

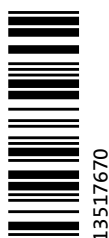
Engineer/ PLC Designer



Application Samples Inverter 8400/9400 _ _ _

Software Manual

EN



Inhalt

1	About this documentation	7
1.1	Document history	10
1.2	Conventions used	11
1.3	Terminology used	12
1.4	Notes used	13
2	Safety instructions	14
3	Conditions	16
3.1	System requirements	16
3.2	Wiring devices (Controllers, inverters, motors)	17
3.3	Establishing communication with the controller	18
4	Lenze Application Samples	19
4.1	Overview of the application samples	20
4.2	Where to find the application samples	21
5	Using sample projects in the »Engineer«	22
6	Using sample projects in the »PLC Designer«	24
6.1	Control configuration (EtherCAT/CAN)	26
6.2	Sample project with the "speed actuating drive" application	27
6.3	Sample project with the "table positioning" application	28
6.4	Program blocks (PRG)	29
6.5	Visualisations of the "speed actuating drive" application	30
6.5.1	Information (start page)	31
6.5.2	Automatic mode	32
6.5.3	Manual mode	33
6.5.4	Service mode	34
6.5.5	Parameter	35
6.6	Visualisations of the "table positioning" application	36
6.6.1	Information (start page)	37
6.6.2	Automatic mode	38
6.6.3	Manual mode	39
6.6.4	Service mode	40
6.6.5	Homing mode	41
6.6.6	Parameter	42
6.7	Extending sample projects (adding devices)	43
6.7.1	Adding EtherCAT field devices	43
6.7.2	Adding CAN field devices	46

7	Device descriptions.....	50
7.1	Inverter Drive 8400 BaseLine C.....	50
7.2	Inverter Drive 8400 StateLine C	52
7.3	Inverter Drive 8400 HighLine C.....	54
7.4	Inverter Drive 8400 TopLine C.....	56
7.5	Servo Drive 9400 HighLine.....	58
7.5.1	Indices for "table positioning" technology application	59
7.5.2	Indices for the other technology applications.....	60
7.6	Servo Drive 9400 Regenerative Power Supply Module	61
7.7	Lenze Generic Drive	63

8	L_LCB_LogicControlBasic function library	65
8.1	Overview of the functions and function blocks	66
8.2	Function blocks	67
8.2.1	L_LCB_AXIS_REF	67
8.2.2	L_LCB_AXIS_REF_CAN (internal use)	68
8.2.3	L_LCB_AXIS_REF_ETC (internal use)	68
8.2.4	L_LCB_8400Drive	69
8.2.5	L_LCB_9400Drive	70
8.2.6	L_LCB_GenericDrive	72
8.2.7	L_LCB_ActuatorSpeed	73
8.2.8	L_LCB_TablePositioning	77
8.2.9	L_LCB_Norm_aToNorm_n	83
8.2.10	L_LCB_Norm_nToNorm_a	83
8.2.11	L_LCB_Norm_nToSpeed_s	84
8.2.12	L_LCB_Speed_nToNorm_r	84
8.2.13	L_LCB_Speed_rToNorm_n	85
8.2.14	L_LCB_Speed_sToNorm_n	85
8.2.15	L_LCB_Speed_sToSpeed_v	86
8.2.16	L_LCB_Speed_vToSpeed_s	86
8.2.17	L_LCB_GetAxisData	87
8.2.18	L_LCB_GetPosition	88
8.2.19	L_LCB_GetSpeed	89
8.2.20	L_LCB_SetAxisData	90
8.2.21	L_LCB_SetPosition	92
8.2.22	L_LCB_SetSpeed	93
8.2.23	L_LCB_AccToUnit	94
8.2.24	L_LCB_PosToUnit	95
8.2.25	L_LCB_SpeedToUnit	96
8.2.26	L_LCB_UnitToAcc	97
8.2.27	L_LCB_UnitToPos	98
8.2.28	L_LCB_UnitToSpeed	99
8.2.29	L_LCB_TaskCycle	100
8.3	Structures (Structs)	101
8.3.1	L_LCB_AxisData	101
8.4	Enumerations (Enums)	102
8.4.1	L_LCB_AXISMODE	102
8.4.2	L_LCB_Error	102
8.4.3	L_LCB_LOGICDEVICE	102

9	L_DCO_DriveCommunication function library	103
9.1	Overview of the functions and function blocks	104
9.2	Function blocks	105
9.2.1	L_DCO_AXIS_REF_BASE	105
9.2.2	L_DCO_ReadDriveParameter	106
9.2.3	L_DCO_ReadDriveParameterString	108
9.2.4	L_DCO_TransferDriveParameter	110
9.2.5	L_DCO_WriteDriveParameter	112
9.3	Structures (Structs)	114
9.3.1	L_DCO_SDOData	114
9.3.2	L_DCO_TransferData	114
9.4	Enumerations (Enums)	115
9.4.1	L_DCO_CommState	115
9.4.2	L_DCO_Error	116
9.5	Unions	118
9.5.1	L_DCO_ConvertData	118
10	L_DAC_DataConversion function library	119
10.1	Overview of the functions and function blocks	119
10.2	Bit operations	120
10.2.1	L_DAC_GetBitOfByte	120
10.2.2	L_DAC_GetBitOfDWord	120
10.2.3	L_DAC_GetBitOfWord	121
10.2.4	L_DAC_ResetBitOfByte	121
10.2.5	L_DAC_ResetBitOfDWord	122
10.2.6	L_DAC_ResetBitOfWord	122
10.2.7	L_DAC_SetBitOfByte	123
10.2.8	L_DAC_SetBitOfDWord	123
10.2.9	L_DAC_SetBitOfWord	124
10.3	Bit splitter	125
10.3.1	L_DAC_BitsToByte	125
10.3.2	L_DAC_BitsToDWord	125
10.3.3	L_DAC_BitsToWord	126
10.3.4	L_DAC_ByteToBits	126
10.3.5	L_DAC_DWordToBits	127
10.3.6	L_DAC_WordToBits	127

Contents

10.4	Type converter	128
10.4.1	L_DAC_2BytesToWord	128
10.4.2	L_DAC_2WordsToDWord	128
10.4.3	L_DAC_4BytesToDWord	129
10.4.4	L_DAC_DWordTo2Words	129
10.4.5	L_DAC_DWordTo4Bytes	130
10.4.6	L_DAC_WordTo2Bytes	130
11	Automatically generated functions (project information)	131
11.1	GetBooleanProperty	131
11.2	GetCompany	132
11.3	GetNumberProperty	132
11.4	GetTextProperty	133
11.5	GetTitle	133
11.6	GetVersion	134
11.7	GetVersionProperty	134
12	Index	135
	<i>Your opinion is important to us.</i>	<i>138</i>

1 About this documentation

This documentation describes the Lenze application samples for an easy commissioning of Inverter Drives 8400 and Servo Drives 9400 within the scope of the Lenze "Controller-based Automation" system.

The application samples are predefined sample projects for the Lenze Engineering tools »PLC Designer« and »Engineer«.

Depending on the application, the sample projects can be extended as desired and adapted to the requirements in each case.

This documentation 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 Controller (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« (backup, restore, 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:

Planning / configuration / technical data	
☐	Product catalogues • Controller-based Automation • Controllers • Inverter Drives/Servo Drives
Mounting 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
Parameter setting / 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 and reference manuals • i700 application sample • Application Samples 8400/9400 • FAST Application Template Lenze/PackML • 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 configure, install, commission and maintain a Lenze automation system.

Information regarding the validity

The information in this documentation applies to:

Engineering tool	from software version
»PLC Designer«	3.3
»Engineer«	2.13

The Lenze engineering tools are available for download at:

www.lenze.com → Download → Software Downloads

Screenshots/application examples

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

1.1

Document history

Version			Description
2.1	08/2016	TD11	Notes on use added ▶ L_LCB_SetAxisData (📖 90) ▶ L_DCO_AXIS_REF_BASE (📖 105) ▶ L_DCO_ReadDriveParameter (📖 106) ▶ L_DCO_ReadDriveParameterString (📖 108) ▶ L_DCO_TransferDriveParameter (📖 110) ▶ L_DCO_WriteDriveParameter (📖 112)
2.0	02/2016	TD11	General revision and updates
1.4	06/2015	TD11	Updated: • L_LCB_GetAxisData (📖 87)
1.3	06/2014	TD11	• Overview of the application samples (📖 20) supplemented. • Updated representation of the function blocks.
1.2	06/2012	TD11	Notes on use added ▶ L_LCB_ActuatorSpeed (📖 73) ▶ L_LCB_TablePositioning (📖 77)
1.1	05/2011	TD11	• Device descriptions (📖 50) supplemented. • L_LCB_LogicControlBasic function library (📖 65) extended.
1.0	04/2011	TD11	First edition

1 About this documentation

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 separator	Point	The decimal point is always used. For example: 1234.56
Hexadecimal number	0x	For hexadecimal numbers, the "0x" prefix is used. Example: 0x60F4
Binary number	0b	For binary numbers, the "0b" prefix is used. Example: 0b00010111
Text		
Version information	Blue text colour	All information that only applies to or from a certain software version of the inverter is marked correspondingly in this documentation. Example: This function extension is available from software version V3.0 onwards!
Program name	» «	Lenze »PLC Designer«...
Window	<i>italics</i>	The <i>message window</i> ... / The <i>Options</i> dialog box ...
Variable names		By setting <i>bEnable</i> 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. It is activated with a mouse-click in this online documentation.
Page reference	(11)	Reference to further information: Page number in PDF file.
Icons		
Step-by-step instructions		Step-by-step instructions are indicated by a pictograph.

1 About this documentation

1.3 Terminology used

1.3 Terminology used

Term	Meaning
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.
Engineering PC	The Engineering PC and the Engineering tools installed serve to configure and parameterise the system. The Engineering PC communicates with the controller via Ethernet.
 EtherCAT	EtherCAT® (Ethernet for Controller and Automation Technology) 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.
FAST	Lenze Application Software
Inverters	Generic term for Lenze frequency inverters, servo inverters
PLC	Programmable Logic Controller (German designation: SPS - Speicherprogrammierbare Steuerung)

1.4

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!	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.

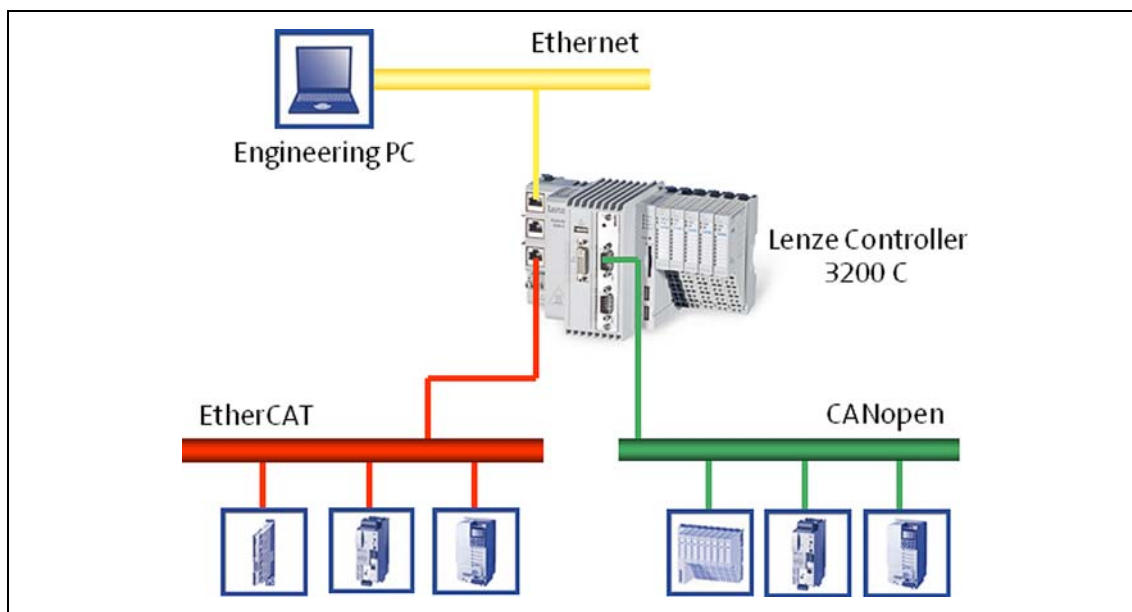
Protective measures

- 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 Conditions

The conditions described in this chapter must be fulfilled in order that an application sample can be used.

3.1 System requirements



[3-1] Example: system structure with a Controller 3200 C

Components	Engineering PC	Lenze Controller/Inverter	
Hardware	PC/notebook	Controller: • 3200 C / p500 / c300 / p300 Inverter: • 8400 BaseLine (CAN) • 8400 StateLine (CAN, EtherCAT) • 9400 HighLine (CAN, EtherCAT)	
Operating system of	Windows®	Controller:	
		3200C	Windows® CE
		p500	
		c300	Windows® Embedded Compact 7
		p300	
Engineering tool	»PLC Designer« from version 3.3 »Engineer« from version 2.13	• Controller: Application Software "FAST Runtime" (formerly "L-force Logic" (LPC 1000)) • Inverter: Firmware is device-dependent.	
Further requirements	-	Selectable sample applications for CAN or EtherCAT	

Detailed information on the system requirements for the Lenze engineering tools can be found on the download pages of the engineering tools at:

www.lenze.com → Download → Software Downloads

3.2

Wiring devices (Controllers, inverters, motors)



More information on the electrical connections can be found in the documentation supplied. Please read the mounting instructions supplied with the devices before you start working!

The mounting instructions contains safety instructions which must be observed!

Before you can start working with a sample project, the hardware components need to be connected with each other.

- Connect the Controller with the fieldbus nodes (inverters).
For this purpose, use a suitable fieldbus cable.
- Connect the devices (Controllers, inverters, motors) with the corresponding voltage supply via the power terminals.
- Wire the control terminals.
- Connect USB diagnostic adapter.
- Switch on the voltage supply of the fieldbus nodes (inverters).

When the green LED "DRV-RDY" is blinking and the red LED "DRV-ERR" is off, the inverter is ready to start and you can continue with the commissioning.

3.3 Establishing communication with the controller

Connect the Engineering PC to the Lenze Controller via a network cable. The »PLC Designer« accesses the Controller via Ethernet.

Detailed information on establishing fieldbus communication and on commissioning the Lenze "Controller-based Automation" system can be found in the following sets of documentation:



Communication manual Controller-based Automation EtherCAT

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



Communication manual for Controller-based Automation with CANopen

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

4 Lenze Application Samples

The Lenze application samples are predefined sample projects for the Engineering tools »Engineer« and »PLC Designer«.

The sample projects contain standard system configurations that represent typical applications. Depending on the application, the sample projects can be extended as desired and adapted to the respective requirements.

This way you can develop an executable Lenze automation system with little effort.

Structure of the application samples

Each sample project consists of:

- Ready-made projects in the »Engineer«
- Program code and visualisations in the »PLC Designer«

Possible operating modes are:

- Automatic
- Manual
- Service
- Homing

4.1

Overview of the application samples

Product/ application	Project name	Bus system	Lenze Engineering tool	
			»PLC Designer« 	»Engineer« 
Inverter Drives 8400				
Baseline				
"Speed actuating drive"	LAS_40_INTF_Can_84BL_Speed_ [...]	CAN	●	●
StateLine				
"Speed actuating drive"	LAS_40_INTF_Can_84SL_Speed_ [...]	CAN	●	●
	LAS_40_INTF_ETC_84SL_Speed_ [...]	EtherCAT	●	●
HighLine				
"Table positioning"	LAS_40_INTF_Can_84HL_TabPos_ [...]	CAN	●	●
	LAS_40_INTF_ETC_84HL_TabPos_ [...]	EtherCAT	●	●
Servo Drives 9400				
HighLine				
"Speed actuating drive"	LAS_40_INTF_Can_94HL_Speed_ [...]	CAN	●	●
	LAS_40_INTF_ETC_94HL_Speed_ [...]	EtherCAT	●	●
"Table positioning"	LAS_40_INTF_Can_94HL_TabPos_ [...]	CAN	●	●
	LAS_40_INTF_ETC_94HL_TabPos_ [...]	EtherCAT	●	●

Name of the application samples

LAS_<Sort key>_<Category>_<Bus system>_<Inverter>_<Application>_<Version>

Keyword	Entry	Meaning
Sort key	40	Application samples with interface function (interface connection) for connecting Lenze inverters.
Category	INTF	
Bus system	Can	Project for CAN
	ETC	Project for EtherCAT
Inverters	84BL	Inverter Drive 8400 BaseLine
	84SL	Inverter Drive 8400 Stateline
	84BH	Inverter Drive 8400 HighLine
	94HL	Servo Drive 9400 HighLine
Application	Speed	Application "speed actuating drive"
	TabPos	Application "table positioning"
Version	xyyy	Sample version • Example: Entry '0200' => Version '2.0'

4.2 Where to find the application samples:

After the »Engineer« and »PLC Designer« have been installed successfully, you can find the application samples in the **Windows® start menu** under:

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



Tip!

The »Engineer« sample projects are archived in ZIP format to allow you to send them for instance by e-mail.

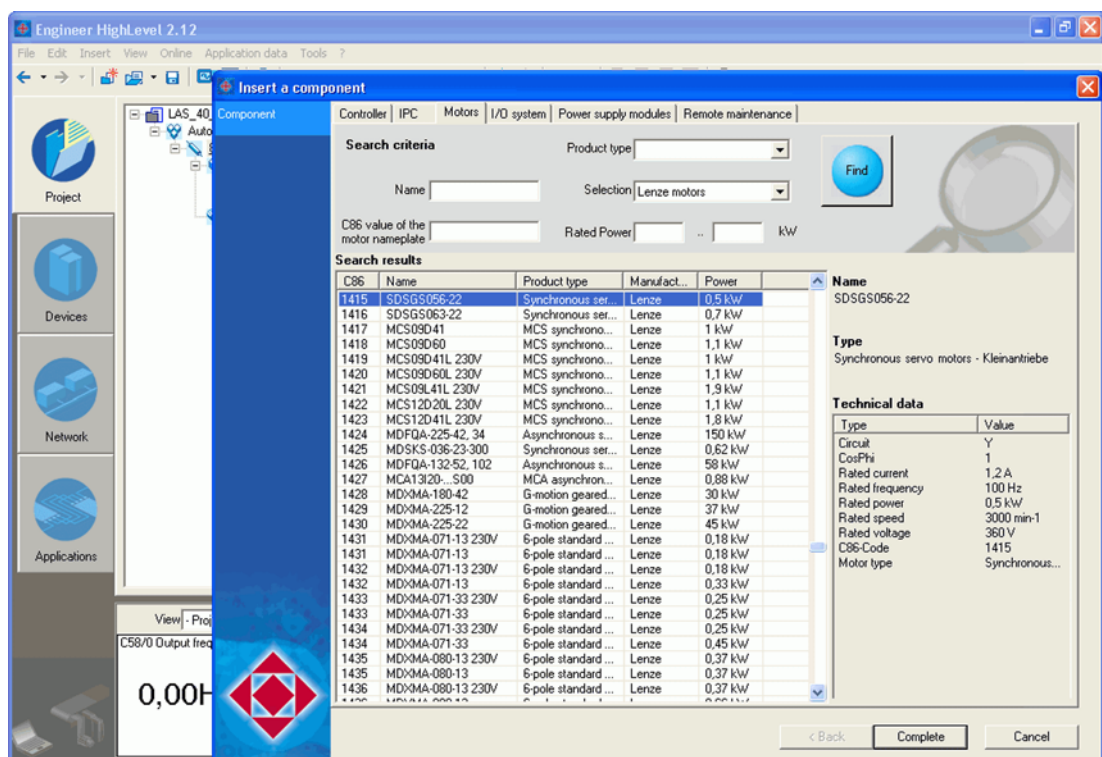
The »Engineer« supports project saving (**File → Save archive**) and project opening (**File → Open archive**) in ZIP format.

5 Using sample projects in the »Engineer«



How to proceed:

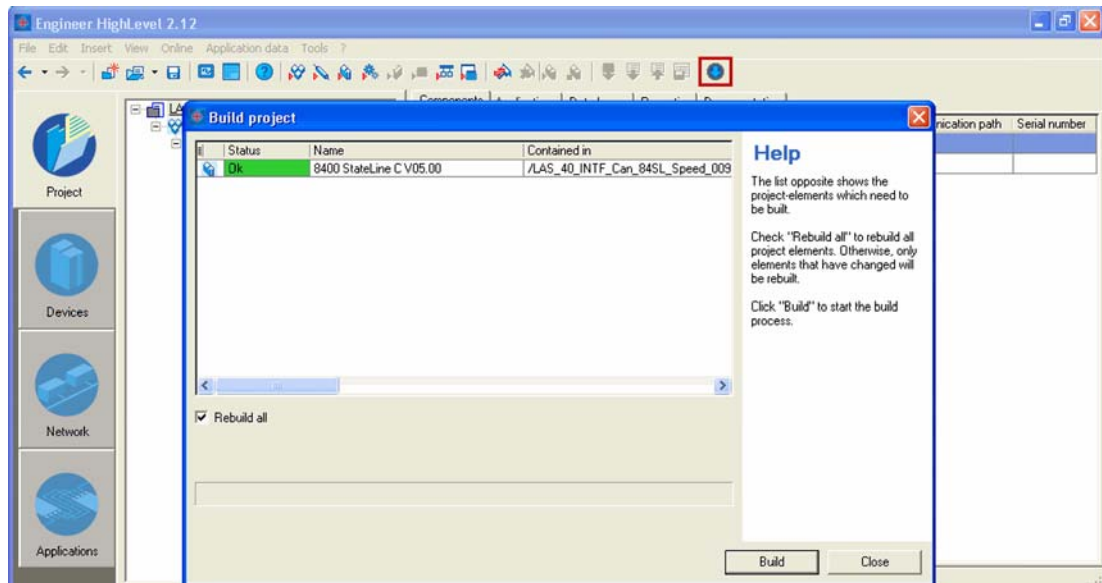
1. Start the »Engineer«.
2. Open the desired sample project (ZIP archive).
 - Double-click the ZIP archive in the **Windows® start menu** under:
Start → All programs → Lenze → AppSamples → ...
- or:
- Menu command: **File → Open archive**
3. Mark the axis (inverter) in the project tree and execute the **Insert → Component** menu command.
4. Select the desired motor in the appearing selection dialog under the "Motors" tab.



5. Insert the selected motor into the project tree by clicking the **Complete** button.
6. The menu commands **Insert → Components** or **Insert → Gearbox** can be used to insert further devices or a gearbox.
7. Mark or remove devices that are not required, either using the delete key or the menu command.

5 Using sample projects in the »Engineer«

8. Update the project with the  button.



- Select the "Rebuild all" option.
- Click the **Build** button to update the project.

9. Go online with the  button.

After a connection to the inverter has been established, the following status is displayed in the status line:



10. Download the parameter set to the device with the  button.

This command has the effect that the application and the parameter settings of the »Engineer« project overwrite the parameter settings in the inverter.

11. Optional: Adapt desired communication settings (e.g. network address, baud rate).
12. Save the parameter set with the  button.
13. Switch on the mains.

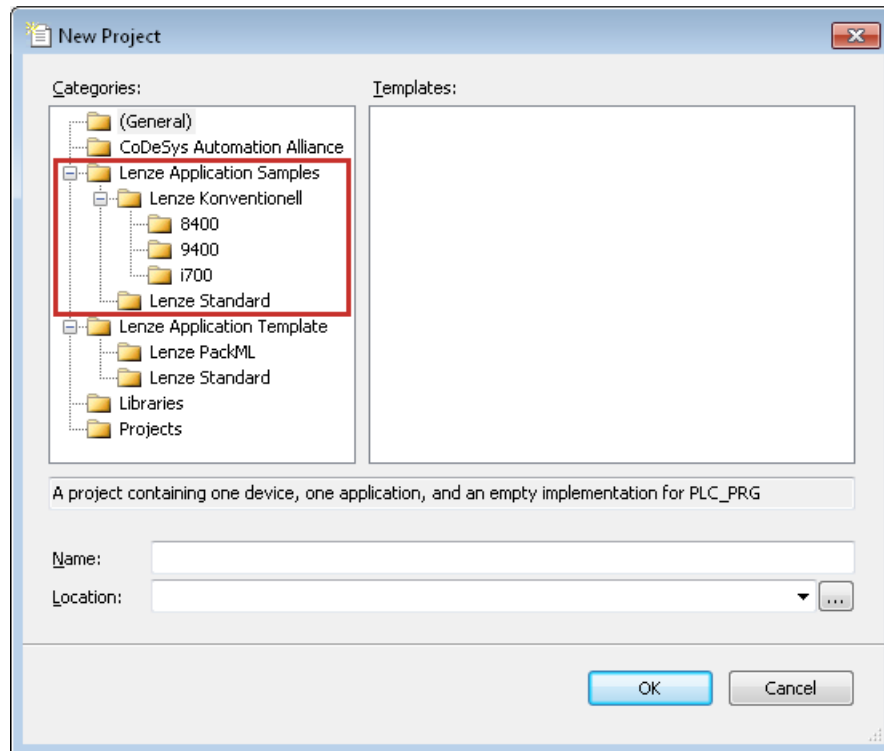
The communication settings are transmitted to the device.

6 Using sample projects in the »PLC Designer«



How to proceed:

1. Start the »PLC Designer«.
2. Create a new project with the **File → New project** menu command.
3. Select a sample project from the folders '8400' or '9400' under the "Lenze Application Samples" category.



4. Open the sample project with the **OK** button.
 - ▶ [Overview of the application samples](#) (20)
 - ▶ [Sample project with the "speed actuating drive" application](#) (27)
 - ▶ [Sample project with the "table positioning" application](#) (28)
5. Extend and adapt the sample projects, depending on the application.
 - ▶ [Extending sample projects \(adding devices\)](#) (43)Adapt ▶ [Program blocks \(PRG\)](#) (29).
6. Compile the PLC program code with the **Build → Compile** menu command or with the **<F11>** function key.
 - If errors occur during the compilation, they can be located and corrected on the basis of the »PLC Designer« error messages.
Afterwards, recompile 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. Use the menu command **Online → Log in** or **<Alt>+<F8>** to log in on the Controller.
 - For this, the PLC program must be error-free.
 - With the log-in, the [Control configuration \(EtherCAT/CAN\)](#) (□ 26) and the PLC program are loaded into the Controller. Any configuration or a PLC program that is possibly available are overwritten.
8. Use the menu command **Debug → Start** or the function key **<F5>** to start the PLC program.



Tip!

Install the PLC program as "**boot project**" in order to load it automatically after the device is restarted.

Menu command: **Online → Generate boot application for L-force Controller**

6.1 Control configuration (EtherCAT/CAN)

EtherCAT

- The EtherCAT master is the control unit of the system configuration.
- The inverter (station ID = 1001) is attached below the EtherCAT master.
- Below the inverter, the axis data is available in the form of an additional node (LC_Drive) via the function block instance [L_LCB_AXIS_REF](#) (📖 67).



Communication manual Controller-based Automation EtherCAT

Here you can find detailed information on the commissioning of EtherCAT components.

CAN

- The CAN master (node ID = 127) is the control unit of the system configuration.
- The inverter (node ID = 5) is attached below the CAN master.
- Below the inverter, the axis data is available in the form of an additional node (LC_Drive) via the function block instance [L_LCB_AXIS_REF](#) (📖 67).
- One PDO for CAN-Input and one PDO for CAN-Output is used, respectively.
- The transmission mode is sync-controlled (10 ms).



Communication manual for Controller-based Automation with CANopen

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

Extension of the control configuration

Further information on how to extend the control configuration can be found here:

- ▶ [Extending sample projects \(adding devices\)](#) (📖 43)

6.2 Sample project with the "speed actuating drive" application

Here, the following function libraries and function blocks are used:

- ▶ [L_LCB_LogicControlBasic function library](#) (📖 65)
- ▶ [L_DCO_DriveCommunication function library](#) (📖 103)
- ▶ [L_LCB_ActuatorSpeed](#) (📖 73)

Overview of the functions

Mode	Description
Automatic	In the automatic mode, a sequence runs over an infinite loop. Selection: 1. Travel 3 seconds with speed '1'. 2. Then travel 5 seconds with speed '2'. 3. Afterwards travel with speed '1' again.
Manual	In manual mode, the inverter can be controlled manually by setting individual control bits (xJog1Set, xJog2Set, xDriveSetQsp, xResetError, ...). The inverter can be controlled in manual mode via the control bits "xJog1Set" and "xJog2Set", for instance, for cleaning or changing the tool.
Service	The service mode serves to adjust the inverter. Selection: 1. Travel 1 second in positive direction. 2. Stop 1 second. 3. Travel 1 second in negative direction.

6.3 Sample project with the "table positioning" application

Here, the following function libraries and function blocks are used:

- ▶ [L_LCB_LogicControlBasic function library](#) (📖 65)
- ▶ [L_DCO_DriveCommunication function library](#) (📖 103)
- ▶ [L_LCB_TablePositioning](#) (📖 77)
- ▶ [L_LCB_SetAxisData](#) (📖 90)

Overview of the functions

Mode	Description
Automatic	In the automatic mode, a sequence runs over an infinite loop. Selection: When the drive is referenced via the home mode (see below) ... 1.travel to position '1', 2.travel to position '2', 3.travel to position '1' again.
Manual	In manual mode, the inverter can be controlled manually by setting individual control bits (xManualPos, xManualNeg, xDriveSetQsp, xResetError, ...). The inverter can be controlled in manual mode via the control bits "xManualPos" and "xManualNeg", for instance, for cleaning or changing the tool.
Service	The service mode serves to adjust the inverter. Selection: 1.Travel 1 second in positive direction. 2.Stop 1 second. 3.Travel 1 second in negative direction.
Homing	In homing mode, the drive is referenced. Either the home position is set directly or a homing is started.

6.4

Program blocks (PRG)

Main (PRG)
The "Main (PRG)" program contains the basic program for processing the program sequences: initialisation of the inverter and selection of the different modes.

