration corresponds to the physical bus structure. The master configuration of the stack is compared to the actual bus structure.		
Selection list(Lenze setting printed in bold)		**Info**
0	**No match**	The master configuration does not correspond to the bus structure.
1	OK	The master configuration corresponds to the bus structure.
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer		

C281/2

Parameter | Name: **C281/2 | ECAT master: State**		Data type: UNSIGNED_8 Index: 24294.2 = 0x5EE6.0x02
Display of the current state of the master		
Selection list (read only)		
0	Unknown	
1	Init	
2	Pre-Operational	
3	Bootstrap mode	The Bootstrap mode is not supported
4	Safe-operational	
8	Operational	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer		

C281/5

Parameter | Name: **C281/5 | ECAT master: State info**	Data type: UNSIGNED_32 Index: 24294.5 = 0x5EE6.0x05
Display of additional information on the current state of the master The bits are set to the value 1 when the respective states are reached.	

Value is bit-coded:	
Bit 0	Master ok
Bit 1	Reserved 1
Bit 2	Reserved 2
Bit 3	Reserved 3
Bit 4	Init
Bit 5	Pre-Operational
Bit 6	Safe-operational
Bit 7	Operational
Bit 8	Slaves in requested state
Bit 9	Master in requested state
Bit 10	Bus Scan Match
Bit 11	Reserved 4
Bit 12	DC: Activated
Bit 13	DC: Synchronised
Bit 14	DC: Busy
Bit 15	Reserved 5
Bit 16	Link Up
Bit 17	Reserved 6
...	...
Bit 31	Reserved 20

☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer

C281/6

Parameter | Name: **C281/6 | ECAT bus scan**	Data type: UNSIGNED_8 Index: 24294.6 = 0x5EE6.0x06
Activation of the fieldbus scan The fieldbus scan updates all EtherCAT codes.	

Selection list(Lenze setting printed in bold)	
0	**No Operation**
1	The bus is scanned

☑ Read access ☑ Write access ☐ CINH ☐ PLC-STOP ☑ No transfer

C282/2

Parameter | Name: **C282/2 | ECAT DC: Slave sync deviation limit**	Data type: UNSIGNED_32 Index: 24293.2 = 0x5EE5.0x02
Permissible deviation of the distributed clocks of all devices in nanoseconds. If the permissible deviation is exceeded, the master will initiate a resynchronisation of the distributed clocks.	

Display range (min. value | unit | max. value)		
0	ns	4294967295

☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer

C282/3

Parameter | Name: **C282/3 | ECAT DC: Current deviation**			Data type: INTEGER_32 Index: 24293.3 = 0x5EE5.0x03
Current maximum deviation of the distributed clocks of all devices in nanoseconds.			
Display range (min. value | unit | max. value)			
-2147483647	ns	2147483647	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/3

Parameter | Name: **C286/3 | ECAT bus: No. of slaves**			Data type: UNSIGNED_32 Index: 24289.3 = 0x5EE1.0x03
Number of slaves connected to the fieldbus			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/4

Parameter | Name: **C286/4 | ECATBus: No. of DC slaves**			Data type: UNSIGNED_32 Index: 24289.4 = 0x5EE1.0x04
Number of slaves connected to the fieldbus and supported by distributed clocks			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/5

Parameter | Name: **C286/5 | ECAT config.: No. of slaves**			Data type: UNSIGNED_32 Index: 24289.5 = 0x5EE1.0x05
Number of slaves configured in the master configuration file			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/6

Parameter | Name: **C286/6 | ECAT config.: No. of mailbox slaves**			Data type: UNSIGNED_32 Index: 24289.6 = 0x5EE1.0x06
Number of mailbox slaves configured in the master configuration file			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/7

Parameter | Name: **C286/7 | ECAT counter: Tx frames**			Data type: UNSIGNED_32 Index: 24289.7 = 0x5EE1.0x07
Number of sent frames			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/8

Parameter | Name: **C286/8 | ECAT counter: Rx frames**			Data type: UNSIGNED_32 Index: 24289.8 = 0x5EE1.0x08
Number of received frames			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/9

Parameter | Name: **C286/9 | ECAT counter: Lost frames**			Data type: UNSIGNED_32 Index: 24289.9 = 0x5EE1.0x09
Number of lost frames			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/10

Parameter | Name: **C286/10 | ECAT counter: Cyclic frames**			Data type: UNSIGNED_32 Index: 24289.10 = 0x5EE1.0x0A
Number of cyclic frames			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/11

Parameter | Name: **C286/11 | ECAT counter: Cyclic datagrams**			Data type: UNSIGNED_32 Index: 24289.11 = 0x5EE1.0x0B
Number of cyclic datagrams			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/12

Parameter | Name: **C286/12 | ECAT counter: Acyclic frames**			Data type: UNSIGNED_32 Index: 24289.12 = 0x5EE1.0x0C
Number of acyclic frames			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/13

Parameter | Name: **C286/13 | ECAT counter: Acyclic datagrams**			Data type: UNSIGNED_32 Index: 24289.13 = 0x5EE1.0x0D
Number of acyclic datagrams			
Display range (min. value | unit | max. value)			
0		4294967295	
☑ Read access ☐ Write access ☐ CINH ☐ PLC-STOP ☐ No transfer			

C286/14

Parameter | Name: **C286/14 | ECAT Clear specific counters**		Data type: UNSIGNED_32 Index: 24289.14 = 0x5EE1.0x0E
Reset frame and datagram counters (C1086/7 ... 13)		
Selection list(Lenze setting printed in bold)		
0	**No Operation**	
1	Reset - All counters	
2	Reset - Tx frame counters	
4	Reset - Rx frame counters	
8	Reset - Lost frame counters	
16	Clear Cyclical Frame Counter	
32	Clear Cyclical Datagram Counter	
64	Reset - Acyclic frame counters	
128	Reset - Acyclic datagram counters	
☑ Read access ☑ Write access ☐ CINH ☐ PLC-STOP ☑ No transfer		

Index

Figures

A

B

C

D

T

U

V

W

FEEDBACK

Your opinion is important to us

These instructions were created to the best of our knowledge and belief to give you the best possible support for handling our product.

If you have suggestions for improvement, please e-mail us to:

feedback-docu@lenze.com

Thank you for your support.

Your Lenze documentation team

Lenze SE
Postfach 10 13 52 · 31763 Hameln
Hans-Lenze-Straße 1 · 31855 Aerzen
GERMANY
Hannover HRB 204803
Phone +49 5154 82-0
Fax +49 5154 82-2800
sales.de@lenze.com
www.lenze.com

Service
Lenze Service GmbH
Breslauer Straße 3, 32699 Extertal
GERMANY
Phone 008000 24 46877 (24 h helpline)
Fax +49 5154 82-1112
service.de@lenze.com