

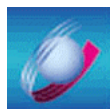
Lenze

Manual

IEC 61131-3

inside

***Global Drive
PLC Developer Studio***



Global Drive

Function library

LenzeIOSystem.lib

The **LenzeIOSystem.lib** and **LenzeIOSystemVxxxx.lib** function libraries can be used for the following Lenze PLC devices:

	Type	As of hardware version	As of software version
Drive PLC	EPL10200_xI	VC	2.x
Drive PLC	EPL10200_xT	VC	2.x
9300 Servo PLC	EVS93xx_xI	2K	2.x
9300 Servo PLC	EVS93xx_xT	2K	2.x
ECSxA	ECSxAxxx	1C	7.0

Important note:

The software is supplied to the user as described in this document. Any risks resulting from its quality or use remain the responsibility of the user. The user must provide all safety measures protecting against possible maloperation.

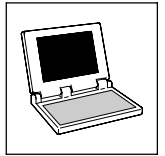
We do not take any liability for direct or indirect damage, e.g. profit loss, order loss or any loss regarding business.

© 2006 Lenze Drive Systems GmbH

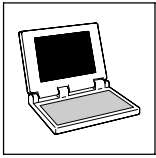
No part of this documentation may be copied or made available to third parties without the explicit written approval of Lenze Drive Systems GmbH.

All information given in this documentation has been carefully selected and tested for compliance with the hardware and software described. Nevertheless, discrepancies cannot be ruled out. We do not accept any responsibility or liability for any damage that may occur. Required corrections will be included in updates of this documentation.

All product names mentioned in this documentation are trademarks of the corresponding owners.



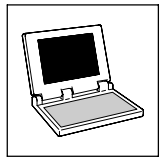
1	Preface and general information	1-1
1.1	About this Manual	1-1
1.1.1	Conventions used in this Manual	1-1
1.1.2	Structure of the descriptions	1-2
1.1.3	Pictographs used in this Manual	1-2
1.1.4	Terminology used	1-2
1.2	Version identifiers of the function library	1-3
2	Introduction	2-1
2.1	Introduction	2-1
2.2	Design concept	2-1
2.3	Minimum configuration	2-2
3	Application examples	3-1
3.1	Modular decentralised I/O system	3-1
3.2	Compact decentralised I/O system	3-5
4	Function blocks for parameterisation	4-1
4.1	L_IOParComGuarding - Monitoring function	4-2
4.2	L_IOParAlnModule - Parameterisation function	4-4
4.3	L_IOParAOutModule - Parameterisation function	4-7
4.4	L_IOParAlAOModule - Parameterisation function	4-9
4.5	L_IOParSSIModule - Parameterisation function	4-12
4.6	L_IOParCounterModule - Counter function	4-14
4.7	L_IOParCounterDIModule - Counter function	4-16
4.8	L_IOParPDO15 - Parameterisation function	4-18
4.9	L_IOParPDO610 - Parameterisation function	4-22
4.10	L_IOParCompactModule - Parameterisation function	4-26
5	Function blocks for process data processing	5-1
5.1	L_IOData15 - Coordinate data	5-2
5.2	L_IOData610 - Coordinate data	5-5
5.3	L_IOCompactModule - Coordinate data	5-8
5.4	L_IOCCounterDataToIO - Counter function	5-11
5.5	L_IOCCounterDataFromIO - Counter function	5-13
5.6	L_IOCCounterDIModuleDataToIO - Counter function	5-14
5.7	L_IOCCounterDIModuleDataFromIO - Counter function	5-15
5.8	L_IOSSIDataToIO - Counter function	5-16
5.9	L_IOSSIDataFromIO - Counter function	5-17
5.10	L_IOConvByteToByteArray - Conversion function	5-18
5.11	L_IOConvByteArrayToByte - Conversion function	5-19
5.12	L_IODInModule - Bit conversion	5-20
5.13	L_IODOutModule - Bit conversion	5-21
5.14	L_IOAlnModule - Signal conversion	5-22
5.15	L_IOAOutModule - Signal conversion	5-23



Function library LenzeIOSystem.lib

Contents

6	Appendix	6-1
6.1	Signal functions for analog inputs EPM-T310	6-1
6.2	Signal functions for analog outputs EPM-T320	6-4
6.3	Signal functions for analog inputs and outputs EPM-T330	6-6
6.4	Measured value scaling	6-9
6.5	Selecting the counter modes EPM-T410	6-10
7	Index	7-1



1 Preface and general information

1.1 About this Manual

This Manual contains information on the **LenzeIOSystem.lib** and **LenzeIOSystemVxxxx** function libraries for the **Drive PLC Developer Studio**.

1.1.1 Conventions used in this Manual

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

Variable identifiers

are printed in italics in the explanatory text:

- "By means of *bReset...*"



Tip!

Information about the conventions used for variables of Lenze system blocks, function blocks and functions can be obtained from the appendix of the DDS online documentation "Introduction into IEC 61131-3 programming". The conventions ensure universal and uniform labelling and support the readability of PLC programs.

Lenze functions/function blocks

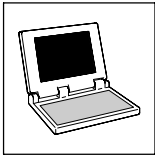
can be identified by the name. They always begin with an "L_":

- "By means of the **L_IOPar...**"FB

Program listings

are itemised in the "Courier" font, the keywords being printed bold:

- "**IF** (ReturnValue < 0) **THEN...**"



Function library LenzelOSystem.lib

Preface and general information

About this Manual

1.1.2 Structure of the descriptions

The descriptions of the individual functions/function blocks as well as system blocks in this Manual are structured in a standardised manner according to the following pattern:

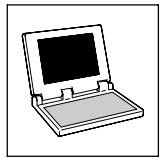
	Function	Function block (FB)/ system block (SB)
①	Headline with function and identifier	
②	Declaration of the function: - Data type of the return value - Function identifier - List of the transfer parameters	-
③	Short description with the most important features	
④	Graph showing the function including all corresponding variables - Transfer parameters - Return value	System block/function block chart including all corresponding variables - Input variables - Output variables
⑤	Table giving information on the transfer parameters: - Identifiers - Data type - Possible settings - Information	Table giving information about input and output variables: - Identifiers - Data type - Variable type - Possible settings - Information
⑥	Table giving information on the return value: - Data type of the return value - Possible return values and their meaning	-
⑦	Further information (comments, tips, application examples, etc.)	

1.1.3 Pictographs used in this Manual

	Pictographs used	Signal words
Warning of material damage		Stop! Warns of potential damage to material . Possible consequences if disregarded: Damage of the controller/drive system or its environment.
More notes		Tip! Note! Indicates a tip or note.

1.1.4 Terminology used

Term	In this Manual used for
DDS	Drive PLC Developer Studio
FB	Function block
GDC	Global Drive Control (parameter setting program from Lenze)
Parameter codes	Codes for setting the function of a function block
PLC	9300 servo PLC - Drive PLC - ECSxA "Application" axis module
SB	System block



1.2 Version identifiers of the function library

The version of the function library can be found under the global constant `C_w[Function library name]Version`.

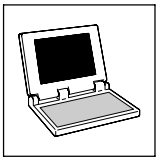
Version identifiers as of PLC software version 7.x:

Constant	Meaning	Example value
<code>C_w[FunctionLibraryName]VersionER</code>	External Release	01
<code>C_w[FunctionLibraryName]VersionEL</code>	External Level	05
<code>C_w[FunctionLibraryName]VersionIR</code>	Internal Release	00
<code>C_w[FunctionLibraryName]VersionBN</code>	Build No.	00

Version: 01 05 00 00

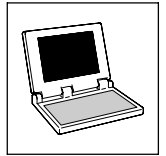
The value of this constant is a hexadecimal code.

- In the example, "01050000" stands for version "1.05".



Function library LenzeIOSystem.lib

Preface and general information



2 Introduction

2.1 Introduction

The user is offered suitable function blocks in function libraries to enable decentralised I/O system support by Lenze PLC products.

These function blocks are integrated in an IEC 61131 program and assist in the parameterisation of the I/O system as well as in process data processing. Furthermore, they provide digital and analog I/O system input and output information.

LenzeIOSystem.lib

This library provides all setting options. User configuration via user codes is not required.

2.2 Design concept

The function libraries contain two types of function blocks.

- Parameter data blocks
- Process data blocks

Parameter data blocks

These function blocks allow the modification of index information in the decentralised I/O system and the transmission of specified data. These data are transmitted only once to the I/O system during the initialisation phase.

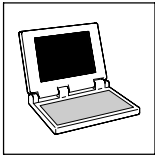
- Parameterising communication
 - These blocks parameterise communication relationships (Identifier assignment for PDO communication, transmission method, transmission period, etc.) between PLC and I/O system.
 - These settings are mandatory for data exchange.
- Parameterising modules and monitoring functions
 - The function blocks are required for individual parameterisation of analog modules, individual parameterisation of counter modules, communication monitoring

Process data blocks

These function blocks provide assistance for evaluating and further processing the current process data information from the I/O system inputs in the PLC program.

Information and data can be transferred into the process via the output modules.

- Process data blocks to transmit and receive I/O data between PLC and I/O system.
- Process data blocks for I/O signal conversion.



Function library *LenzeIOSystem.lib*

Introduction

Minimum configuration

2.3 Minimum configuration

A minimum configuration must be established to integrate the I/O system, consisting of one parameter data block **L_IOParPDO15** to parameterise communication relationships and one process data block **L_IOData15** to evaluate the input/output information.

The information of the parameter block is transmitted to the I/O system and the function block for PLC I/O system process data exchange.

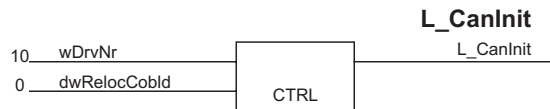
Data exchange between the output of the parameter block and the input of the process data block is possible with the help of a global variable.



Note!

The function libraries for the decentralised I/O system are based on unassigned CAN objects, so that the CAN driver must be initialised via block **L_CANInit**.

The block **L_CANInit** is located in the Lenze library **LenzeCanDrv.lib**.



10 = System bus

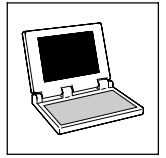
0 = ID range 832..895 (Servo PLC only)



Note!

The identifier range is limited in the case of the 9300 Servo PLC.

Use of function blocks **L_IOParPDO15** and **L_IOParPDO610** requires adjustment of the Cob IDs from PDO4 onwards.



3 Application examples

The examples were generated with DDS version 2.0. Examples concerning the I/O system are given in the directory C:\Programme\Lenze\DDS_P_2_20\Projects\IOSystem\Samples

3.1 Modular decentralised I/O system

The modular decentralised I/O system with eight digital inputs and outputs is integrated into a PLC program with the help of LenzelIOSystem.lib.

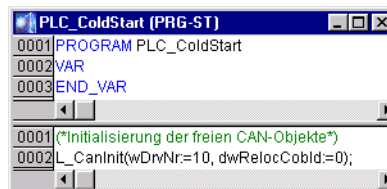
During this process, the following steps must be carried out in the given order:

1. Initialise the CAN driver
2. Parameterise the communication conditions
3. Program the process data exchange between PLC and I/O system
4. Initialise the codes

Initialising the CAN driver

Communication between the PLC and the I/O system is based on "unassigned CAN objects" (control, alarm, diagnostics...). These objects are available in DDS from version 2.x onwards.

To initialise the unassigned CAN objects, the function **L_CANInit** must be requested once only in the system POU **PLC_ColdStart**.

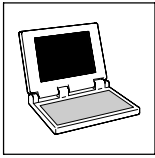


Parameterising the communication conditions

Function block **L_IOParPDO15** for parameterising the communication conditions of PDO1-PDO5 is integrated into the SFC editor and necessarily is to be used in every project. This function block specifies the communication parameters in the I/O system.

The values are transferred to the process data block **L_IOData15** as a structure (STRUCT) in order to match the communication parameters of the devices (controller and I/O system).

On program execution, this step is executed only once at the beginning.

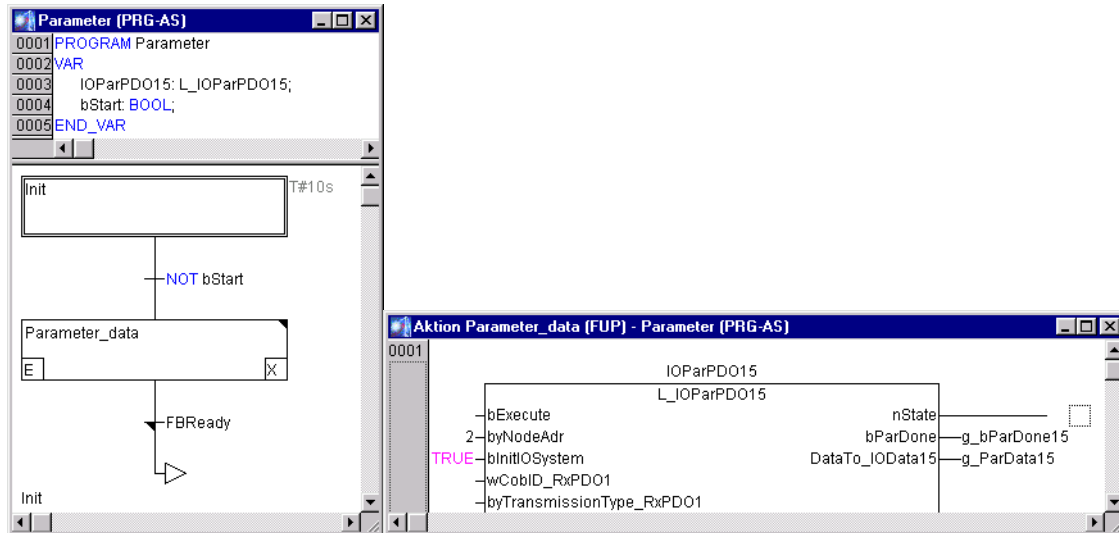


Function library LenzeIOSystem.lib

Application examples

Modular decentralised I/O system

Then the **Parameter** program that must be generated in the SFC editor follows. With the invocation of the function block instance, the step **Parameter_data** has the action generated in the FBD editor.

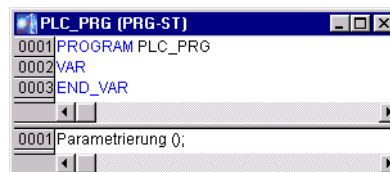


Note!

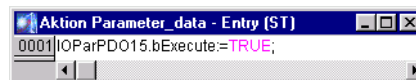
- Enter a minimum time of ten seconds for step **Init** via the menu command **Extras→Step attributes....** Correct entry: T#10s

Make sure that step **Parameter_data** is not executed before the initialisation phase of the I/O system is complete. This is achieved with a minimum time of ten seconds.

As this POU must be processed cyclically, the program is invoked in the PLC_PRG (main program). The sequential function chart ensures a cyclic task.



The parameterisation process is started via the initial action in step **Parameter_data**.

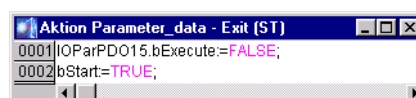


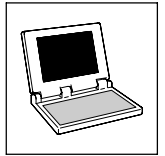
Once the parameterisation is complete, the final action in step **Parameter_data** sets input **bExecute** to FALSE and variable **bStart** to TRUE.

Variable **bStart** ensures that step **Parameter_data** is executed

- on switching on
- on reset and subsequent starting.

When the parameterisation is complete, the PLC_PRG (main program) stops in step **Init**. It does not impose an extra burden on the target system during subsequent program operation.

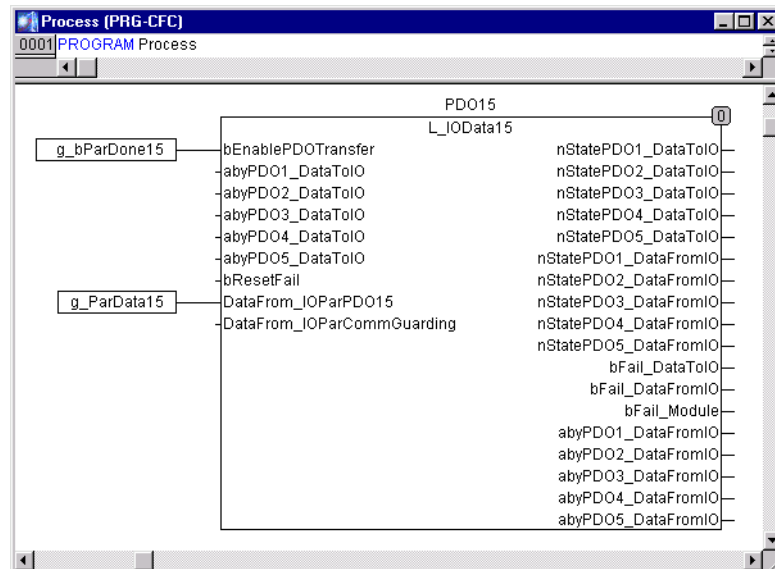




Programming the process data exchange between PLC and I/O system

The CFC editor is used to generate a program named **Process**.

- Declare an instance of function block **L_IOData15** in **Process**.
- Generate two inputs in the CFC editor, assign names *g_bParDone15* and *g_ParData15* and attach these to the corresponding inputs of function block **L_IOData15** as shown in the figure.



Several conversion blocks are available to evaluate input data and to transfer output data.

Function block **L_IOData15** transmits output data to the I/O system and receives input data from there.

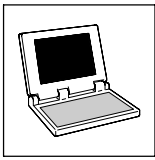
At input *DataFrom_IOParPDO15*, CAN communication relationships between PLC and I/O system are transmitted from parameterisation block **L_IOParPDO15** to process data block **L_IOData15**.

Process data transmission is started after parameterisation via a High signal at input *bEnablePDOTransfer*.