Automatic (PRG) / Home (PRG) / Manual (PRG) / Service (PRG)
Here, the programs for the single modes of the application are contained.

- Home (PRG) not in case of "speed actuating drive"!

Drive (PRG)

- Depending on the sample project, the "Drive (PRG)" program controls the inverter with the function block [L_LCB_ActuatorSpeed](#) (□ 73) or [L_LCB_TablePositioning](#) (□ 77).
- Axis data can be accessed via the function block instance [L_LCB_AXIS_REF](#) (□ 67).
- For "table positioning", prepared machine constants are made available via the [L_LCB_SetAxisData](#) (□ 90) function block. – Not in case of "speed actuating drive"!

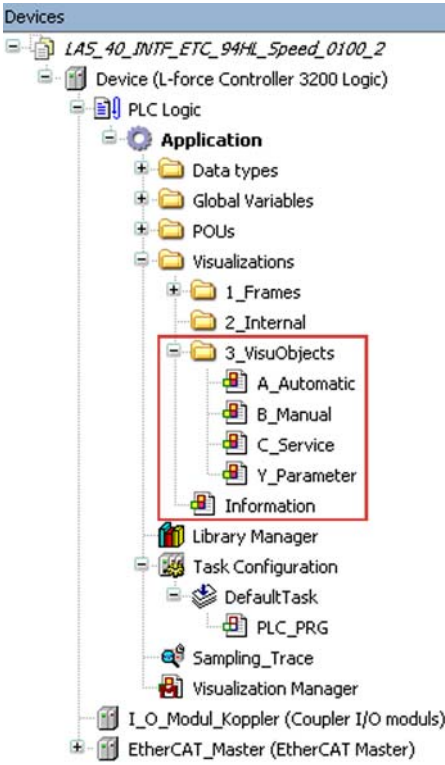
Parameter_Transfer (PRG)
The "Parameter_Transfer (PRG)" program contains calls of the parameter function blocks [L_DCO_ReadDriveParameter](#) (□ 106) and [L_DCO_WriteDriveParameter](#) (□ 112) which serve to read or write single parameters.

PLC_PRG (PRG)
The "PLC_PRG (PRG)" program contains all program calls.

[6-1] **Sample project:** Table positioning - Servo Drive 9400 HighLine - EtherCAT

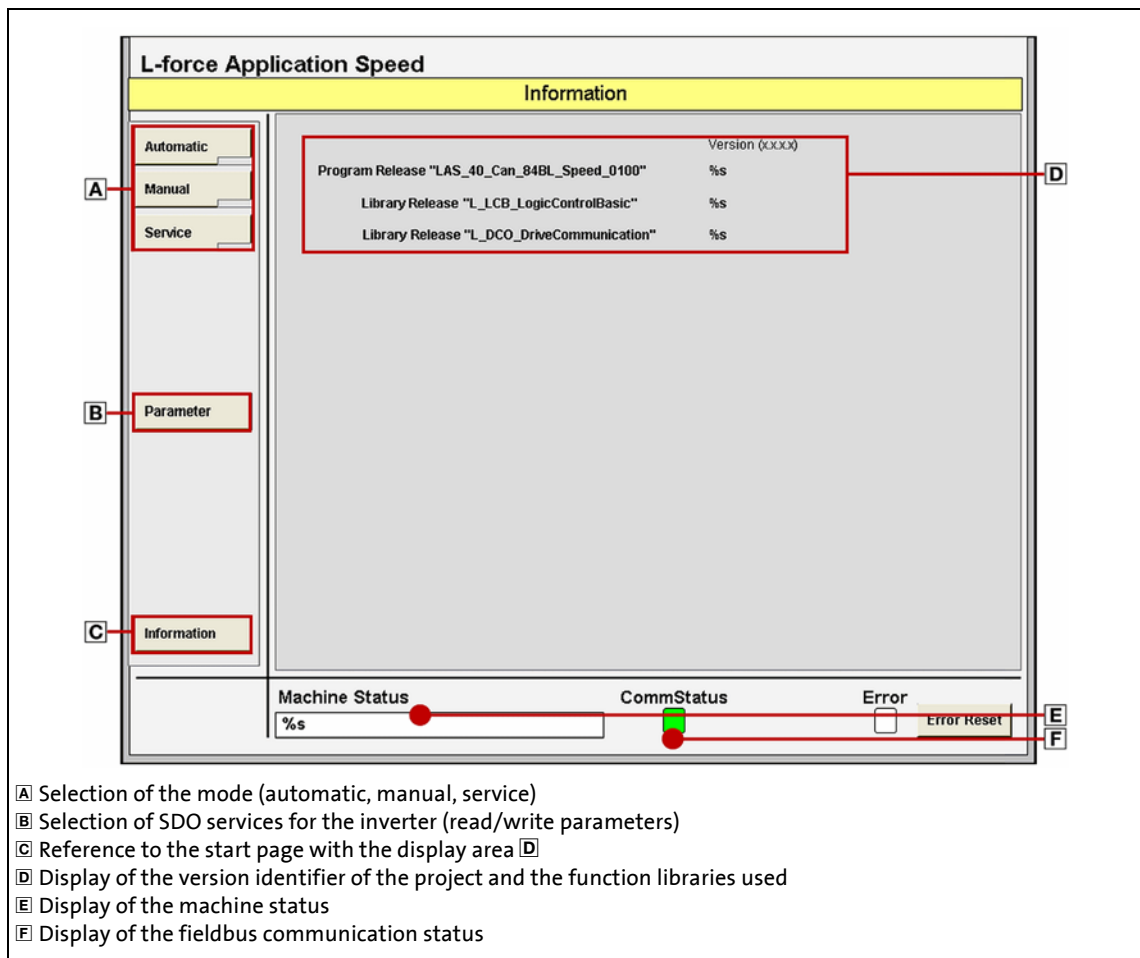
6.5 Visualisations of the "speed actuating drive" application

The sample project with the "speed actuating drive" application contains visualisations for operating, setting and displaying parameters and for displaying errors.

	"Information" (start page) provides general information on the sample project. In addition, there are visualisations for the following operating modes: • Automatic • Manual • Service "Y_Parameter" is the visualisation of the parameter function blocks for reading or writing single parameters.
--	---

[6-2] Sample project: Speed actuating drive - Servo Drive 9400 HighLine - EtherCAT

6.5.1 Information (start page)



[6-3] Start page of the visualisation in the "speed actuating drive" application

Display of error messages

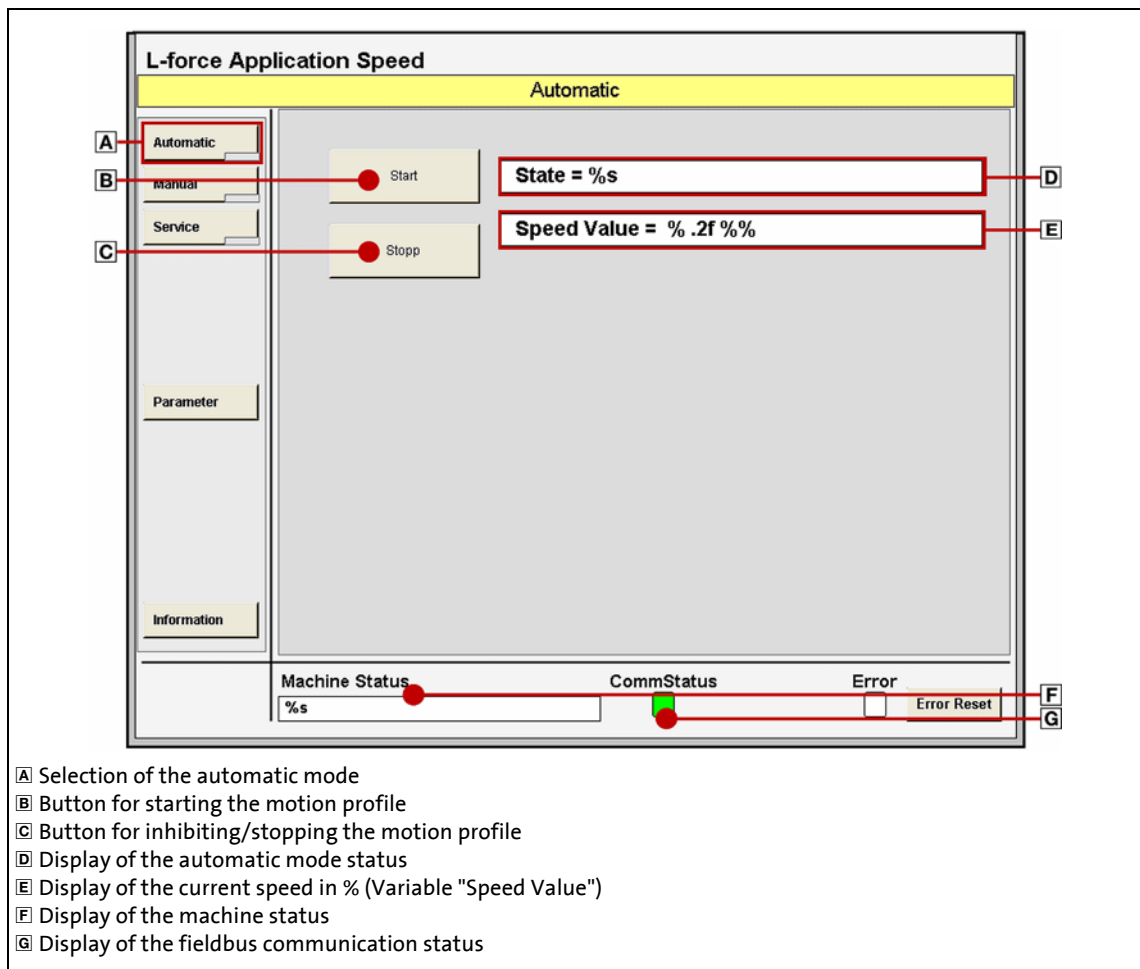
The **CommStatus** (F) control field shows the fieldbus communication status.

In case of a faulty fieldbus communication, the Controller and the inverters have to be restarted since the fieldbus communication is not reset automatically.

The **Error** control field shows errors in the application and errors in the inverter.

The **Error Reset** button resets the error messages.

6.5.2 Automatic mode



[6-4] Visualisation of the automatic mode in the "speed actuating drive" application

Display of error messages

The **CommStatus** (G) control field shows the fieldbus communication status.

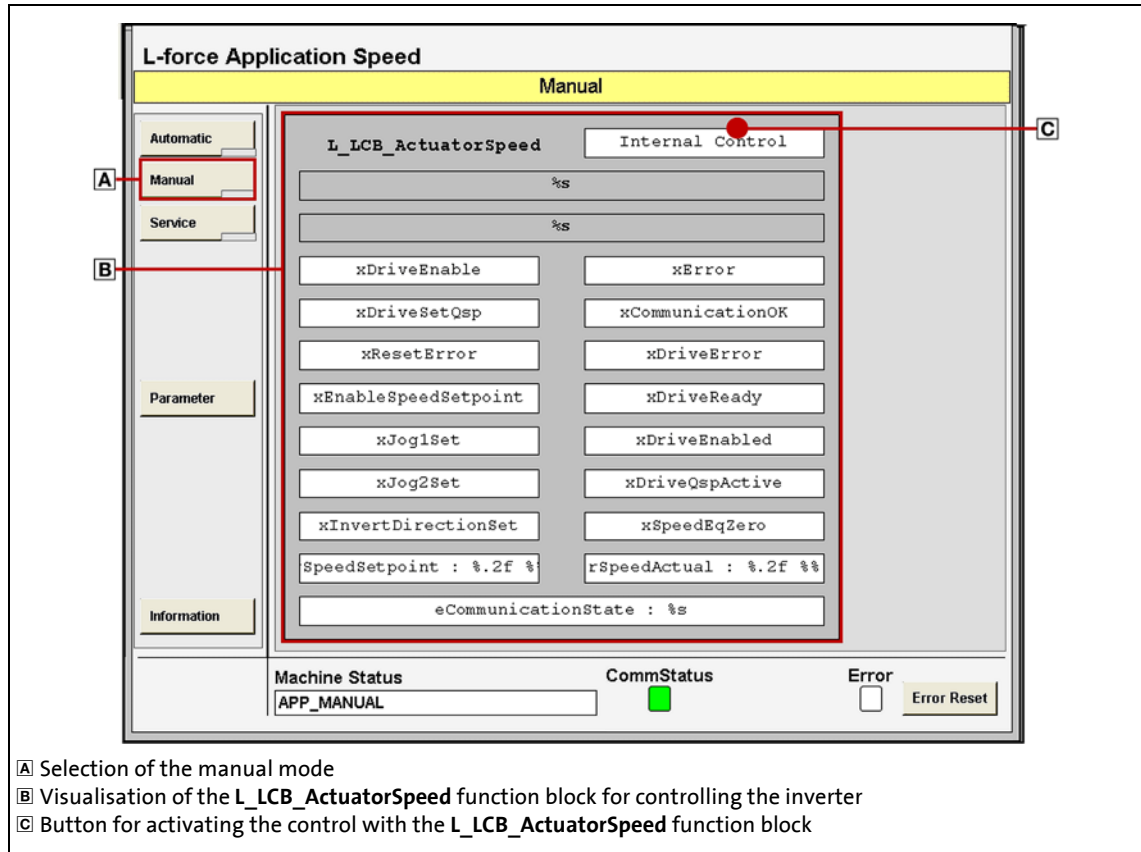
In case of a faulty fieldbus communication, the Controller and the inverters have to be restarted since the fieldbus communication is not reset automatically.

The **Error** control field shows errors in the application and errors in the inverter.

The **Error Reset** button resets the error messages.

6.5.3 Manual mode

The inverter can be controlled via the visualisation of the [L_LCB_ActuatorSpeed](#) (73) function block in manual mode.

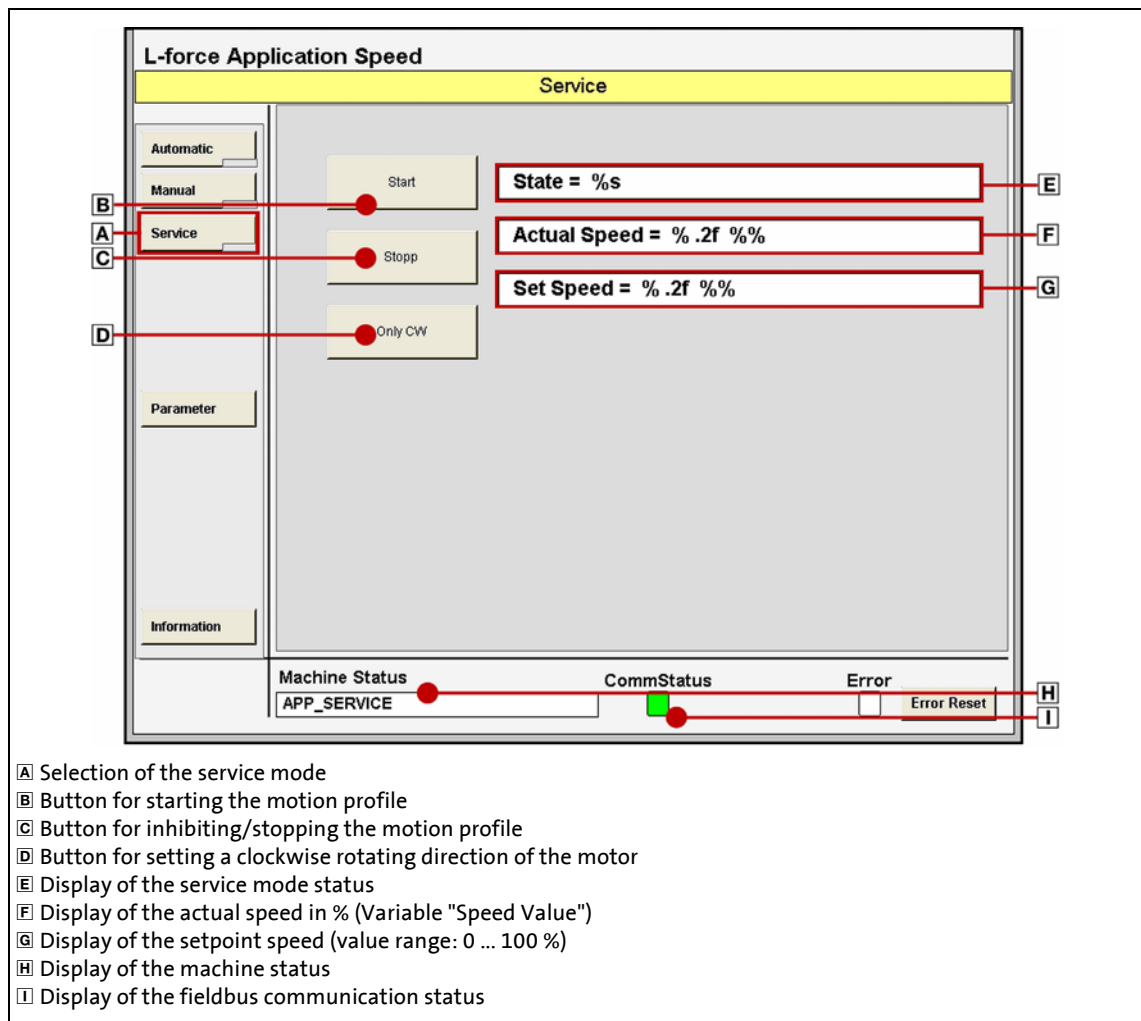


[6-5] Visualisation of the manual mode in the "speed actuating drive" application

- Activate the **Internal Control** (C) button to operate the inverter via the visualisation.
- Enable the inverter via the **xDriveEnable** input to be able to define setpoints manually.
- Click the **xDriveSetQsp** button to activate quick stop (QSP).

6.5.4

Service mode



[6-6] Visualisation of the service mode in the "speed actuating drive" application

Display of error messages

The **CommStatus** (I) control field shows the fieldbus communication status.

In case of a faulty fieldbus communication, the Controller and the inverters have to be restarted since the fieldbus communication is not reset automatically.

The **Error** control field shows errors in the application and errors in the inverter.

The **Error Reset** button resets the error messages.

6.5.5 Parameter

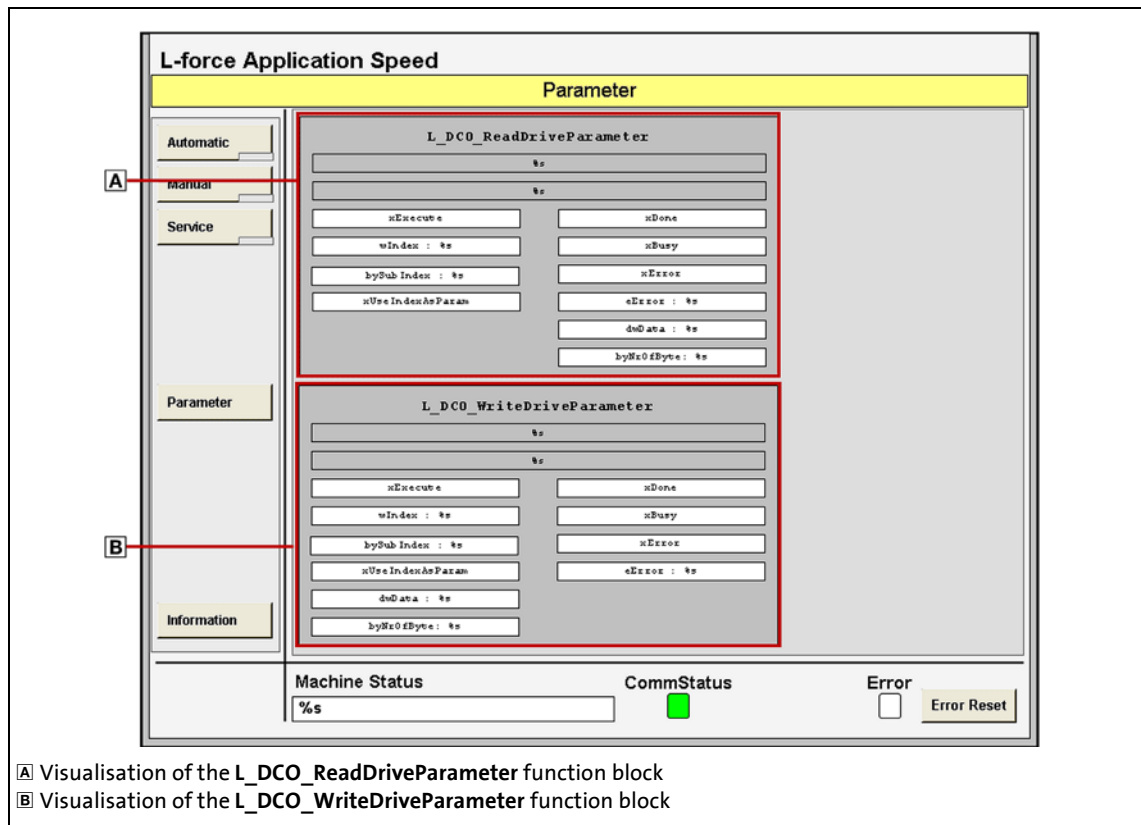
The **Parameter** button serves to open the visualisation of the parameter function blocks:

▶ [L_DCO_ReadDriveParameter](#) (106)

▶ [L_DCO_WriteDriveParameter](#) (112)

These function blocks serve to read or write single parameters.

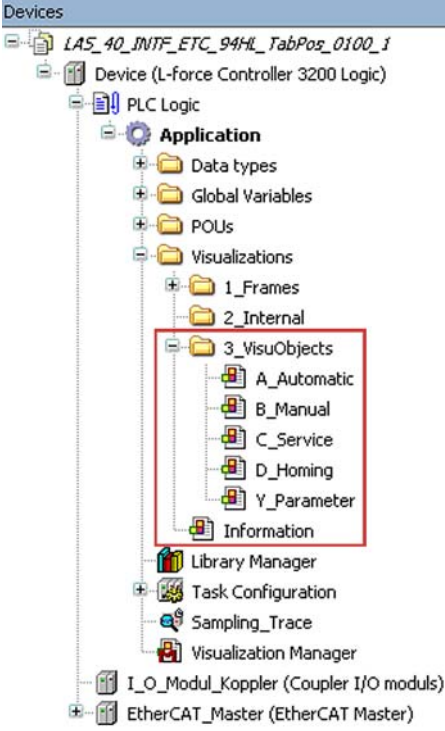
The mode selected last remains active.



[6-7] Visualisation of the parameter function blocks in the "speed actuating drive" application

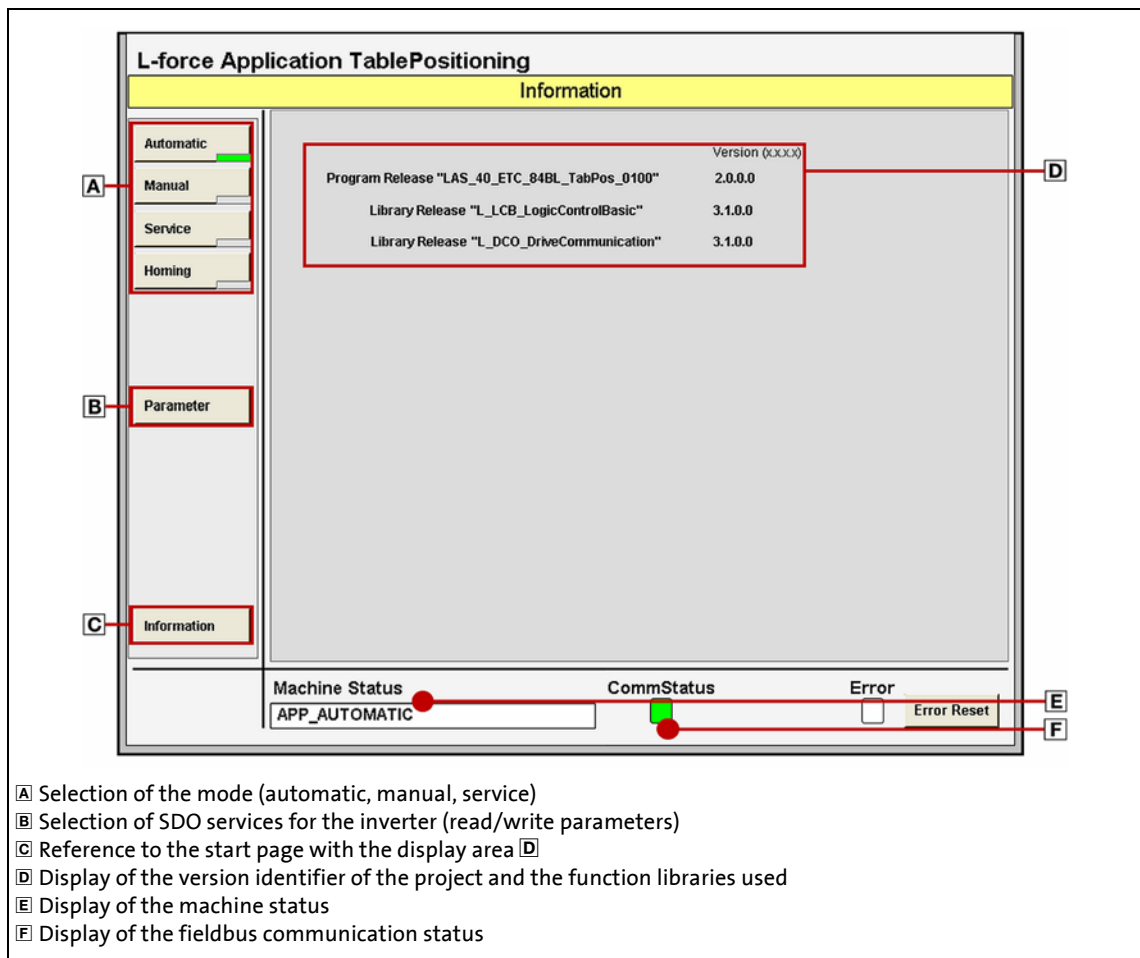
6.6 Visualisations of the "table positioning" application

The sample project with the "table positioning" application contains visualisations for operating, setting and displaying parameters and for displaying errors.

	"Information" (start page) provides general information on the sample project. In addition, there are visualisations for the following operating modes: • Automatic • Manual • Service • Homing "Y_Parameter" is the visualisation of the parameter function blocks for reading or writing single parameters.
--	---

[6-8] **Sample project:** Table positioning - Servo Drive 9400 HighLine - EtherCAT

6.6.1 Information (start page)



[6-9] Start page of the visualisation in the "table positioning" application

Display of error messages

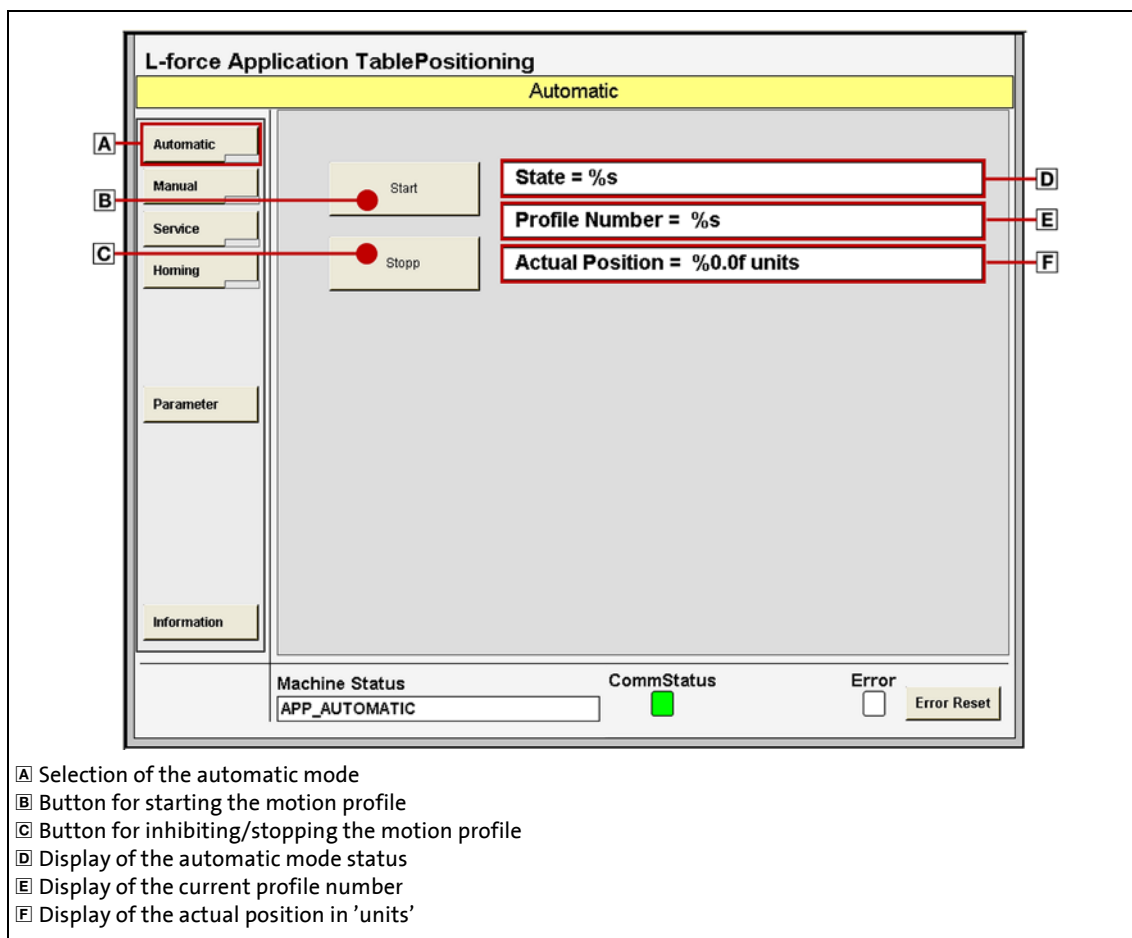
The **CommStatus** (F) control field shows the fieldbus communication status.

In case of a faulty fieldbus communication, the Controller and the inverters have to be restarted since the fieldbus communication is not reset automatically.

The **Error** control field shows errors in the application and errors in the inverter.

The **Error Reset** button resets the error messages.

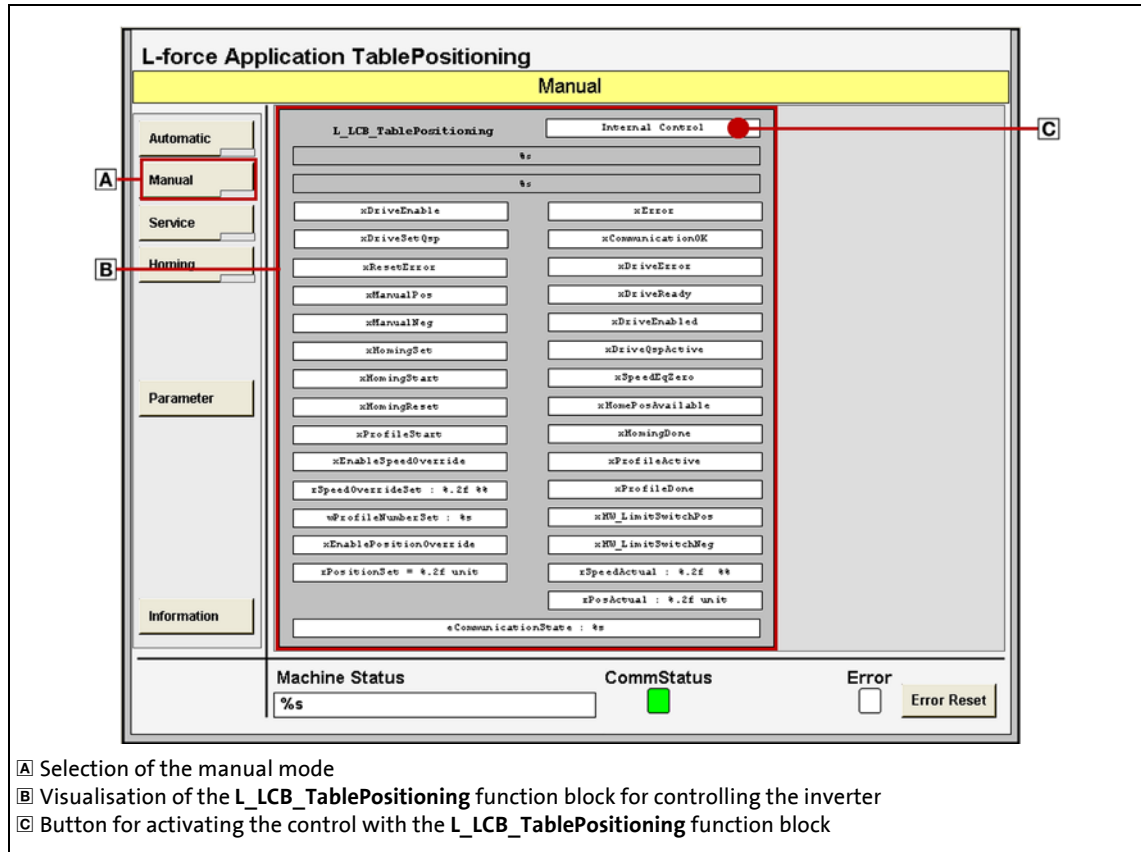
6.6.2 Automatic mode



[6-10] Visualisation of the automatic mode in the "table positioning" application

6.6.3 Manual mode

The inverter can be controlled manually via the visualisation of the **L_LCB_TablePositioning** function block.



[6-11] Visualisation of the manual mode in the "table positioning" application

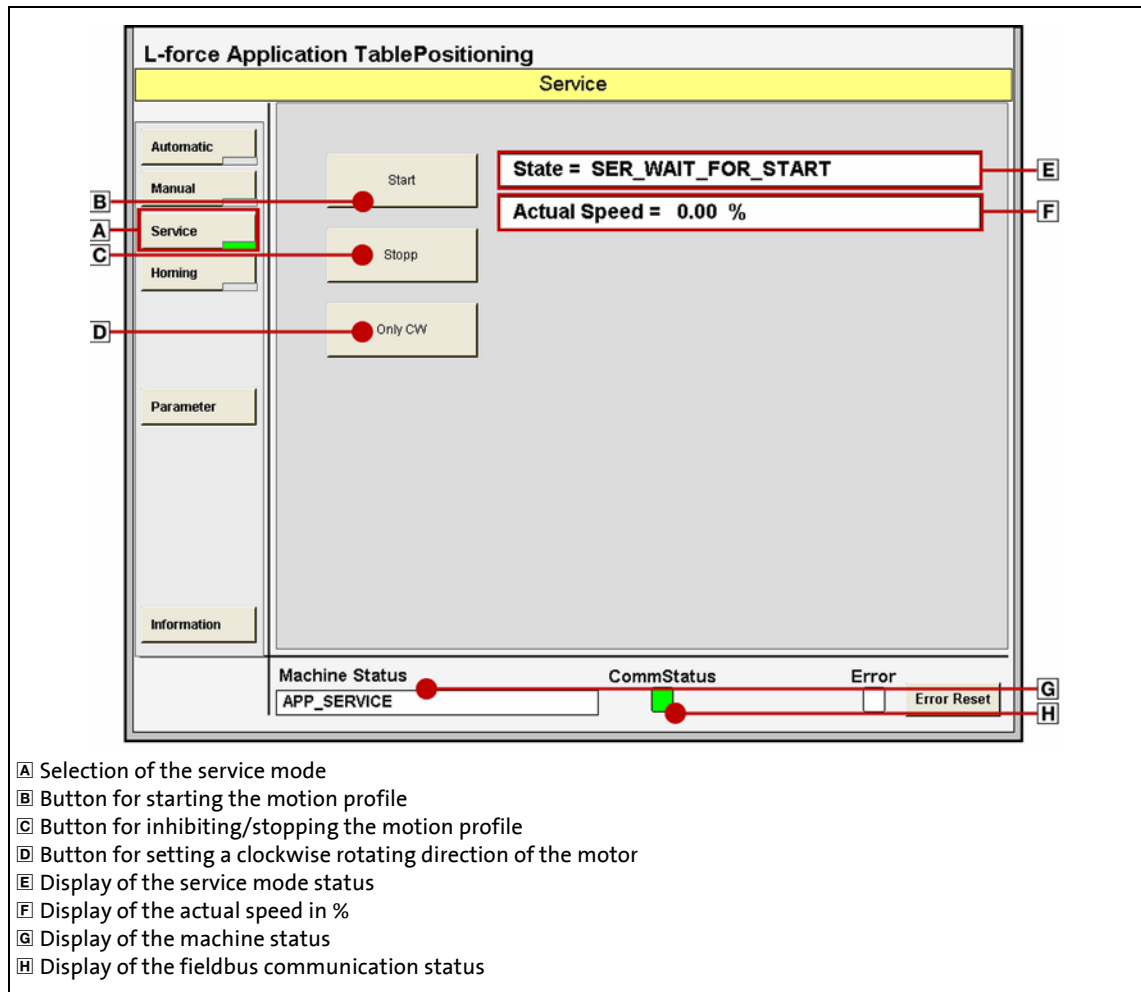
- Activate the **Internal Control** (C) button to operate the inverter via the visualisation.
- Enable the inverter via the **xDriveEnable** input to be able to define setpoints manually.
- Click the **xDriveSetQsp** button to activate quick stop (QSP).

Detailed information on the **L_LCB_TablePositioning** function block can be found here:

► [L_LCB_TablePositioning](#) (73)

6.6.4

Service mode



[6-12] Visualisation of the service mode in the "table positioning" application

Display of error messages

The **CommStatus** (H) control field shows the fieldbus communication status.

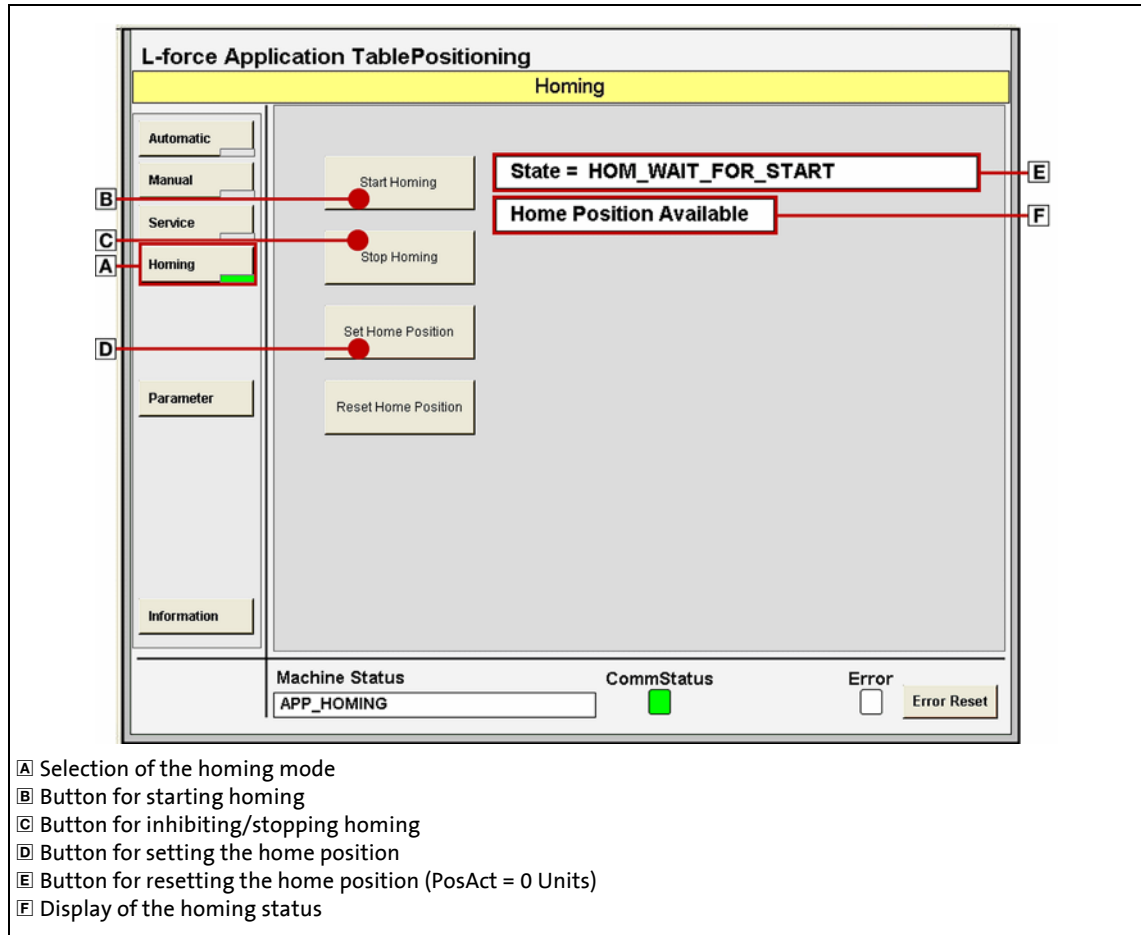
In case of a faulty fieldbus communication, the Controller and the inverters have to be restarted since the fieldbus communication is not reset automatically.

The **Error** control field shows errors in the application and errors in the inverter.

The **Error Reset** button resets the error messages.

6.6.5 Homing mode

The active Lenze standard setting in the inverter is homing mode '12' (travel to negative limit switch / digital input 3).



[6-13] Visualisation of the homing mode in the "table positioning" application

6.6.6 Parameter

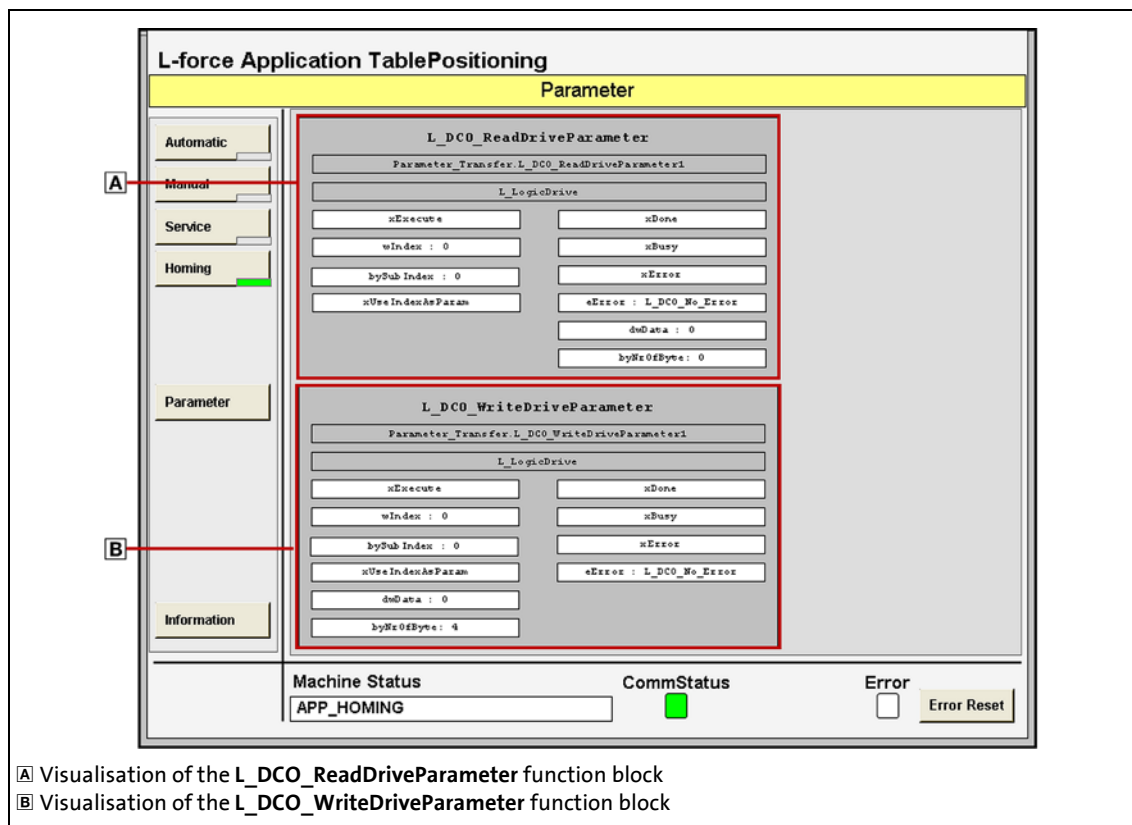
The **Parameter** button serves to open the visualisation of the parameter function blocks:

▶ [L_DCO_ReadDriveParameter](#) (106)

▶ [L_DCO_WriteDriveParameter](#) (112)

These function blocks serve to read or write single parameters.

The mode selected last remains active.



[6-14] Visualisation of the parameter function blocks in the "table positioning" application

6.7 Extending sample projects (adding devices)

This chapter informs you on how to extend the control configuration manually.

Follow the procedure described in the following sections to individually extend an existing sample project.

Adapt the communication settings for each individual node.

6.7.1 Adding EtherCAT field devices



Note!

- 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.



Tip!

Feldbus-Scan

The »PLC Designer« provides a "fieldbus scan" that automatically detects the devices connected to the fieldbus. For this purpose, go to the context menu of the EtherCAT master and execute the **Start Search** command.

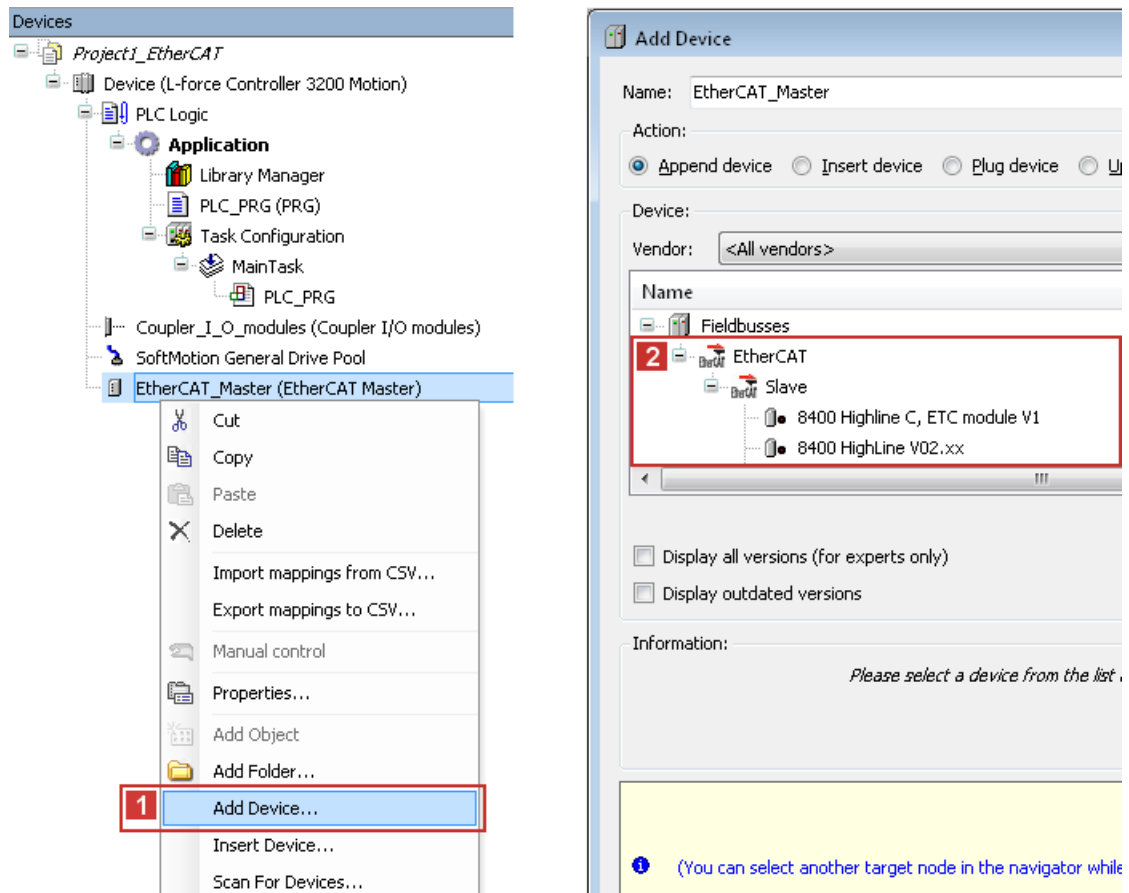
Missing devices or devices of other manufacturers

In order to integrate missing devices or devices of other manufacturers, the corresponding device description files of the manufacturer are required. In the »PLC Designer«, device description files of *.XML, *.devdesc.XML, *.EDS, *.DCF and *.GSx type can be imported via the **Tools → Device repository...** menu command.



How to add EtherCAT field devices manually:

1. Use the **1 Add device** command in the context menu to add an EtherCAT slave.



Select a field device from the **2** selection list. You can only select devices the EtherCAT device description files of which have been imported in the »PLC Designer«.

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

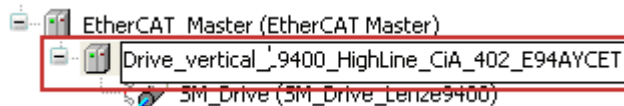
2. Give the inserted slaves suitable names (e.g. "Drive_vertical").

The names must ...

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

You can enter a name by clicking the element.

Example:



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

Below the inverter, the axis data is available in the form of an additional node (LC_Drive) via the function block instance [L_LCB_AXIS_REF](#) (67).

3. Connect a function block with the axis data instance **LC_Drive** from the control configuration in order to establish a connection to the inverter.

Transmitted process data are written automatically to the axis data. Hence, no manual assignment of the process data to the application is required.

▶ [Connecting a function block to an inverter](#) (65)



Communication manual "Controller-based Automation EtherCAT"

Observe further information on commissioning:

- "Fieldbus scan" / import of device description files
- Synchronisation with "Distributed Clocks"
- Settings of cycle times and motion parameters
- EtherCAT I/O mapping
- Modular machine configuration (from release 3.10)
- **L_IODrvEtherCAT** function library
- Mixed operation - EtherCAT with other bus systems
- EtherCAT diagnostics

6.7.2

Adding CAN field devices

**Note!**

The configuration of a Lenze Controller in a CANopen network must be created in the »PLC Designer«, because the complete configuration is written to the connected slaves when a controller is started. This process overwrites the previous slave settings.

**Tip!****Missing devices or devices of other manufacturers**

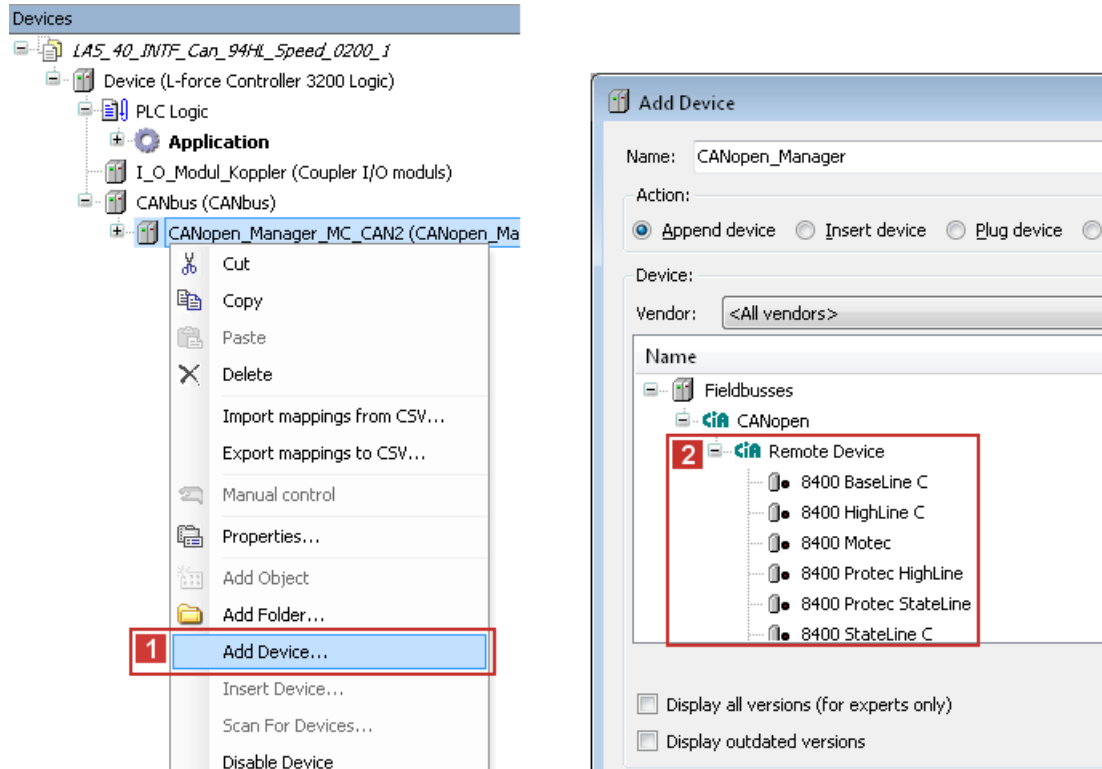
In order to integrate missing devices or devices of other manufacturers, the corresponding device description files of the manufacturer are required. In the »PLC Designer«, device description files of *.XML, *.devdesc.XML, *.EDS, *.DCF and *.GSx type can be imported via the **Tools → Device repository...** menu command.

In addition, the general device description "[Lenze Generic Drive](#)" ([book 63](#)) is available for integrating other Lenze devices. Control takes place via the function block [L_LCB_GenericDrive](#) ([book 72](#)).



How to add the CAN field devices:

1. Go to the context menu of the "CANopen Manager" and use the **1 Add device** command to insert a logic device (slave) below the "CANopen Manager" (master).



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

For EDS files created in the »Engineer«, the field device appears in the selection list with the same name as during the export of the EDS file in the »Engineer«, extended by the name of the interface and device type.

2. Repeat the **1 Add Device** command until all slaves connected to the fieldbus are included in the control configuration.

As an alternative, you can **copy** and **insert** an already inserted node via the context menu of the device.

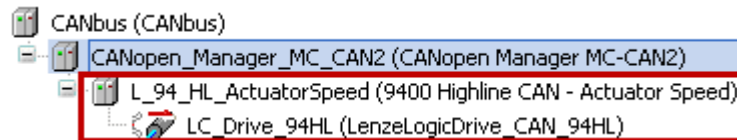
3. Give the inserted slaves suitable names (e.g. "Drive_vertical").

The names must ...

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

You can enter a name by clicking the element.

Example:



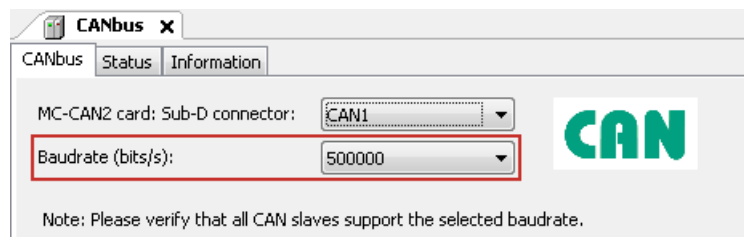
Below the inverter, the axis data is available in the form of an additional node (LC_Drive) via the function block instance [L_LCB_AXIS_REF](#) (67).

4. Connect a function block with the axis data instance **LC_Drive** from the control configuration in order to establish a connection to the inverter.

Transmitted process data are written automatically to the axis data. Hence, no manual assignment of the process data to the application is required.

► [Connecting a function block to an inverter](#) (65)

5. Use the **CANbus** tab to set the baud rate.

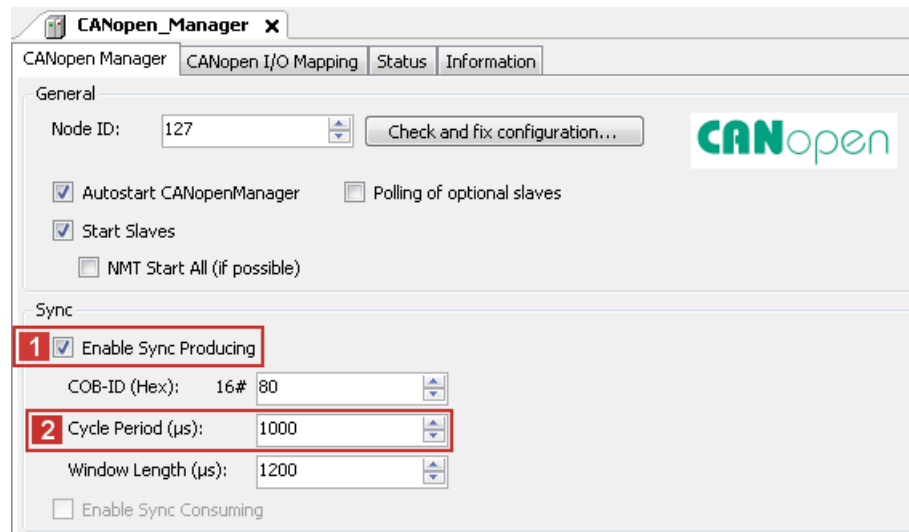


Note!

The baud rate set in the »PLC Designer« overwrites the baud rate set for the field devices via »Engineer« or »EASY Starter«.

In a CANopen network, set the same baud rate for all nodes.

6. Use the **CANopen_Manager** tab to set the parameters for Sync generation.



The sync producing is required if ...

- at least one PDO with sync-controlled processing is used on the bus;
- the applications are to run in synchronism in several field devices;
- Motion devices are to be operated on the fieldbus.

If you want to use CAN synchronisation, tick the input field **1 Enable Sync Producing**.

Go to the input field and set the **2 Cycle Period**.



Communication manual for "Controller-based Automation CANopen"

Observe further information on commissioning:

- Import of device description files
- Settings of CAN parameters and PDO mapping
- Special features of Inverter Drives 8400, Servo Drives 9400 and I/O system 1000
- Mixed operation of CANopen with other bus systems
- CAN diagnostics

7 Device descriptions

This chapter contains information on the device descriptions of the Inverter Drives 8400 and Servo Drives 9400 available in the »PLC Designer«.

If an inverter is inserted into the control configuration, the axis data is available below the device in the form of an additional node (LC_Drive) via the function block instance [L_LCB_AXIS_REF](#) (67).

Information on how to extend the [Control configuration \(EtherCAT/CAN\)](#) (26) in the »PLC Designer« can be found here:

- ▶ [Extending sample projects \(adding devices\)](#) (43)

7.1 Inverter Drive 8400 BaseLine C



Communication with the Inverter Drives 8400 BaseLine C is only possible via the CAN bus (CANopen).

Available technology applications

- None

Sample project

- LAS_40_INTF_Can_84BL_Speed_0100

Overview of the indices

The received process data are transmitted into the *awReceiveData* element of the [L_LCB_AXIS_REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L_LCB_AXIS_REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LP_CanIn1_wCtrl	0x5E66/01	-	-
awTransmitData2	LP_CanIn1_wIn2	0x5E66/02	-	-
awTransmitData3	LP_CanIn1_wIn3	0x5E66/03	-	-
awTransmitData4	LP_CanIn1_wIn4	0x5E66/04	-	-
awTransmitData5	LP_CanIn2_wIn1	0x5E66/05	-	-
awTransmitData6	LP_CanIn2_wIn2	0x5E66/06	-	-
awTransmitData7	LP_CanIn2_wIn3	0x5E66/07	-	-
awTransmitData8	LP_CanIn2_wIn4	0x5E66/08	-	-
awTransmitData9	-	-	-	-
awTransmitData10	-	-	-	-
awTransmitData11	-	-	-	-
awTransmitData12	-	-	-	-
awTransmitData13	-	-	-	-
awTransmitData14	-	-	-	-
awTransmitData15	-	-	-	-
awTransmitData16	-	-	-	-

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awReceiveData1	LP_CanOut1 _wState	0x5C9B/01	-	-
awReceiveData2	LP_CanOut1 _wOut2	0x5C9B/02	-	-
awReceiveData3	LP_CanOut1 _wOut3	0x5C9B/03	-	-
awReceiveData4	LP_CanOut1 _wOut4	0x5C9B/04	-	-
awReceiveData5	LP_CanOut2 _wOut1	0x5C9B/05	-	-
awReceiveData6	LP_CanOut2 _wOut2	0x5C9B/06	-	-
awReceiveData7	LP_CanOut2 _wOut3	0x5C9B/07	-	-
awReceiveData8	LP_CanOut2 _wOut4	0x5C9B/08	-	-
awReceiveData9	-	-	-	-
awReceiveData10	-	-	-	-
awReceiveData11	-	-	-	-
awReceiveData12	-	-	-	-
awReceiveData13	-	-	-	-
awReceiveData14	-	-	-	-
awReceiveData15	-	-	-	-
awReceiveData16	-	-	-	-

7.2

Inverter Drive 8400 StateLine C



Communication with the Inverter Drive 8400 StateLine C is possible via the CAN bus (CANopen) and EtherCAT.