Data conversion for evaluating individual channels

The program **Process** can be extended as follows:

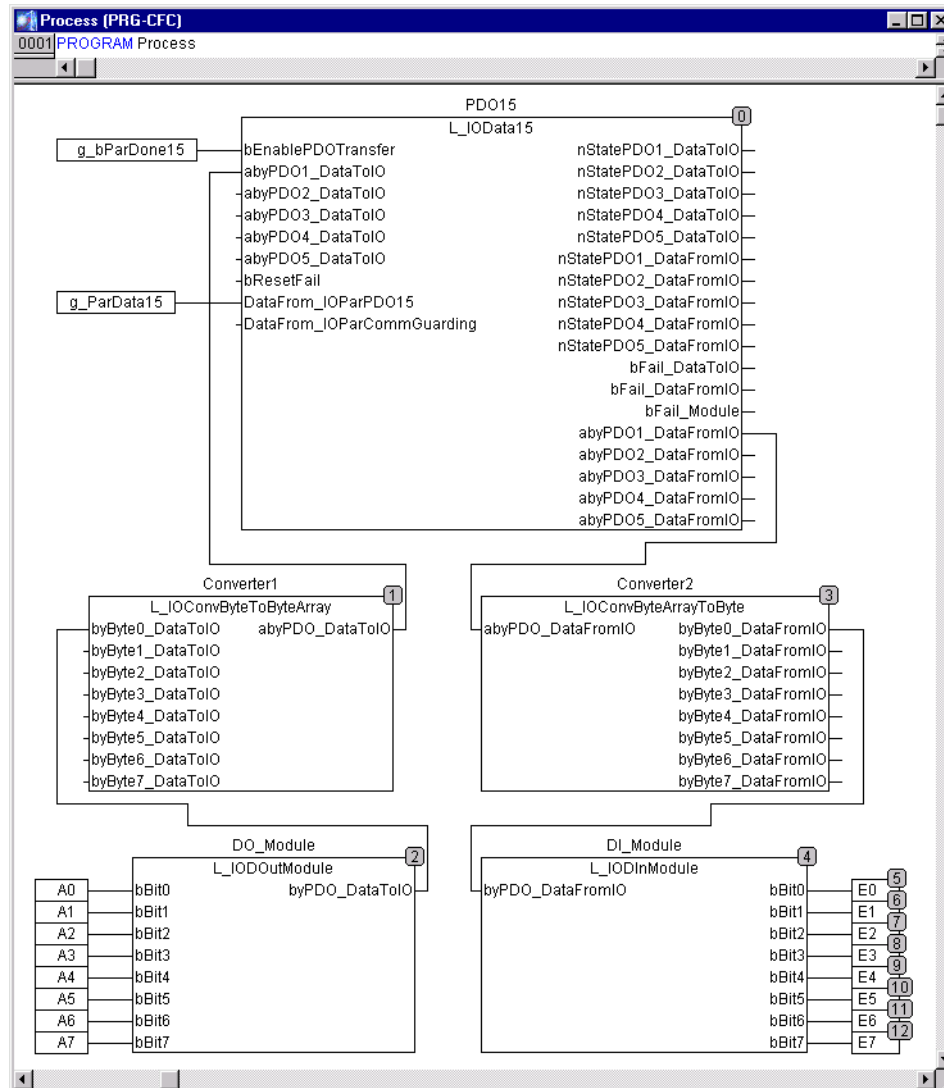
- Use the menu command **Extras**→**Order** to indicate the sequence.
- Drag the individual function blocks to their positions in order to make the program sequence comprehensible.



Function library *LenzeIOSystem.lib*

Application examples

Modular decentralised I/O system



Function block **L_IODData15** supplies the transmit and receive data as 8 byte information via the CAN telegram.

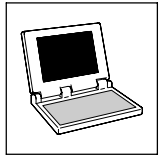
The two converters convert the information into individual bytes.

A digital input and output module maps the individual bits.

Initialising the codes

Initialise the following codes in the PLC

- Code C0352
Subcode 0 = 1
Defines PLC as a CAN master so that the configuration enters the Operational state.
- Code C0356
Subcode 1 = 10000
Increases the boot-up time to 10s
- Code C2104
Subcode 0 = 1
Starts the PLC program automatically after switching on the mains



3.2 Compact decentralised I/O system

The compact module EPM - T832 with 24 digital inputs and 8 digital outputs is integrated into a PLC program with the help of LenzelOSystem.lib.

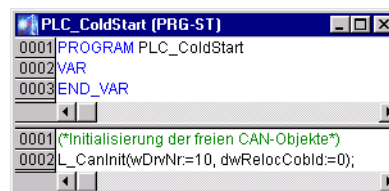
During this process, the following steps must be carried out in the given order:

1. Initialise the CAN driver
2. Parameterise the communication conditions
3. Program the process data exchange between PLC and I/O system
4. Initialise the codes

Initialising the CAN driver

Communication between the PLC and the I/O system is based on "unassigned CAN objects" (control, alarm, diagnostics...). These objects are available in DDS from version 2.x onwards.

To initialise the unassigned CAN objects, the function **L_CANInit** must be requested once only in the system POU **PLC_ColdStart**.



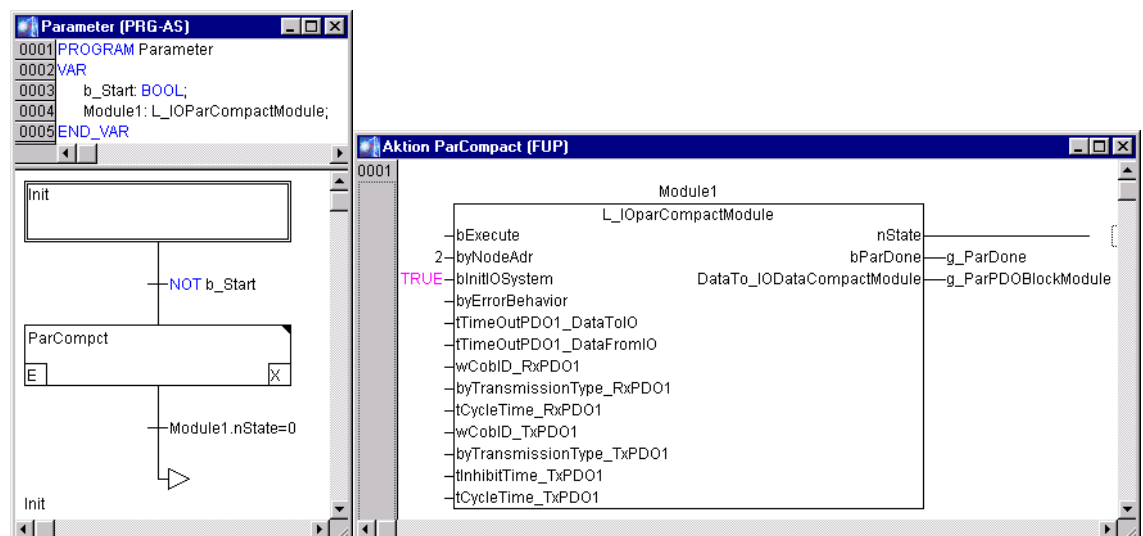
Parameterising the communication conditions

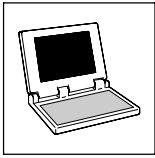
Function block **L_IOParCompactModule** for parameterising the communication conditions is integrated into the SFC editor and necessarily is to be used in every project. This function block specifies the communication parameters in the I/O system.

The values are transferred to the process data block **L_IOCompactModule** as a structure (STRUCT) in order to match the communication parameters of the devices (controller and I/O system).

On program execution, this step is executed once only at the beginning.

Then the **Parameter** program that must be generated in the SFC editor follows. With the invocation of the function block instance, the step **ParCompact** has the action generated in the FBD editor.





Function library *LenzeIOSystem.lib* Application examples

Compact decentralised I/O system

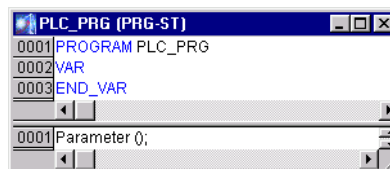


Note!

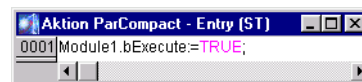
- Enter a minimum time of ten seconds for step **Init** via the menu command **Extras→Step attributes....** Correct entry: T#10s

Make sure that step **Parameter_data** is not executed before the initialisation phase of the I/O system is complete. This is achieved with a minimum time of ten seconds.

As this POU must be processed cyclically, the program is invoked in the PLC_PRG (main program). The sequential function chart ensures a cyclic task.



The parameterisation process is started via the initial action in step **ParCompact**.

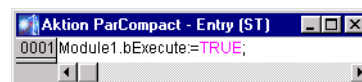


Once the parameterisation is complete, the final action in step **ParCompact** sets input *bExecute* to FALSE and variable *bStart* to TRUE.

Variable *bStart* ensures that step **ParCompact** is executed

- on switching on
- on reset and subsequent starting.

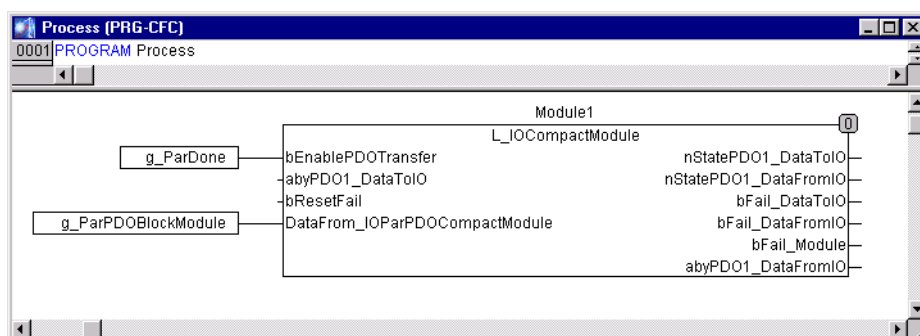
When the parameterisation is complete, the PLC_PRG (main program) stops in step **Init**. It does not impose an extra burden on the target system during subsequent program operation.



Programming the process data exchange between PLC and I/O system

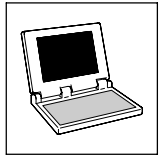
The CFC editor is used to generate a program named **Process**.

- Declare an instance of function block **L_IOCompactModule** in **Process**.
- Generate two inputs in the CFC editor, assign names *g_bParDone15* and *g_ParData15* and attach these to the corresponding inputs of function block **L_IOCompactModule** as shown in the figure.



Several conversion blocks are available to evaluate input data and to transfer output data.

Function block **L_IOCompactModule** transmits output data to the I/O system and receives input data from there.



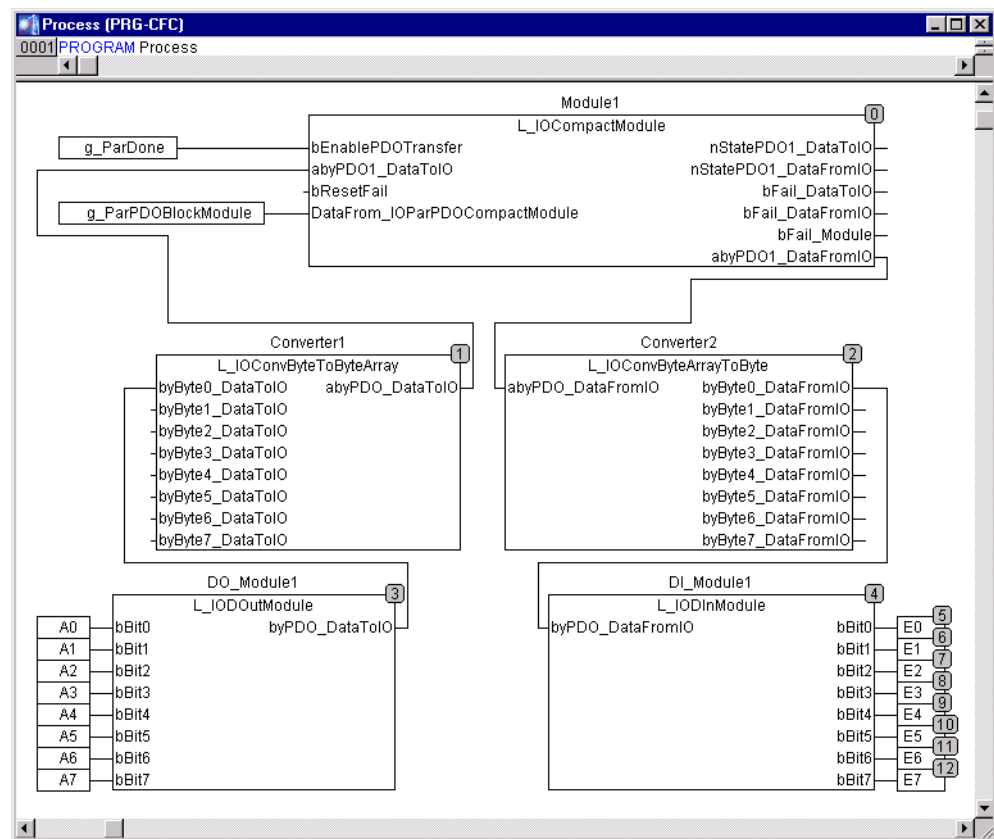
At input *DataFrom_IOParPDO15*, CAN communication relationships between PLC and I/O system are transmitted from parameterisation block **L_IOParCompactModule** to process data block **L_IOCompactModule**.

Process data transmission is started after parameterisation via a High signal at input *bEnablePDOTransfer*.

Data conversion for evaluating individual channels

The program **Process** can be extended as follows:

- Use the menu command **Extras→Order** to indicate the sequence.
- Drag the individual function blocks to their positions in order to make the program sequence comprehensible.

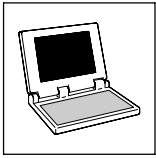


Function block **L_IOCompactModule** supplies the transmit and receive data as 8*8 byte information via the CAN telegram.

The two converters convert the information into individual bits.

A digital input and output module maps the individual bits.

If the example for eight inputs / outputs is to be extended to 24 inputs, two further function blocks **L_IODInModule** must be generated and linked via converter **L_IOConvByteArrayToByte**.



Function library LenzeIOSystem.lib

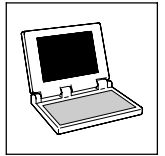
Application examples

Compact decentralised I/O system

Initialising the codes

Initialise the following codes in the PLC

- Code C0352
Subcode 0 = 1
Defines PLC as a CAN master so that the configuration enters the Operational state.
- Code C0356
Subcode 1 = 10000
Increases the boot-up time to 10s
- Code C2104
Subcode 0 = 1
Starts the PLC program automatically after switching on the mains

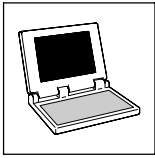


4 Function blocks for parameterisation

This chapter contains information on function blocks for parameterisation.

The following functions and modules are parameterised:

- Monitoring function
- Analog input modules
- Analog output modules
- Counter modules
- Link between PLC control and I/O system
- Link between PLC control and compact module



Function library *LenzeIOSystem.lib*

Function blocks

L_IOParComGuarding - Monitoring function

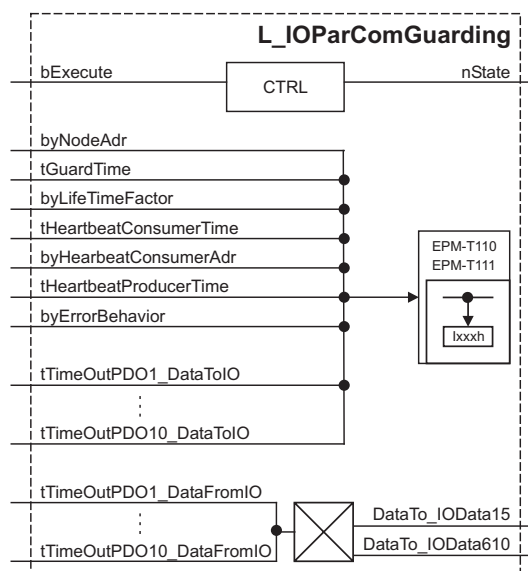
4.1 L_IOParComGuarding - Monitoring function

This function block parameterises, via the input setting, the monitoring functions provided by the I/O system.

Included are

- NodeGuarding, Heartbeat and a Lenze-specific monitoring function,
- setting the output channels' switching behaviour in the case of an error via input *byErrorBehavior*.

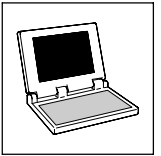
The function block output indicates the transmission status.



Note!

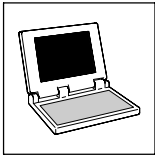
Use the FBs of *LenzeCanDSxDrv.lib* for evaluating the functions NodeGuarding and Heartbeat.

⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bExecute	Bool	Execute write request FALSE ↗ TRUE Write request is executed
byNodeAdr	Byte	Device address: 1 - 63
tGuardTime	Time	Nodeguarding: Time interval. An NMT master must poll the I/O system within the guard time period. Default t = 0 ms t = 0 - 65535 ms
byLifeTimeFactor	Byte	Lifeguarding: Life Time Factor x Guard Time = Life Time. Once this period has expired, the I/O system is switched into a predefined status.
tHeartbeatConsumerTime	Time	Heartbeat: Monitoring interval. The I/O system is switched into a predefined status if no telegram is received from the Heartbeat Producer within this interval. t = 0 - 65535 ms
byHeartbeatConsumerAdr	Byte	Node ID of the device to be monitored. (HeartbeatProducer Node ID)



⇒ Inputs (Variable type: VAR_INPUT)										
Variable name	Data type	Information/possible settings								
tHeartbeatProducerTime	Time	Set the transmission time for one heartbeat telegram t = 0 - 65535ms								
byErrorBehavior	Byte	Set the bus status / behaviour on critical device error <table><tr><td>0</td><td>Pre-Operational (default)</td></tr><tr><td>1</td><td>No status change</td></tr><tr><td>2</td><td>Stop</td></tr><tr><td>3</td><td>Reset after 2 seconds</td></tr></table>	0	Pre-Operational (default)	1	No status change	2	Stop	3	Reset after 2 seconds
0	Pre-Operational (default)									
1	No status change									
2	Stop									
3	Reset after 2 seconds									
tTimeOutPDO1_DataToIO ... tTimeOutPDO10_DataToIO	Time	Monitoring period for process input data (Lenze-specific). If no data are received at the selected PDO within the monitoring period, the I/O system assumes the status set under <i>byErrorBehavior</i> . <table><tr><td>Default</td><td>t = 0 ms t = 0 - 65535 ms</td></tr></table>	Default	t = 0 ms t = 0 - 65535 ms						
Default	t = 0 ms t = 0 - 65535 ms									
tTimeOutPDO1_DataFromIO ... tTimeOutPDO10_DataFromIO	Time	Monitoring period for process output data. If no data are transmitted from the I/O system to the PLC at the selected PDO within the monitoring period, function block L_IODataXXX signals a time monitoring error. <table><tr><td>Default</td><td>t = 0 ms t = 0 - 65535 ms</td></tr></table>	Default	t = 0 ms t = 0 - 65535 ms						
Default	t = 0 ms t = 0 - 65535 ms									

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit command is executed.
		1 Transmit command sent - processing in process
		2001 Error occurred during bus transfer
		2002 Transmit command timed out
		2004 Access denied - wrong target system: Subindex does not exist
		2005 Access denied - wrong target system: Index does not exist
		2013 Access denied - wrong target system: Invalid value range
		2111 Access denied - wrong target system: More than 4 digits specified after comma, or no fixed-comma-format index
		2117 Communication channel does not exist - also refer Set byCom channel
		2120 Too many modules connected - parameter transfer aborted
		-10xx Incorrect value at function block input. XX := Number of faulty input (counting direction down), parameterisation impossible. Remedy: Change input value, restart data transmission
		DataTO_IOData15
DataTO_IOData610	Struct	Structure for process data block L_IOData610 Transfer process output data monitoring periods to process data block L_IOData610 . Global struct-type variable



Function library *LenzeIOSystem.lib*

Function blocks

L_IOParAlnModule - Parameterisation function

4.2 *L_IOParAlnModule* - Parameterisation function

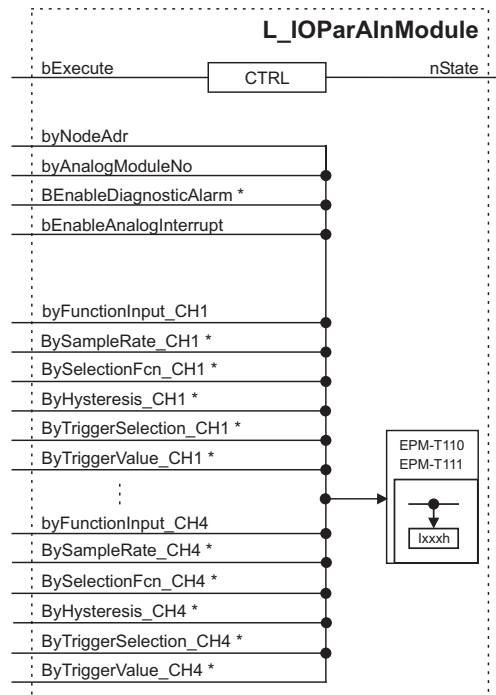
By means of the **L_IOParAlnModule** function block, the analog input modules (EPM-T310 / 311 / T312) are parameterised. Parameterisation is started by a high edge on the input *bExecute*.

Each module has four channels, whereby each channel can be set individually.

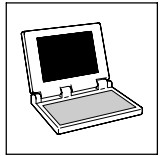
The following is provided:

- Voltage measurement
- Current measurement
- Resistance measurement
- Temperature measurement

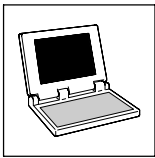
On the output of the function block, the transmission status and the information on the connected modules are displayed.



* Input/function is exclusively supported by type EPM-T310.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bExecute	BOOL	Execute write request FALSE ↗ TRUE Write request is executed
byNodeAdr	Byte	Device address: 1 - 63
byAnalogModuleNo	Byte	Slot number • The analog module that is mounted first has the slot number 1. • Maximally 16 analog modules can be attached.
bEnableDiagnosticAlarm	BOOL	Enable diagnostics alarm TRUE Diagnostics alarm is enabled
bEnableAnalogInterrupt	BOOL	Enable analog data transmission TRUE Data transmission is enabled
byFunctionInput_CH1 ... byFunctionInput_CH4	Byte	Parameterise function Default ±10V function code = 3B _h 6-1
bySampleRate_CH1 ... bySampleRate_CH4	Byte	Select number of conversions per second 0 15 conversions 1 30 conversions 2 60 conversions 3 123 conversions 4 168 conversions 5 202 conversions 6 3.7 conversions 7 7.5 conversions
bySelectionFcn_CH1 ... bySelectionFcn_CH4	Byte	Selection 0 Selection deactivated 1 Use 2 of 3 values 2 Use 4 of 6 values
byHysteresis_CH1 ... byHysteresis_CH4	Byte	Hysteresis 0 Hysteresis deactivated 1 Hysteresis ±8 2 Hysteresis ±16
byTriggerSelection_CH1 ... byTriggerSelection_CH4	Byte	Transmission result 0 Use default settings 1 Upper limit value exceeded 2 Lower limit value not reached 3 Input value has changed by a defined value.
byTriggerValue_CH1 ... byTriggerValue_CH4	DWORD	Value selection for a transmission event 2 Default

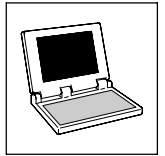


Function library *LenzeIOSystem.lib*

Function blocks

L_IOParAlnModule - Parameterisation function

Outputs ⇨ (Variable type: VAR_OUTPUT)		
Identifiers	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit request is executed. The module is parameterised.
		1 Transmit request dropped Processing is running
		2001 Error occurred during bus transfer
		2002 Transmit request timed out
		2004 Access denied Incorrect target system: index does not exist
		2005 Access denied Incorrect target system: subindex does not exist
		2013 Access denied Incorrect target system: invalid value range
		2111 Access denied Incorrect target system: more than 4 decimal positions specified or no fixed point format index
		2117 Communication channel does not exist byCom-Channelsetting
		-10xx Incorrect value at function block input. XX := number of the faulty input (counting direction downwards), parameterisation impossible. Remedy: change input value, restart data transmission



4.3 L_IOParAOutModule - Parameterisation function

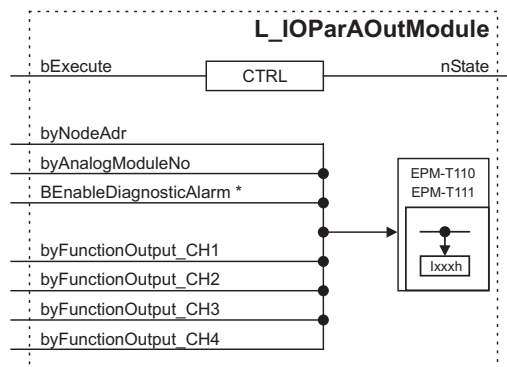
By means of the **L_IOParAOutModule** function block, the analog output modules (EPM-T320 / T321 / T322) are parameterised. Parameterisation is started by a high edge on the input *bExecute*.

Each module has four channels, whereby each channel can be set individually.

The following is provided:

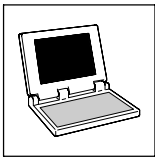
- Voltage measurement
- Current measurement

The function block output indicates the transmission status.



* Input/function is exclusively supported by type EPM-T320.

⇔ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bExecute	BOOL	Execute write request FALSE → TRUE Write request is executed
byNodeAdr	Byte	Device address: 1 - 63
byAnalogModuleNo	Byte	Slot number • The analog module that is mounted first has the slot number 1. • Maximally 16 analog modules can be attached.
bEnableDiagnosticAlarm	BOOL	Enable diagnostics alarm TRUE Diagnostics alarm is enabled
byFunctionOutput_CH1 ... byFunctionOutput_CH4	Byte	Parameterise function Default ±10V function code = 01 _h □ 6-4

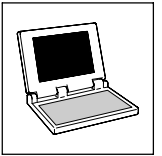


Function library *LenzeIOSystem.lib*

Function blocks

L_IOParAOutModule - Parameterisation function

Outputs ⇨ (Variable type: VAR_OUTPUT)		
Identifiers	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit request is executed. The module is parameterised.
		1 Transmit request dropped Processing is running
		2001 Error occurred during bus transfer
		2002 Transmit request timed out
		2004 Access denied Incorrect target system: subindex does not exist
		2005 Access denied Incorrect target system: index does not exist
		2013 Access denied Incorrect target system: invalid value range
		2111 Access denied Incorrect target system: more than 4 decimal positions specified or no fixed point format index
		2117 Communication channel does not exist byCom-Channelsetting
		-10xx Incorrect value at function block input. XX := number of faulty input (counting direction down), parameterisation impossible. Remedy: change input variable, restart data transmission



4.4 L_IOParAIAOModule - Parameterisation function

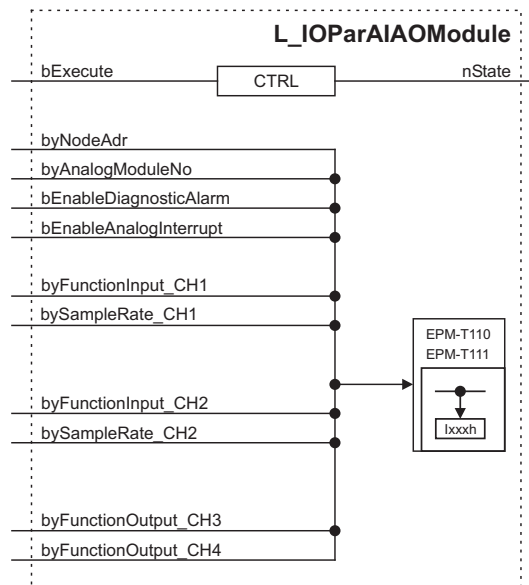
Function block **L_IOParAIAOModule** parameterises the analog input and output modules EPM-T330, a High edge at input *bExecute* starting the parameterisation process.

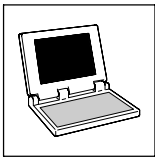
Every module has four channels that can each be set individually. Two analog inputs and two analog outputs are available.

The following functions can be set:

- Voltage measuring
- Current measuring
- Voltage output
- Current output

The function block output indicates the transmission status and information on linked modules.



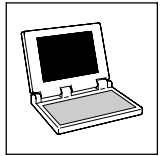


Function library *LenzeIOSystem.lib*

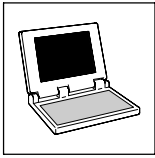
Function blocks

L_IOParAIAOModule - Parameterisation function

⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bExecute	Bool	Execute write request FALSE ↗ TRUE Write request is executed
byNodeAdr	Byte	Device address: 1 - 63
byAnalogModuleNo	Byte	Slot number • The first analog module installed is assigned slot number 1. • A maximum of 16 analog modules may be plugged.
bEnableDiagnosticAlarm	Bool	Enable diagnostic alarm TRUE Diagnostic alarm is enabled
bEnableAnalogInterrupt	Bool	Enable analog data transmission TRUE Data transmission is enabled
byFunctionInput_CH1 ... byFunctionInput_CH2	Byte	Parameterise function Default ±10V function parameter = 01 _h 6-6
bySampleRate_CH1 ... bySampleRate_CH2	Byte	Select number of conversions per second 0 15 conversions 1 30 conversions 2 60 conversions 3 123 conversions 4 168 conversions 5 202 conversions 6 3.7 conversions 7 7.5 conversions
byFunctionOutput_CH3 ... byFunctionOutput_CH4	Byte	Parameterise function Default ±10V function code = 01 _h 6-6



Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit command is executed. The module is parameterised.
		1 Transmit command sent Processing in process
		2001 Error occurred during bus transfer
		2002 Transmit command timed out
		2004 Access denied Wrong target system: Index does not exist
		2005 Access denied Wrong target system: Subindex does not exist
		2013 Access denied Wrong target system: Invalid value range
		2111 Access denied Wrong target system: More than 4 digits specified after comma, or no fixed-comma-format index
		2117 Communication channel does not exist Set byCom channel
		-10xx Incorrect value at function block input. XX := Number of the faulty input (counting direction down), parameterisation impossible. Remedy: Change input value, restart data transmission



Function library *LenzeIOSystem.lib*

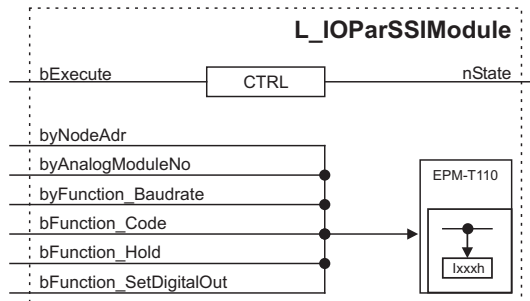
Function blocks

L_IOParSSIModule - Parameterisation function

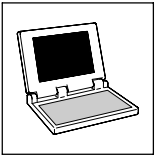
4.5 L_IOParSSIModule - Parameterisation function

The SSI interface (SSI = Synchronous Serial Interface) is a synchronously clocked interface.

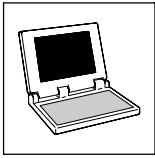
Function block **L_IOParSSIModule** converts the serial information of the measured values into parallel information that is subsequently supplied to the control system.



⇒ Inputs		(Variable type: VAR_INPUT)	
Variable name	Data type	Information/possible settings	
bExecute	Bool	Execute write request	
		FALSE ↗ TRUE	Write request is executed
byNodeAdr	Byte	Device address: 1 - 63	
byAnalogModuleNo	Byte	Slot number • The first analog module installed is assigned slot number 1. • A maximum of 9 modules may be plugged.	
byFunction_Baudrate	Byte	Baud rate setting	
		0	300 kbps
		1	100 kbps
		2	300 kbps
		3	600 kbps
		4	300 kbps
bFunction_Code	Bool	Coding	
		Default 0	Binary code
		1	Gray code
bFunction_Hold	Bool	Hold function	
		0	Not active
		1	Active
bFunction_SetDigitalOut	Bool	Digital outputs	
		0	Not active
		1	Active



Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit command is executed. The module is parameterised.
		1 Transmit command sent Processing in process
		2001 Error occurred during bus transfer
		2002 Transmit command timed out
		2004 Access denied Wrong target system: Subindex does not exist
		2005 Access denied Wrong target system: Index does not exist
		2013 Access denied Wrong target system: Invalid value range
		2111 Access denied Wrong target system: More than 4 digits specified after comma, or no fixed-comma-format index
		2117 Communication channel does not exist Set byCom channel
		-10xx Incorrect value at function block input. XX := Number of faulty input (counting direction down), parameterisation impossible. Remedy: Change input value, restart data transmission



Function library *LenzeIOSystem.lib*

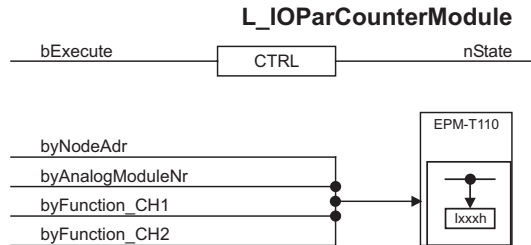
Function blocks

L_IOParCounterModule - Counter function

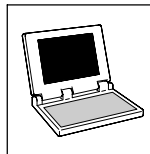
4.6 L_IOParCounterModule - Counter function

The counter modules are parameterised via function block **L_IOParCounterModule**.

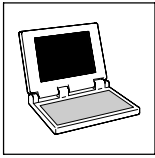
- The counter function is specified via inputs *byFunction_CH1* and *byFunction_CH2*.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bExecute	Bool	Execute write request FALSE ↗ TRUE Write request is executed
byNodeAdr	Byte	Device address: 1 - 63
byAnalogModuleNo	Byte	Slot number - The first analog module installed is assigned slot number 1. - A maximum of 16 analog modules may be plugged.
byFunction_CH1	Byte	Parameterise function Default (2 * 32 bit counters) function code 00 _h 6-10
byFunction_CH2	Byte	Parameterise function Default (2 * 32 bit counters) function code 00 _h 6-10



Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit command is executed. The module is parameterised.
		1 Transmit command sent Processing in process
		2001 Error occurred during bus transfer
		2002 Transmit command timed out
		2004 Access denied Wrong target system: Subindex does not exist
		2005 Access denied Wrong target system: Index does not exist
		2013 Access denied Wrong target system: Invalid value range
		2111 Access denied Wrong target system: More than 4 digits specified after comma, or no fixed-comma-format index
		2117 Communication channel does not exist Set byCom channel
		-10xx Incorrect value at function block input. XX := Number of faulty input (counting direction down), parameterisation impossible. Remedy: Change input value, restart data transmission



Function library *LenzeIOSystem.lib*

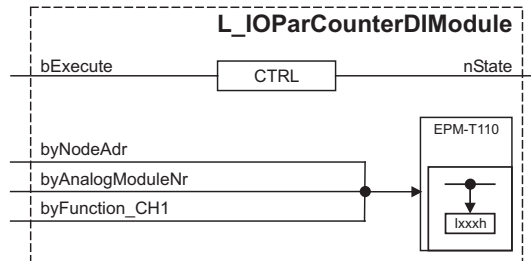
Function blocks

L_IOParCounterDIModule - Counter function

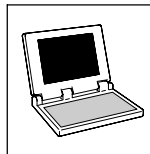
4.7 L_IOParCounterDIModule - Counter function

The counter function of module EPM-T430 is parameterised via function block **L_IOParCounterDIModule**.

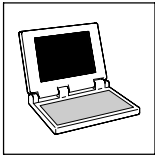
- The counter function is specified via inputs *byFunction_CH1*.