Available technology applications

- Speed actuating drive
- Switch-off positioning without feedback

Sample projects

- LAS_40_INTF_Can_84SL_Speed_0200
- LAS_40_INTF_ETC_84SL_Speed_0100

Overview of the indices

The received process data are transmitted into the *awReceiveData* element of the [L LCB AXIS REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L LCB AXIS REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LP_CanIn1_wCtrl	0x5E66/01	MciIn_wCtrl	0x5C93/01
awTransmitData2	LP_CanIn1_wIn2	0x5E66/02	MciIn_wIn2	0x5C93/02
awTransmitData3	LP_CanIn1_wIn3	0x5E66/03	MciIn_wIn3	0x5C93/03
awTransmitData4	LP_CanIn1_wIn4	0x5E66/04	MciIn_wIn4	0x5C93/04
awTransmitData5	LP_CanIn2_wIn1	0x5E66/05	MciIn_wIn5	0x5C93/05
awTransmitData6	LP_CanIn2_wIn2	0x5E66/06	MciIn_wIn6	0x5C93/06
awTransmitData7	LP_CanIn2_wIn3	0x5E66/07	MciIn_wIn7	0x5C93/07
awTransmitData8	LP_CanIn2_wIn4	0x5E66/08	MciIn_wIn8	0x5C93/08
awTransmitData9	LP_CanIn3_wIn1	0x5E66/09	MciIn_wIn9	0x5C93/09
awTransmitData10	LP_CanIn3_wIn2	0x5E66/0A	MciIn_wIn10	0x5C93/0A
awTransmitData11	LP_CanIn3_wIn3	0x5E66/0B	MciIn_wIn11	0x5C93/0B
awTransmitData12	LP_CanIn3_wIn4	0x5E66/0C	MciIn_wIn12	0x5C93/0C
awTransmitData13	-	-	MciIn_wIn13	0x5C93/0D
awTransmitData14	-	-	MciIn_wIn14	0x5C93/0E
awTransmitData15	-	-	MciIn_wIn15	0x5C93/0F
awTransmitData16	-	-	MciIn_wIn16	0x5C93/10
awReceiveData1	LP_CanOut1_wState	0x5C9B/01	MciOut_wState	0x5C92/01
awReceiveData2	LP_CanOut1_wOut2	0x5C9B/02	MciOut_wOut2	0x5C92/02
awReceiveData3	LP_CanOut1_wOut3	0x5C9B/03	MciOut_wOut3	0x5C92/03
awReceiveData4	LP_CanOut1_wOut4	0x5C9B/04	MciOut_wOut4	0x5C92/04
awReceiveData5	LP_CanOut2_wOut1	0x5C9B/05	MciOut_wOut5	0x5C92/05

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awReceiveData6	LP_CanOut2_wOut2	0x5C9B/06	MciOut_wOut6	0x5C92/06
awReceiveData7	LP_CanOut2_wOut3	0x5C9B/07	MciOut_wOut7	0x5C92/07
awReceiveData8	LP_CanOut2_wOut4	0x5C9B/08	MciOut_wOut8	0x5C92/08
awReceiveData9	LP_CanOut3_wOut1	0x5C9B/09	MciOut_wOut9	0x5C92/09
awReceiveData10	LP_CanOut3_wOut2	0x5C9B/0A	MciOut_wOut10	0x5C92/0A
awReceiveData11	LP_CanOut3_wOut3	0x5C9B/0B	MciOut_wOut11	0x5C92/0B
awReceiveData12	LP_CanOut3_wOut4	0x5C9B/0C	MciOut_wOut12	0x5C92/0C
awReceiveData13	-	-	MciOut_wOut13	0x5C92/0D
awReceiveData14	-	-	MciOut_wOut14	0x5C92/0E
awReceiveData15	-	-	MciOut_wOut15	0x5C92/0F
awReceiveData16	-	-	MciOut_wOut16	0x5C92/10

7.3

Inverter Drive 8400 HighLine C



Communication with the Inverter Drive 8400 HighLine C is possible via the CAN bus (CANopen) and EtherCAT.

Available technology applications

- Speed actuating drive
- Switch-off positioning without feedback
- Table positioning without feedback

Sample projects

- LAS_40_INTF_Can_84HL_TabPos_0200
- LAS_40_INTF_ETC_84HL_TabPos_0100

Overview of the indices

The received process data are transmitted into the *awReceiveData* element of the [L_LCB_AXIS_REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L_LCB_AXIS_REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LP_CanIn1_wCtrl	0x5E66/01	MciIn_wCtrl	0x5C93/01
awTransmitData2	LP_CanIn1_wIn2	0x5E66/02	MciIn_wIn2	0x5C93/02
awTransmitData3	LP_CanIn1_wIn3	0x5E66/03	MciIn_wIn3	0x5C93/03
awTransmitData4	LP_CanIn1_wIn4	0x5E66/04	MciIn_wIn4	0x5C93/04
awTransmitData5	LP_CanIn2_wIn1	0x5E66/05	MciIn_wIn5	0x5C93/05
awTransmitData6	LP_CanIn2_wIn2	0x5E66/06	MciIn_wIn6	0x5C93/06
awTransmitData7	LP_CanIn2_wIn3	0x5E66/07	MciIn_wIn7	0x5C93/07
awTransmitData8	LP_CanIn2_wIn4	0x5E66/08	MciIn_wIn8	0x5C93/08
awTransmitData9	LP_CanIn3_wIn1	0x5E66/09	MciIn_wIn9	0x5C93/09
awTransmitData10	LP_CanIn3_wIn2	0x5E66/0A	MciIn_wIn10	0x5C93/0A
awTransmitData11	LP_CanIn3_wIn3	0x5E66/0B	MciIn_wIn11	0x5C93/0B
awTransmitData12	LP_CanIn3_wIn4	0x5E66/0C	MciIn_wIn12	0x5C93/0C
awTransmitData13	-	-	MciIn_wIn13	0x5C93/0D
awTransmitData14	-	-	MciIn_wIn14	0x5C93/0E
awTransmitData15	-	-	MciIn_wIn15	0x5C93/0F
awTransmitData16	-	-	MciIn_wIn16	0x5C93/10
awReceiveData1	LP_CanOut1_wState	0x5C9B/01	MciOut_wState	0x5C92/01
awReceiveData2	LP_CanOut1_wOut2	0x5C9B/02	MciOut_wOut2	0x5C92/02
awReceiveData3	LP_CanOut1_wOut3	0x5C9B/03	MciOut_wOut3	0x5C92/03
awReceiveData4	LP_CanOut1_wOut4	0x5C9B/04	MciOut_wOut4	0x5C92/04

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awReceiveData5	LP_CanOut2_wOut1	0x5C9B/05	MciOut_wOut5	0x5C92/05
awReceiveData6	LP_CanOut2_wOut2	0x5C9B/06	MciOut_wOut6	0x5C92/06
awReceiveData7	LP_CanOut2_wOut3	0x5C9B/07	MciOut_wOut7	0x5C92/07
awReceiveData8	LP_CanOut2_wOut4	0x5C9B/08	MciOut_wOut8	0x5C92/08
awReceiveData9	LP_CanOut3_wOut1	0x5C9B/09	MciOut_wOut9	0x5C92/09
awReceiveData10	P_CanOut3_wOut2	0x5C9B/0A	MciOut_wOut10	0x5C92/0A
awReceiveData11	P_CanOut3_wOut3	0x5C9B/0B	MciOut_wOut11	0x5C92/0B
awReceiveData12	P_CanOut3_wOut4	0x5C9B/0C	MciOut_wOut12	0x5C92/0C
awReceiveData13	-	-	MciOut_wOut13	0x5C92/0D
awReceiveData14	-	-	MciOut_wOut14	0x5C92/0E
awReceiveData15	-	-	MciOut_wOut15	0x5C92/0F
awReceiveData16	-	-	MciOut_wOut16	0x5C92/10

7.4

Inverter Drive 8400 TopLine C



Communication with the Inverter Drive 8400 TopLine C is possible via the CAN bus (CANopen) and EtherCAT.

Available technology applications

- Speed actuating drive
- Switch-off positioning without feedback
- Table positioning without feedback (with sequence positioning)

Advanced functions

Function blocks for ...

- Positioning sequence control
- Electrical shaft (speed and angular synchronism)
- Dancer control
- Mains failure control

Sample projects

- None

Overview of the indices

The received process data are transmitted into the *awReceiveData* element of the [L_LCB_AXIS_REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L_LCB_AXIS_REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LP_CanIn1_wCtrl	0x5E66/01	MciIn_wCtrl	0x5C93/01
awTransmitData2	LP_CanIn1_wIn2	0x5E66/02	MciIn_wIn2	0x5C93/02
awTransmitData3	LP_CanIn1_wIn3	0x5E66/03	MciIn_wIn3	0x5C93/03
awTransmitData4	LP_CanIn1_wIn4	0x5E66/04	MciIn_wIn4	0x5C93/04
awTransmitData5	LP_CanIn2_wIn1	0x5E66/05	MciIn_wIn5	0x5C93/05
awTransmitData6	LP_CanIn2_wIn2	0x5E66/06	MciIn_wIn6	0x5C93/06
awTransmitData7	LP_CanIn2_wIn3	0x5E66/07	MciIn_wIn7	0x5C93/07
awTransmitData8	LP_CanIn2_wIn4	0x5E66/08	MciIn_wIn8	0x5C93/08
awTransmitData9	LP_CanIn3_wIn1	0x5E66/09	MciIn_wIn9	0x5C93/09
awTransmitData10	LP_CanIn3_wIn2	0x5E66/0A	MciIn_wIn10	0x5C93/0A
awTransmitData11	LP_CanIn3_wIn3	0x5E66/0B	MciIn_wIn11	0x5C93/0B
awTransmitData12	LP_CanIn3_wIn4	0x5E66/0C	MciIn_wIn12	0x5C93/0C
awTransmitData13	-	-	MciIn_wIn13	0x5C93/0D
awTransmitData14	-	-	MciIn_wIn14	0x5C93/0E
awTransmitData15	-	-	MciIn_wIn15	0x5C93/0F
awTransmitData16	-	-	MciIn_wIn16	0x5C93/10
awReceiveData1	LP_CanOut1_wState	0x5C9B/01	MciOut_wState	0x5C92/01

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awReceiveData2	LP_CanOut1_wOut2	0x5C9B/02	MciOut_wOut2	0x5C92/02
awReceiveData3	LP_CanOut1_wOut3	0x5C9B/03	MciOut_wOut3	0x5C92/03
awReceiveData4	LP_CanOut1_wOut4	0x5C9B/04	MciOut_wOut4	0x5C92/04
awReceiveData5	LP_CanOut2_wOut1	0x5C9B/05	MciOut_wOut5	0x5C92/05
awReceiveData6	LP_CanOut2_wOut2	0x5C9B/06	MciOut_wOut6	0x5C92/06
awReceiveData7	LP_CanOut2_wOut3	0x5C9B/07	MciOut_wOut7	0x5C92/07
awReceiveData8	LP_CanOut2_wOut4	0x5C9B/08	MciOut_wOut8	0x5C92/08
awReceiveData9	LP_CanOut3_wOut1	0x5C9B/09	MciOut_wOut9	0x5C92/09
awReceiveData10	P_CanOut3_wOut2	0x5C9B/0A	MciOut_wOut10	0x5C92/0A
awReceiveData11	P_CanOut3_wOut3	0x5C9B/0B	MciOut_wOut11	0x5C92/0B
awReceiveData12	P_CanOut3_wOut4	0x5C9B/0C	MciOut_wOut12	0x5C92/0C
awReceiveData13	-	-	MciOut_wOut13	0x5C92/0D
awReceiveData14	-	-	MciOut_wOut14	0x5C92/0E
awReceiveData15	-	-	MciOut_wOut15	0x5C92/0F
awReceiveData16	-	-	MciOut_wOut16	0x5C92/10

7.5

Servo Drive 9400 HighLine



Communication with the Servo Drive 9400 HighLine is possible via the CAN bus (CANopen) and EtherCAT.

Available technology applications

From firmware version 1.0.1 onwards:

- Empty application

Note!

After creating the »Engineer« application, the indices have to be configured manually in the »PLC Designer« since no predefined port variables (indices) are defined here.

From firmware version 4.0 onwards:

- Actuator speed
- Actuator - Torque

From firmware version 5.0 onwards:

- Table positioning
- Electronic gearbox
- Synchronism with mark synchronisation
- Positioning sequence control

From firmware version 8.0 onwards:

- CiA 402 (für SoftMotion applications)

Advanced functions

- Function blocks for cam function

Sample projects

- LAS_40_INTF_Can_94HL_Speed_0200
- LAS_40_INTF_Can_94HL_TabPos_0200
- LAS_40_INTF_ETC_94HL_Speed_0100
- LAS_40_INTF_ETC_94HL_TabPos_0100

7.5.1 Indices for "table positioning" technology application

The received process data are transmitted into the *awReceiveData* element of the [L LCB AXIS REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L LCB AXIS REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LPortControl1	0xA580/02	LPortAxisIn1_wControl	0xA580/01
awTransmitData2	LPort16In1	0xA540/02	LPortAxisIn1_nIn1	0xA540/01
awTransmitData3	LPort16In2	0xA540/03	LPortAxisIn1_dnIn2	0xA640/01
awTransmitData4	LPort16In3	0xA540/04		
awTransmitData5	LPort32In1	0xA640/02	LPortControl1	0xA580/02
awTransmitData6			LPortControl2	0xA580/03
awTransmitData7	LPort32In2	0xA640/03	LPort32In1	0xA640/02
awTransmitData8				
awTransmitData9	LPort32In3	0xA640/04	LPort32In2	0xA640/03
awTransmitData10				
awTransmitData11	LPortControl2	0xA580/03	LPort32In3	0xA640/04
awTransmitData12	LPortAxisIn1_wControl	0xA580/01		
awTransmitData13	LPortAxisIn1_nIn1	0xA540/01	LPort16In1	0xA540/02
awTransmitData14	LPortAxisIn1_dnIn2	0xA640/01	LPort16In2	0xA540/03
awTransmitData15			LPort16In3	0xA540/04
awTransmitData16	-	-	-	-
awReceiveData1	LPortAxisOut1_wStatus	0xA100/01	LPortAxisOut1_wStatus	0xA100/01
awReceiveData2	LPortAxisOut1_nOut1	0xA0C0/01	LPortAxisOut1_nOut1	0xA0C0/01
awReceiveData3	LPortAxisOut1_dnOut2	0xA1C0/01	LPortAxisOut1_dnOut2	0xA1C0/01
awReceiveData4				
awReceiveData5	LPortStatus1	0xA100/02	LPortStatus1	0xA100/02
awReceiveData6	LPortStatus2	0xA100/03	LPortStatus2	0xA100/03
awReceiveData7	LPort32Out1	0xA1C0/02	LPort32Out1	0xA1C0/02
awReceiveData8				
awReceiveData9	LPort32Out2	0xA1C0/03	LPort32Out2	0xA1C0/03
awReceiveData10				
awReceiveData11	LPort32Out3	0xA1C0/04	LPort32Out3	0xA1C0/04
awReceiveData12				
awReceiveData13	LPort16Out1	0xA0C0/02	LPort16Out1	0xA0C0/02
awReceiveData14	LPort16Out2	0xA0C0/03	LPort16Out2	0xA0C0/03
awReceiveData15	LPort16Out3	0xA0C0/04	LPort16Out3	0xA0C0/04
awReceiveData16	-	-	-	-

7.5.2 Indices for the other technology applications

The received process data are transmitted into the *awReceiveData* element of the [L LCB AXIS REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L LCB AXIS REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LPortAxisIn1_wControl	0xA580/01	LPortAxisIn1_wControl	0xA580/01
awTransmitData2	LPortAxisIn1_nIn1	0xA540/01	LPortAxisIn1_nIn1	0xA540/01
awTransmitData3	LPortAxisIn1_dnIn2	0xA640/01	LPortAxisIn1_dnIn2	0xA640/01
awTransmitData4				
awTransmitData5	LPortControl1	0xA580/02	LPortControl1	0xA580/02
awTransmitData6	LPortControl2	0xA580/03	LPortControl2	0xA580/03
awTransmitData7	LPort32In1	0xA640/02	LPort32In1	0xA640/02
awTransmitData8				
awTransmitData9	LPort32In2	0xA640/03	LPort32In2	0xA640/03
awTransmitData10				
awTransmitData11	LPort32In3	0xA640/04	LPort32In3	0xA640/04
awTransmitData12				
awTransmitData13	LPort16In1	0xA540/02	LPort16In1	0xA540/02
awTransmitData14	LPort16In2	0xA540/03	LPort16In2	0xA540/03
awTransmitData15	LPort16In3	0xA540/04	LPort16In3	0xA540/04
awTransmitData16	-	-	-	-
awReceiveData1	LPortAxisOut1_wStatus	0xA100/01	LPortAxisOut1_wStatus	0xA100/01
awReceiveData2	LPortAxisOut1_nOut1	0xA0C0/01	LPortAxisOut1_nOut1	0xA0C0/01
awReceiveData3	LPortAxisOut1_dnOut2	0xA1C0/01	LPortAxisOut1_dnOut2	0xA1C0/01
awReceiveData4				
awReceiveData5	LPortStatus1	0xA100/02	LPortStatus1	0xA100/02
awReceiveData6	LPortStatus2	0xA100/03	LPortStatus2	0xA100/03
awReceiveData7	LPort32Out1	0xA1C0/02	LPort32Out1	0xA1C0/02
awReceiveData8				
awReceiveData9	LPort32Out2	0xA1C0/03	LPort32Out2	0xA1C0/03
awReceiveData10				
awReceiveData11	LPort32Out3	0xA1C0/04	LPort32Out3	0xA1C0/04
awReceiveData12				
awReceiveData13	LPort16Out1	0xA0C0/02	LPort16Out1	0xA0C0/02
awReceiveData14	LPort16Out2	0xA0C0/03	LPort16Out2	0xA0C0/03
awReceiveData15	LPort16Out3	0xA0C0/04	LPort16Out3	0xA0C0/04
awReceiveData16	-	-	-	-

7.6

Servo Drive 9400 Regenerative Power Supply Module



Communication with the Servo Drive 9400 Regenerative Power Supply Module is possible via the CAN bus (CANopen) and EtherCAT.

Available technology applications

- Application based on the technology application VR module V3.0

Sample projects

- None

Overview of the indices

The received process data are transmitted into the *awReceiveData* element of the [L_LCB_AXIS_REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L_LCB_AXIS_REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	LPortAxisIn1_wControl	0xA580/01	LPortAxisIn1_wControl	0xA580/01
awTransmitData2	LPortAxisIn1_nIn1	0xA540/01	LPortAxisIn1_nIn1	0xA540/01
awTransmitData3	LPortAxisIn1_dnIn2	0xA640/01	LPortAxisIn1_dnIn2	0xA640/01
awTransmitData4				
awTransmitData5	LPortControl1	0xA580/02	LPortControl1	0xA580/02
awTransmitData6	LPortControl2	0xA580/03	LPortControl2	0xA580/03
awTransmitData7	LPort32In1	0xA640/02	LPort32In1	0xA640/02
awTransmitData8				
awTransmitData9	LPort32In2	0xA640/03	LPort32In2	0xA640/03
awTransmitData10				
awTransmitData11	LPort32In3	0xA640/04	LPort32In3	0xA640/04
awTransmitData12				
awTransmitData13	LPort16In1	0xA540/02	LPort16In1	0xA540/02
awTransmitData14	LPort16In2	0xA540/03	LPort16In2	0xA540/03
awTransmitData15	LPort16In3	0xA540/04	LPort16In3	0xA540/04
awTransmitData16	-	-	-	-
awReceiveData1	LPortAxisOut1_wStatus	0xA100/01	LPortAxisOut1_wStatus	0xA100/01
awReceiveData2	LPortAxisOut1_nOut1	0xA0C0/01	LPortAxisOut1_nOut1	0xA0C0/01
awReceiveData3	LPortAxisOut1_dnOut2	0xA1C0/01	LPortAxisOut1_dnOut2	0xA1C0/01
awReceiveData4				
awReceiveData5	LPortStatus1	0xA100/02	LPortStatus1	0xA100/02
awReceiveData6	LPortStatus2	0xA100/03	LPortStatus2	0xA100/03
awReceiveData7	LPort32Out1	0xA1C0/02	LPort32Out1	0xA1C0/02
awReceiveData8				

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awReceiveData9	LPort32Out2	0xA1C0/03	LPort32Out2	0xA1C0/03
awReceiveData10				
awReceiveData11	LPort32Out3	0xA1C0/04	LPort32Out3	0xA1C0/04
awReceiveData12				
awReceiveData13	LPort16Out1	0xA0C0/02	LPort16Out1	0xA0C0/02
awReceiveData14	LPort16Out2	0xA0C0/03	LPort16Out2	0xA0C0/03
awReceiveData15	LPort16Out3	0xA0C0/04	LPort16Out3	0xA0C0/04
awReceiveData16	-	-	-	-

7.7

Lenze Generic Drive



The "@Lenze Generic Drive" serves to control any Lenze inverter on the CAN bus (CANopen).

Control takes place via the function block [L LCB GenericDrive](#) (72).

Sample projects

- None

Overview of the indices



Note!

The process data have to be mapped in the inverter.

The device description of the Lenze Generic Drive serves to define wildcards (e.g. CAN IN1 W1) to which the process data will be transmitted.

The received process data are transmitted into the *awReceiveData* element of the [L LCB AXIS REF](#) instance (67).

The process data to be sent are transmitted by the *awTransmitData* element of the [L LCB AXIS REF](#) instance (67).

The table shows the mapping of the indices to variables.

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awTransmitData1	CAN IN1 W1	0x5CA7/01	-	-
awTransmitData2	CAN IN1 W2	0x5CA7/02	-	-
awTransmitData3	CAN IN1 W3	0x5CA7/03	-	-
awTransmitData4	CAN IN1 W4	0x5CA7/04	-	-
awTransmitData5	CAN IN2 W1	0x5CA8/01	-	-
awTransmitData6	CAN IN2 W2	0x5CA8/02	-	-
awTransmitData7	CAN IN2 W3	0x5CA8/03	-	-
awTransmitData8	CAN IN2 W4	0x5CA8/04	-	-
awTransmitData9	CAN IN3 W1	0x5CB0/01	-	-
awTransmitData10	CAN IN3 W2	0x5CB0/02	-	-
awTransmitData11	CAN IN3 W3	0x5CB0/03	-	-
awTransmitData12	CAN IN3 W4	0x5CB0/04	-	-
awTransmitData13	-	-	-	-
awTransmitData14	-	-	-	-
awTransmitData15	-	-	-	-
awTransmitData16	-	-	-	-
awReceiveData1	CAN OUT1 W1	0x5CAD/01	-	-
awReceiveData2	CAN OUT1 W2	0x5CAD/02	-	-
awReceiveData3	CAN OUT1 W3	0x5CAD/03	-	-
awReceiveData4	CAN OUT1 W4	0x5CAD/04	-	-
awReceiveData5	CAN OUT2 W1	0x5CAE/01	-	-
awReceiveData6	CAN OUT2 W2	0x5CAE/02	-	-

Application variable	Name CAN	Index CAN	Name EtherCAT	Index EtherCAT
awReceiveData7	CAN OUT2 W3	0x5CAE/03	-	-
awReceiveData8	CAN OUT2 W4	0x5CAE/04	-	-
awReceiveData9	CAN OUT3 W1	0x5CAF/01	-	-
awReceiveData10	CAN OUT3 W2	0x5CAF/02	-	-
awReceiveData11	CAN OUT3 W3	0x5CAF/30	-	-
awReceiveData12	CAN OUT3 W4	0x5CAF/04	-	-
awReceiveData13	-	-	-	-
awReceiveData14	-	-	-	-
awReceiveData15	-	-	-	-
awReceiveData16	-	-	-	-

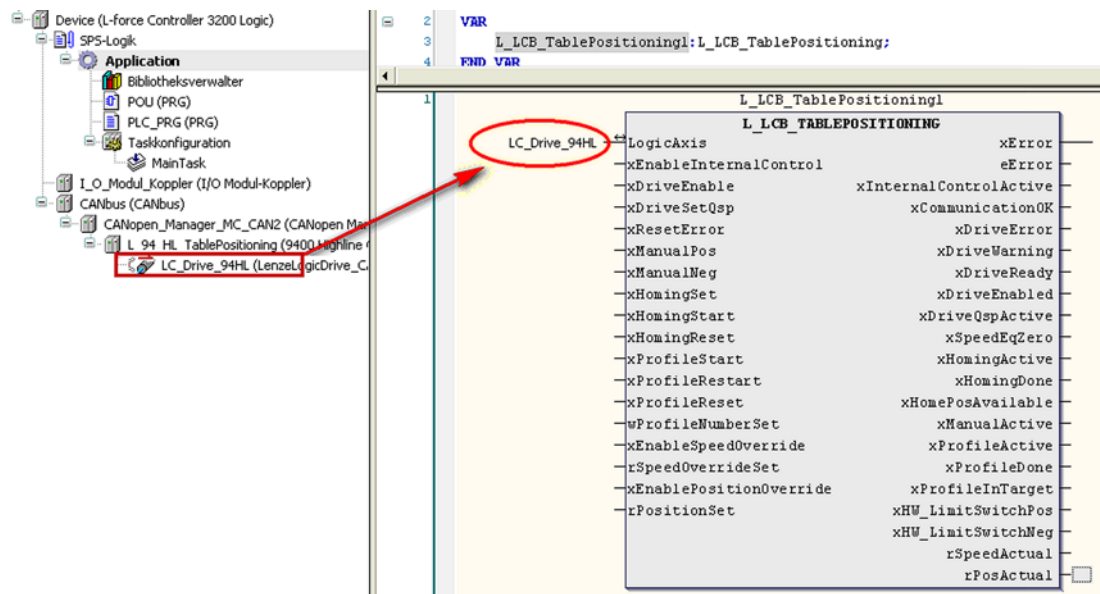
8 L_LCB_LogicControlBasic function library

The **L_LCB_LogicControlBasic** library contains function blocks for a device-independent control of inverters on the Logic bus:

Function block	Description
▶ L_LCB_ActuatorSpeed	Control of an inverter with the "speed actuating drive" application sample
▶ L_LCB_TablePositioning	Control of an inverter with the "table positioning" application sample
▶ L_LCB_8400Drive	Control of an Inverter Drive 8400
▶ L_LCB_9400Drive	Control of a Servo Drive 9400
▶ L_LCB_GenericDrive	Control of any Lenze inverter with any application

Connecting a function block to an inverter

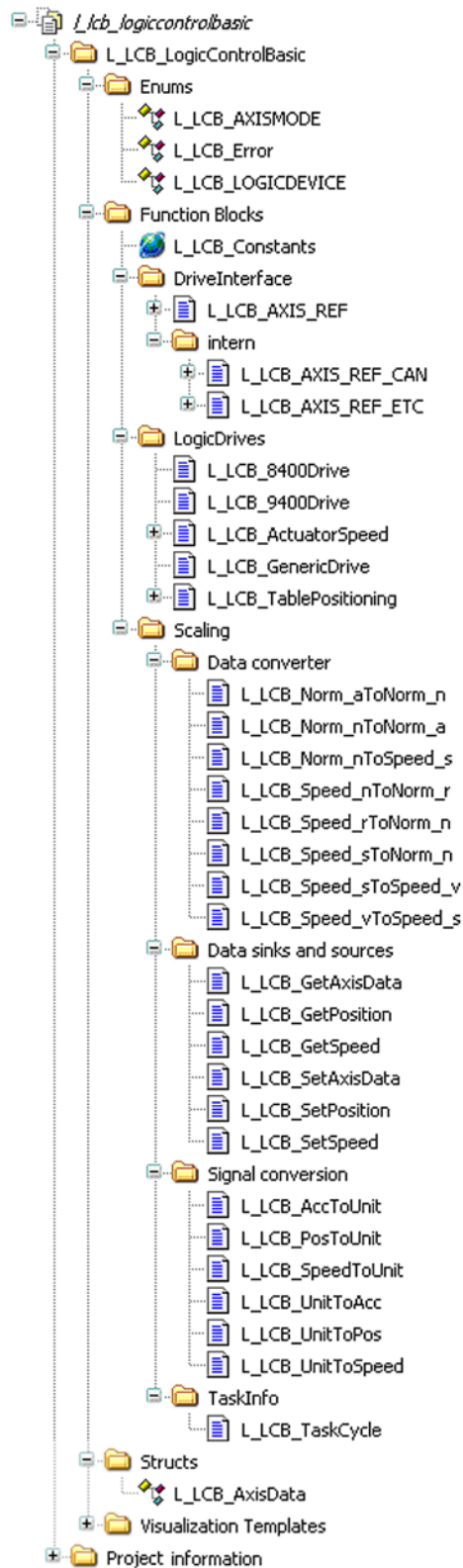
In order to connect a function block to an inverter, the axis data instance from the control configuration has to be connected to the function block.



[8-1] **Model connection:** Axis data instance of a 9400 HighLine with the **L_LCB_TablePositioning** function block

8.1 Overview of the functions and function blocks

The functions and function blocks of the **L_LCB_LogicControlBasic** library are divided into different groups.



Enums

- ▶ [L_LCB_AXISMODE](#) (102)
- ▶ [L_LCB_Error](#) (102)
- ▶ [L_LCB_LOGICDEVICE](#) (102)

Function Blocks

DriveInterface

- ▶ [L_LCB_AXIS_REF](#) (67)
- ▶ [L_LCB_AXIS_REF_CAN \(internal use\)](#) (68)
- ▶ [L_LCB_AXIS_REF_ETC \(internal use\)](#) (68)

LogicDrives

- ▶ [L_LCB_8400Drive](#) (69)
- ▶ [L_LCB_9400Drive](#) (70)
- ▶ [L_LCB_ActuatorSpeed](#) (73)
- ▶ [L_LCB_GenericDrive](#) (72)
- ▶ [L_LCB_TablePositioning](#) (77)

Data converter

- ▶ [L_LCB_Norm_aToNorm_n](#) (83)
- ▶ [L_LCB_Norm_nToNorm_a](#) (83)
- ▶ [L_LCB_Norm_nToSpeed_s](#) (84)
- ▶ [L_LCB_Speed_nToNorm_r](#) (84)
- ▶ [L_LCB_Speed_rToNorm_n](#) (85)
- ▶ [L_LCB_Speed_sToNorm_n](#) (85)
- ▶ [L_LCB_Speed_sToSpeed_v](#) (86)
- ▶ [L_LCB_Speed_vToSpeed_s](#) (86)

Data sinks and sources

- ▶ [L_LCB_GetAxisData](#) (87)
- ▶ [L_LCB_GetPosition](#) (88)
- ▶ [L_LCB_GetSpeed](#) (89)
- ▶ [L_LCB_SetAxisData](#) (90)
- ▶ [L_LCB_SetPosition](#) (92)
- ▶ [L_LCB_SetSpeed](#) (93)

Signal conversion

- ▶ [L_LCB_AccToUnit](#) (94)
- ▶ [L_LCB_PosToUnit](#) (95)
- ▶ [L_LCB_SpeedToUnit](#) (96)
- ▶ [L_LCB_UnitToAcc](#) (97)
- ▶ [L_LCB_UnitToPos](#) (98)
- ▶ [L_LCB_UnitToSpeed](#) (99)

TaskInfo

- ▶ [L_LCB_TaskCycle](#) (100)

Struts

- ▶ [L_LCB_AxisData](#) (101)

8.2 Function blocks

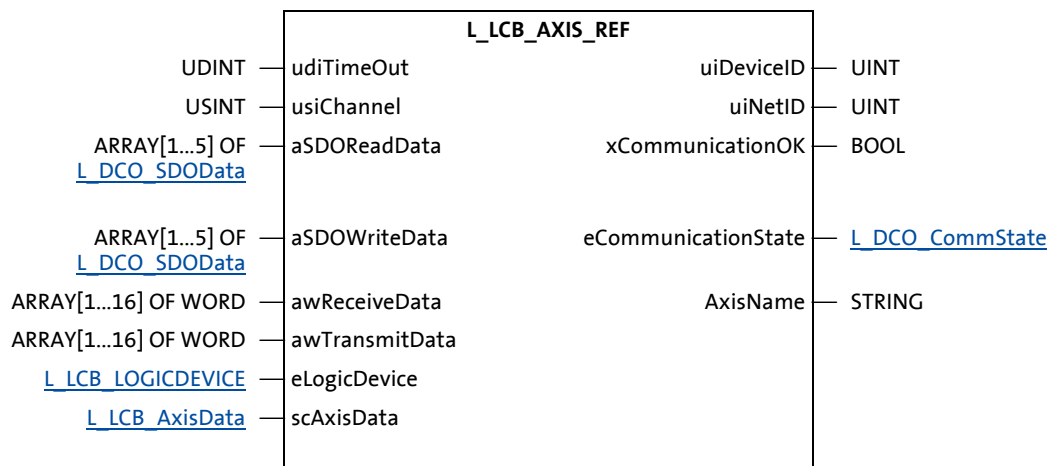
8.2.1 L_LCB_AXIS_REF

This function block instance is a logic drive in the application.

Every Logic axis implemented in the control configuration is an extended instance of **L_LCB_AXIS_REF**. A manual instancing of this function block is not required.

Via this function block, the application is provided with process data of the connected Logic axis.

L_LCB_AXIS_REF is an extension of the [L_DCO_AXIS_REF_BASE](#) (105) function block.



Inputs

Identifier/data type	Information/possible settings
udiTimeOut UDINT	Timeout period for the SDO communication • Initial value: 10000
usiChannel USINT	SDO channel to be used • Initial value: 1
aSdoReadData ARRAY[1...5] OF L_DCO_SDOData	Exchange for SDO communication (internal use) • Initial value: 0
aSdoWriteData ARRAY[1...5] OF L_DCO_SDOData	Exchange for SDO communication (internal use) • Initial value: 0
awReceiveData ARRAY[1...16] OF WORD	Interface for the process data received by the inverter. • Initial value: 0
awTransmitData ARRAY[1...16] OF WORD	Interface for the process data to be transmitted to the inverter. • Initial value: 0
eLogicDrive L_LCB_LOGICDEVICE	Type of the connected inverter
scAxisData L_LCB_AxisData	Machine parameters of the connected inverter for converting machine units into internal units.

Outputs

Identifier/data type	Value/meaning	
uiDeviceId UINT	Station/node address of the axis in the used bus system • Initial value: 0	
uiNetId UINT	Network/master instance number of the used bus system • Initial value: 0	
xCommunicationOK BOOL	TRUE	Bus communication available
	FALSE	No bus communication active
eCommunicationState L_DCO_CommState	Busstatus • Initial value: 0	
AxisName STRING	Axis designation	

8.2.2 L_LCB_AXIS_REF_CAN (internal use)

Fieldbus-specific extension of the [L_LCB_AXIS_REF](#) function block for the CAN bus system.

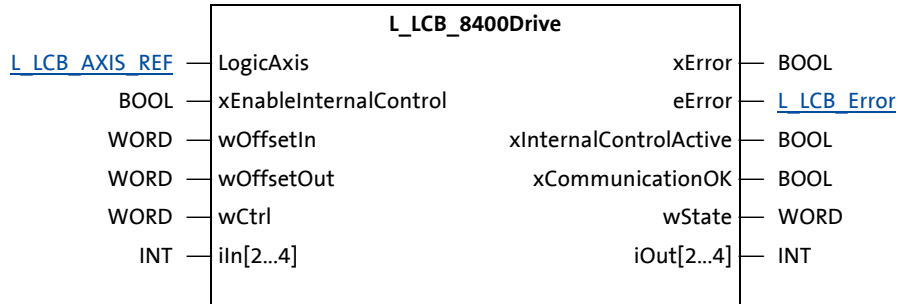
8.2.3 L_LCB_AXIS_REF_ETC (internal use)

Fieldbus-specific extension of the [L_LCB_AXIS_REF](#) function block for the EtherCAT bus system.

8.2.4 L_LCB_8400Drive

This function block serves to actuate Inverter Drives 8400.

Communication with the devices is possible via the CAN or EtherCAT bus systems.



Inputs

Identifier/data type	Information/possible settings	
LogicAxis L_LCB_AXIS_REF	Structural data for drive control	
xEnableInternalControl BOOL	TRUE	The internal control of the inverter can be activated via the "Internal Control" button in the visualisation.
xOffsetIn WORD	Offset of the access to process data for inputs by <i>wOffsetIn</i> x 8 bytes Examples: • '0' for CanIn PDO 1 • '1' for CanIn PDO 2	
xOffsetOut WORD	Offset of the access to process data for outputs by <i>wOffsetOut</i> x 8 bytes Examples: • '0' for CanOut PDO 1 • '1' for CanOut PDO 2	
wCtrl WORD	Interface of the Inverter Drive 8400	
iIn[2...4] INT	Interfaces of the Inverter Drive 8400	

Outputs

Identifier/data type	Value/meaning	
xError BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)	
xInternalControlActive BOOL	TRUE	The <i>xEnableInternalControl</i> and the "Internal Control" button in the visualisation are activated.
xCommunicationOK BOOL	TRUE	Bus communication available
	FALSE	No bus communication active
wState WORD	Interface of the Inverter Drive 8400	
iOut[2...4] INT	Interfaces of the Inverter Drive 8400	

8.2.5 L_LCB_9400Drive

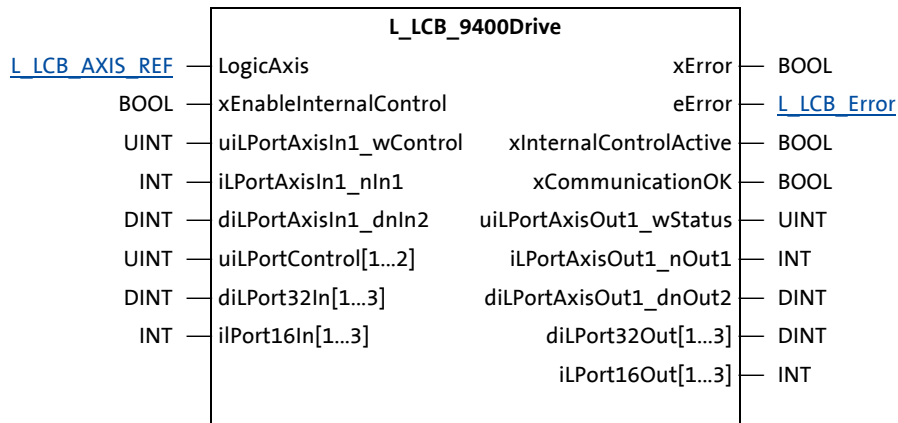
This function block serves to control Servo Drives 9400 with any application.

**Note!**

For an error-free function of the function block, the port configuration of the application needs to be maintained on the inverter!

No I/O ports must be deleted or moved.

Communication with the devices is possible via the CAN or EtherCAT bus systems.

**Inputs**

Identifier/data type	Information/possible settings	
LogicAxis L_LCB_AXIS_REF	Structural data for drive control	
xEnableInternalControl BOOL	TRUE	The internal control of the inverter can be activated via the "Internal Control" button in the visualisation.
uiLPortAxisIn1_wControl UINT	<i>LPortAxisIn1.wControl</i> inverter of the Servo Drive 9400	
iLPortAxisIn1_nIn1 INT	<i>LPortAxisIn1.nIn1</i> interface of the Servo Drive 9400	
diLPortAxisIn1_dnIn2 DINT	<i>LPortAxisIn1.dnIn2</i> interface of the Servo Drive 9400	
uiLPortControl1 UINT	<i>LPortControl1</i> interface of the Servo Drive 9400	
uiLPortControl2 UINT	<i>LPortControl2</i> interface of the Servo Drive 9400	
diLPort32In1 DINT	<i>LPort32In1</i> interface of the Servo Drive 9400	
diLPort32In2 DINT	<i>LPort32In2</i> interface of the Servo Drive 9400	
diLPort32In3 DINT	<i>LPort32In3</i> interface of the Servo Drive 9400	
iLPort16In1 INT	<i>LPort16In1</i> interface of the Servo Drive 9400	

Identifier/data type	Information/possible settings
iLPort16In2 INT	<i>LPort16In2</i> interface of the Servo Drive 9400
iLPort16In3 INT	<i>LPort16In3</i> interface of the Servo Drive 9400

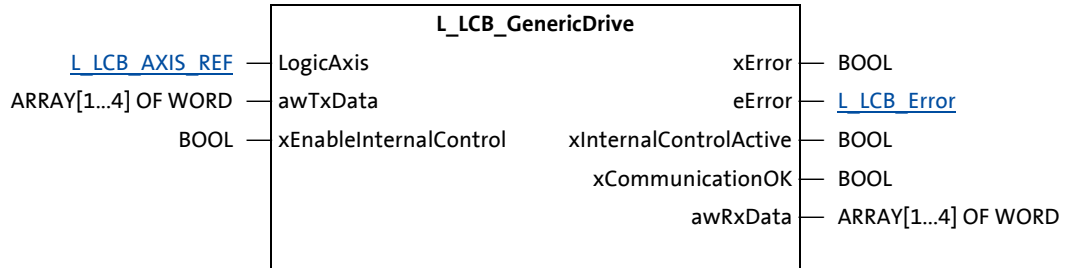
Outputs

Identifier/data type	Value/meaning
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If xError = TRUE: Error number for the error that has occurred. ► Error messages (102)
xInternalControlActive BOOL	TRUE The <i>xEnableInternalControl</i> and the "Internal Control" button in the visualisation are activated.
xCommunicationOK BOOL	TRUE Bus communication available
	FALSE No bus communication active
uiLPortAxisOut1_wStatus UINT	<i>LPortAxisOut1.wStatus</i> interface of the Servo Drive 9400
diLPortAxisOut1_dnOut1 INT	<i>LPortAxisOut1.nOut1</i> interface of the Servo Drive 9400
diLPortAxisOut1_dnOut2 DINT	<i>LPortAxisOut1.dnOut2</i> interface of the Servo Drive 9400
uiLPortStatus1 UINT	<i>LPortStatus1</i> interface of the Servo Drive 9400
uiLPortStatus2 UINT	<i>LPortStatus2</i> interface of the Servo Drive 9400
diLPort32Out1 DINT	<i>LPort32Out1</i> interface of the Servo Drive 9400
diLPort32Out2 DINT	<i>LPort32Out2</i> interface of the Servo Drive 9400
diLPort32Out3 DINT	<i>LPort32Out3</i> interface of the Servo Drive 9400
iLPort16Out1 INT	<i>LPort16Out1</i> interface of the Servo Drive 9400
iLPort16Out2 INT	<i>LPort16Out2</i> interface of the Servo Drive 9400
iLPort16Out3 INT	<i>LPort16Out3</i> interface of the Servo Drive 9400

8.2.6 L_LCB_GenericDrive

This function block serves to control any Lenze inverter.

Communication with the device is possible via CAN.



Input

Identifier/data type	Information/possible settings	
LogicAxis L_LCB_AXIS_REF	Structural data for drive control	
awTxData ARRAY[1...4] OF WORD	Process data to be transmitted to the inverter	
xEnableInternalControl BOOL	TRUE	The internal control of the inverter can be activated via the "Internal Control" button in the visualisation.

Outputs

Identifier/data type	Value/meaning	
xError BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)	
xInternalControlActive BOOL	TRUE	The <i>xEnableInternalControl</i> and the "Internal Control" button in the visualisation are activated.
xCommunicationOK BOOL	TRUE	Bus communication available
	FALSE	No bus communication active
awRxData ARRAY[1...4] OF WORD	Process data received by the inverter	

8.2.7 L_LCB_ActuatorSpeed

This function block serves to control Lenze inverters with the "speed actuating drive" application.

**Note!**

The function block only works in combination with a "speed actuating drive" sample project (marked with "_Speed").

► [Overview of the application samples](#) (20)

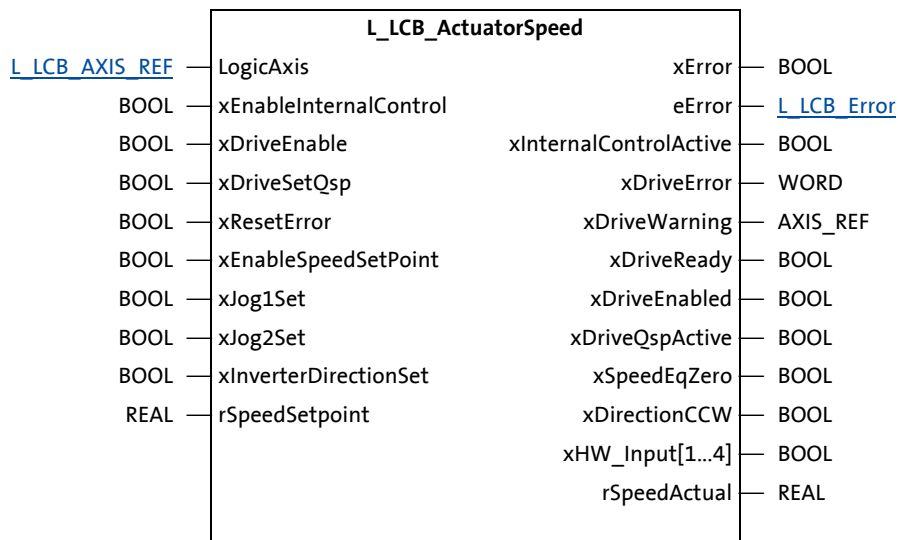
For creating a function block interconnection, the following modules can be used:

► [L_LCB_8400Drive](#) (69)

► [L_LCB_9400Drive](#) (70)

► [L_LCB_GenericDrive](#) (72)

Communication with the devices is possible via the CAN or EtherCAT bus systems.

**Inputs**

Identifier/data type	Information/possible settings	
LogicAxis L_LCB_AXIS_REF	Structural data for drive control	
xEnableInternalControl BOOL	TRUE	The internal control of the inverter can be activated via the "Internal Control" button in the visualisation.
xDriveEnable BOOL	TRUE: Deactivate controller inhibit (inverted)	
xDriveSetQSP BOOL	TRUE: Activate quick stop	
xResetError BOOL	TRUE: Reset error message (acknowledged)	
xEnableSpeedSetpoint BOOL	TRUE: Speed control activated • The inputs <i>xJogSet1</i> , <i>xJogSet2</i> and <i>rSpeedSetPoint</i> are evaluated.	
xJog1Set BOOL	TRUE: Jog 1 activated • Speed setpoint 1 requested • Active if <i>xEnableSpeedSetpoint</i> = TRUE	

Identifier/data type		Information/possible settings
xJog2Set	BOOL	TRUE: Jog 2 activated - Speed setpoint 2 requested - Active if <i>xEnableSpeedSetpoint</i> = TRUE
xInvertDirectionSet	BOOL	TRUE: Inversion of the direction of rotation of the motor (clockwise rotation/ counter-clockwise rotation)
rSpeedSetpoint	REAL	Setpoint selection for the speed in [%] - Active if... <i>xEnableSpeedSetpoint</i> = TRUE <u>and</u> <i>xJog1Set</i> = FALSE <u>and</u> <i>xJog2Set</i> = FALSE

Outputs

Identifier/data type		Value/meaning
xError	BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError	L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ▶ Error messages (102)
xCommunicationOK	BOOL	TRUE Bus communication available
		FALSE No bus communication active
xInternalControlActive	BOOL	TRUE The <i>xEnableInternalControl</i> and the "Internal Control" button in the visualisation are activated.
xDriveError	BOOL	Status signal "Error in the inverter"
xDriveWarning	BOOL	TRUE: Inverter is in "Warning" device state
xDriveReady	BOOL	Status signal "Inverter is ready for operation"
xDriveEnabled	BOOL	TRUE: Inverter has been enabled
xDriveQSPActive	BOOL	TRUE: Quick stop (QSP) active
xSpeedEqZero	BOOL	TRUE: Speed is zero
xDirectionCCW	BOOL	Display of the direction of rotation of the motor TRUE:CCW rotation is active
xHW_Input1	BOOL	Digital input 1 is active
xHW_Input2	BOOL	Digital input 2 is active
xHW_Input3	BOOL	Digital input 3 is active
xHW_Input4	BOOL	Digital input 4 is active
rSpeedActual	REAL	Current speed of the motor shaft in [%] -199.99 ... 199.99 %

8.2.7.1 Assignment of the input ports

Identifier/data type	Servo Drive 9400 HighLine	Inverter Drive 8400 HighLine	WORD
xDriveEnable BOOL	NOT Bit 0	Bit 0 and 3	WORD 0
xDriveSetQSP BOOL	Bit 1	Bit 2	
xResetError BOOL	Bit 10	Bit 7	
xEnableSpeedSetpoint BOOL	Bit 2	-	
xJog1Set BOOL	Bit 3	Bit 12 and xEnableSpeedSetpoint	
xJog2Set BOOL	Bit 4	Bit 13 and xEnableSpeedSetpoint	
xInvertDirectionSet BOOL	Bit 5	Bit 15	
	xCtrl06	xCtrl01	
	xCtrl07	xCtrl04	
	xCtrl08	xCtrl05	
	xCtrl09	xCtrl06	
	-	xCtrl08	
	xCtrl11/xSetErrorSet	xCtrl09	
	xCtrl12	xCtrl10	
	xCtrl13/xEnableManual	xCtrl11	
	xCtrl14/xManualPositive	xCtrl14/ xSetError	
	xCtrl15/xManualNegative	Free	
	Free	rSpeedSetpoint	WORD 1
	rSpeedSetpoint	Free	WORD 2
		Free	WORD 3

8.2.7.2 Assignment of the output ports

Identifier/data type	Servo Drive 9400 HighLine	Inverter Drive 8400 HighLine	WORD
xDriveError BOOL	Bit 0	Bit 0	WORD 0
xDriveWarning BOOL	Bit 12	Bit 12	
xDriveReady BOOL	Bit 2	Bit 2	
xDriveEnabled BOOL	NOT Bit 7	Bit 7	
xDriveQSPActive BOOL	Bit 3	Bit 3	
xSpeedEqZero BOOL	Bit 6	Bit 6	
xDirectionCCW BOOL	-	-	
xHW_Input1 BOOL	Bit 4	Bit 4/Digital_In1	
xHW_Input2 BOOL	Bit 5	Bit 5/Digital_In2	
xHW_Input3 BOOL	Bit 14	Bit 14/Digital_In3	
xHW_Input4 BOOL	Bit 15	Bit 15/Digital_In4	
	xStat08/SF_active	xStat08/StatusCodeBit1	
	xStat09	xStat09/StatusCodeBit1	
	xStat10	xStat10/StatusCodeBit1	
	xStat11	xStat11/StatusCodeBit1	
	xStat13	xStat13/Message	
	Free	rSpeedActual	WORD 1
	rSpeedActual	Free	WORD 2

8.2.8 L_LCB_TablePositioning

This function block serves to control Inverter Drives 8400 HighLine and Servo Drives 9400 HighLine with the "table positioning" application.



Note!

The function block only works in combination with a "table positioning" sample project (marked with "_TabPos").

▶ [Overview of the application samples](#) (20)

For creating a function block interconnection, the following modules can be used:

▶ [L_LCB_8400Drive](#) (69)

▶ [L_LCB_9400Drive](#) (70)

▶ [L_LCB_GenericDrive](#) (72)

Communication with the devices is possible via the CAN or EtherCAT bus systems.

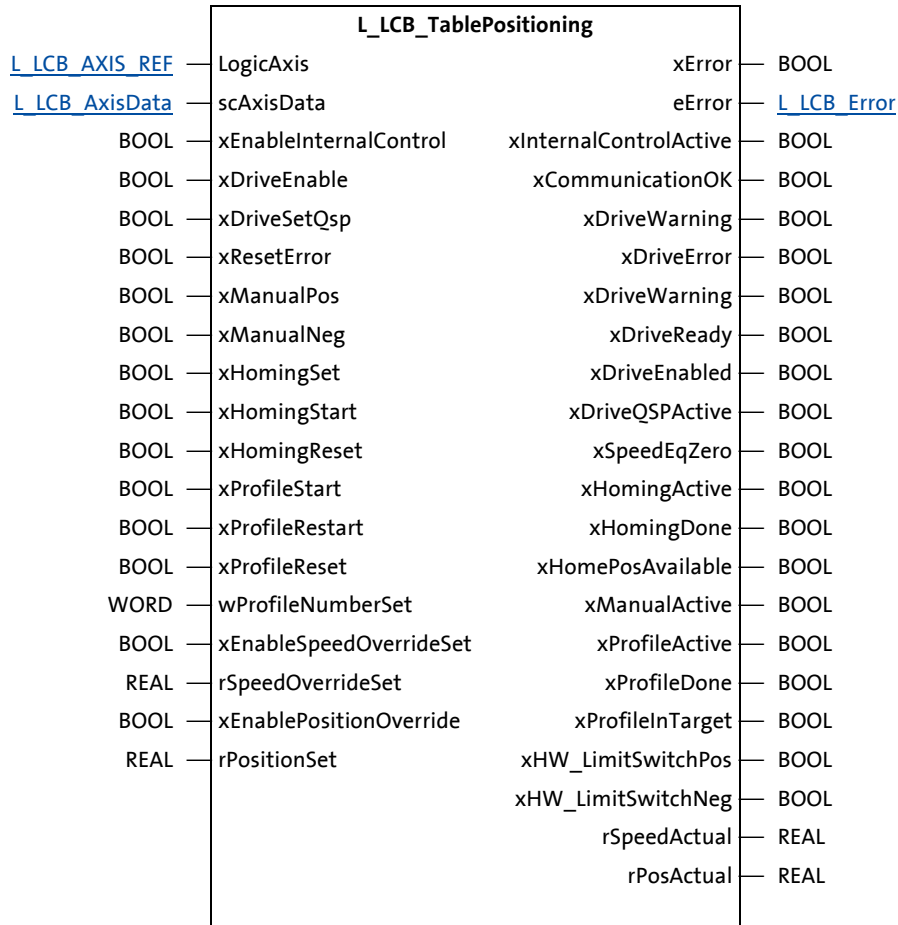
Activating "position override"

In order to be able to execute a positioning, the inverter has to be enabled (set input *xDriveEnable* to TRUE) and the home position has to be set (e.g. via the *xHomingSet* input).



How to activate a "position override":

1. Set the *xEnablePositionOverride* input to TRUE.
2. Set the desired setpoint position at the *rPositionSet* input.
3. Set the number of the profile data set for the positioning at the *wProfileNumberSet* input.
4. Start the positioning process by setting the *xProfileStart* input to TRUE.
 - The position is written to the code/parameter of the selected profile data set.
 - Positioning is carried out with the speed and acceleration of the selected profile data set.



Inputs

Identifier/data type	Information/possible settings	
LogicAxis L_LCB_AXIS_REF	Structural data for drive control	
scAxisData L_LCB_AxisData	Machine parameters of the inverter to be controlled	
xEnableInternalControl BOOL	TRUE	The internal control of the inverter can be activated via the "Internal Control" button in the visualisation.
xDriveEnable BOOL	TRUE	Deactivate controller inhibit (inverted)
xDriveSetQSP BOOL	TRUE	Activate quick stop (QSP)
xResetError BOOL	TRUE	Reset (acknowledge) error message
xManualPos BOOL	TRUE	Manual jog in positive direction (CW rotation)
xManualNeg BOOL	TRUE	Manual jog in negative direction (CCW rotation)
xHomingSet BOOL	TRUE	Set home position
xHomingStart BOOL	TRUE	Start homing

Identifier/data type	Information/possible settings
xHomingReset BOOL	TRUE Reset home position
xProfileStart BOOL	Start/stop positioning
	TRUE Start/restart positioning process - Define the profile to be travelled at the <i>wProfileNumberSet</i> input. - During an active positioning process, another profile can already be defined via <i>wProfileNumberSet</i> which will be executed after the restart (renewed state change from TRUE to FALSE). - A positioning process previously aborted will be continued.
	FALSE Abort the active positioning process.
xProfileRestart BOOL	FALSE↗TRUE Update of the profile data of a currently active profile considering the distance already covered.
xProfileReset BOOL	FALSE↗TRUE A profile previously aborted cannot be continued. When <i>xProfileStart</i> = TRUE, a profile will be restarted.
wProfileNumberSet WORD	Profile number of the profile to be executed from the profile data management - Servo Drives 9400: 1 ... 75 - Inverter Drives 8400: 3 ... 15
xEnableSpeedOverride BOOL	TRUE Switch on speed override
rSpeedOverrideSet REAL	Value for the speed override in [%] - Unit: % - Value range: 0.00 ... 199.99
xEnablePositionOverride BOOL	TRUE Switch on "Position Override" Note: The second receive PDO (RPDO) has to be activated in the »PLC Designer« for the CAN bus system ("PDO Mapping" tab). This setting is not required for the EtherCAT bus system.
rPositionSet REAL	Position selection Unit: units

Outputs

Identifier/data type	Value/meaning
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. Error messages (102)
xCommunicationOK BOOL	TRUE Bus communication available
	FALSE No bus communication active
xInternalControlActive BOOL	TRUE The <i>xEnableInternalControl</i> and the "Internal Control" button in the visualisation are activated.
xDriveError BOOL	TRUE An error is active in the inverter (acknowledgement required).
xDriveWarning BOOL	TRUE The inverter is in "Warning" device state.
xDriveReady BOOL	TRUE The inverter is ready for operation.
xDriveEnabled BOOL	TRUE The inverter is enabled.
xDriveQSPActive BOOL	TRUE A quick stop (QSP) is active by application.
xSpeedEqZero BOOL	TRUE The inverter is at standstill The speed value in code C00019 must be observed!