⇒ Inputs			(Variable type: VAR_INPUT)	
Variable name	Data type	Information/possible settings		
bExecute	Bool	Execute write request		
		FALSE ↗ TRUE	Write request is executed	
byNodeAdr	Byte	Device address: 1 - 63		
byAnalogModuleNo	Byte	Slot number		
		• The first analog module installed is assigned slot number 1 • A maximum of 16 analog modules may be plugged.		
byFunction_CH1	Byte	Counter function		
		0	Fourfold pulse evaluation	
		1	Pulse and direction evaluation	
		2	Counter upwards / downwards	
		3	Frequency measuring	
		4	Period measuring	



Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit command is executed. The module is parameterised.
		1 Transmit command sent Processing in process
		2001 Error occurred during bus transfer
		2002 Transmit command timed out
		2004 Access denied Wrong target system: Subindex does not exist
		2005 Access denied Wrong target system: Index does not exist
		2013 Access denied Wrong target system: Invalid value range
		2111 Access denied Wrong target system: More than 4 digits specified after comma, or no fixed-comma-format index
		2117 Communication channel does not exist Set byCom channel
		-10xx Incorrect value at function block input. XX := Number of faulty input (counting direction down), parameterisation impossible. Remedy: Change input value, restart data transmission



Function library LenzelOSystem.lib

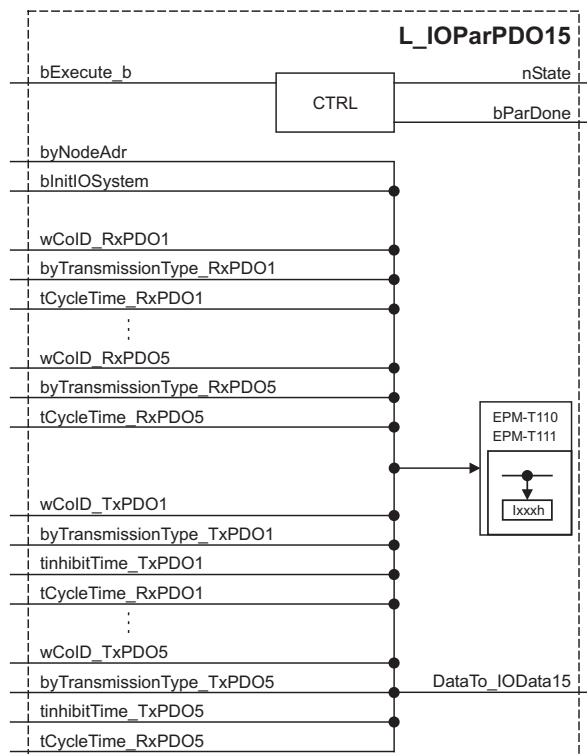
Function blocks

L_IOParPDO15 - Parameterisation function

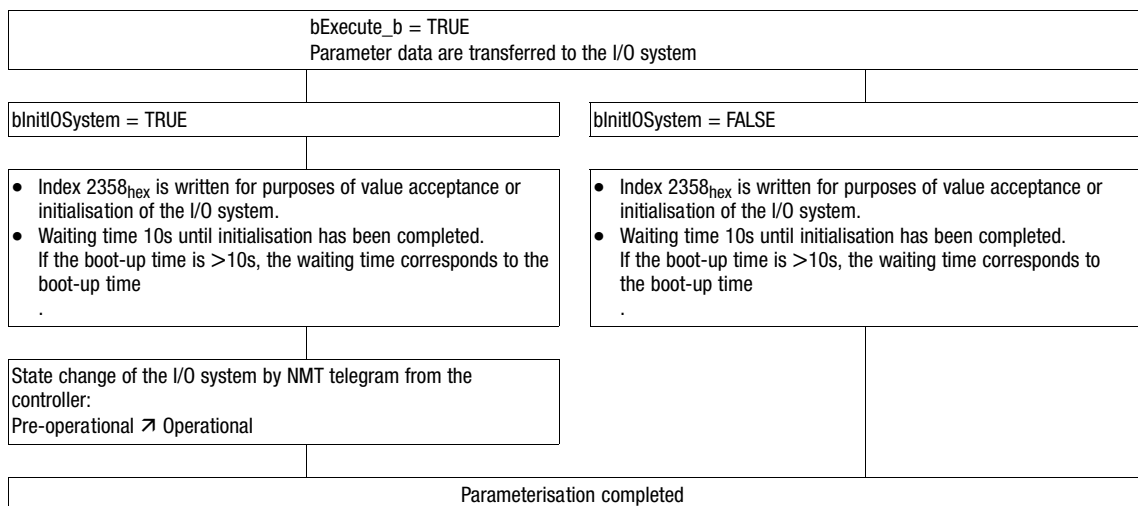
4.8 L_IOParPDO15 - Parameterisation function

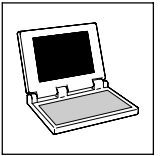
This function block serves to set communication-relevant parameters between the PLC control and the I/O system for PDO 1 - 5.

- Via 3 or 4 input values, a transmission/reception object is parameterised.
- The output *nState* displays the parameterisation status.
- Via the output *DataTo_IOData15*, information is transferred to the **L_IOData15** process data block in a fixed structure.



Internal function





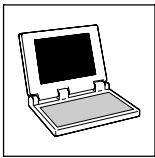
Note!

Features and use of the **L_IOParPDO15** FB.

- Use the **L_IOParPDO15** FB last in a sequencer consisting of several FBs for parameterisation. By means of the **L_IOParPDO15** FB, parameterisation of the I/O system is carried out.
- For sending the NMT telegram, the controller has to be in the *Operational* CAN status, therefore parameterise the controller as master. After the boot-up time has expired, the state change from *Pre-operational* to *Operational* is automatically effected.
- If several IO systems (stations) are parameterised with standard values via the **L_IOParPDO15** FB on the system bus, an identifier collision between the individual PDOs of these stations can occur. Therefore check the identifiers used.

After changing to the *Operational* CAN status, the current process image of the I/O system is sent.

⇒ Inputs		(Variable type: VAR_INPUT)			
Variable name	Data type	Information/possible settings			
bExecute	BOOL	Execute write request			
		FALSE ↗ TRUE		Write request is executed	
byNodeAdr	Byte	Device address: 1 - 63			
bInitIOSystem	BOOL	Initialise I/O system after parameterisation			
		TRUE		• Value acceptance by reset node • I/O system is re-initialised, state change to Operational	
		FALSE		• No value acceptance • I/O system is not initialised	
wCobID_RxPDO1 ... wCobID_RxPDO5	WORD	Setting identifiers Default values: input wiring = 0 RxPDO = receive identifiers from the I/O system Identifiers can be optionally set by value selection. If the input wiring is "0", the identifiers are composed as follows. Note! The basic identifiers are not settled in the pre-assigned area of the standard system bus. ID range: 384-831 By this, crossovers of the ID with the free CAN objects which the IO-lib is based on are avoided.			
		RxPDOX	Identifier =	Basic ID +	Node address
		RxPDO1	837	835	2
		RxPDO2		840	
		RxPDO3		845	
		RxPDO4		850	
		RxPDO5		855	
byTransmissionType_RxPDO1 ... byTransmissionType_RxPDO5	Byte	Transmission mode			
		0	Event-controlled		
		1	Time-controlled		
		2	Event-controlled with superimposed cycle time		
tCycleTime_RxPDO1 ... tCycleTime_RxPDO5	Time	Cycle time: t = 0 - 65535 ms			

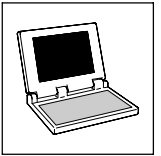


Function library *LenzeIOSystem.lib*

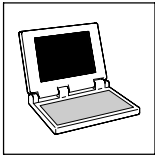
Function blocks

L_IOParPDO15 - Parameterisation function

⇒ Inputs	(Variable type: VAR_INPUT)																											
Variable name	Data type	Information/possible settings																										
wCobID_TxPDO1 ... wCobID_TxPDO5	WORD	Setting identifiers Default values: input wiring = 0 TxPDO = receive identifiers from the I/O system Identifiers can be optionally set by value selection. If the input wiring is "0", the identifiers are composed as follows. Note! The basic identifiers are not settled in the pre-assigned area of the standard system bus. ID range: 384-831 By this, crossovers of the ID with the free CAN objects which the IO-lib is based on are avoided. <table><tr><td>TxPDOX</td><td>Identifier =</td><td>Basic ID +</td><td>Node address</td></tr><tr><td>TxPDO1</td><td>838</td><td>836</td><td>2</td></tr><tr><td>TxPDO2</td><td></td><td>841</td><td></td></tr><tr><td>TxPDO3</td><td></td><td>846</td><td></td></tr><tr><td>TxPDO4</td><td></td><td>851</td><td></td></tr><tr><td>TxPDO5</td><td></td><td>856</td><td></td></tr></table>			TxPDOX	Identifier =	Basic ID +	Node address	TxPDO1	838	836	2	TxPDO2		841		TxPDO3		846		TxPDO4		851		TxPDO5		856	
TxPDOX	Identifier =	Basic ID +	Node address																									
TxPDO1	838	836	2																									
TxPDO2		841																										
TxPDO3		846																										
TxPDO4		851																										
TxPDO5		856																										
byTransmissionType_TxPDO1 ... byTransmissionType_TxPDO5	Byte	Transmission mode <table><tr><td>0</td><td>Event-controlled</td></tr><tr><td>1</td><td>Time-controlled</td></tr><tr><td>2</td><td>Event-controlled with superimposed cycle time</td></tr></table>			0	Event-controlled	1	Time-controlled	2	Event-controlled with superimposed cycle time																		
0	Event-controlled																											
1	Time-controlled																											
2	Event-controlled with superimposed cycle time																											
tinhibitTime_TxPDO1 ... tinhibitTime_TxPDO5	Time	Inhibit time: t = 0 - 65535 ms																										
tCycleTime_TxPDO1 ... tCycleTime_TxPDO5	Time	Cycle time: t = 0 - 65535 ms																										



Outputs ⇨		(Variable type: VAR_OUTPUT)	
Identifiers	Data type	Value/meaning	
nState	Integer	States If a transmission error occurs during the parameterisation process or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.	
		0	Transmit request is executed. The module is parameterised.
		1	Transmit request dropped Processing is running
		2001	Error occurred during bus transfer
		2002	Transmit request timed out
		2004	Access denied Incorrect target system: index does not exist
		2005	Access denied Incorrect target system: subindex does not exist
		2013	Access denied Incorrect target system: invalid value range
		2111	Access denied Incorrect target system: More than 4 decimal positions specified or no fixed point format index
		2117	Communication channel does not exist byCom-Channelsetting
		-10xx	Incorrect value at function block input. XX := number of faulty input (counting direction down), parameterisation impossible. Remedy: change input variable, restart data transmission
		nState	Integer
-150	Can bus is not in the "Operational" status.		
-121	Incorrect driver number		
-120	CAN driver for free CAN objects at present is not initialised.		
-119	The transmit request memory is full. The transmit request could not be entered anymore.		
-118	No free CAN channel available.		
-12	Message identifier is beyond the permissible range.		
-11	<i>pIOAddress</i> pointer does not point to PLC-RAM.		
0	Transmit or reception request has been carried out.		
1	Request has not yet been completed and still is available in the request memory.		
10	Telegram length is greater than 8 bytes. Telegram length is limited to 8 bytes.		
200	Data in the reception memory have been overwritten.		
bParDone	BOOL	Parameterisation	
		TRUE	Parameterisation completed
DataTO_IOData15	Struct	Structure for the L_IOData15 process data block Global variable of the "Struct" type	



Function library LenzelOSystem.lib

Function blocks

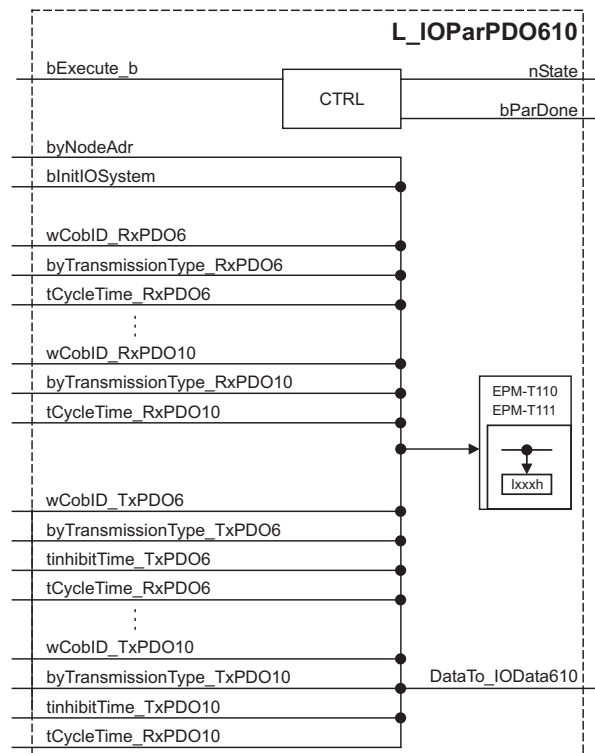
L_IOParPDO610 - Parameterisation function

4.9

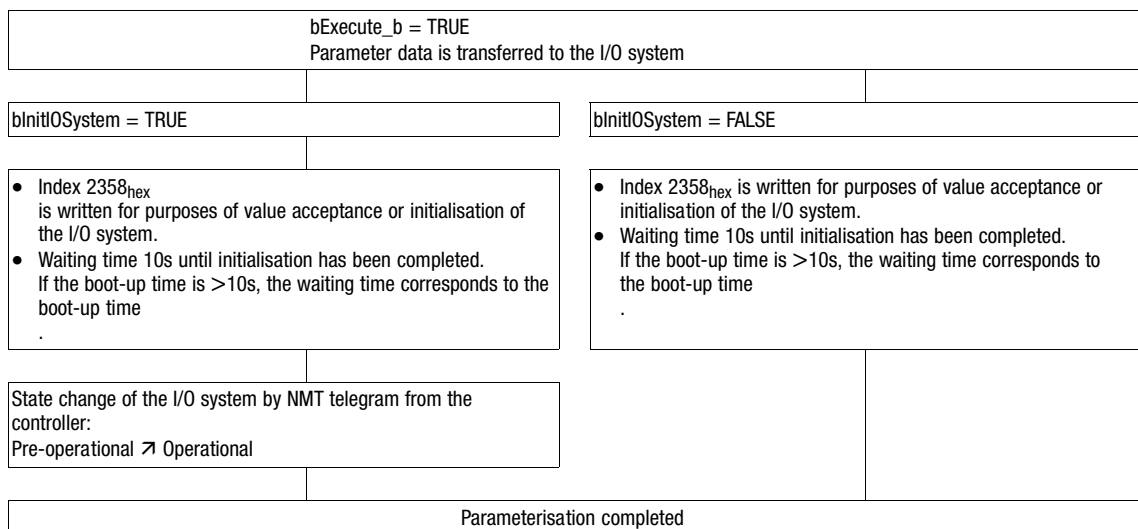
L_IOParPDO610 - Parameterisation function

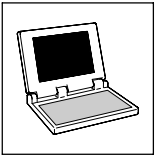
This function block serves to set communication-relevant parameters between the PLC control and the I/O system for PDO 6 - 10.

- Via 3 or 4 input values, a transmission/reception object is parameterised.
- The output *nState* display the parameterisation status.
- Via the output *DataTo_IOData610*, information is transferred to the **L_IOData610** process data block in a fixed structure.



Internal function





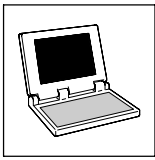
Note!

Features and use of the **L_IOParPDO610** FB.

- Use the **L_IOParPDO610** FB last in a sequencer consisting of several FBs for parameterisation. By means of the **L_IOParPDO610** FB, parameterisation of the I/O system is carried out.
- Parameterise the controller as master.
After the boot-up time has expired, for the master the change from the *Pre-operational* to the *Operational* status is effected automatically.
- For sending the NMT telegram, the controller has to be in the *Operational* CAN status.
- If several IO systems (stations) are parameterised with standard values via the **L_IOParPDO610** FB on the system bus, an identifier collision between the individual PDOs of these stations can occur. Therefore check the identifiers used.

After changing to the *Operational* CAN status, the current process image of the I/O system is sent.

⇒ Inputs		(Variable type: VAR_INPUT)			
Variable name	Data type	Information/possible settings			
bExecute	BOOL	Execute write request			
		FALSE ↗ TRUE		Write request is executed	
byNodeAdr	Byte	Device address: 1 - 63			
bInitIOSystem	BOOL	Initialise I/O system after parameterisation			
		TRUE		● Value acceptance by reset node	
		FALSE		● No value acceptance ● I/O system is not initialised	
wCobID_RxPDO6 ... wCobID_RxPDO10	WORD	Setting identifiers Default values: input wiring = 0 RxPDO = receive identifiers from the I/O system Identifiers can be optionally set by value selection. If the input wiring is "0", the identifiers are composed as follows. Note: The basic identifiers are not settled in the pre-assigned area of the standard system bus. ID range: 384-831 By this, crossovers of the ID with the free CAN objects which the IO-lib is based on are avoided.			
		RxPDOX	Identifier =	Basic ID +	Node address
		RxPDO6	862	860	2
		RxPDO7		865	
		RxPDO8		870	
		RxPDO9		875	
		RxPDO10		880	
byTransmissionType_RxPDO6 ... byTransmissionType_RxPDO10	Byte	Transmission mode			
		0	Event-controlled		
		1	Time-controlled		
		2	Event-controlled with superimposed cycle time		
tCyceTime_RxPDO6 ... tCyceTime_RxPDO10	Time	Cycle time: t = 0 - 65535 ms			

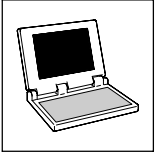


Function library *LenzeIOSystem.lib*

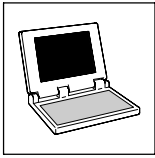
Function blocks

L_IOParPDO610 - Parameterisation function

⇒ Inputs (Variable type: VAR_INPUT)																				
Variable name	Data type	Information/possible settings																		
wCobID_TxPDO6 ... wCobID_TxPDO10	WORD	Setting identifiers Default values: input wiring = 0 TxPDO = receive identifiers from the I/O system Identifiers can be optionally set by value selection. If the input wiring is "0", the identifiers are composed as follows. Note: The basic identifiers are not settled in the pre-assigned area of the standard system bus. ID range: 384-831 By this, crossovers of the ID with the free CAN objects which the IO-lib is based on are avoided. <table><tr><th>TxPDOX</th><th>Identifier =</th><th>Basic ID +</th><th>Node address</th></tr><tr><td>TxPDO6</td><td rowspan="5">863</td><td>861</td><td rowspan="6">2</td></tr><tr><td>TxPDO7</td><td>866</td></tr><tr><td>TxPDO8</td><td>871</td></tr><tr><td>TxPDO9</td><td>876</td></tr><tr><td>TxPDO10</td><td>881</td></tr></table>			TxPDOX	Identifier =	Basic ID +	Node address	TxPDO6	863	861	2	TxPDO7	866	TxPDO8	871	TxPDO9	876	TxPDO10	881
TxPDOX	Identifier =	Basic ID +	Node address																	
TxPDO6	863	861	2																	
TxPDO7		866																		
TxPDO8		871																		
TxPDO9		876																		
TxPDO10		881																		
byTransmissionType_TxPDO6 ... byTransmissionType_TxPDO10	Byte	Transmission mode <table><tr><td>0</td><td>Event-controlled</td></tr><tr><td>1</td><td>Time-controlled</td></tr><tr><td>2</td><td>Event-controlled with superimposed cycle time</td></tr></table>			0	Event-controlled	1	Time-controlled	2	Event-controlled with superimposed cycle time										
0	Event-controlled																			
1	Time-controlled																			
2	Event-controlled with superimposed cycle time																			
tinhibitTime_TxPDO6 ... tinhibitTime_TxPDO10	Time	Inhibit time: t = 0 - 65535 ms																		
tCyceTime_TxPDO6 ... tCyceTime_TxPDO10	Time	Cycle time: t = 0 - 65535 ms																		