Identifier/data type		Value/meaning	
xHomingActive	BOOL	TRUE	Homing is activated.
xHomingDone	BOOL	TRUE	Homing has been stopped.
xHomePosAvailable	BOOL	TRUE	The home position is known.
xManualActive	BOOL	TRUE	Manual jog is activated.
xProfileActive	BOOL	TRUE	Positioning is activated.
xProfileDone	BOOL	TRUE	Positioning has been stopped.
xProfileInTarget	BOOL	TRUE	The target position has been reached.
xHW_LimitSwitchPos	BOOL	TRUE	The positive hardware limit switch has been approached.
xHW_LimitSwitchNeg	BOOL	TRUE	The negative hardware limit switch has been approached.
rSpeedActual	REAL	Current velocity • Unit: %	
rPositionActual	REAL	Current position • Unit: units	

8.2.8.1 Assignment of the input ports

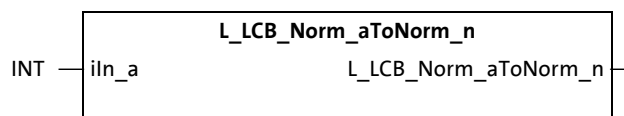
Identifier/data type	Servo Drive 9400 HighLine	Inverter Drive 8400 HighLine	WORD
xDriveEnable BOOL	NOT Bit 0	Bit 3	WORD 0
xDriveSetQSP BOOL	Bit 1	Bit 15	
xResetError BOOL	Bit 10	Bit 11	
xManualPos BOOL	Bit 14	Bit 12	
xManualNeg BOOL	Bit 15	Bit 13	
xHomingSet BOOL	Bit 12	Bit 10	
xHomingStart BOOL	Bit 3	Bit 8	
xHomingReset BOOL	Bit 4	Bit 14	
xProfileStart BOOL	Bit 5	Bit 0	
	Bit 6	Bit 1	
	-	Bit 8	
xProfileRestart BOOL	Bit 7	Bit 0	
	-	Bit 1	
	-	Bit 8	
xProfileReset BOOL	Bit 8	Bit 0	
	-	Bit 1	
	-	Bit 8	
xEnableSpeedOverride BOOL	Bit 9	Bit 9	
xEnablePositionOverride REAL	Bit 2	Bit 2	
		wProfileNumberSet.0 Bit 4	
		wProfileNumberSet.0 Bit 5	
		wProfileNumberSet.0 Bit 6	
		wProfileNumberSet.0 Bit 7	
	rSpeedOverrideSet	rSpeedOverrideSet	WORD 1
	wProfileNumberSet	rPositionSet	WORD 2

8.2.8.2 Assignment of the output ports

Identifier/data type	Servo Drive 9400 HighLine	Inverter Drive 8400 HighLine	WORD
xDriveError BOOL	Bit 0	Bit 0	WORD 0
xDriveWarning BOOL	Bit 9	Bit 15	
xDriveReady BOOL	Bit 2	Bit 4	
xDriveEnabled BOOL	NOT Bit 1	NOT Bit 3	
xDriveQSPActive BOOL	Bit 3	Bit 14	
xSpeedEqZero BOOL	Bit 4	Bit 11	
xHomingActive BOOL	Bit 5	Bit 9	
xHomingDone BOOL	Bit 6	Bit 6	
xHomePosAvailable BOOL	Bit 7	Bit 10	
xManualActive BOOL	Bit 8	Bit 9	
xProfileActive BOOL	Bit 10	Bit 9	
xProfileDone BOOL	Bit 11	Bit 7	
xProfileInTarget BOOL	Bit 12	Bit 8	
xHW_LimitSwitchPos BOOL	Bit 14	Bit 12	
xHW_LimitSwitchNeg BOOL	Bit 15	Bit 13	
rSpeedActual REAL			WORD 1
rPositionActual REAL			WORD 2

8.2.9 L_LCB_Norm_aToNorm_n

This function converts a 16 bit input signal into a 32 bit output signal.



Input

Identifier/data type	Information/possible settings
iln_a INT	Input signal • 100 % = $2^{14} = 16384$



Note!

If the input information has a value of '-32768', the function block internally limits the value '-32767' and copies it into the High Word of the *dnOut_n* output.

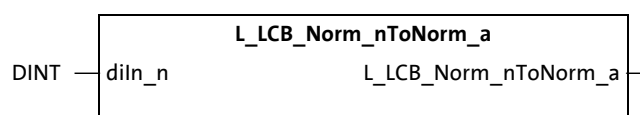
Therefore, this function block cannot be used to combine values of the "Low Word" and "High Word" data type in order to form a value of "DWORD" type.

Output signal

L_LCB_Norm_aToNorm_n (100 % = $2^{30} = 1073741824$)

8.2.10 L_LCB_Norm_nToNorm_a

This function converts a 32 bit input signal into a 16 bit output signal.



Input

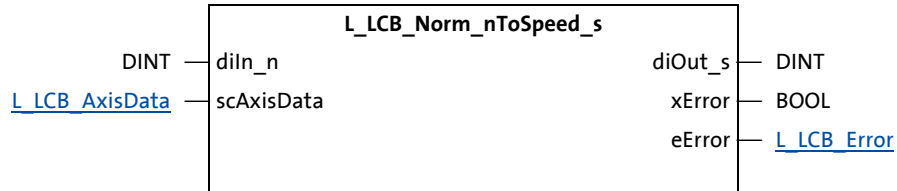
Identifier/data type	Information/possible settings
diIn_n DINT	Input signal • Unit: % (100 % = $2^{30} = 1073741824$)

Output signal

L_LCB_Norm_nToNorm_a (100 % = $2^{14} = 16384$)

8.2.11 L_LCB_Norm_nToSpeed_s

This function block uses the transmitted machine parameters and converts a 32-bit input signal into a 32-bit speed signal.



Input

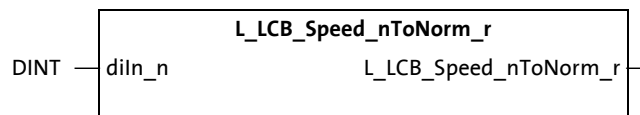
Identifier/data type	Information/possible settings
diIn_n DINT	Input signal • Unit: % ($100\% = 2^{30} = 1073741824$)
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diOut_s DINT	Output signal • Unit: rpm ($15000\text{ rpm} = 2^{26} = 67108864$)
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ▶ Error messages (102)

8.2.12 L_LCB_Speed_nToNorm_r

This function converts a 32-bit input signal into a speed signal of "REAL" type.



Input

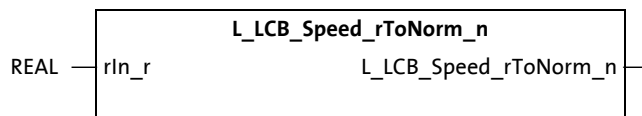
Identifier/data type	Information/possible settings
diIn_n DINT	Input signal • Unit: % ($100\% = 2^{30} = 1073741824$)

Output signal

L_LCB_Speed_nToNorm_r (REAL 0.0 ... 100.0 %)

8.2.13 L_LCB_Speed_rToNorm_n

This function converts an input signal of "REAL" type into a 32-bit speed signal.



Input

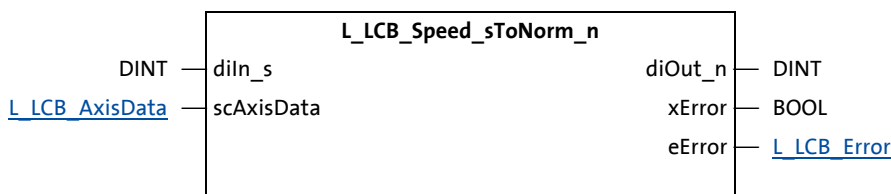
Identifier/data type	Information/possible settings
rln_r REAL	Input signal • Unit: % (0 ... 100 %)

Output signal

L_LCB_Speed_rToNorm_n (0 ... 2^{30} = 0 ... 1073741824)

8.2.14 L_LCB_Speed_sToNorm_n

This function block uses the transmitted machine parameters and converts a 32-bit speed signal into a 32-bit output signal.



Inputs

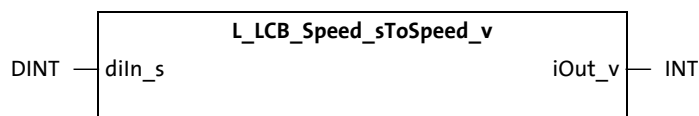
Identifier/data type	Information/possible settings
diIn_s DINT	Input signal • Unit: rpm (15000 rpm = 2^{26} = 67108864)
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diOut_n DINT	Output signal • Unit: % (100 % = 2^{30} = 1073741824)
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.15 L_LCB_Speed_sToSpeed_v

This function converts a 32-bit speed signal into a 16-bit speed signal.



Input

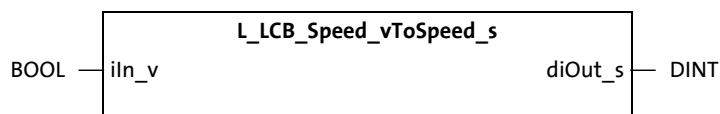
Identifier/data type	Information/possible settings
diIn_s DINT	Input signal • Unit: rpm (15000 rpm $\equiv 2^{26} = 67108864$)

Output

Identifier/data type	Value/meaning
iOut_v INT	Output signal • Unit: rpm (15000 rpm $\equiv 2^{14} = 16384$)

8.2.16 L_LCB_Speed_vToSpeed_s

This function converts a 16-bit speed signal into a 32-bit speed signal.



Input

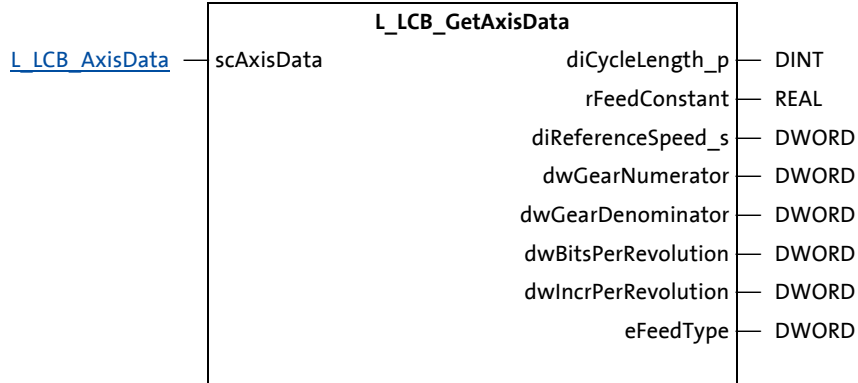
Identifier/data type	Information/possible settings
iIn_v INT	Input signal • Unit: rpm (15000 rpm $\equiv 2^{14} = 16384$)

Output

Identifier/data type	Value/meaning
diOut_s DINT	Output signal • Unit: rpm (15000 rpm $\equiv 2^{26} = 67108864$)

8.2.17 L_LCB_GetAxisData

This function block reads out selected machine parameters from the axis data transmitted and provides them to the following function blocks, requiring these data for internal calculations.



Input

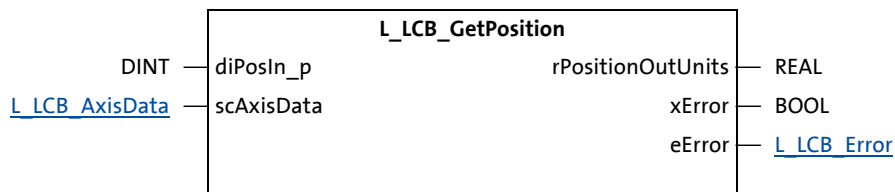
Identifier/data type	Information/possible settings
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diCycleLength_p DINT	Cycle • Unit: Increments
rFeedConstant REAL	Cycle
diReferenceSpeed_s DWORD	Reference speed as speed signal
dwGearNumerator DWORD	Gearbox factor numerator
dwGearDenominator DWORD	Gearbox factor denominator
dwBitsPerRevolution DWORD	Resolution of an encoder revolution • Unit: Bit (10 ... 24 bits)
dwIncrPerRevolution DWORD	Resolution of one revolution • Unit: Increments
eFeedType L_LCB_AXISMODE	Travel range
	0 Unlimited
	1 Limited
	2 Modulo

8.2.18 L_LCB_GetPosition

This function block used the transmitted machine parameters and converts the transferred position value in increments into a position value in the real unit of the machine.



Inputs

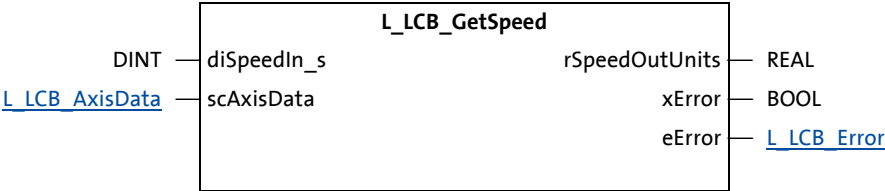
Identifier/data type	Information/possible settings
diPosIn_p DINT	Position value - Unit: Increments - Scaling: 1 encoder revolution = 2^{16} increments (or according to scAxisData)
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
rPositionOutUnits REAL	Display of the position value of the <i>diPosIn_p</i> input signal in the real unit of the machine - Unit: units - Display in code C04273
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. Error messages (102)

8.2.19 L_LCB_GetSpeed

This function block uses the transmitted machine parameters and converts the transferred speed into a velocity value.



Inputs

Identifier/data type	Information/possible settings
diSpeedIn_s DINT	Speed Unit: rpm (15000 rpm = 2²⁶ = 67108864)
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
rSpeedOutUnits REAL	Speed value Display in code C04283
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ▶ Error messages (102)

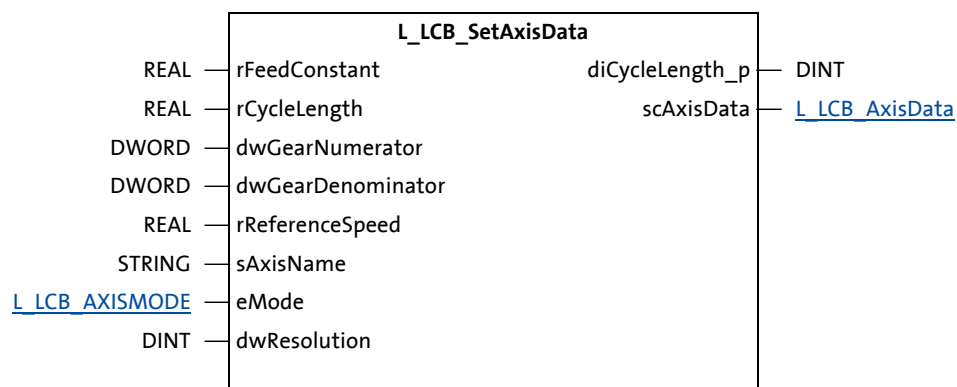
8.2.20 L_LCB_SetAxisData

This function block serves to map the machine parameters of a higher-level drive. The module prepares the machine parameters for the internal representation that are specified in the physical units of the machine via parameters.

The module outputs a pointer to the data structure [L_LCB_AxisData](#) (□ 101) at the *scAxisData* output with the prepared machine parameters.

In order to pass the machine parameters to an function block which requires these data for internal calculations, the output *scAxisData* is to be connected to the input with the same name of the corresponding function block.

Parameters that are entered via this function block are not directly transmitted to an inverter. These parameters are only used inside the PLC application for the movement-performing function blocks (e.g. L_LCB_Tablepositioning).



Inputs

Identifier/data type	Information/possible settings
rFeedConstant REAL	Feed constant - Unit: units - Initial value: 360.0
rCycleLength REAL	Cycle - Unit: units - Initial value: 360.0
dwGearNumerator DWORD	Gearbox factor numerator Initial value: 1
dwGearDenominator DWORD	Gearbox factor denominator - Initial value: 1 Example : Gearbox ratio 10:1, i.e. the motor rotates 10 times faster than the output. The parameter setting for this is as follows: - dwGearNumerator = 1 - dwGearDenominator = 10
rReferenceSpeed REAL	Reference speed
sAxisName STRING(16)	Axis name

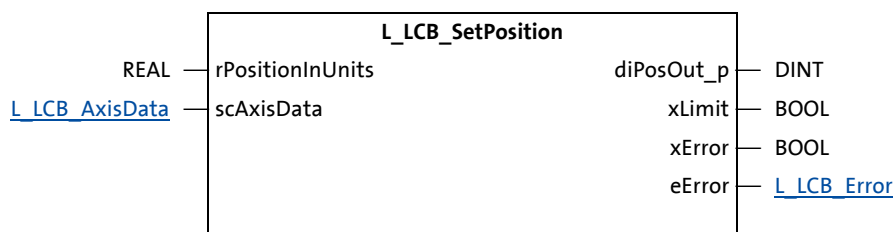
Identifier/data type	Information/possible settings						
eMode L_LCB_AXISMODE	Travel range Initial value: 0 (unlimited) <table> <tr> <td>0</td><td>Unlimited</td></tr> <tr> <td>1</td><td>Reserved</td></tr> <tr> <td>2</td><td>Modulo</td></tr> </table>	0	Unlimited	1	Reserved	2	Modulo
0	Unlimited						
1	Reserved						
2	Modulo						
dwResolution DINT	Resolution of one revolution Example: With a resolution of 65535 increments per revolution, the following value has to be set: dwResolution = 16 ($2^{16} = 65536$)						

Outputs

Identifier/data type	Value/meaning
diCycleLength_p DINT	Cycle - Unit: Increments - The cycle in the real unit of the machine can be viewed in the corresponding code.
scAxisData L_LCB_AxisData	Machine parameters - Pointer to the data structure L_LCB_AxisData (□ 101) which contains the machine parameters in the internal measuring system. - This interface provides the machine parameters of other function blocks which need this data for internal calculations.

8.2.21 L_LCB_SetPosition

This function block converts a position in 'units' to a position in 'increments' on the basis of the transmitted machine parameters and provides them at the *diPosOut_p* output for further processing.



Inputs

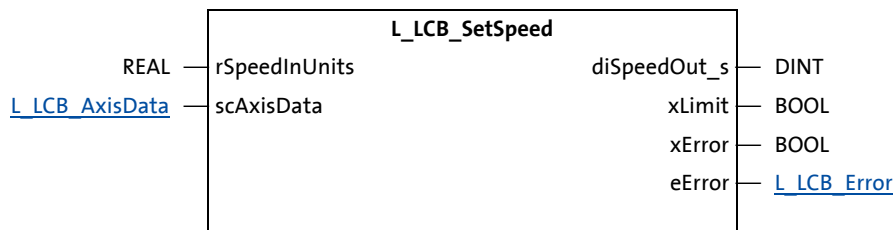
Identifier/data type	Information/possible settings
rPositionInUnits REAL	Position • Unit: units
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning	
diPosOut_p DINT	Position • Unit: Increments	
xLimit BOOL	TRUE	The output signal is limited to the displayable value range ± 214748.3647 .
xError BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)	

8.2.22 L_LCB_SetSpeed

This function block converts a velocity into a speed on the basis of the transmitted machine parameters and provides them at the *dnSpeedOut_s* output for further processing.



Inputs

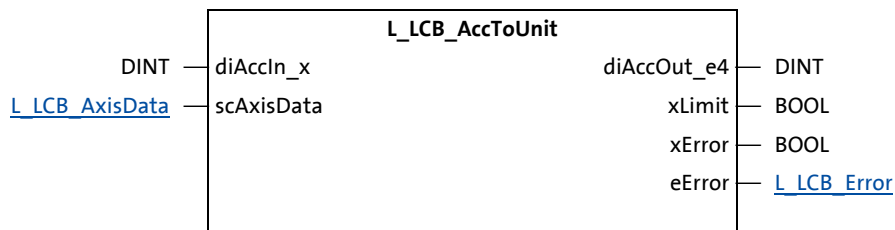
Identifier/data type	Information/possible settings
rSpeedInUnits REAL	Velocity • Unit: units
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diSpeedOut_s DINT	Output of the velocity as speed • Unit: rpm ($15000 \text{ rpm} \approx 2^{26} = 67108864$)
xLimit BOOL	TRUE The output signal is limited to the displayable value range ± 214748.3647 .
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.23 L_LCB_AccToUnit

This function block converts an acceleration value in the internal unit [rpm/s] into the real unit of the machine on the basis of the machine parameters.



Inputs

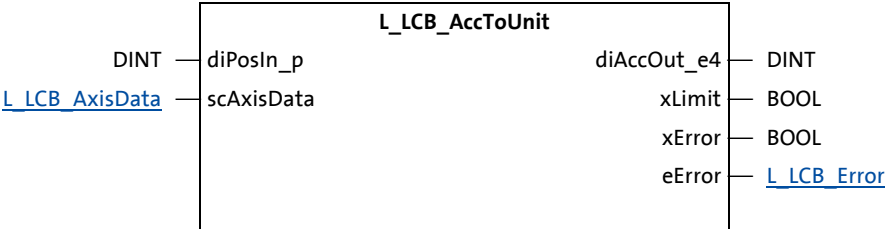
Identifier/data type	Information/possible settings
diAccIn_x DINT	Acceleration as speed variation/time • Unit: rpm/s (15000 rpm/s = 2^{22} = 4194304)
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diAccOut_e4 DINT	Acceleration in the real unit of the machine • Unit: Units/s ² • Output value with fixed point and four decimal positions
xLimit BOOL	TRUE The output signal is limited to the displayable value range ± 214748.3647 .
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.24 L_LCB_PosToUnit

This function block converts a position value in increments into the real unit of the machine on the basis of the transmitted machine parameters.



Inputs

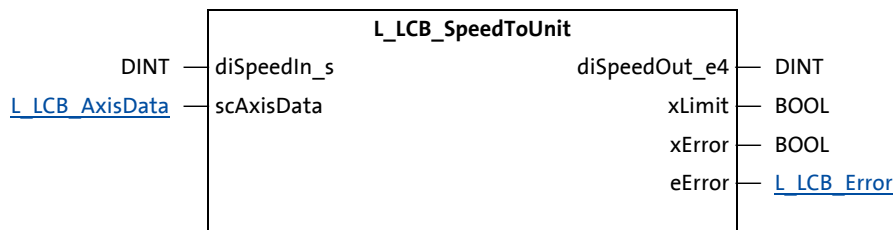
Identifier/data type	Information/possible settings
diPosIn_p DINT	Position • Unit: Increments
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diPosOut_e4 DINT	Position in the real unit of the machine • Unit: units • Output value with fixed point and four decimal positions
xLimit BOOL	TRUE The output signal is limited to the displayable value range ±214748.3647.
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.25 L_LCB_SpeedToUnit

This function block converts a velocity value in the internal unit [rpm] into the real unit of the machine on the basis of the machine parameters.



Inputs

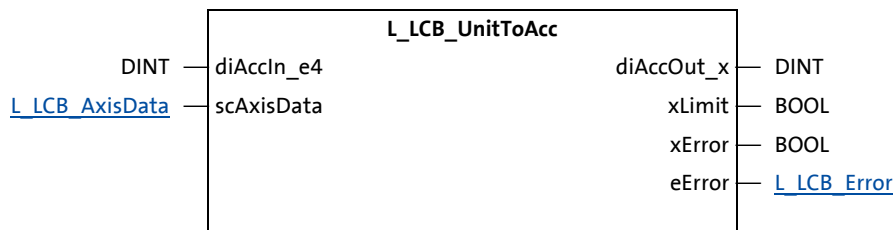
Identifier/data type	Information/possible settings
diSpeedIn_s DINT	Velocity as speed • Unit: rpm (15000 rpm = 2^{26} = 67108864)
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diSpeedOut_e4 DINT	Velocity in the real unit of the machine • Unit: units/s • Output value with fixed point and four decimal positions
xLimit BOOL	TRUE The output signal is limited to the displayable value range ± 214748.3647 .
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.26 L_LCB_UnitToAcc

This function block converts an acceleration value in the real unit of the machine into the internal unit [rpm] on the basis of the transmitted parameters.



Inputs

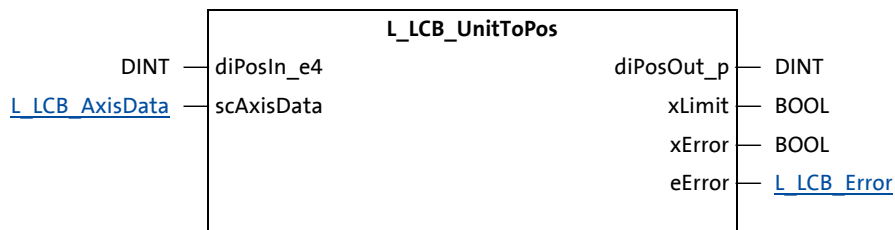
Identifier/data type	Information/possible settings
diAccIn_e4 DINT	Acceleration in the real unit of the machine - Unit: Units/s² - Default value with fixed point and four decimal positions
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diAccOut_x DINT	Acceleration as speed variation/time Unit: rpm/s (15000 rpm/s = 2²² = 4194304)
xLimit BOOL	TRUE The output signal is limited to the displayable value range ±214748.3647.
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.27 L_LCB_UnitToPos

This function block converts a position value in the real unit of the machine into increments on the basis of the transmitted machine parameters.



Inputs

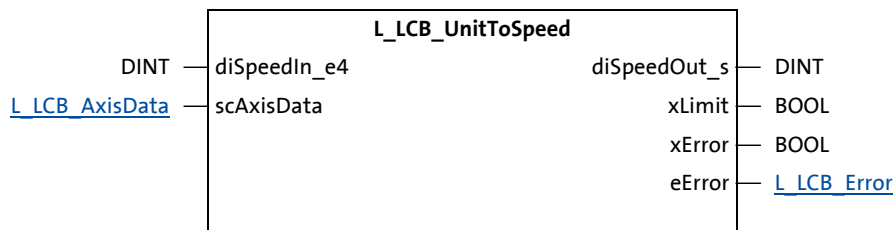
Identifier/data type	Information/possible settings
diPosIn_e4 DINT	Position in the real unit of the machine - Unit: units - Default value with fixed point and four decimal positions
scAxisData L_LCB_AxisData	Machine parameters

Outputs

Identifier/data type	Value/meaning
diPosOut_p DINT	Position Unit: Increments
xLimit BOOL	TRUE The output signal is limited to the displayable value range ± 214748.3647 .
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.28 L_LCB_UnitToSpeed

This function block converts a velocity value in the real unit of the machine into the internal unit [rpm] on the basis of the transmitted parameters.



Inputs

Identifier/data type	Information/possible settings
diSpeedIn_e4 DINT	Velocity in the real unit of the machine - Unit: units/s - Default value with fixed point and four decimal positions
scAxisData L_LCB_AxisData	Machine parameters

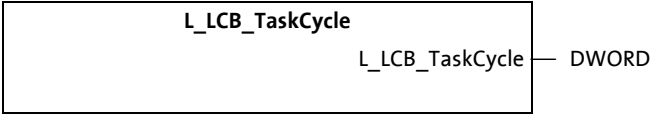
Outputs

Identifier/data type	Value/meaning
diSpeedOut_s DINT	Velocity as speed Unit: rpm ($15000 \text{ rpm} \approx 2^{26} = 67108864$)
xLimit BOOL	TRUE The output signal is limited to the value range that can be displayed.
xError BOOL	TRUE An error has occurred. For details see <i>eError</i> .
eError L_LCB_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (102)

8.2.29

L_LCB_TaskCycle

This function determines the cycle time of the task.



Output signal

Identifier/data type	Value/meaning
L_LCB_TaskCycle DWORD	Task cycle time • Unit: ms

8.3 Structures (Structs)

8.3.1 L_LCB_AxisData

Identifier/data type	Value/meaning
dwAxisID DWORD	Axis assignment
rFeedConstant REAL	Feed constant
rInvFeedConstant REAL	Inverse feed constant
rCycleLength REAL	Cycle length (only Modulo operation)
dwGearNumerator DWORD	Gearbox factor numerator
dwGearDenominator DWORD	Gearbox factor denominator
rMaxSpeed REAL	Maximum speed
rReferenceSpeed REAL	Reference speed
diReferenceSpeed_s DINT	Reference speed
sAxisName STRING[16]	Axis name
eMode L_LCB_AXISMODE	Travel range
	0 Unlimited
	1 Limited
	2 Modulo
dwResolution DWORD	Encoder resolution
rEncoderGear REAL	Resolution of an encoder revolution
rInvEncoderGear REAL	Inverted resolution of one encoder revolution
diCycleLength_p DINT	Cycle
rUnitsToPos REAL	Internal calculation value
rPosToUnits REAL	Internal calculation value
rUnitsToSpeed REAL	Internal calculation value
rSpeedToUnits REAL	Internal calculation value
rTimeUnit REAL	Internal calculation value

8.4 Enumerations (Enums)

8.4.1 L_LCB_AXISMODE

The **L_LCB_AXISMODE** type describes the travel range of the inverter in the machine parameters.

Value	Designator	Description
0	L_LCB_AM_UNLIMITED	Unlimited travel range
1	L_LCB_AM_LIMITED	Limited travel range
2	L_LCB_AM_MODULO	Modulo travel range

8.4.2 L_LCB_Error

The **L_LCB_Error** type describes the possible error messages that can be generated at the *eError* output of the function blocks of the **L_LCB_LogicControlBasic** library.

Error messages

Error number (dec)	Designator	Description
0	L_LCB_NO_ERROR	No error
20101	L_LCB_AXIS_IS_NO_AXIS_REF	The connected Axis_REF instance is invalid.
20102	L_LCB_CON_DRIVE_INVALID	Invalid drive identification of L_LCB_LOGICDEVICE type
20110	L_LCB_AXISDATA_INVALID	The connected machine parameters are invalid.

8.4.3 L_LCB_LOGICDEVICE

The **L_LCB_LOGICDEVICE** type describes the code of connected logic devices.

Value	Designator	Description
0	L_LCB_DeviceDefault	Unknown device
110	L_LCB_Device84_BL	ID for an Inverter Drive 8400 BaseLine
111	L_LCB_Device84_SL	ID for an Inverter Drive 8400 StateLine
112	L_LCB_Device84_HL	ID for an Inverter Drive 8400 HighLine
120	L_LCB_Device94_HL	ID for a Servo Drive 9400 HighLine

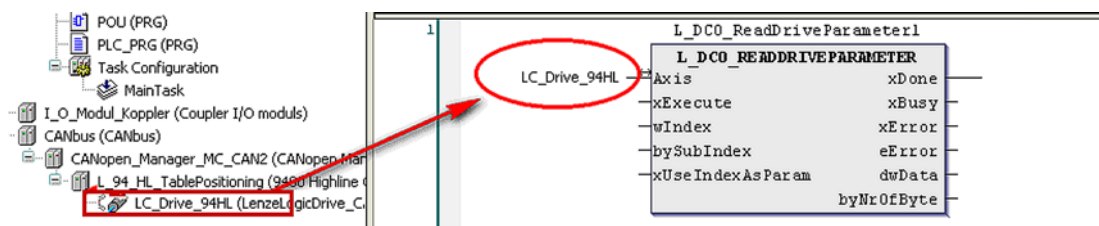
9 L_DCO_DriveCommunication function library

The **L_DCO_DriveCommunication** library contains function blocks for a bus-independent parameter communication.

The function blocks can be used together with logic axes of the inverter series 8400 and 9400 as well as in FAST Motion applications with inverters of the i-series (e.g. i500, i700).