Outputs ⇔		(Variable type: VAR_OUTPUT)
Identifiers	Data type	Value/meaning
nState	integer	States If a transmission error occurs during the parameterisation process or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0Transmit request is executed. The module is parameterised.
		1Transmit request dropped Processing is running
		2001Error occurred during bus transfer
		2002Transmit request timed out
		2004Access denied Incorrect target system: index does not exist
		2005Access denied Incorrect target system: subindex does not exist
		2013Access denied Incorrect target system: invalid value range
		2111Access denied Incorrect target system: More than 4 decimal positions specified or no fixed point format index
		2117Communication channel does not exist byCom-Channelsetting
		-10xxIncorrect value at function block input. XX := number of faulty input (counting direction down), parameterisation impossible. Remedy: change input variable, restart data transmission
		nState
-150Can bus is not in the "Operational" status.		
-121Incorrect driver number		
-120CAN driver for free CAN objects at present is not initialised.		
-119The transmit request memory is full. The transmit request could not be entered anymore.		
-118No free CAN channel available.		
-12Message identifier is beyond the permissible range.		
-11 <i>pIOAddress</i> pointer does not point to PLC-RAM.		
0Transmit or reception request has been carried out.		
1Request has not yet been completed and still is available in the request memory.		
10Telegram length is greater than 8 bytes. Telegram length is limited to 8 bytes.		
200Data in the reception memory have been overwritten.		
bParDone	BOOL	Parameterisation TRUEParameterisation completed
DataTO_IOData610	Struct	Structure for the L_IOData610 process data block Global variable of the "Struct" type



Function library *LenzeIOSystem.lib*

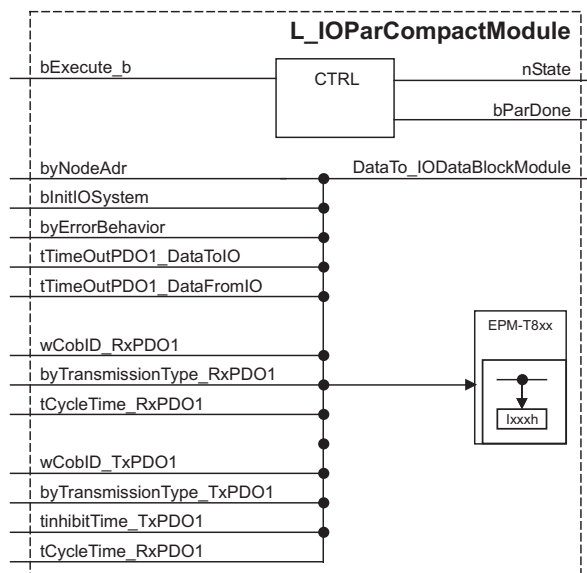
Function blocks

L_IOParCompactModule - Parameterisation function

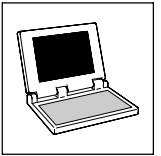
4.10 L_IOParCompactModule - Parameterisation function

This function block assists in the setting of parameters relevant to PLC control compact module communication.

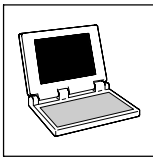
- Parameterising a function for process data exchange time monitoring.
- Output *nState* indicates the parameterisation process status.
- Use output *DataTo_IODataBlockModule* to transfer fixed-structure information to process data block **L_IOCompactModule**.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bExecute	Bool	Execute write request FALSE → TRUE Write request is executed
byNodeAdr	Byte	Device address: 1 - 63
bInitIOSystem	Bool	Initialise I/O system after parameterisation TRUE • Value accepted via Reset Node • I/O system is reinitialised, status changes to OPERATIONAL FALSE • No value accepted • I/O system is not initialised
byErrorBehavior	Byte	Set the bus states Behaviour during a critical device error 0 Pre-Operational 1 No status change 2 Stop 3 Reset after 2 seconds
tTimeOutPDO1_DataToIO	Time	Monitoring period for process input data (Lenze-specific). If no data are received at the selected PDO within the monitoring period, the I/O system assumes the state set under <i>byErrorBehavior</i> . Default t = 0 ms (monitoring off) t = 0 - 65535 ms
tTimeOutPDO1_DataFromIO	Time	Monitoring period for process output data. If no data are transmitted from the I/O system to the PLC at the selected PDO within the monitoring period, function block L_IODataXXX signals a time monitoring error. Default t = 0 ms (monitoring off) t = 0 - 65535 ms



⇒ Inputs		(Variable type: VAR_INPUT)																		
Variable name	Data type	Information/possible settings																		
wCobID_RxPDO1	Word	Set identifiers Default values: Input setting = 0 RxPDO = I/O system receive identifiers Identifiers can be given any setting with a specified value. The identifiers are composed as follows if the input setting = 0. Note! The basic identifiers are outside the preassigned standard system bus range. ID range: 384-831 This prevents IDs from coinciding with unassigned CAN objects on which the IO-lib is based. <table><tr><td>RxPDOX</td><td>Identifier =</td><td>Basic ID +</td><td colspan="2">Node address</td></tr><tr><td>RxPDO1</td><td>837</td><td>835</td><td colspan="2">2</td></tr></table>				RxPDOX	Identifier =	Basic ID +	Node address		RxPDO1	837	835	2						
RxPDOX	Identifier =	Basic ID +	Node address																	
RxPDO1	837	835	2																	
byTransmissionType_RxPDO1	Byte	Transmission method <table><tr><td>0</td><td colspan="4">Event-controlled</td></tr><tr><td>1</td><td colspan="4">Time-controlled</td></tr><tr><td>2</td><td colspan="4">Event-controlled with superimposed cycle time</td></tr></table>				0	Event-controlled				1	Time-controlled				2	Event-controlled with superimposed cycle time			
0	Event-controlled																			
1	Time-controlled																			
2	Event-controlled with superimposed cycle time																			
tCyceTime_TxPDO6 ... tCyceTime_TxPDO10	Time	Cycle time t = 0 - 65535 ms																		
wCobID_TxPDO1	Word	Set identifiers Default values: Input setting = 0 TxPDO = I/O system receive identifiers Identifiers can be given any setting with a specified value. The identifiers are composed as follows if the input setting = 0. The basic identifiers are outside the preassigned standard system bus range. ID range: 384-831 This prevents IDs from coinciding with unassigned CAN objects on which the IO-lib is based. <table><tr><td>TxPDOX</td><td>Identifier =</td><td>Basic ID +</td><td colspan="2">Node address</td></tr><tr><td>TxPDO1</td><td>838</td><td>836</td><td colspan="2">2</td></tr></table>				TxPDOX	Identifier =	Basic ID +	Node address		TxPDO1	838	836	2						
TxPDOX	Identifier =	Basic ID +	Node address																	
TxPDO1	838	836	2																	
byTransmissionType_TxPDO1	Byte	Transmission method <table><tr><td>0</td><td colspan="4">Event-controlled</td></tr><tr><td>1</td><td colspan="4">Time-controlled</td></tr><tr><td>2</td><td colspan="4">Event-controlled with superimposed cycle time</td></tr></table>				0	Event-controlled				1	Time-controlled				2	Event-controlled with superimposed cycle time			
0	Event-controlled																			
1	Time-controlled																			
2	Event-controlled with superimposed cycle time																			
tInhibitTime_TxPDO6 ... tInhibitTime_TxPDO10	Time	Inhibit time t = 0 - 65535 ms																		
tCyceTime_TxPDO6 ... tCyceTime_TxPDO10	Time	Cycle time t = 0 - 65535 ms																		

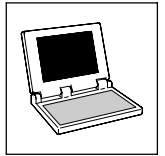


Function library *LenzeIOSystem.lib*

Function blocks

L_IOParCompactModule - Parameterisation function

Outputs ⇨ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		0 Transmit command is executed. The module is parameterised.
		1 Transmit command sent Processing in process
		2001 Error occurred during bus transfer
		2002 Transmit command timed out
		2004 Access denied Wrong target system: Index does not exist
		2005 Access denied Wrong target system: Subindex does not exist
		2013 Access denied Wrong target system: Invalid value range
		2111 Access denied Wrong target system: More than 4 digits specified after comma, or no fixed-comma-format index
		2117 Communication channel does not exist Set byCom channel
		-10xx Incorrect value at function block input. XX := Number of faulty input (counting direction down), parameterisation impossible. Remedy: Change input value, restart data transmission
nState	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i> . Any errors are reset via an edge reversal at input <i>bExecute</i> or on program restart.
		-150 CAN bus not in Operational status.
		-121 Wrong driver number
		-120 CAN driver for unassigned CAN objects currently not initialised.
		-119 Transmit command memory is full. Transmit command could not be entered.
		-118 No free CAN channel available.
		-12 Message identifier outside permissible range.
		-11 <i>pIOAddress</i> pointer does not point at PLC RAM.
		0 Transmit and/or receive command is executed.
		1 Command has not yet been completed and is still in the command memory.
bParDone	Byte	Parameterisation status
		TRUE Parameterisation executed
DataTO_IODataCompactmodule	Struct	Structure for process data block L_IOCompactModule Global struct-type variable



5 Function blocks for process data processing

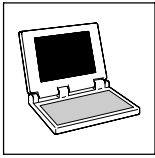
This chapter contains information on function blocks for coordination, support and conversion.

The following functions and modules are coordinated:

- Digital and analog input and output data

The following functions and modules are supported:

- Analog input modules
- Analog output modules
- Counter modules
- Link between PLC control and I/O system
- Link between PLC control and compact module



Function library *LenzeIOSystem.lib*

Function blocks

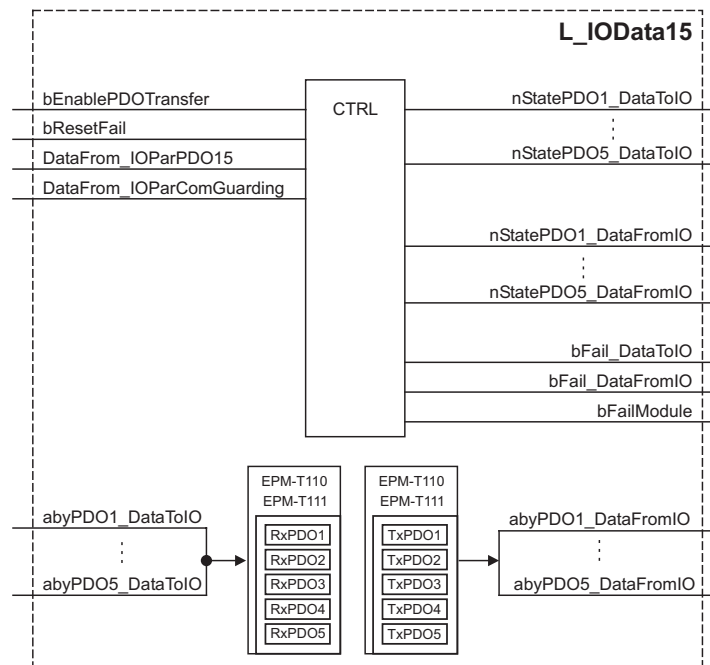
L_IOData15 - Coordinate data

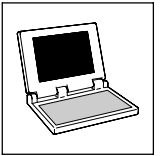
5.1 L_IOData15 - Coordinate data

Function block **L_IOData15** coordinates digital and analog input and/or output data.

The block represents the interface between PLC and I/O system.

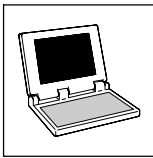
- Via inputs *abyPDO_DataToIO*, data are transmitted from the control to the I/O system (digital/analog output channels).
- The states of the digital and analog input channels are transmitted to the control via outputs *abyPDO_DataFromIO*.
- This function block is made up of unassigned CAN objects so that the status display at outputs *nState...* is the same as for the CAN library's transmit and receive blocks.





⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bEnablePD0Transfer	Bool	Data transfer TRUE PDO data transfer active
abyPD01_DataToIO ... abyPD05_DataToIO	Byte array	I/O system receive objects • Transfer process data from PLC to I/O system (digital/analog outputs)
bResetFail	Bool	Reset output data <i>bFail_DataToIO</i> , <i>bFail_DataFromIO</i> and <i>bFail_Module</i> TRUE Output data reset
DataFrom_IOParPD015	Struct	Structural data of parameter block L_IOParPD015 are transferred to process data block L_IOData15 via this input. Global struct-type variable
DataFrom_IOParComGuarding	Struct	Transfer monitoring times <i>tTimeout_DataFromIO</i> for PDO 1-5. Structural data of parameter block L_IOParComGuarding . Global struct-type variable

Outputs ⇒ (Variable type: VAR_OUTPUT)																												
Identifier	Data type	Value/meaning																										
nStatePDO1_DataFromIO ... nStatePDO5_DataFromIO	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i>. <table><tr><td>-150</td><td>CAN bus not in Operational status.</td></tr><tr><td>-121</td><td>Wrong driver number</td></tr><tr><td>-120</td><td>CAN driver for unassigned CAN objects currently not initialised.</td></tr><tr><td>-119</td><td>Transmit command memory is full. Transmit command could not be entered.</td></tr><tr><td>-118</td><td>No free CAN channel available.</td></tr><tr><td>-12</td><td>Message identifier outside permissible range.</td></tr><tr><td>-11</td><td><i>pIOAddress</i> pointer does not point at PLC RAM.</td></tr><tr><td>0</td><td>Transmit and/or receive command is executed.</td></tr><tr><td>1</td><td>Command has not yet been completed and is still in the command memory.</td></tr><tr><td>2</td><td>Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete </td></tr><tr><td>10</td><td>Telegram longer than 8 bytes. Telegram length limited to 8 bytes.</td></tr><tr><td>200</td><td>Data in receive memory have been overwritten.</td></tr><tr><td>-30xx</td><td>TimeOut monitoring (xx ≅ PDOxx) Data transfer from I/O system to PLC failed. Data arrived late or not at all.</td></tr></table>	-150	CAN bus not in Operational status.	-121	Wrong driver number	-120	CAN driver for unassigned CAN objects currently not initialised.	-119	Transmit command memory is full. Transmit command could not be entered.	-118	No free CAN channel available.	-12	Message identifier outside permissible range.	-11	<i>pIOAddress</i> pointer does not point at PLC RAM.	0	Transmit and/or receive command is executed.	1	Command has not yet been completed and is still in the command memory.	2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete	10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.	200	Data in receive memory have been overwritten.	-30xx	TimeOut monitoring (xx ≅ PDOxx) Data transfer from I/O system to PLC failed. Data arrived late or not at all.
-150	CAN bus not in Operational status.																											
-121	Wrong driver number																											
-120	CAN driver for unassigned CAN objects currently not initialised.																											
-119	Transmit command memory is full. Transmit command could not be entered.																											
-118	No free CAN channel available.																											
-12	Message identifier outside permissible range.																											
-11	<i>pIOAddress</i> pointer does not point at PLC RAM.																											
0	Transmit and/or receive command is executed.																											
1	Command has not yet been completed and is still in the command memory.																											
2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete																											
10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.																											
200	Data in receive memory have been overwritten.																											
-30xx	TimeOut monitoring (xx ≅ PDOxx) Data transfer from I/O system to PLC failed. Data arrived late or not at all.																											

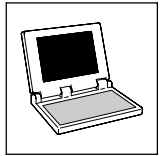


Function library *LenzeIOSystem.lib*

Function blocks

L_IOData15 - Coordinate data

Outputs ⇨																										
(Variable type: VAR_OUTPUT)																										
Identifier	Data type	Value/meaning																								
nStatePD01_DataToIO ... nStatePD05_DataToIO	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i>. <table><tr><td>-150</td><td>CAN bus not in Operational status.</td></tr><tr><td>-121</td><td>Wrong driver number</td></tr><tr><td>-120</td><td>CAN driver for unassigned CAN objects currently not initialised.</td></tr><tr><td>-119</td><td>Transmit command memory is full. Transmit command could not be entered.</td></tr><tr><td>-118</td><td>No free CAN channel available.</td></tr><tr><td>-12</td><td>Message identifier outside permissible range.</td></tr><tr><td>-11</td><td><i>pIOAddress</i> pointer does not point at PLC RAM.</td></tr><tr><td>0</td><td>Transmit and/or receive command is executed.</td></tr><tr><td>1</td><td>Command has not yet been completed and is still in the command memory.</td></tr><tr><td>2</td><td>Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete </td></tr><tr><td>10</td><td>Telegram longer than 8 bytes. Telegram length limited to 8 bytes.</td></tr><tr><td>200</td><td>Data in receive memory have been overwritten.</td></tr></table>	-150	CAN bus not in Operational status.	-121	Wrong driver number	-120	CAN driver for unassigned CAN objects currently not initialised.	-119	Transmit command memory is full. Transmit command could not be entered.	-118	No free CAN channel available.	-12	Message identifier outside permissible range.	-11	<i>pIOAddress</i> pointer does not point at PLC RAM.	0	Transmit and/or receive command is executed.	1	Command has not yet been completed and is still in the command memory.	2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete	10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.	200	Data in receive memory have been overwritten.
-150	CAN bus not in Operational status.																									
-121	Wrong driver number																									
-120	CAN driver for unassigned CAN objects currently not initialised.																									
-119	Transmit command memory is full. Transmit command could not be entered.																									
-118	No free CAN channel available.																									
-12	Message identifier outside permissible range.																									
-11	<i>pIOAddress</i> pointer does not point at PLC RAM.																									
0	Transmit and/or receive command is executed.																									
1	Command has not yet been completed and is still in the command memory.																									
2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete																									
10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.																									
200	Data in receive memory have been overwritten.																									
bFail_DataToIO	Bool	Transmit command to I/O system <table><tr><td>TRUE</td><td>Transmit command to I/O system is not correct</td></tr></table>	TRUE	Transmit command to I/O system is not correct																						
TRUE	Transmit command to I/O system is not correct																									
bFail_DataFromIO	Bool	Receive command from I/O system <table><tr><td>TRUE</td><td>Receive command from I/O system is not correct</td></tr></table>	TRUE	Receive command from I/O system is not correct																						
TRUE	Receive command from I/O system is not correct																									
bFail_Module	Bool	Module error <table><tr><td>TRUE</td><td> • Module diagnostics active • Emergency object indicates error </td></tr></table>	TRUE	• Module diagnostics active • Emergency object indicates error																						
TRUE	• Module diagnostics active • Emergency object indicates error																									
abyPD01_DataFromIO ... abyPD05_DataFromIO	Byte array	I/O system transmit objects Transfer process data from I/O system to PLC (digital/analog inputs)																								

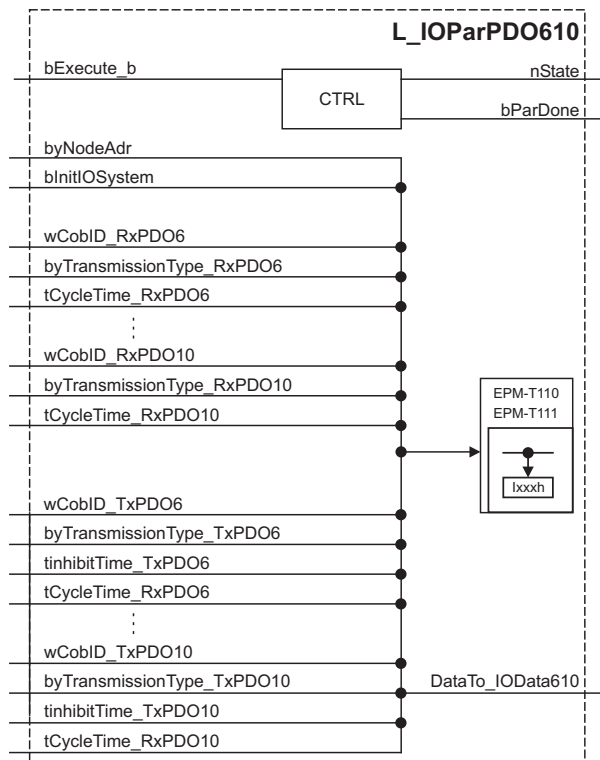


5.2 L_IOData610 - Coordinate data

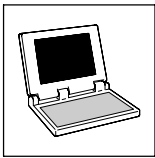
Function block **L_IOData610** coordinates digital and analog input and/or output data for process data objects 6 - 10.

The block represents the interface between PLC and I/O system.

- Via inputs *abyPDO_DataToIO*, data are transmitted from the control to the I/O system (digital/analog output channels).
- The states of the digital and analog input channels are transmitted to the control via outputs *abyPDO_DataFromIO*.
- This function block is made up of unassigned CAN objects so that the status display at outputs *nState...* is the same as for the CAN library's transmit and receive blocks.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bEnablePDOTransfer	Bool	Data transfer TRUE PDO data transfer active
abyPDO6_DataToIO ... abyPDO10_DataToIO	Byte array	I/O system receive objects • Transfer process data from PLC to I/O system (digital/analog outputs)
bResetFail	Bool	Reset output data <i>bFail_DataToIO</i> and <i>bFail_DataFromIO</i> TRUE Output data reset
DataFrom_IOParPDO610	Struct	Structural data of parameter block L_IOParPDO610 are transferred to process data block L_IOData610 via this input. Global struct-type variable
DataFrom_IOParComGuarding	Struct	Transfer monitoring times <i>tTimeOut_DataFromIO</i> for PDO 6-10. Structural data of parameter block L_IOParComGuarding . Global struct-type variable

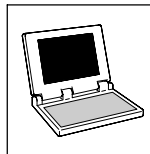


Function library *LenzeIOSystem.lib*

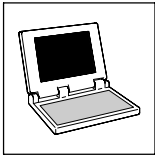
Function blocks

L_IOData610 - Coordinate data

Outputs ⇨ (Variable type: VAR_OUTPUT)																										
Identifier	Data type	Value/meaning																								
nStatePDO6_DataToIO ... nStatePDO10_DataToIO	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i>. <table><tr><td>-150</td><td>CAN bus not in Operational status.</td></tr><tr><td>-121</td><td>Wrong driver number</td></tr><tr><td>-120</td><td>CAN driver for unassigned CAN objects currently not initialised.</td></tr><tr><td>-119</td><td>Transmit command memory is full. Transmit command could not be entered.</td></tr><tr><td>-118</td><td>No free CAN channel available.</td></tr><tr><td>-12</td><td>Message identifier outside permissible range.</td></tr><tr><td>-11</td><td><i>pIOAddress</i> pointer does not point at PLC RAM.</td></tr><tr><td>0</td><td>Transmit and/or receive command is executed.</td></tr><tr><td>1</td><td>Command has not yet been completed and is still in the command memory.</td></tr><tr><td>2</td><td>Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete </td></tr><tr><td>10</td><td>Telegram longer than 8 bytes. Telegram length limited to 8 bytes.</td></tr><tr><td>200</td><td>Data in receive memory have been overwritten.</td></tr></table>	-150	CAN bus not in Operational status.	-121	Wrong driver number	-120	CAN driver for unassigned CAN objects currently not initialised.	-119	Transmit command memory is full. Transmit command could not be entered.	-118	No free CAN channel available.	-12	Message identifier outside permissible range.	-11	<i>pIOAddress</i> pointer does not point at PLC RAM.	0	Transmit and/or receive command is executed.	1	Command has not yet been completed and is still in the command memory.	2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete	10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.	200	Data in receive memory have been overwritten.
-150	CAN bus not in Operational status.																									
-121	Wrong driver number																									
-120	CAN driver for unassigned CAN objects currently not initialised.																									
-119	Transmit command memory is full. Transmit command could not be entered.																									
-118	No free CAN channel available.																									
-12	Message identifier outside permissible range.																									
-11	<i>pIOAddress</i> pointer does not point at PLC RAM.																									
0	Transmit and/or receive command is executed.																									
1	Command has not yet been completed and is still in the command memory.																									
2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete																									
10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.																									
200	Data in receive memory have been overwritten.																									



Outputs ⇒ (Variable type: VAR_OUTPUT)																												
Identifier	Data type	Value/meaning																										
nStatePDO6_DataFromIO ... nStatePDO10_DataFromIO	Integer	States If a transmission error occurs during the parameterisation process, or the input data are incorrect, the parameterisation process is aborted and an error number is indicated at output <i>nState</i>. <table><tr><td>-150</td><td>CAN bus not in Operational status.</td></tr><tr><td>-121</td><td>Wrong driver number</td></tr><tr><td>-120</td><td>CAN driver for unassigned CAN objects currently not initialised.</td></tr><tr><td>-119</td><td>Transmit command memory is full. Transmit command could not be entered.</td></tr><tr><td>-118</td><td>No free CAN channel available.</td></tr><tr><td>-12</td><td>Message identifier outside permissible range.</td></tr><tr><td>-11</td><td><i>pIOAddress</i> pointer does not point at PLC RAM.</td></tr><tr><td>0</td><td>Transmit and/or receive command is executed.</td></tr><tr><td>1</td><td>Command has not yet been completed and is still in the command memory.</td></tr><tr><td>2</td><td>Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete </td></tr><tr><td>10</td><td>Telegram longer than 8 bytes. Telegram length limited to 8 bytes.</td></tr><tr><td>200</td><td>Data in receive memory have been overwritten.</td></tr><tr><td>-30xx</td><td>TimeOut monitoring (xx ≙ PDOxx) Data transfer from I/O system to PLC failed. Data arrived late or not at all.</td></tr></table>	-150	CAN bus not in Operational status.	-121	Wrong driver number	-120	CAN driver for unassigned CAN objects currently not initialised.	-119	Transmit command memory is full. Transmit command could not be entered.	-118	No free CAN channel available.	-12	Message identifier outside permissible range.	-11	<i>pIOAddress</i> pointer does not point at PLC RAM.	0	Transmit and/or receive command is executed.	1	Command has not yet been completed and is still in the command memory.	2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete	10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.	200	Data in receive memory have been overwritten.	-30xx	TimeOut monitoring (xx ≙ PDOxx) Data transfer from I/O system to PLC failed. Data arrived late or not at all.
-150	CAN bus not in Operational status.																											
-121	Wrong driver number																											
-120	CAN driver for unassigned CAN objects currently not initialised.																											
-119	Transmit command memory is full. Transmit command could not be entered.																											
-118	No free CAN channel available.																											
-12	Message identifier outside permissible range.																											
-11	<i>pIOAddress</i> pointer does not point at PLC RAM.																											
0	Transmit and/or receive command is executed.																											
1	Command has not yet been completed and is still in the command memory.																											
2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete																											
10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.																											
200	Data in receive memory have been overwritten.																											
-30xx	TimeOut monitoring (xx ≙ PDOxx) Data transfer from I/O system to PLC failed. Data arrived late or not at all.																											
bFail_DataToIO	Bool	Transmit command to I/O system <table><tr><td>TRUE</td><td>Transmit command to I/O system is not correct</td></tr></table>	TRUE	Transmit command to I/O system is not correct																								
TRUE	Transmit command to I/O system is not correct																											
bFail_DataFromIO	Bool	Receive command from I/O system <table><tr><td>TRUE</td><td>Receive command from I/O system is not correct</td></tr></table>	TRUE	Receive command from I/O system is not correct																								
TRUE	Receive command from I/O system is not correct																											
abyPDO6_DataFromIO ... abyPDO10_DataFromIO	Byte array	I/O system transmit objects Transfer process data from I/O system to PLC (digital/analog inputs)																										



Function library *LenzeIOSystem.lib*

Function blocks

L_IOCompactModule - Coordinate data

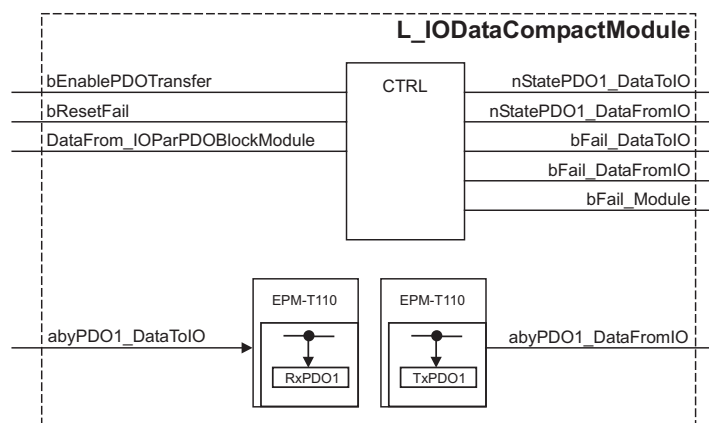
5.3

L_IOCompactModule - Coordinate data

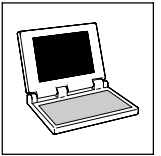
Function block **L_IOCompactModule** supplies the digital input and/or output data of the I/O system and is used for compact module EPM-T83X.

The block represents the interface between PLC and I/O system.

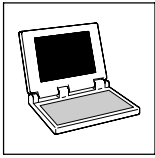
- Via inputs *abyPDO_DataToIO*, data are transmitted from the control to the I/O system (digital/analog output channels).
- The states of the digital and analog input channels are transmitted to the control via outputs *abyPDO_DataFromIO*.
- This function block is made up of unassigned CAN objects so that the status display at outputs *nState...* is the same as for the CAN library's transmit and receive blocks.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bEnablePDOTransfer	Bool	Data transfer TRUE PDO data transfer active
abyPDO1_DataToIO	Byte array	I/O system receive object • Transfer process data from PLC to I/O system (digital outputs)
bResetFail	Bool	Reset output data <i>bFail_DataToIO</i> , <i>bFail_DataFromIO</i> and <i>bFail_Module</i> TRUE Output data reset
DataFrom_IOParPDOCompactModule	Struct	Structural data of parameter block L_IOParPDOCompactModule . Global struct-type variable.



Outputs ⇨		(Variable type: VAR_OUTPUT)	
Identifier	Data type	Value/meaning	
nStatePDO1_DataToIO	Integer	States Indicates actual status of the transmit objects.	
		-150	CAN bus not in Operational status.
		-121	Wrong driver number
		-120	CAN driver for unassigned CAN objects currently not initialised.
		-119	Transmit command memory is full. Transmit command could not be entered.
		-118	No free CAN channel available.
		-12	Message identifier outside permissible range.
		-11	pIOAddress pointer does not point at PLC RAM.
		0	Transmit and/or receive command is executed.
		1	Command has not yet been completed and is still in the command memory.
		2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete
		10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.
		200	Data in receive memory have been overwritten.
nStatePDO1_DataFromIO	Integer	States Indicates actual status of the receive objects.	
		-150	CAN bus not in Operational status.
		-121	Wrong driver number
		-120	CAN driver for unassigned CAN objects currently not initialised.
		-119	Transmit command memory is full. Transmit command could not be entered.
		-118	No free CAN channel available.
		-12	Message identifier outside permissible range.
		-11	pIOAddress pointer does not point at PLC RAM.
		0	Transmit and/or receive command is executed.
		1	Command has not yet been completed and is still in the command memory.
		2	Data transmission disabled • bEnablePDOTransfer = False • I/O system parameterisation not yet complete
		10	Telegram longer than 8 bytes. Telegram length limited to 8 bytes.
		200	Data in receive memory have been overwritten.
		-30xx	TimeOut monitoring Data transfer from I/O system to PLC failed. Data arrived late or not at all.

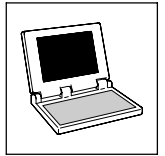


Function library *LenzeIOSystem.lib*

Function blocks

L_IOCompactModule - Coordinate data

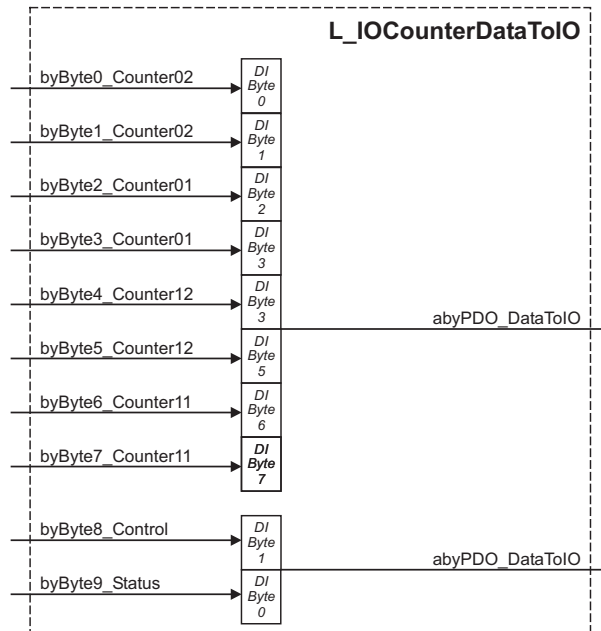
Outputs ⇨ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
bFail_DataToIO	Bool	Transmit command to I/O system
		TRUE Transmit command to I/O system is not correct
bFail_DataFromIO	Bool	Receive command from I/O system
		TRUE Receive command from I/O system is not correct
bFail_Module	Bool	Module error
		TRUE - Module diagnostics active - Emergency object indicates error
abyPDO1_DataFromIO	Byte array	I/O system transmit object Transfer process data from I/O system to PLC (digital inputs)



5.4 L_IOCounterDataToIO - Counter function

The counter module function is set via parameterisation data block

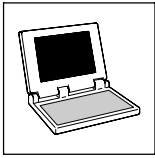
L_IOParCounterDataToIO. Depending on this parameterisation, the counter may be preassigned values for counting or comparison. Function block **L_IOCounterDataToIO** is used for this preassignment.



Assignment of bytes 0...7 is dependent on the selected mode.

The counter mode default value is zero, resulting in the following assignment.

- Counter 0 preassignment with counting values is assisted by inputs
 - byByte0_Counter02 *LowByte Bit16...23*
 - byByte1_Counter02 *HighByte Bit24...32*
 - byByte2_Counter01 *LowByte Bit0...7*
 - byByte3_Counter01 *HighByte Bit8...15*
- Setting of respective values for counter 1 is via inputs
 - byByte4_Counter12 *LowByte Bit16...23*
 - byByte5_Counter12 *HighByte Bit24...32*
 - byByte6_Counter11 *LowByte Bit0...7*
 - byByte7_Counter11 *HighByte Bit8...15*

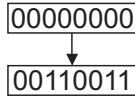


Function library LenzelOSystem.lib

Function blocks

L_I0CounterDataToIO - Counter function

- After an edge reversal at input *byByte8_Control* from Low to High, the specified values are transmitted to the counter module.
Selection in control byte



byByte9_Status

This input may be used to store the last count value in the module, or to start counting at 0 on restart.

- 1 = Count value is stored
- 2 = Count value is cleared on restart (default)
- The written data are supplied at the following outputs:
 - *abyPDO_Data1ToIO*
 - *abyPDO_Data2ToIO*

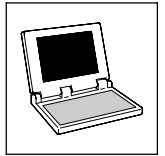


Note!

The counter module takes up two Rx and two Tx PDOs.