Connecting a function block to an inverter

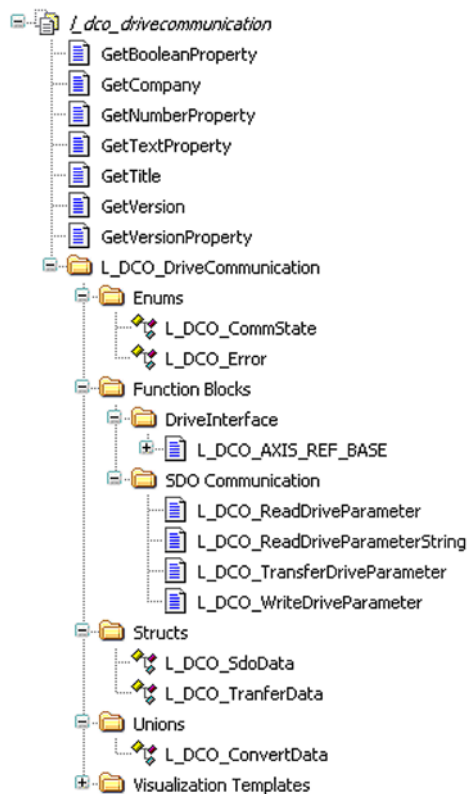
In order to connect a function block to an inverter, the axis data instance from the control configuration has to be connected to the function block.



[9-1] **Model connection:** Axis data instance of a 9400 HighLine with the **L_DCO_ReadDriveParameter** function block

9.1 Overview of the functions and function blocks

The functions and function blocks of the **L_DCO_DriveCommunication** library are divided into different groups.



Automatically generated functions (project information)

- ▶ [GetBooleanProperty](#) (131)
- ▶ [GetCompany](#) (132)
- ▶ [GetNumberProperty](#) (132)
- ▶ [GetTextProperty](#) (133)
- ▶ [GetTitle](#) (133)
- ▶ [GetVersion](#) (134)
- ▶ [GetVersionProperty](#) (134)

Enums

- ▶ [L_DCO_CommState](#) (115)
- ▶ [L_DCO_Error](#) (116)

Function Blocks

DriveInterface

- ▶ [L_DCO_AXIS_REF_BASE](#) (105)

SDO Communication

- ▶ [L_DCO_ReadDriveParameter](#) (106)
- ▶ [L_DCO_ReadDriveParameterString](#) (108)
- ▶ [L_DCO_TransferDriveParameter](#) (110)
- ▶ [L_DCO_WriteDriveParameter](#) (112)

Structs

- ▶ [L_DCO_SDOData](#) (114)
- ▶ [L_DCO_TransferData](#) (114)

Unions

- ▶ [L_DCO_ConvertData](#) (118)

9.2 Function blocks

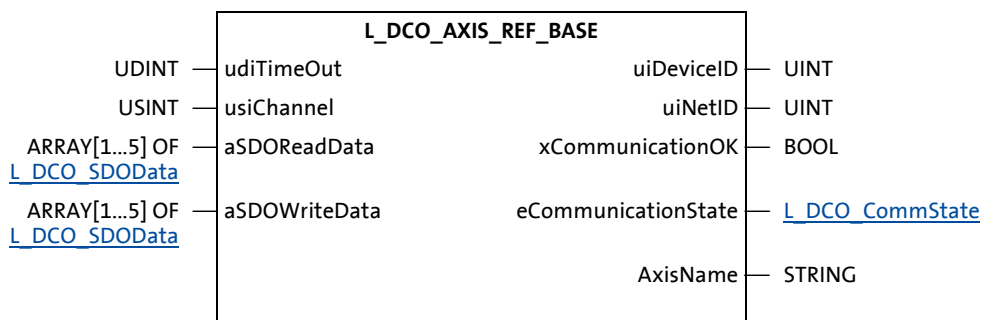
9.2.1 L_DCO_AXIS_REF_BASE

This function block provides the basic communication information of the Logic axis.

The function block can be used together with logic axes of the inverter series 8400 and 9400 as well as in FAST Motion applications with inverters of the i-series (e.g. i500, i700).

Every Logic axis implemented in the control configuration is an extended instance of **L_DCO_AXIS_REF_BASE**. A manual instancing of this function block is not required.

The function block has routines that serve as drivers.



Inputs

Identifier/data type	Information/possible settings
udiTimeOut UDINT	Timeout period for the SDO communication • Initial value: 10000
usiChannel USINT	SDO channel to be used • Initial value: 1
aSdoReadData ARRAY[1...5] OF L_DCO_SDOData	Exchange for SDO communication (internal use) • Initial value: 0
aSdoWriteData ARRAY[1...5] OF L_DCO_SDOData	Exchange for SDO communication (internal use) • Initial value: 0

Outputs

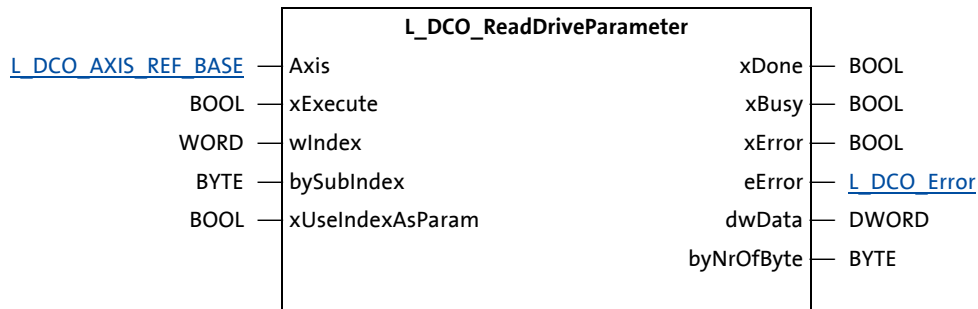
Identifier/data type	Value/meaning
uiDeviceId UINT	Station/node address at the used bus system • Initial value: 0
uiNetId UINT	Network/master instance number of the used bus system. • Initial value: 0
xCommunicationOK BOOL	TRUE Bus communication available
	FALSE No bus communication active
eCommunicationState L_DCO_CommState	Bus state • Initial value: 0
AxisName STRING	Instance name

9.2.2 L_DCO_ReadDriveParameter

This function block reads the parameters out of a Lenze inverter and provides them to the application.

- Parameter values of max. 32 bits can be read.
- The parameter to be read can be defined as index or code number.
- The function block is independent of the used bus system.

The function block can be used together with logic axes of the inverter series 8400 and 9400 as well as in FAST Motion applications with inverters of the i-series (e.g. i500, i700).



Inputs

Identifier/data type	Information/possible settings	
Axis L_DCO_AXIS_REF_BASE	Connection of the axis data of the Logic axis	
xExecute BOOL	Activation of the "Read parameters" function	
	FALSE → TRUE	Transmit parameter read request
	FALSE	Outputs are reset. • Outputs are reset after request has been completed • Status signals are set for at least one cycle.
wIndex WORD	Index of the parameter to be read • If <i>xUseIndexAsParam</i> = TRUE: Indication of the parameter/code number	
bySubIndex BYTE	Subindex of the parameter to be read	
xUseIndexAsParam BOOL	TRUE	Selection of the indices according to Lenze parameters • The value given at <i>wIndex</i> is interpreted as parameter/code number.
	FALSE	Selection of the indices according to CANopen

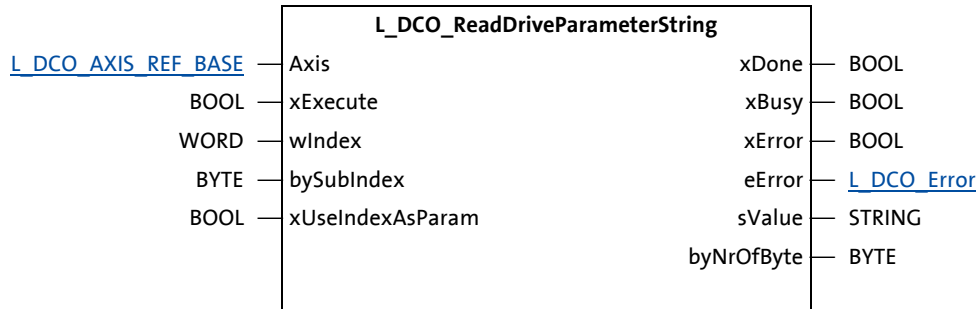
Outputs

Identifier/data type		Value/meaning	
xDone	BOOL	TRUE	Read request completed
		FALSE	Read request is currently being executed
xBusy	BOOL	TRUE	The read request has been transmitted to the system. The function block waits for the response of the request.
		FALSE	No read request active
xError	BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError	L_DCO_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (□ 116)	
dwData	DWORD	Read parameter value	
byNrOfByte	BYTE	Length of the read value	

9.2.3 L_DCO_ReadDriveParameterString

This function block reads a parameter of "STRING" data type of a Lenze inverter which is connected to the Controller via a bus system and provides this to the application.

The function block can be used together with logic axes of the inverter series 8400 and 9400 as well as in FAST Motion applications with inverters of the i-series (e.g. i500, i700).



Inputs

Identifier/data type	Information/possible settings	
Axis L_DCO_AXIS_REF_BASE	Connection of the axis data of the Logic axis	
xExecute BOOL	Activation of the "Read parameters" function	
	FALSE → TRUE	Transmit parameter read request
	FALSE	Outputs are reset. • Outputs are reset after request has been completed • Status signals are set for at least one cycle.
wIndex WORD	Index of the parameter to be read • If <i>xUseIndexAsParam</i> = TRUE: Indication of the parameter/code number	
bySubIndex BYTE	Subindex of the parameter to be read	
xUseIndexAsParam BOOL	TRUE	Selection of the indices according to Lenze parameters • The value given at <i>wIndex</i> is interpreted as parameter/code number.
	FALSE	Selection of the indices according to CANopen

Outputs

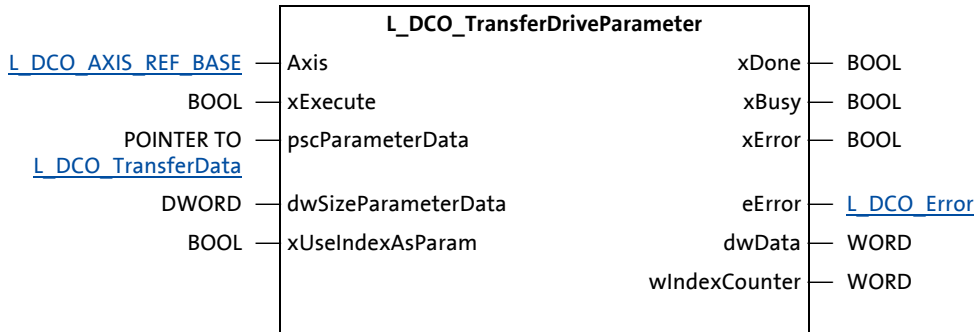
Identifier/data type		Value/meaning	
xDone	BOOL	TRUE	Read request completed
		FALSE	Read request is currently being executed
xBusy	BOOL	TRUE	The read request has been transmitted to the system. The function block waits for the response of the request.
		FALSE	No read request active
xError	BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError	L_DCO_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (□ 116)	
dwData	DWORD	Read parameter value	
byNrOfByte	BYTE	Length of the read value	

9.2.4 L_DCO_TransferDriveParameter

This function block transmits parameters of the Lenze inverter that is connected to the Controller via a bus system.

The parameters to be transmitted are defined via the array structure "ARRAY[1...n] OF L_DCO_TransferData" (see sample program [\[9-2\]](#) ([□ 111](#))).

The function block can be used together with logic axes of the inverter series 8400 and 9400 as well as in FAST Motion applications with inverters of the i-series (e.g. i500, i700).



Inputs

Identifier/data type	Information/possible settings	
Axis L_DCO_AXIS_REF_BASE	Connection of the axis data of the Logic axis	
xExecute BOOL	Activation of the "Transmit parameters" function	
	FALSE → TRUE	Transmit parameter request
	FALSE	Outputs are reset. • Outputs are reset after request has been completed • Status signals are set for at least one cycle.
pscParameterData POINTER TO L_DCO_TransferData	Here, the address of the array structure of L_DCO_TransferData (□ 114) type is transmitted	
dwSizeParameterData DWORD	Size of the array structure to be transmitted containing the codes.	
xUseIndexAsParam BOOL	TRUE	Selection of the indices according to Lenze parameters • The value given at <i>wIndex</i> is interpreted as parameter/code number.
	FALSE	Selection of the indices according to CANopen

Outputs

Identifier/data type	Value/meaning	
xDone BOOL	TRUE	Parameter request completed
	FALSE	Parameter request is currently being executed
xBusy BOOL	TRUE	The parameter request has been transmitted to the system. The function block waits for the response of the request.
	FALSE	No parameter request active
xError BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError L_DCO_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ► Error messages (□ 116)	
dwData DWORD	Read parameter value	
wIndexCounter WORD	Number of the processed parameter requests • In the event of an error, the number of the request that has caused the transmission error is indicated here.	

```

PROGRAM Demo
VAR
L_DCO_TransferDriveParameter1 : L_DCO_TransferDriveParameter;
scTransferData : ARRAY[1..5] OF L_DCO_TransferData

:= [
(wIndex := 11, bySubIndex := 0, dwData := 0, byNrOfByte := 4, xEnable := TRUE,
xWriteAccess:= FALSE ),
(wIndex := 11, bySubIndex := 0, dwData := 0, byNrOfByte := 4, xEnable := TRUE,
xWriteAccess:= FALSE ),
(wIndex := 11, bySubIndex := 0, dwData := 0, byNrOfByte := 4, xEnable := TRUE,
xWriteAccess:= FALSE ),
(wIndex := 11, bySubIndex := 0, dwData := 0, byNrOfByte := 4, xEnable := TRUE,
xWriteAccess:= FALSE ),
(wIndex := 11, bySubIndex := 0, dwData := 0, byNrOfByte := 4, xEnable := TRUE,
xWriteAccess:= FALSE ) ];

END_VAR

...

// Instance call
L_DCO_TransferDriveParameter1.pscParameterData := ADR(scTransferData);
L_DCO_TransferDriveParameter1.dwSizeParameterData := SIZEOF(scTransferData);
L_DCO_TransferDriveParameter1(Axis:=LC_Drive_94HL);

```

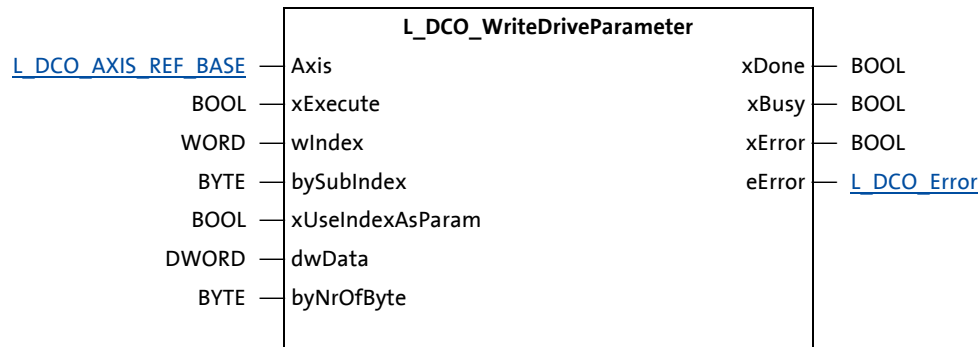
[9-2] **Sample program:** Use of the **L_DCO_TransferDriveParameter** function block

9.2.5 L_DCO_WriteDriveParameter

This function block writes a parameter of a Lenze inverter that is connected to the Controller via a bus system.

- Parameter values of max. 32 bits can be written.
- The parameter to be written can be defined as index or code number.
- The function block is independent of the used bus system.

The function block can be used together with logic axes of the inverter series 8400 and 9400 as well as in FAST Motion applications with inverters of the i-series (e.g. i500, i700).



Inputs

Identifier/data type	Information/possible settings	
Axis L_DCO_AXIS_REF_BASE	Connection of the axis data of the Logic axis	
xExecute BOOL	Activation of the "Write parameters" function	
	FALSE → TRUE	Transmit parameter write request
	FALSE	Outputs are reset. • Outputs are reset after request has been completed • Status signals are set for at least one cycle.
wIndex WORD	Index of the parameter to be written	
bySubIndex BYTE	Subindex of the parameter to be written	
xUseIndexAsParam BOOL	TRUE	Selection of the indices according to Lenze parameters • The value given at <i>wIndex</i> is interpreted as parameter/code number.
	FALSE	Selection of the indices according to CANopen
dwData DWORD	Parameter value to be written	
byNrOfByte BYTE	Number of the bytes to be written	

Outputs

Identifier/data type		Value/meaning	
xDone	BOOL	TRUE	Write request completed
		FALSE	Write request is currently being executed
xBusy	BOOL	TRUE	The write request has been transmitted to the system. The function block waits for the response of the request.
		FALSE	No write request active
xError	BOOL	TRUE	An error has occurred. For details see <i>eError</i> .
eError	L_DCO_Error	If <i>xError</i> = TRUE: Error number for the error that has occurred. ▶ Error messages (116)	

9.3 Structures (Structs)

9.3.1 L_DCO_SDOData

The **L_DCO_SDOData** structure serves to exchange parameter write requests and parameter read requests.

This structure is only used internally!

Identifier/data type	Value/meaning
wParam WORD	Index number of the parameter to be processed • Initial value: 0
bySubIndex BYTE	Subindex of the parameter to be processed • Initial value: 0
wState WORD	Status of the parameter request • Initial value: 0
dwOwner DWORD	Identification of the orderer • Initial value: 0
dwData DWORD	Transmitted/read data • Initial value: 0
psData POINTER OF BYTE	Pointer to a byte into which the read data is to be written. • Initial value: 0
uDataLength UNIT	Data length of the parameter • Initial value: 0
wError WORD	Error number • Initial value: 0

9.3.2 L_DCO_TransferData

The **L_DCO_TransferData** structure describes a request for transmitting a parameter.

Using the [L_DCO_TransferDriveParameter](#) (□ 110) function block serves to transmit several parameters from a drive via an "ARRAY[1...n] OF L_DCO_TransferData".

Identifier/data type	Description
wIndex WORD	Index number of the parameter to be transmitted • Initial value: 0
bySubIndex BYTE	Subindex of the parameter to be transmitted • Initial value: 0
xWriteAccess BOOL	Write or read access • Initial value: FALSE
	TRUE Write access
	FALSE Read access
xEnable BOOL	Activation of the parameter request • Initial value: FALSE
	TRUE Parameter request is processed
	FALSE Parameter request is ignored
byNrOfByte BYTE	Number of the transmitted/read bytes • Initial value: 0
dwData DWORD	Transmitted/read data • Initial value: 0

9.4 Enumerations (Enums)

9.4.1 L_DCO_CommState

The **L_DCO_CommState** type describes possible states that can be adopted by the fieldbus nodes (master and slaves).

The communication status is available via an instance of the [L_DCO_AXIS_REF_BASE](#) (105) function block and its extensions.

The respective function block instance is provided in the device editor.

Value	Bezeichner	Description
0	L_DCO_STATE_BOOT	"Boot process" state
10	L_DCO_STATE_INIT	"Initialisation" state
20	L_DCO_STATE_RESET_COMM	"Communication reset" state
30	L_DCO_STATE_RESET_APP	"Reset" state
40	L_DCO_STATE_PREOPERATIONAL	"Preoperational" state
50	L_DCO_STATE_STOPPED	"Stopped" state
70	L_DCO_STATE_OPERATIONAL	"Operational" state
80	L_DCO_STATE_UNKNOWN	"Unknown" state
90	L_DCO_STATE_NOT_AVAIL	"Not available" state

9.4.2 L_DCO_Error

The **L_DCO_Error** type describes the possible error messages that can be generated at the *eError* output of the function blocks of the **L_DCO_DriveCommunication** library.

Error messages

Error number (dec)	Bezeichner	Description
0	L_DCO_No_Error	No error reported
20001	L_DCO_OTHERERROR	General error
20002	L_DCO_INVALIDINDEX	Invalid SDO index
20003	L_DCO_INVALIDOFFSET	Invalid offset value during access to process data image
20004	L_DCO_INVALIDSIZE	Invalid length value... • when the process data image is accessed • when data is saved in a data area
20005	L_DCO_INVALIDDATA	Invalid data
20006	L_DCO_NOTREADY	Internal software error • Cause not reported
20007	L_DCO_BUSY	The master is busy at the moment and cannot process the API function. • Repeat the function at a later time.
20008	L_DCO_NOMEMORY	Not enough application memory available.
20009	L_DCO_INVALIDPARM	Incorrect call of a API function by invalid parameters.
20010	L_DCO_INVALIDSTATE	Invalid state
20011	L_DCO_TIMEOUT	Time-out
20012	L_DCO_INVALID_SLAVE_STATE	Impermissible mailbox commands with current slave state.
20013	L_DCO_LINK_DISCONNECTED	EtherCAT cable is not connected to control.
20015	L_DCO_ALL_CHANNELS_BUSY	All channels for processing parameter request are busy.
20016	L_DCO_DATA_REQUEST_ERROR	General error reported at request
20017	L_DCO_BUS_OFF	Bus not initialised
20018	L_DCO_SLAVE_ERROR	Slave not addressable
20020	L_DCO_SDO_ABORTCODE_TOGGLE	Unchanged state of the toggle bit • Abort Code 0x05030000
20021	L_DCO_SDO_ABORTCODE_TIMEOUT	Time-out of the SDO protocol • Abort code 0x05040000
20022	L_DCO_SDO_ABORTCODE_CCS_SCS	Invalid or unknown specification symbol for the client/server command • Abort Code 0x05040001
20023	L_DCO_SDO_ABORTCODE_BLK_SIZE	Invalid block size (only on "Block mode") • Abort Code 0x05040002
20024	L_DCO_SDO_ABORTCODE_SEQNO	Invalid sequence number (only in "Block mode") • Abort Code 0x05040003
20025	L_DCO_SDO_ABORTCODE_CRC	CRC error (only in "Block mode") • Abort Code 0x05040004
20026	L_DCO_SDO_ABORTCODE_MEMORY	Not enough memory space available in the main memory • Abort code 0x05040005
20027	L_DCO_SDO_ABORTCODE_ACCESS	Access to an object denied • Abort Code 0x06010000

Error number (dec)	Bezeichner	Description
20028	L_DCO_SDO_ABORTCODE_WRITEONLY	Read access to a write-protected object • Abort Code 0x06010001
20029	L_DCO_SDO_ABORTCODE_READONLY	Write access to a write-protected object • Abort Code 0x06010002
20030	L_DCO_SDO_ABORTCODE_INDEX	Object not found in the object directory. • Abort Code 0x06020000
20031	L_DCO_SDO_ABORTCODE_PDO_MAP	An object cannot be mapped into the PDO • Abort code 0x06040041
20032	L_DCO_SDO_ABORTCODE_PDO_LEN	Number/length of the mapped objects exceeds the PDO length. • Abort Code 0x06040042
20033	L_DCO_SDO_ABORTCODE_P_INCOMP	General message for parameter incompatibility • Abort Code 0x06040043
20034	L_DCO_SDO_ABORTCODE_I_INCOMP	General message for parameter incompatibility • Abort Code 0x06040043
20035	L_DCO_SDO_ABORTCODE_HARDWARE	Failed access by hardware error • Abort Code 0x06060000
20036	L_DCO_SDO_ABORTCODE_DATA_SIZE	Data type or parameter length differ • Abort code 0x06070010
20037	L_DCO_SDO_ABORTCODE_DATA_SIZE1	Incorrect data type (The parameter length is too large) • Abort Code 0x06070012
20038	L_DCO_SDO_ABORTCODE_DATA_SIZE2	Incorrect data type (The parameter length is too small) • Abort Code 0x06070013
20039	L_DCO_SDO_ABORTCODE_OFFSET	Missing subindex • Abort Code 0x06090011
20040	L_DCO_SDO_ABORTCODE_DATA_RANGE	The value range for parameters is too great (only for write access) • Abort Code 0x06090030
20041	L_DCO_SDO_ABORTCODE_DATA_RANGE1	The parameter value is too high • Abort code 0x06090031
20042	L_DCO_SDO_ABORTCODE_DATA_RANGE2	The parameter value is too low • Abort Code 0x06090032
20043	L_DCO_SDO_ABORTCODE_MINMAX	The maximum value is lower than the minimum value • Abort Code 0x06090036
20044	L_DCO_SDO_ABORTCODE_GENERAL	General error • Abort code 0x08000000
20045	L_DCO_SDO_ABORTCODE_TRANSFER	Error when saving/transmitting the data in the application • Abort Code 0x08000020
20046	L_DCO_SDO_ABORTCODE_TRANSFER1	Error when saving/transmitting the data in the application (caused by local control) • Abort Code 0x08000021
20047	L_DCO_SDO_ABORTCODE_TRANSFER2	Error when saving/transmitting the data in the application (caused by current device state) • Abort Code 0x08000022
20048	L_DCO_SDO_ABORTCODE_DICTIONARY	Dynamic object directory generation has failed or no object directory available • Abort code 0x08000023
20049	L_DCO_SDO_ABORTCODE_UNKNOWN	Internal slave error (cause unknown)
20060	L_DCO_TRANSFERTABLE_INVALID	Invalid structure L_DCO_TransferData (114) transmitted to L_DCO_TransferDriveParameter (110) function block.

9.5

Unions

9.5.1

L_DCO_ConvertData

Provides different access possibilities to a 32-bit memory area.

Identifier/data type	Description
abData ARRAY OF BYTE	Transmitted/read data
awData ARRAY OF WORD	Transmitted/read data
dwData DWORD	Transmitted/read data

10

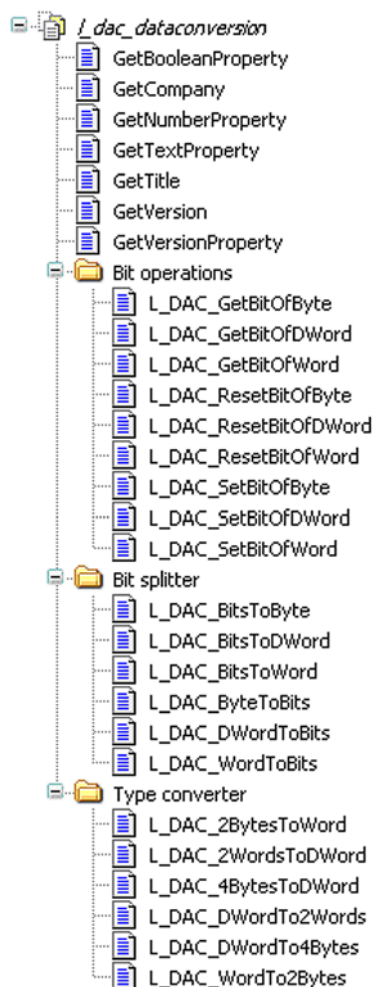
L_DAC_DataConversion function library

The L_DAC_DataConversion library contains functions and function blocks for converting data.

10.1

Overview of the functions and function blocks

The functions and function blocks of the L_DAC_DataConversion library are divided into different groups.



Automatically generated functions (project information)

- ▶ [GetBooleanProperty](#) (131)
- ▶ [GetCompany](#) (132)
- ▶ [GetNumberProperty](#) (132)
- ▶ [GetTextProperty](#) (133)
- ▶ [GetTitle](#) (133)
- ▶ [GetVersion](#) (134)
- ▶ [GetVersionProperty](#) (134)

Bit operations

- ▶ [L_DAC_GetBitOfByte](#) (120)
- ▶ [L_DAC_GetBitOfDWord](#) (120)
- ▶ [L_DAC_GetBitOfWord](#) (121)
- ▶ [L_DAC_ResetBitOfByte](#) (121)
- ▶ [L_DAC_ResetBitOfDWord](#) (122)
- ▶ [L_DAC_ResetBitOfWord](#) (122)
- ▶ [L_DAC_SetBitOfByte](#) (123)
- ▶ [L_DAC_SetBitOfDWord](#) (123)
- ▶ [L_DAC_SetBitOfWord](#) (124)

Bit splitter

- ▶ [L_DAC_BitsToByte](#) (125)
- ▶ [L_DAC_BitsToDWord](#) (125)
- ▶ [L_DAC_BitsToWord](#) (126)
- ▶ [L_DAC_ByteToBits](#) (126)
- ▶ [L_DAC_DWordToBits](#) (127)
- ▶ [L_DAC_WordToBits](#) (127)

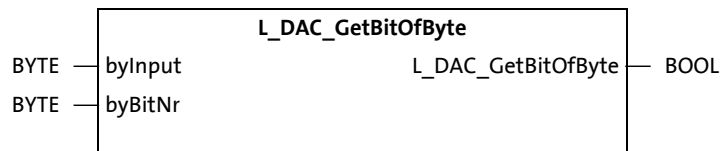
Type converter

- ▶ [L_DAC_2BytesToWord](#) (128)
- ▶ [L_DAC_2WordsToDWord](#) (128)
- ▶ [L_DAC_4BytesToDWord](#) (129)
- ▶ [L_DAC_DWordTo2Words](#) (129)
- ▶ [L_DAC_DWordTo4Bytes](#) (130)
- ▶ [L_DAC_WordTo2Bytes](#) (130)

10.2 Bit operations

10.2.1 L_DAC_GetBitOfByte

This function returns the state ('TRUE' or 'FALSE') of a single bit within a "BYTE" value.



Inputs

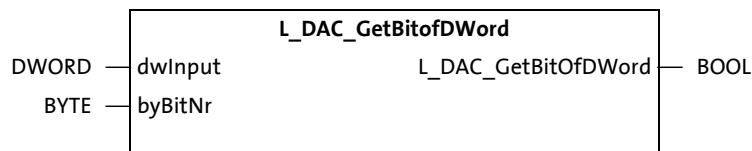
Identifier/data type	Information/possible settings
byInput BYTE	Input signal
byBitNr BYTE	Number (0 ... 7) of the bit of <i>byInput</i> , the status of which is to be determined.

Output

Identifier/data type	Value/meaning
L_DAC_GetBitOfByte BOOL	State ('TRUE' or 'FALSE') of bit <i>byBitNr</i> from the <i>byInput</i> input signal.

10.2.2 L_DAC_GetBitOfDWord

This function returns the state ('TRUE' or 'FALSE') of a single bit within a "DWORD" value.



Inputs

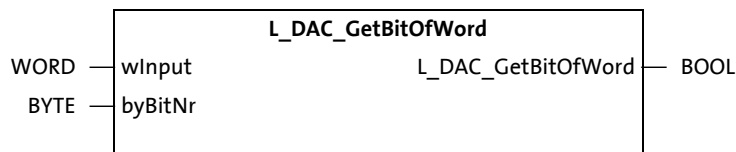
Identifier/data type	Information/possible settings
dwInput DWORD	Input signal
byBitNr BYTE	Number (0 ... 31) of the bit of <i>dwInput</i> , the status of which is to be determined.

Output

Identifier/data type	Value/meaning
L_DAC_GetBitOfDWord BOOL	State ('TRUE' or 'FALSE') of bit <i>byBitNr</i> from the <i>dwInput</i> input signal.

10.2.3 L_DAC_GetBitOfWord

This function returns the state ('TRUE' or 'FALSE') of a single bit within a "WORD" value.



Inputs

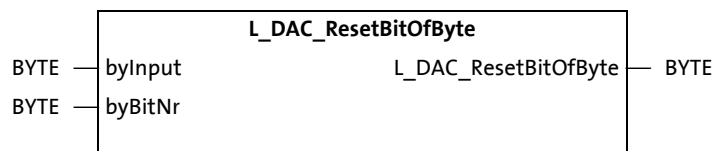
Identifier/data type	Information/possible settings
wInput WORD	Input signal
byBitNr BYTE	Number (0 ... 15) of the bit of dwInput, the status of which is to be determined.

Output

Identifier/data type	Value/meaning
L_DAC_GetBitOfWord BOOL	State ('TRUE' or 'FALSE') of bit <i>byBitNr</i> from the <i>wInput</i> input signal.

10.2.4 L_DAC_ResetBitOfByte

This function sets the state of a single bit to 'FALSE' with a "BYTE" value.



Inputs

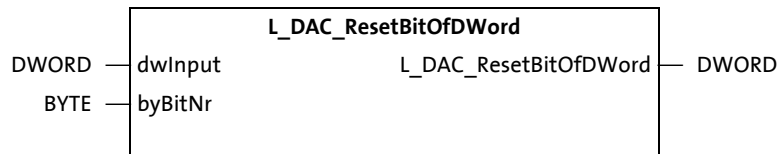
Identifier/data type	Information/possible settings
byInput BYTE	Input signal
byBitNr BYTE	Number (0 ... 7) of the bit that is to be set to 'FALSE'.

Output

Identifier/data type	Value/meaning
L_DAC_ResetBitOfByte BYTE	Output signal The value results from bit <i>byBitNr</i> set to 'FALSE' from the <i>byInput</i> input signal.

10.2.5 L_DAC_ResetBitOfDWord

This function sets the state of a single bit to 'FALSE' with a "DWORD" value.



Inputs

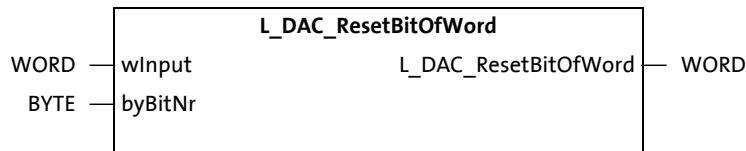
Identifier/data type	Information/possible settings
dwInput DWORD	Input signal
byBitNr BYTE	Number (0 ... 31) of the bit that is to be set to 'FALSE'.

Output

Identifier/data type	Value/meaning
L_DAC_ResetBitOfDWord DWORD	Output signal The value results from bit <i>byBitNr</i> set to 'FALSE' from the <i>dwInput</i> input signal.

10.2.6 L_DAC_ResetBitOfWord

This function sets the state of a single bit to 'FALSE' with a "WORD" value.



Inputs

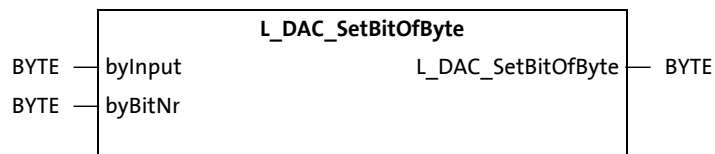
Identifier/data type	Information/possible settings
wInput WORD	Input signal
byBitNr BYTE	Number (0 ... 15) of the bit that is to be set to 'FALSE'.

Output

Identifier/data type	Value/meaning
L_DAC_ResetBitOfWord WORD	Output signal The value results from bit <i>byBitNr</i> set to 'FALSE' from the <i>wInput</i> input signal.

10.2.7 L_DAC_SetBitOfByte

This function sets the state of a single bit to 'TRUE' with a "BYTE" value.



Inputs

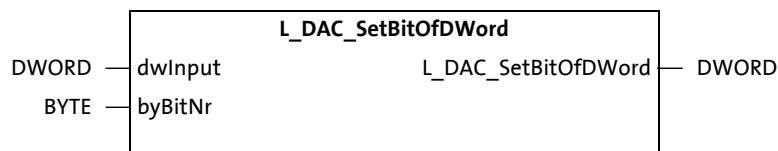
Identifier/data type	Information/possible settings
byInput BYTE	Input signal
byBitNr BYTE	Number (0 ... 7) of the bit that is to be set to 'True'.

Output

Identifier/data type	Value/meaning
L_DAC_SetBitOfByte BYTE	Output signal The value results from bit <i>byBitNr</i> set to 'TRUE' from the <i>byInput</i> input signal.

10.2.8 L_DAC_SetBitOfDWord

This function sets the state of a single bit to 'TRUE' within a "DWORD" value.



Inputs

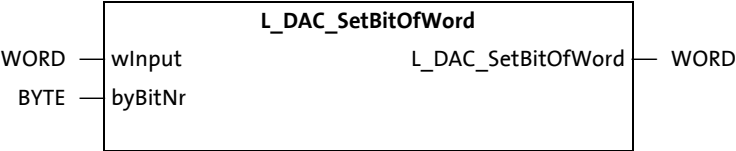
Identifier/data type	Information/possible settings
dwInput DWORD	Input signal
byBitNr BYTE	Number (0 ... 31) of the bit that is to be set to 'True'.

Output

Identifier/data type	Value/meaning
L_DAC_SetBitOfDWord BYTE	Output signal The value results from bit <i>byBitNr</i> set to 'TRUE' from the <i>dwInput</i> input signal.

10.2.9 L_DAC_SetBitOfWord

This function sets the state of a single bit to 'TRUE' within a "WORD" value.



Inputs

Identifier/data type	Information/possible settings
wInput WORD	Input signal
byBitNr BYTE	Number (0 ... 15) of the bit that is to be set to 'True'.

Output

Identifier/data type	Value/meaning
L_DAC_SetBitOfWord BYTE	Output signal • The value results from bit <i>byBitNr</i> set to 'TRUE' from the <i>wInput</i> input signal.

10.3 Bit splitter

10.3.1 L_DAC_BitsToByte

This function block converts 8 input bits to a value of the "BYTE" type.



Inputs

Identifier/data type	Information/possible settings
xBit[0...7] BOOL	Input signal of bits 0 ... 7

Outputs

Identifier/data type	Value/meaning
byOutput BYTE	Output value resulting from the input bits <i>xBit[0...7]</i> .

10.3.2 L_DAC_BitsToDWord

This function block converts 32 input bits into a value of the "DWORD" type.



Inputs

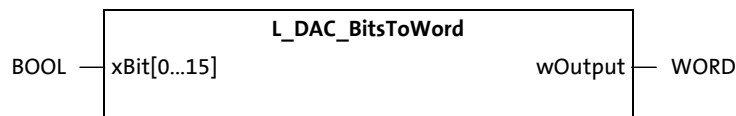
Identifier/data type	Information/possible settings
xBit[0...31] BOOL	Input signal of bits 0 ... 31

Outputs

Identifier/data type	Value/meaning
dwOutput DWORD	Output value resulting from the input bits <i>xBit[0...31]</i> .

10.3.3 L_DAC_BitsToWorld

This function block converts 16 input bits into a value of the "WORD" type.



Inputs

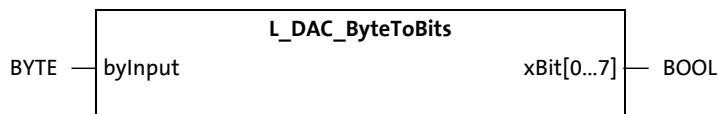
Identifier/data type	Information/possible settings
xBit[0...15] BOOL	Input signal of bits 0 ... 15

Outputs

Identifier/data type	Value/meaning
wOutput WORD	Output value resulting from the input bits <i>xBit[0...15]</i> .

10.3.4 L_DAC_ByteToBits

This function block outputs the 8 corresponding bit values for an input value of the "BYTE" type.



Inputs

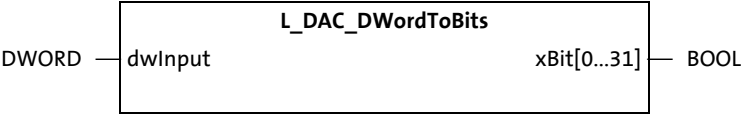
Identifier/data type	Information/possible settings
byInput BYTE	Input signal

Outputs

Identifier/data type	Value/meaning
xBit[0...7] BOOL	Output values for the bits 0 ... 7 resulting from the <i>byInput</i> input value. • Valency: $2^0 \dots 2^7$

10.3.5 L_DAC_DWordToBits

This function block outputs the 32 corresponding bit values for an input value of the "DWORD" type.



Inputs

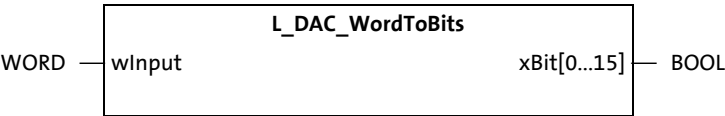
Identifier/data type	Information/possible settings
dwInput DWORD	Input signal

Outputs

Identifier/data type	Value/meaning
xBit[0...31] BOOL	Output values for the bits 0 ... 31 resulting from the <i>dwInput</i> input value. • Valency: $2^0 \dots 2^{31}$

10.3.6 L_DAC_WordToBits

This function block outputs the 16 corresponding bit values for an input value of the "WORD" type.



Inputs

Identifier/data type	Information/possible settings
wInput WORD	Input signal

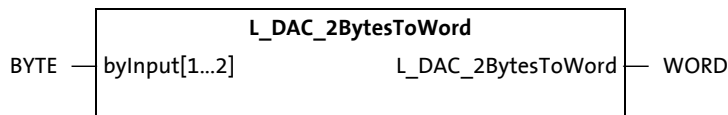
Outputs

Identifier/data type	Value/meaning
xBit[0...15] BOOL	Output values for the bits 0 ... 15 resulting from the <i>wInput</i> input value. • Valency: $2^0 \dots 2^{15}$

10.4 Type converter

10.4.1 L_DAC_2BytesToWord

This function converts two input values of the "BYTE" type into an output value of the "WORD" type.



Inputs

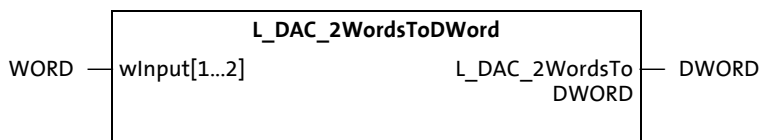
Identifier/data type	Information/possible settings
byInput1 BYTE	Input value 1 = bit 0 ... 7 (byte 1)
byInput2 BYTE	Input value 2 = bit 8 ... 15 (byte 2)

Outputs

Identifier/data type	Information/possible settings
L_DAC_2BytesToWord WORD	Output value resulting from the input values <i>byInput1</i> (byte 1) and <i>byInput2</i> (byte 2).

10.4.2 L_DAC_2WordsToDWord

This function converts two input values of the "WORD" type into an output value of the "DWORD" type.



Inputs

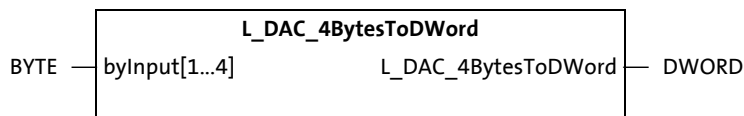
Identifier/data type	Information/possible settings
wInput1 WORD	Input value 1 = bit 0 ... 7 (word 1)
wInput2 WORD	Input value 2 = bit 8 ... 15 (word 2)

Outputs

Identifier/data type	Information/possible settings
L_DAC_2WordsToDWord DWORD	Output value resulting from the input values <i>wInput1</i> (word 1) and <i>wInput2</i> (word 2).

10.4.3 L_DAC_4BytesToDWord

This function converts four input values of the "BYTE" into an output value of the "DWORD" type.



Inputs

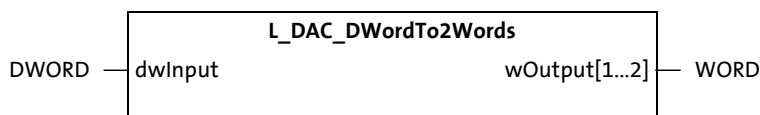
Identifier/data type	Information/possible settings
byInput1 BYTE	Input value 1 = bit 0 ... 7 (byte 1)
byInput2 BYTE	Input value 2 = bit 8 ... 15 (byte 2)
byInput3 BYTE	Input value 3 = bit 16 ... 23 (byte 3)
byInput4 BYTE	Input value 4 = bit 24 ... 31 (byte 4)

Outputs

Identifier/data type	Information/possible settings
L_DAC_4BytesToDWord DWORD	Output value resulting from the input values <i>byInput[1...4]</i> (bytes 1 ... 4).

10.4.4 L_DAC_DWordTo2Words

This function block converts an input value of the "DWORD" type into two output values of the "WORD" type.



Inputs

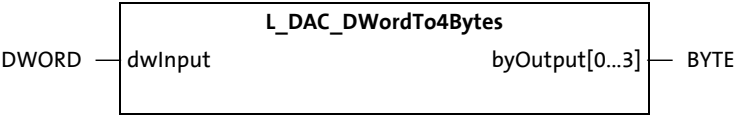
Identifier/data type	Information/possible settings
dwInput DWORD	Input value

Outputs

Identifier/data type	Value/meaning
wOutput1 WORD	Output value 1 = bit 0...15 of <i>dwInput</i>
wOutput2 WORD	Output value 2 = bit 16...31 of <i>dwInput</i>

10.4.5 L_DAC_DWordTo4Bytes

This function block converts an input value of the "DWORD" type into four output values of the "BYTE" type.



Inputs

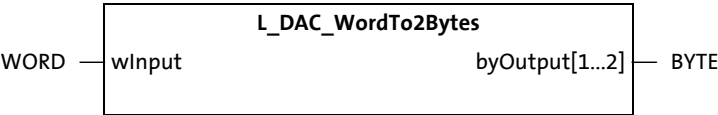
Identifier/data type	Information/possible settings
dwInput DWORD	Input value

Outputs

Identifier/data type	Value/meaning
byOutput0 BYTE	Output value 1 = bit 0 ... 7 of <i>dwInput</i>
byOutput1 BYTE	Output value 2 = bit 8 ... 15 of <i>dwInput</i>
byOutput2 BYTE	Output value 3 = bit 16 ... 23 of <i>dwInput</i>
byOutput3 BYTE	Output value 4 = bit 24 ... 31 of <i>dwInput</i>

10.4.6 L_DAC_WordTo2Bytes

This function block converts an input value of the "WORD" type into two output values of the "BYTE" type.



Inputs

Identifier/data type	Information/possible settings
wInput WORD	Input value

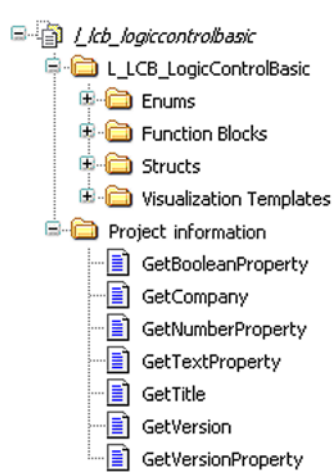
Outputs

Identifier/data type	Value/meaning
byOutput1 BYTE	Output value 1 = bit 0 ... 7 of <i>wInput</i>
byOutput2 BYTE	Output value 2 = bit 8 ... 15 of <i>wInput</i>

11 Automatically generated functions (project information)

This section informs about the functions automatically generated by the »PLC Designer«.

The output values can be found under the **project → Project information** menu command.



Project information

- ▶ [GetBooleanProperty](#) (📖 131)
- ▶ [GetCompany](#) (📖 132)
- ▶ [GetNumberProperty](#) (📖 132)
- ▶ [GetTextProperty](#) (📖 133)
- ▶ [GetTitle](#) (📖 133)
- ▶ [GetVersion](#) (📖 134)
- ▶ [GetVersionProperty](#) (📖 134)

11.1 GetBooleanProperty

This function provides a Boolean project information of the library.



Note!

This function must always be used with the corresponding name area of the related library, e.g. "**L_DAC**.GetBooleanProperty(stKey)".



Inputs

Identifier/data type	Information/possible settings
stKey WSTRING	Key of the project information

Outputs

Identifier/data type	Value/meaning
GetBooleanProperty BOOL	Value of the project information

11.2

GetCompany

This function provides the name of the company that has created this library.

**Note!**

This function must always be used with the corresponding name area of the related library, e.g. "**L_DAC**.GetCompany ()".

**Outputs**

Identifier/data type	Value/meaning
GetCompany WSTRING	Name of the company that created the library.

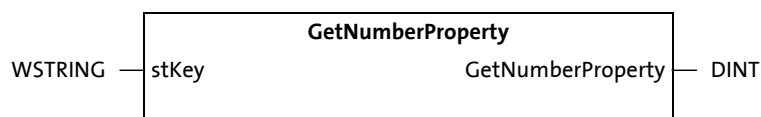
11.3

GetNumberProperty

This function provides a numerical project information of the library.

**Note!**

This function must always be used with the corresponding name area of the related library, e.g. "**L_DAC**.GetNumberProperty (stKey)".

**Inputs**

Identifier/data type	Information/possible settings
stKey WSTRING	Key of the project information

Outputs

Identifier/data type	Value/meaning
GetNumberProperty DINT	Value of the project information

11.4

GetTextProperty

This function provides a textual project information of the library.

**Note!**

This function must always be used with the corresponding name area of the related library, e.g. `"L_DAC.GetTextProperty(stKey)"`.

**Inputs**

Identifier/data type	Information/possible settings
stKey WSTRING	Key of the project information • "Company" - name of the library provider • "Title" - name of the library • "DefaultNamespace" - name area of the library • "Project" - name of the library • "Description" - description of the library • "Author" - author of the library

Outputs

Identifier/data type	Value/meaning
GetTextProperty WSTRING	Value of the project information

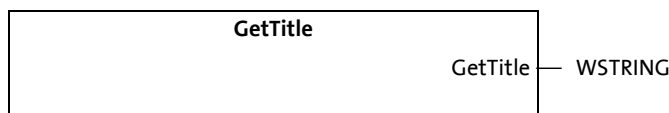
11.5

GetTitle

This function provides the title of the library.

**Note!**

This function must always be used with the corresponding name area of the related library, e.g. `"L_DAC.GetTitle"`.

**Outputs**

Identifier/data type	Value/meaning
GetTitle WSTRING	Name of the library

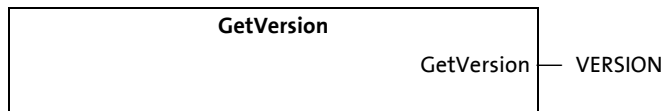
11.6

GetVersion

The function provides the version of the library.

**Note!**

This function must always be used with the corresponding name area of the related library, e.g. "**L_DAC**.GetVersion".

**Outputs**

Identifier/data type	Value/meaning
GetVersion VERSION	Version of the library

11.7

GetVersionProperty

The function provides the version of the library.

**Note!**

This function must always be used with the corresponding name area of the related library, e.g. "**L_DAC**.GetVersion(stKey)".

**Inputs**

Identifier/data type	Information/possible settings
stKey WSTRING	Key of the project information • Extended versioning information of the library

Outputs

Identifier/data type	Value/meaning
GetVersionProperty VERSION	Version of the library

12 Index

A

Activate position override [77](#)
Adding CAN field devices [46](#)
Adding devices (EtherCAT) [43](#)
Adding EtherCAT field devices [43](#)
Application examples [9](#)
Application notes (representation) [13](#)
Application Samples [19](#)
Application samples - program blocks [29](#)
Application Samples - visualisations [30](#), [36](#)
Application Samples, name [20](#)
Application samples, overview [20](#)
Application samples, structure [19](#)
Assignment of the input ports (L_LCB_ActuatorSpeed) [75](#)
Assignment of the input ports (L_LCB_TablePositioning) [81](#)
Assignment of the output ports (L_LCB_ActuatorSpeed) [76](#)
Assignment of the output ports (L_LCB_TablePositioning) [82](#)
Automatic ("speed actuating drive" visualisation) [32](#)
Automatic ("table positioning" visualisation) [38](#)
Automatically generated functions (project information) [131](#)

B

Bit operations [120](#)
Bit splitter [125](#)

C

CAN control configuration [26](#)
CANbus [48](#)
CANopen Manager [49](#)
Conditions [16](#)
Connecting a function block [65](#), [103](#)
Connection of function blocks [65](#), [103](#)
Control configuration (EtherCAT/CAN) [26](#)
Conventions used [11](#)

D

Device descriptions [50](#)
Devices of other manufacturers [43](#)
Document history [10](#)

E

E-mail to Lenze [138](#)
Enumerations (L_DCO_DriveCommunication) [115](#)
Enumerations (L_LCB_LogicControlBasic) [102](#)
Error messages (L_LCB_Error) [102](#), [116](#)
Establishing communication with the controller [18](#)
EtherCAT control configuration [26](#)
Extending sample projects (adding devices) [43](#)

F

Feedback to Lenze [138](#)
Feldbus-Scan (EtherCAT) [43](#)
Function blocks (L_DCO_DriveCommunication) [105](#)
Function blocks (L_LCB_LogicControlBasic) [67](#)
Functions and function blocks (overview L_DAC_DataConversion) [119](#)
Functions and function blocks (overview L_DCO_DriveCommunication) [104](#)
Functions and function blocks (overview L_LCB_LogicControlBasic) [66](#)

G

Generic Drive [63](#)
GetBooleanProperty [131](#)
GetCompany [132](#)
GetNumberProperty [132](#)
GetTextProperty [133](#)
GetTitle [133](#)
GetVersion [134](#)
GetVersionProperty [134](#)

H

Homing ("table positioning" visualisation) [41](#)

I

Information ("speed actuating drive" visualisation) [31](#)
Information ("table positioning" visualisation) [37](#)
Inverter Drive 8400 Baseline C [50](#)
Inverter Drive 8400 HighLine C [54](#)
Inverter Drive 8400 StateLine C [52](#)
Inverter Drive 8400 TopLine C [56](#)

L

[L_DAC_2BytesToWord](#) [128](#)
[L_DAC_2WordsToDWord](#) [128](#)
[L_DAC_4BytesToDWord](#) [129](#)
[L_DAC_BitsToByte](#) [125](#)
[L_DAC_BitsToDWord](#) [125](#)
[L_DAC_BitsToWord](#) [126](#)
[L_DAC_ByteToBits](#) [126](#)
[L_DAC_DataConversion](#) [119](#)
[L_DAC_DataConversion function library](#) [119](#)
[L_DAC_DWordTo2Words](#) [129](#)
[L_DAC_DWordTo4Bytes](#) [130](#)
[L_DAC_DWordToBits](#) [127](#)
[L_DAC_GetBitOfByte](#) [120](#)
[L_DAC_GetBitOfDWord](#) [120](#)
[L_DAC_GetBitOfWord](#) [121](#)
[L_DAC_ResetBitOfByte](#) [121](#)
[L_DAC_ResetBitOfDWord](#) [122](#)
[L_DAC_ResetBitOfWord](#) [122](#)
[L_DAC_SetBitOfByte](#) [123](#)
[L_DAC_SetBitOfDWord](#) [123](#)
[L_DAC_SetBitOfWord](#) [124](#)
[L_DAC_WordTo2Bytes](#) [130](#)
[L_DAC_WordToBits](#) [127](#)
[L_DCO_AXIS_REF_BASE](#) [105](#)
[L_DCO_CommState](#) [115](#)
[L_DCO_ConvertData](#) [118](#)
[L_DCO_DriveCommunication](#) [103](#)
[L_DCO_DriveCommunication function library](#) [103](#)
[L_DCO_Error](#) [116](#)
[L_DCO_ReadDriveParameter](#) [106](#)
[L_DCO_ReadDriveParameterString](#) [108](#)
[L_DCO_SDOData](#) [114](#)
[L_DCO_TransferData](#) [114](#)
[L_DCO_TransferDriveParameter](#) [110](#)
[L_DCO_WriteDriveParameter](#) [112](#)
[L_LCB_8400Drive](#) [69](#)
[L_LCB_9400Drive](#) [70](#)
[L_LCB_AccToUnit](#) [94](#)
[L_LCB_ActuatorSpeed](#) [73](#)
[L_LCB_AXIS_REF](#) [67](#)
[L_LCB_AXIS_REF_CAN](#) [68](#)
[L_LCB_AXIS_REF_ETC](#) [68](#)
[L_LCB_AxisData](#) [101](#)
[L_LCB_AXISMODE](#) [102](#)

[L_LCB_Error](#) [102](#)
[L_LCB_GenericDrive](#) [72](#)
[L_LCB_GetAxisData](#) [87](#)
[L_LCB_GetPosition](#) [88](#)
[L_LCB_GetSpeed](#) [89](#)
[L_LCB_LogicControlBasic](#) [65](#)
[L_LCB_LogicControlBasic function library](#) [65](#)
[L_LCB_LOGICDEVICE](#) [102](#)
[L_LCB_Norm_aToNorm_n](#) [83](#)
[L_LCB_Norm_nToNorm_a](#) [83](#)
[L_LCB_Norm_nToSpeed_s](#) [84](#)
[L_LCB_PosToUnit](#) [95](#)
[L_LCB_SetAxisData](#) [90](#)
[L_LCB_SetPosition](#) [92](#)
[L_LCB_SetSpeed](#) [93](#)
[L_LCB_Speed_nToNorm_r](#) [84](#)
[L_LCB_Speed_rToNorm_n](#) [85](#)
[L_LCB_Speed_sToNorm_n](#) [85](#)
[L_LCB_Speed_sToSpeed_v](#) [86](#)
[L_LCB_Speed_vToSpeed_s](#) [86](#)
[L_LCB_SpeedToUnit](#) [96](#)
[L_LCB_TablePositioning](#) [77](#)
[L_LCB_TaskCycle](#) [100](#)
[L_LCB_UnitToAcc](#) [97](#)
[L_LCB_UnitToPos](#) [98](#)
[L_LCB_UnitToSpeed](#) [99](#)
[Lenze Generic Drive](#) [63](#)

M

[Manual \("speed actuating drive" visualisation\)](#) [33](#)
[Manual \("table positioning" visualisation\)](#) [39](#)
[Missing devices or devices of other manufacturers](#) [43](#)

N

[Name of the application samples](#) [20](#)
[Notes used](#) [13](#)

O

[Overview of the application samples](#) [20](#)

P

[Parameters \("speed actuating drive" visualisation\)](#) [35](#)
[Parameters \("table positioning" visualisation\)](#) [42](#)
[Program blocks \(PRG\)](#) [29](#)
[Project information](#) [131](#)

S

Safety instructions [14](#)
Safety instructions (representation) [13](#)
Sample projects - program blocks [29](#)
Sample projects - visualisations [30](#), [36](#)
Screenshots [9](#)
Service ("speed actuating drive" visualisation) [34](#)
Service ("table positioning" visualisation) [40](#)
Servo Drive 9400 HighLine [58](#)
Servo Drive 9400 Regenerative Power Supply Module [61](#)
Setting sync generation [49](#)
Speed actuating drive (assignment of the input ports) [75](#)
Speed actuating drive (assignment of the output ports) [76](#)
Speed actuating drive (program blocks) [29](#)
Speed actuating drive (sample project in the Engineer) [22](#)
Speed actuating drive (sample project in the PLC Designer) [27](#)
Speed actuating drive (visualisations) [30](#)
Structure of the application samples [19](#)
Structures (L_DCO_DriveCommunication) [114](#)
Structures (L_LCB_LogicControlBasic) [101](#)
System requirements [16](#)

T

Table positioning (assignment of the input ports) [81](#)
Table positioning (assignment of the output ports) [82](#)
Table positioning (program blocks) [29](#)
Table positioning (sample project in the Engineer) [22](#)
Table positioning (sample project in the PLC Designer) [28](#)
Table positioning (visualisations) [36](#)
Target group [9](#)
Terms [12](#)
Type converter [128](#)

U

Unions (L_DCO_DriveCommunication) [118](#)
Using application samples (Engineer) [22](#)
Using application samples (PLC Designer) [24](#)
Using sample projects (Engineer) [22](#)
Using sample projects (PLC Designer) [24](#)

V

Validity of the documentation [9](#)
Visualisations "Speed actuating drive" [30](#)
Visualisations "Table positioning" [36](#)

W

Wiring devices (Controllers, inverters, motors) [17](#)
Wiring field devices [17](#)



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.

Perhaps we have not succeeded in achieving this objective in every respect. If you have suggestions for improvement, please e-mail us to:

feedback-docu@lenze.com

Thank you very much for your support.

Your Lenze documentation team

Lenze Automation GmbH
Postfach 10 13 52, 31763 Hameln
Hans-Lenze-Straße 1, 31855 Aerzen
GERMANY
HR Hannover B 205381
 +49 5154 82-0
 +49 5154 82-2800
 lenze@lenze.com
 www.lenze.com

Service

Lenze Service GmbH
Breslauer Straße 3, 32699 Extertal
GERMANY
 008000 24 46877 (24 h helpline)
 +49 5154 82-1112
 service@lenze.com