⇒ Inputs		(Variable type: VAR_INPUT)				
Variable name	Data type	Information/possible settings				
byByte0_Counter02 ... byByte7_Counter11	Byte	Specify start or comparison values to the counter module. Example: The module was parameterised as 2 x 32 bit counter. If the counter is to start at a count value 77, this value must be available at input <i>byByte2_Counter01</i> .				
byByte8_Control	Byte	Accept specified values at the following inputs after Low to High edge reversal. byByte0_Counter02 byByte7_Counter11 (☐ 6-10)				
byByte9_Status	Byte	Count value after restart <table><tr><td>1</td><td>Last count value is stored</td></tr><tr><td>2</td><td>Restart at count value 0 (default)</td></tr></table>	1	Last count value is stored	2	Restart at count value 0 (default)
1	Last count value is stored					
2	Restart at count value 0 (default)					

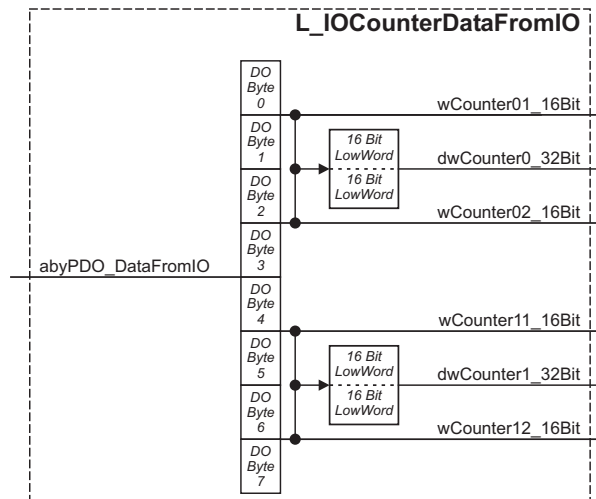
Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
<i>abyPDOData1ToIO</i>	Byte array	Interface with the inputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of information to the counter module.
<i>abyPDOData2ToIO</i>	Byte array	Interface with the inputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of information to the counter module.



5.5 L_IOCounterDataFromIO - Counter function

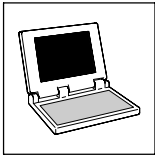
Function block **L_IOCounterDataFromIO** supplies the counter module's counting information.

- Counting information from 4 counters with a 16-bit data width is supplied at the following outputs:
 - wCounter01_16Bit
 - wCounter02_16Bit
 - wCounter11_16Bit
 - wCounter12_16Bit
- Counting information from 2 counters with a 32-bit data width is supplied at the following outputs:
 - dwCounter0_32Bit
 - dwCounter1_32Bit



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
abyPDO_DataFromIO	Byte array	Interface with the outputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of information from the counter module.

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
wCounter01_16Bit wCounter02_16Bit wCounter11_16Bit wCounter12_16Bit	Word	16-bit width counter data
dwCounter0_32Bit dwCounter1_32Bit	Double word	32-bit width counter data



Function library *LenzeIOSystem.lib*

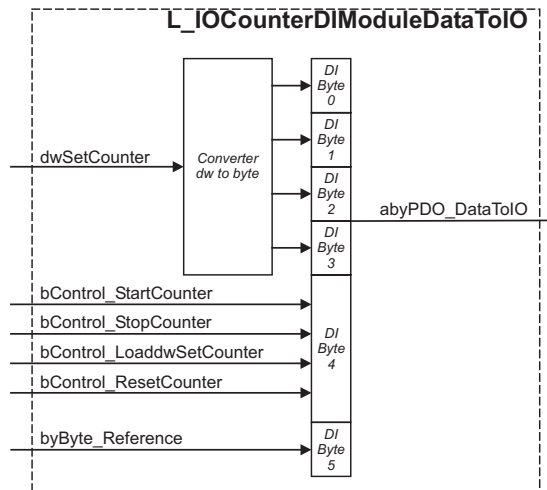
Function blocks

L_IOCounterDIModuleDataToIO - Counter function

5.6 L_IOCounterDIModuleDataToIO - Counter function

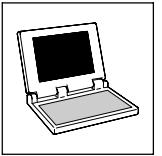
The counter function of module EPM-T430 is set via process data block **L_IOCounterDIModuleDataToIO**. Depending on the function settings for the inputs, the counter may be preassigned values for counting.

In addition, function block **L_IOCounterDIModuleDataToIO** is used for start, stop, set, reset, etc.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
dwSetCounter	Double word	Specified value for counter module
bControl_StartCounter	Bool	Start counter
bControl_StopCounter	Bool	Stop counter
bControl_LoaddwSetCounter	Bool	Specified value is set to counting value
bControl_ResetCounter	Bool	Counter is reset
byByte_Reference	Byte	Reference frequency
		0 16 MHz
		1 8 MHz
		2 4 MHz
		3 1 MHz
		4 100 kHz
		5 10 kHz
		6 1000Hz
		7 100Hz

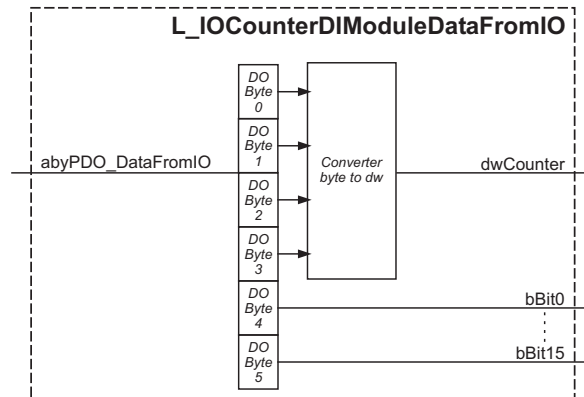
Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
abyPDOData1ToIO	Byte array	Interface with the inputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of information to the counter module.



5.7 L_IOCounterDIModuleDataFromIO - Counter function

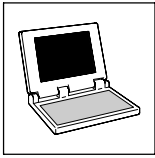
Function block **L_IOCounterDIModuleDataFromIO** supplies the counter module's counting information.

The 32-bit value of the counter is supplied at output *dwCounter*.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
abyPDO_DataFromIO	Byte array	Interface with outputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of count values from the counter module.

Outputs ⇨ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
dwCounter	Double word	Output count value
bBit0 - bBit15	Bool	Output status of digital inputs 0...15



Function library *LenzeIOSystem.lib*

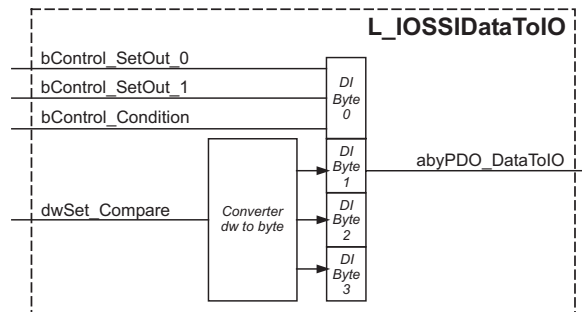
Function blocks

L_IOSSIDataToIO - Counter function

5.8 L_IOSSIDataToIO - Counter function

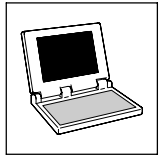
Depending on a comparison value, the two outputs of the module can be set via function block **L_IOSSIDataToIO**.

To the outputs, different comparison functions or comparison values can be assigned, which must be transferred to the SSI module in succession.



⇒ Inputs (Variable type: VAR_INPUT)						
Variable name	Data type	Information/possible settings				
bControl_Set_0	Bool	Output 0 Monitor comparison value <table><tr><td>0</td><td>Off</td></tr><tr><td>1</td><td>On</td></tr></table>	0	Off	1	On
0	Off					
1	On					
bControl_Set_1	Bool	Output 1 Monitor comparison value <table><tr><td>0</td><td>Off</td></tr><tr><td>1</td><td>On</td></tr></table>	0	Off	1	On
0	Off					
1	On					
bControl_Condition	Bool	Set condition of outputs <table><tr><td>0</td><td>Actual value > comparison value</td></tr><tr><td>1</td><td>Actual value < comparison value</td></tr></table>	0	Actual value > comparison value	1	Actual value < comparison value
0	Actual value > comparison value					
1	Actual value < comparison value					
dwSet_Compare	Double word	Enter comparison value				

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
abyPDO_DataToIO	Byte array	Interface with the inputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of information from the counter module.

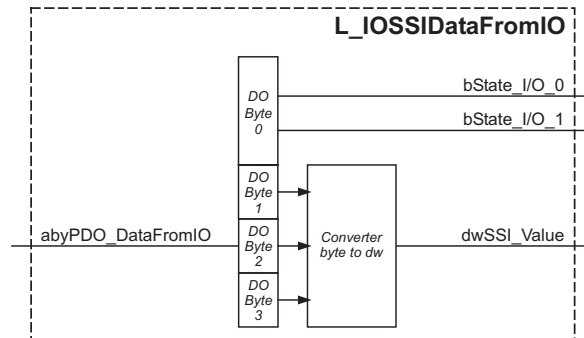


5.9 L_IOSSIDataFromIO - Counter function

Function block **L_IOSSIDataFromIO** supplies the count value of the SSI module at output *dwSSI_Value*.

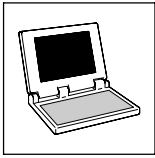
- Output *bStatusIO_0* → Status of I/O channel 0
- Output *bStatusIO_1* → Status of I/O channel 1

The outputs are configured via function block **L_IOSSIDataToIO**.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
abyPDO_DataToIO	Byte array	Interface with outputs of function blocks L_IOPD015 or L_IOPD0610 for the transfer of count values from the SSI module.

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
bState_I/O_0	Bool	Status of I/O channel 0
bState_I/O_1	Bool	Status of I/O channel 1
dwSSI_Value	Double word	Count value of SSI module



Function library *LenzeIOSystem.lib*

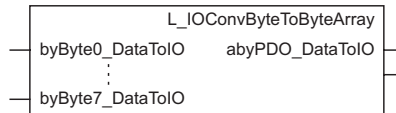
Function blocks

L_IOConvByteToByteArray - Conversion function

5.10 L_IOConvByteToByteArray - Conversion function

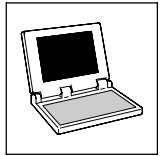
Function block **L_IOConvByteToByteArray** converts 8 bytes of output data into an 8-field byte array.

Due to this, the output data width is identical to exactly one transmit object *PDO_DataToIO* to the I/O system.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
byByte0_DataToIO	Byte	8 bytes digital output data to the I/O system
...		
byByte7_DataToIO		

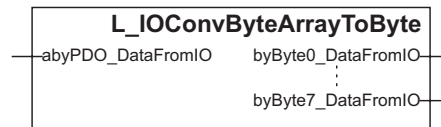
Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
abyPDO_DataToIO	Byte array	8-byte array: Maps one transmit object <i>PDO_DataToIO</i>



5.11 L_IOConvByteArrayToByte - Conversion function

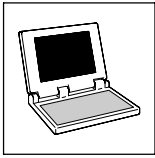
Function block **L_IOConvByteArrayToByte** converts 8 bytes of a receive object into 8 bytes of input data.

Due to that, the input data width is matched to one receive object *PDO_DataFromIO* from the I/O system.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
abyPDO_DataFromIO	Byte array	8-byte array: Maps one receive object <i>PDO_DataFromIO</i>

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
byByte0_DataFromIO ... byByte7_DataFromIO	Byte	Receive object: 8 x 1 bytes of digital/analog input data from the I/O system



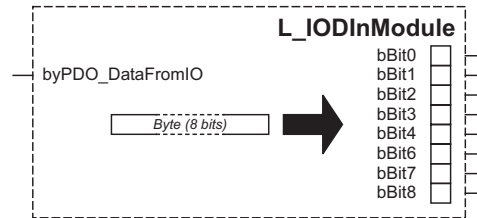
Function library *LenzeIOSystem.lib*

Function blocks

L_IODInModule - Bit conversion

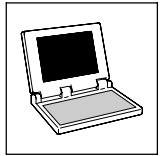
5.12 L_IODInModule - Bit conversion

Function block **L_IODInModule** corresponds to a digital input module (EPM-T210) and converts one data byte into the individual bits 0 - 7.



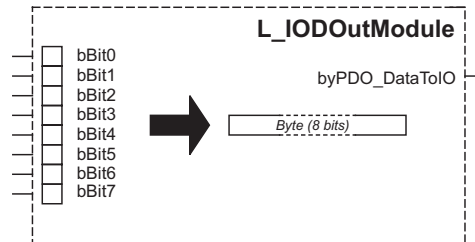
⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
byPDO_DataFromIO	Byte	Digital inputs: 1 byte data = 1 DI module

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
bBit0	Bool	Digital input information, bit 0.
...		...
bBit7		Digital input information, bit 7.



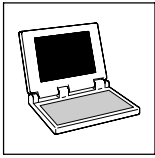
5.13 L_IODOutModule - Bit conversion

Function block **L_IODOutModule** corresponds to a digital output module (EPM-T22X) and converts the individual bits 0 - 7 into one data byte.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
bBit0	Bool	Digital output information, bit 0
...		...
bBit7		Digital output information, bit 7

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifier	Data type	Value/meaning
byPDO_DataFromIO	Byte	Digital outputs: 1 byte data = 1 DO module



Function library LenzelOSystem.lib

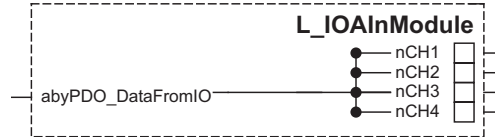
Function blocks

L_IOAInModule - Signal conversion

5.14 L_IOAInModule - Signal conversion

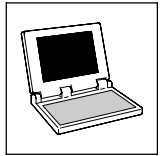
The **L_IOAInModule** function block corresponds to an analog input module (EPM-T310 / T311 / T312) and divides the data sent by the I/O system into the channel information 1 - 4.

The input data are represented in the integer format.



⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
abyPDO_DataFromIO	Byte array	Analog input information of channels 1 - 4

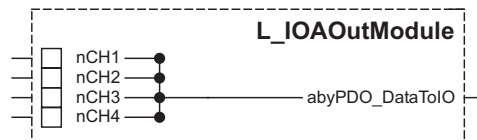
Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifiers	Data type	Value/meaning
nCH1 ... nCH4	Integer	Analog inputs: 4 x 2 bytes data = 1 AI module
		Default ±10V function code 2B _n 6-1



5.15 L_IOAOutModule - Signal conversion

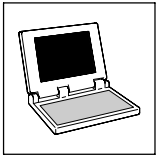
The **L_IOAOutModule** function block corresponds to an analog output module (EPM-T320 / T321 / T322) and bundles the information of channels 1 - 4 into a transmission object, which is sent to the I/O system.

The output data are represented in the integer format.



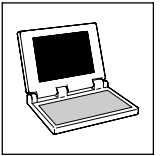
⇒ Inputs (Variable type: VAR_INPUT)		
Variable name	Data type	Information/possible settings
nCH1 ... nCH4	Byte	Analog outputs: 4 x 2 bytes data = 1 AO module
		Default ±10 = function code 01 _h 6-4

Outputs ⇒ (Variable type: VAR_OUTPUT)		
Identifiers	Data type	Value/meaning
abyPDO_DataFromIO	Byte array	Analog output information of channels 1 - 4



Function library LenzeIOSystem.lib

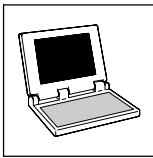
Function blocks



6 Appendix

6.1 Signal functions for analog inputs EPM-T310

Parameter bytes 2/3/4/5	Signal function	Signal range	Format	Tolerance ¹⁾
00 _h	Parameter data in module not overwritten			
01 _h	Temperature measuring with two-wire connection (complement on two)	with PT100 -200.0 {0.1 °C} +850.0	A	±1 °C ²⁾
02 _h		with PT1000 -200.0 {0.1 °C} +500.0		±1 °C ²⁾
03 _h		with NI100 -50.0 {0.1 °C} +250.0		±1 °C ²⁾
04 _h		with NI1000 -50.0 {0.1 °C} +250.0		±1 °C ²⁾
05 _h	Resistance measuring with two-wire connection	at 60 Ω 0 {0.01 Ω} +60.00 {1 dec} 32767	A	±0.2 % of limit value ²⁾
05 _h		at 600 Ω 0 {0.01 Ω} +600.00 {1 dec} 32767		±0.2 % of limit value ²⁾
07 _h		at 3000 Ω 0 {0.01 Ω} +3000.00 {1 dec} 32767		±0.2 % of limit value ²⁾
08 _h		at 6000 Ω 0 {0.01 Ω} +6000.00 {1 dec} 32767		±0.2 % of limit value ²⁾
09 _h	Temperature measuring with four-wire connection (complement on two)	with PT100 -200.0 {0.1 °C} +850.0	A	±0.5 °C
0A _h		with PT1000 -200.0 {0.1 °C} +500.0		±0.5 °C
0B _h		with NI100 -50.0 {0.1 °C} +250.0		±0.5 °C
0C _h		with NI1000 -50.0 {0.1 °C} +250.0		±0.5 °C
0D _h	Resistance measuring with two-wire connection	at 60 Ω 0 {0.01 Ω} +60.00	A	±0.1 % of limit value
0E _h		at 600 Ω 0 {0.01 Ω} +600.00		±0.05 % of limit value
0F _h		at 3000 Ω 0 {0.01 Ω} +3000.00		±0.05 % of limit value
10 _h	Temperature measuring with thermocouple and external compensation ³⁾ (complement on two)	with type J -210.0 {0.1 °C} +850.0	A	±1 °C
11 _h		with type K -270.0 {0.1 °C} +1200.0		±1.5 °C
12 _h		with type N -200.0 {0.1 °C} +1300.0		±1.5 °C
13 _h		with type R -50.0 {0.1 °C} +1760.0		±4 °C
14 _h		with type T -270.0 {0.1 °C} +400.0		±1.5 °C
15 _h		with type S -50.0 {0.1 °C} +1760.0		±5 °C
18 _h	Temperature measuring with thermocouple and internal compensation ⁴⁾ (complement on two)	with type J -210.0 {0.1 °C} +850.0	A	±1.5 °C
19 _h		with type K -270.0 {0.1 °C} +1200.0		±2 °C
1A _h		with type N -200.0 {0.1 °C} +1300.0		±2 °C
1B _h		with type R -50.0 {0.1 °C} +1760.0		±5 °C
1C _h		with type T -270.0 {0.1 °C} +400.0		±2 °C
1D _h		with type S -50.0 {0.1 °C} +1760.0		±5 °C
27 _h	Voltage measuring (complement on two)	0 ... 50 mV 0 {0.01 mV} +50.00 {1 dec} 27648 Lower range limit: - Upper range limit: +59.25 mV 32767 dec	A	±0.1 % of limit value
29 _h	Voltage measuring (complement on two)	±4 V -4.00 {0.01 V} +4.00 V -27648 {1 dec} 27648 dec Lower range limit: -4.74 V -32767 dec Upper range limit: +4.74 V 32767 dec	A	±0.05 % of limit value
2A _h	Voltage measuring (complement on two)	±400 mV -400 {1 mV} +400 -27648 {1 dec} 27648 Lower range limit: -474 mV -32767 dec Upper range limit: +474 mV 32767 dec	A	±0.1 % of limit value
3B _h	Voltage measuring (complement on two)	±10 V -10.00 {0.01 V} +10.00 -16384 {1 dec} 16384	B	±0.2 % of limit value

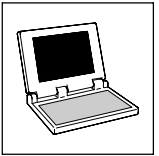


Function library LenzelOSystem.lib

Appendix

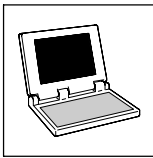
Signal functions for analog inputs EPM-T310

Parameter bytes 2/3/4/5	Signal function		Signal range	Format	Tolerance ¹⁾
			Lower range limit: -11.85 V -20480 dec Upper range limit: +11.85 V 20480 dec		
2C _h	Current measuring (complement on two)	±20 mA	-20.00 {0.01 mA} +20.00 -27648 {1 dec} 27648 Lower range limit: -23.70 mA -32767 dec Upper range limit: +23.70 mA +32767 dec	A	±0.05 % of limit value
2D _h	Current measuring (complement on two)	4 ... 20 mA	4.00 {0.01 mA} 20.00 0 {1 dec} 27648 Lower range limit: 1.185 mA -4864 dec Upper range limit: +22.96 mA +32767 dec	A	±0.05 % of limit value
2E _h	Current measuring	4 ... 20 mA	4.00 {0.01 mA} 20.00 0 {1 dec} 16384 Lower range limit: 1.185 mA -2882 dec Upper range limit: +22.96 mA +20480 dec	B	±0.2 % of limit value
32 _h	Resistance measuring with four-wire connection	at 6000 Ω	0 {0.01 Ω} +6000.00 0 {1 dec} 32767 dec	A	±0.05 % of limit value
33 _h		at 6000 Ω	0 {0.01 Ω} +6000.00 0 {1 dec} 6000 dec		±0.05 % of limit value
35 _h		at 60 Ω	0 {0.01 Ω} +6000.00 0 6000 dec		±0.2 % of limit value ²⁾
36 _h		at 600 Ω	0 {0.01 Ω} +600.00 0 {1 dec} 6000 dec		±0.1 % of limit value ²⁾
37 _h		at 3000 Ω	0 {0.01 Ω} +3000.00 0 {1 dec} 30000 dec		±0.1 % of limit value ²⁾
38 _h		at 6000 Ω	0 {0.01 Ω} +6000.00 0 {1 dec} 6000 dec		±0.1 % of limit value ²⁾



Parameter bytes 2/3/4/5	Signal function		Signal range	Format	Tolerance ¹⁾
3A _h	Current measuring (complement on two)	±20 mA	-20.00 {0.01 mA} +20.00 -16384 {1 dec} 16384 Lower range limit: -23.70 mA -19456 dec Upper range limit: +23.70 mA +19456 dec	A	±0.05 % of limit value
3B _h	Voltage measuring (complement on two)	±10 V	-10.00 {0.01 V} +10.00 -16384 {1 dec} 16384 Lower range limit: -12.5 V -20480 dec Upper range limit: +12.5 V 20480 dec	B	±0.2 % of limit value
3D5 _h	Resistance measuring with four-wire connection	at 60 Ω	0 {0.01 Ω} +60.00 0 {1 dec} 6000 dec	A	±0.1 % of limit value
3E _h		at 600 Ω	0 {0.01 Ω} +600.00 0 {1 dec} 6000 dec		±0.05 % of limit value
3F _h		at 3000 Ω	0 {0.01 Ω} +3000.00 0 {1 dec} 30000 dec		±0.05 % of limit value
57 _h	Voltage measuring	0 ... 50 mV	0 {0.01 mV} +50.00 0 {1 dec} 5000 Upper range limit: +59.25 V 5925 dec	A	±0.1 % of limit value
58 _h	Voltage measuring (complement on two)	±10 V	-10.00 {0.01 V} +10.00 -10000 {1 dec} 10000 Lower range limit: -11.85 V -11850 dec Upper range limit: +11.85 V 11850 dec	A	±0.05 % of limit value
59 _h	Voltage measuring (complement on two)	±4 V	-4.00 {0.01 V} +4.00 V -40000 {1 dec} 40000 Lower range limit: -4.74 V -47400 dec Upper range limit: +4.74 V 47400 dec	A	±0.05 % of limit value
5A _h	Voltage measuring (complement on two)	±400 mV	-400 {1 mV} +400 -40000 {1 dec} 40000 Lower range limit: -474 mV -47400 dec Upper range limit: +474 mV 47400 dec	A	±0.1 % of limit value
5C _h	Current measuring (complement on two)	±20 mA	-20.00 {0.01 mA} +20.00 -20000 {1 dec} 20000 Lower range limit: -23.70 mA -23700 dec Upper range limit: +23.70 mA +23700 dec	A	±0.05 % of limit value
5D _h	Current measuring (complement on two)	4 ... 20 mA	4.00 {0.01 mA} 20.00 0 {1 dec} 16000 Lower range limit: 1.185 mA -2815 dec Upper range limit: +22.96 mA +22960 dec	A	±0.05 % of limit value
FF _h	Analog input deactivated				

- 1) The tolerance values were established at 25 °C ambient temperature and 15 conversions/s. No sensor inaccuracies were taken into consideration.
- 2) Contact and line resistances were not taken into consideration.
- 3) Reference junction compensation to be effected externally.
- 4) Reference junction compensation is effected internally, taking into consideration the terminal temperature. Connect the thermocouple conductors directly to the terminal using a thermocouple extension cable if necessary.



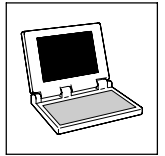
Function library LenzeIOSystem.lib

Appendix

Signal functions for analog outputs EPM-T320

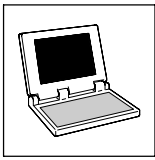
6.2 Signal functions for analog outputs EPM-T320

Parameter bytes 2/3/4/5	Signal function	Signal range	Signal resolution	Tolerance ¹⁾
00 _h	No signal emitted at output			
01 _h	Voltage signal output	±10 V -10.00 {0.01 V} -16384 {1 dec} Lower range limit: -11.85 V -20480 dec Upper range limit: +11.85 V 20480 dec	B	±0.2 %
02 _h	Voltage signal output	+1 ... +5 V 1.0 {0.1 V} 0 {1 dec} Lower range limit: 0 V 20480 dec Upper range limit: +6.0 V 20480 dec	B	±0.05 %
05 _h	Voltage signal output	0 ... +10 V 0 {0.1 V} 0 {1 dec} Lower range limit: - Upper range limit: +12.5 V 20480 dec	B	±0.2 %
09 _h	Voltage signal output (complement on two)	±10 V -10.00 {0.01 V} -27648 {1 dec} Lower range limit: -11.85 V -32767 dec Upper range limit: +11.84 V 32767 dec	A	±0.05 %
0A _h	Voltage signal output (complement on two)	+1 ... +5 V 1.00 {0.01 V} 0 {1 dec} Lower range limit: 0 V -6912 dec Upper range limit: +5.75 V 32767 dec	A	±0.05 %
0D _h	Voltage signal output (complement on two)	0 ... +10 V 0 {0.1 V} 0 {1 dec} Lower range limit: - Upper range limit: +11.5 V 32767 dec	A	±0.2 %
03 _h	Current signal output	±20 mA -20.00 {0.01 mA} -16384 {1 dec} Lower range limit: -23.70 mA -20480 dec Upper range limit: +23.70 mA +20480 dec	B	±0.2 %
04 _h	Current signal output	4 ... 20 mA 4.00 {0.01 mA} 0 {1 dec} Lower range limit: 0 mA -4096 dec Upper range limit: +23.70 mA +20480 dec	B	±0.2 %
06 _h	Current signal output	0 ... 20 mA 0 {0.01 mA} 0 {1 dec} Lower range limit: - Upper range limit: +23.70 mA +20480 dec	B	±0.2 %



Parameter bytes 2/3/4/5	Signal function	Signal range	Signal resolution	Tolerance ¹⁾
0B _h	Current signal output (complement on two)	±20 mA -20.00 {0.01 mA} +20.00 -27648 {1 dec} 27648 Lower range limit: -23.70 mA -32767 dec Upper range limit: +23.70 mA +32767 dec	A	±0.05 %
0C _h	Current signal output (complement on two)	4 ... 20 mA 4.00 {0.01 mA} 20.00 0 {1 dec} 27648 Lower range limit: 0 mA -5530 dec Upper range limit: +22.96 mA +32767 dec	A	±0.05 %
0E _h	Current signal output (complement on two)	0 ... 20 mA 0 {0.01 mA} 20.00 0 {1 dec} 27648 Lower range limit: – Upper range limit: +22.96 mA +32767 dec	A	±0.2 %

¹⁾ The tolerance values were established at 25 °C ambient temperature and 15 conversions/s and are related to the limit value.



Function library LenzeIOSystem.lib

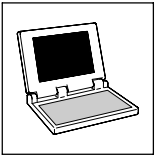
Appendix

Signal functions for analog inputs and outputs EPM-T330

6.3 Signal functions for analog inputs and outputs EPM-T330

Input range (channels 0, 1)

Parameter bytes 2/3/4/5	Signal function	Signal range	Format	Tolerance ¹⁾
00 _h	Parameter data in module not overwritten			
3B _h	Voltage measuring (complement on two)	±10 V -10.00 {0.01 V} -16384 {1 dec} Lower range limit: -12.50 V -20480 _{dec} Upper range limit: +12.50 V +20480 _{dec}	B	±0.2 % of limit value
75 _h	Voltage measuring	0...10 V 0.00 {0.01 V} 0 {1 dec} Lower range limit: - Upper range limit: +12.50 V +20480 _{dec}	B	±0.4 % of limit value
28 _h	Voltage measuring (complement on two)	±10 V -10.00 {0.01 V} -27648 {1 dec} Lower range limit: -11.76 V -32512 _{dec} Upper range limit: +11.76 V +32511 _{dec}	A	±0.2 % of limit value
7A _h	Voltage measuring	1 ... 5V +1.00 {0.01 V} 0 {1 dec} Lower range limit: 0.00 V -6912 _{dec} Upper range limit: +5.704 V +32511 _{dec}	A	±0.6 % of limit value
7D _h	Voltage measuring	0...10 V 0.00 {0.01 V} 0 {1 dec} Lower range limit: - Upper range limit: +11.76 V +32511 _{dec}	A	±0.4 % of limit value
3A _h	Current measuring (complement on two)	±20 mA -20.00 {0.01 mA} -16384 {1 dec} Lower range limit: -25.00 mA -20480 _{dec} Upper range limit: +25.00 mA +20480 _{dec}	B	±0.3 % of limit value
76 _h	Current measuring	0 ... 20 mA 0.00 {0.01 mA} 0 {1 dec} Lower range limit: - Upper range limit: +25.00 mA +20480 _{dec}	B	±0.6 % of limit value
2C _h	Current measuring (complement on two)	±20 mA -20.00 {0.01 mA} -27648 {1 dec} Lower range limit: -23.51 mA -32512 _{dec} Upper range limit: +23.51 mA +32511 _{dec}	A	±0.3 % of limit value
2D _h	Current measuring (complement on two)	4 ... 20 mA +4.00 {0.01 mA} 0 {1 dec} Lower range limit: +1.18 mA -4864 _{dec} Upper range limit: +22.81 mA +32511 _{dec}	A	±0.8 % of limit value

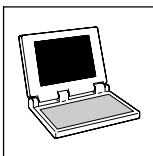


Parameter bytes 2/3/4/5	Signal function	Signal range	Format	Tolerance ¹⁾
7E _h	Current measuring	0 ... 20 mA 0.00 {0.01 mA} +20.00 0 {1 dec} +27648 Lower range limit: - Upper range limit: +23.52 mA +32511 _{dec}	A	±0.6 % of limit value
FF _h	Analog input deactivated			

¹⁾ The tolerance values were established at 25 °C ambient temperature and 15 conversions/s. No sensor inaccuracies were taken into consideration.

Output range (channels 2, 3)

Parameter bytes 2/3/4/5	Signal function	Signal range	Format	Tolerance ¹⁾
00 _h	Parameter data in module not overwritten			
01 _h	Voltage signal output	±10 V -10.00 {0.01 V} +10.00 -16384 {1 dec} +16384 Lower range limit: -12.50 V -20480 _{dec} Upper range limit: +12.50 V +20480 _{dec}	B	±0.2 % of limit value
02 _h	Voltage signal output	1 ... 5V +1.00 {0.01 V} +5.00 0 {1 dec} +16384 Lower range limit: 0.00 V -4096 _{dec} Upper range limit: +6.00 V +20480 _{dec}	B	±0.6 % of limit value
05 _h	Voltage signal output	0...10 V 0.00 {0.01 V} +10.00 0 {1 dec} +16384 Lower range limit: - Upper range limit: +12.50 V +20480 _{dec}	B	±0.4 % of limit value
09 _h	Voltage signal output (complement on two)	±10 V -10.00 {0.01 V} +10.00 -27648 {1 dec} +27648 Lower range limit: -11.76 V -32512 _{dec} Upper range limit: +11.76 V +32511 _{dec}	A	±0.2 % of limit value
0A _h	Voltage signal output (complement on two)	1 ... 5V +1.00 {0.01 V} +5.00 0 {1 dec} +27648 Lower range limit: 0.00 V -6912 _{dec} Upper range limit: +5.704 V +32511 _{dec}	A	±0.6 % of limit value
0D _h	Voltage signal output	0...10 V 0.00 {0.01 V} +10.00 0 {1 dec} +27648 Lower range limit: - Upper range limit: +11.76 V +32511 _{dec}	A	±0.4 % of limit value
03 _h	Current signal output	±20 mA -20.00 {0.01 mA} +20.00 -16384 {1 dec} +16384 Lower range limit: -25.00 mA -20480 _{dec} Upper range limit: +25.00 mA +20480 _{dec}	B	±0.3 % of limit value



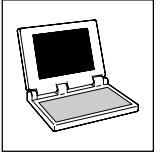
Function library LenzelOSystem.lib

Appendix

Signal functions for analog inputs and outputs EPM-T330

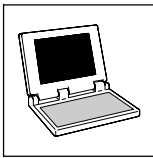
Parameter bytes 2/3/4/5	Signal function	Signal range	Format	Tolerance ¹⁾
04 _h	Current signal output	4 ... 20 mA +4.00 {0.01 mA} +20.00 0 {1 dec} +16384 Lower range limit: +0.00 mA -4096 _{dec} Upper range limit: +24.00 mA +20480 _{dec}	B	±0.8 % of limit value
06 _h	Current signal output	0 ... 20 mA 0.00 {0.01 mA} +20.00 0 {1 dec} +16384 Lower range limit: - Upper range limit: +25.00 mA +20480 _{dec}	B	±0.6 % of limit value
0B _h	Current signal output (complement on two)	±20 mA -20.00 {0.01 mA} +20.00 -27648 {1 dec} +27648 Lower range limit: -23.52 mA -32512 _{dec} Upper range limit: +23.52 mA +32511 _{dec}	A	±0.3 % of limit value
0C _h	Current signal output (complement on two)	4 ... 20 mA +4.00 {0.01 mA} +20.00 0 {1 dec} +27648 Lower range limit: +0.00 mA -6912 _{dec} Upper range limit: +22.81 mA +32511 _{dec}	A	±0.8 % of limit value
0E _h	Current signal output	0 ... 20 mA 0.00 {0.01 mA} +20.00 0 {1 dec} +27648 Lower range limit: - Upper range limit: +23.52 mA +32511 _{dec}	A	±0.6 % of limit value
FF _h	Analog output deactivated			

¹⁾ The tolerance values were established at 25 °C ambient temperature and 15 conversions/s. No sensor inaccuracies were taken into consideration.



6.4 Measured value scaling

Signal range	Signal [U] / [I]	15-bit signal resolution			12-bit signal resolution		
		Decimal value [dec]	Hexadecimal value [hex]	Formulae for calculation	Decimal value [dec]	Hexadecimal value [hex]	Formulae for calculation
±10 V	-10 V	-27648	9400	$\text{dec} = 27648 \cdot \frac{U}{10}$ $U = \text{dec} \cdot \frac{10}{27648}$	-16384	C000	$\text{dec} = 16384 \cdot \frac{U}{10}$ $U = \text{dec} \cdot \frac{10}{16384}$
	-5 V	-13824	CA00		-8192	E000	
	0 V	0	0000		0	0000	
	+5 V	+13824	3600		+8192	2000	
	+10 V	+27648	6C00		+16384	4000	
0 ... 10 V	0 V	0	0000	$\text{dec} = 16384 \cdot \frac{U}{10}$ $U = \text{dec} \cdot \frac{10}{16384}$	0	0000	$\text{dec} = 16384 \cdot \frac{U}{10}$ $U = \text{dec} \cdot \frac{10}{16384}$
	+5 V	+8192	2000		+8192	2000	
	+10 V	+16384	4000		+16384	4000	
1 ... 5V	+1 V	—	—		0	0000	$\text{dec} = 27648 \cdot \frac{U - 1}{4}$ $U = \text{dec} \cdot \frac{4}{16384} + 1$
	+3 V				+8192	2000	
	+5 V				+16384	4000	
±4 V	-4 V	-27648	9400	$\text{dec} = 27648 \cdot \frac{U}{4}$ $U = \text{dec} \cdot \frac{4}{27648}$	—	—	
	0 V	0	0000				
	+4 V	+27648	6C00				
±400 mV	-400 mV	-27648	9400	$\text{dec} = 27648 \cdot \frac{U}{400}$ $U = \text{dec} \cdot \frac{400}{27648}$	—	—	
	0 V	0	0000				
	+400 mV	+27648	6C00				
4 ... 20 mA	+4 mA	0	0000	$\text{dec} = 27648 \cdot \frac{I - 4}{16}$ $U = \text{dec} \cdot \frac{16}{27648} + 1$	0	0000	$\text{dec} = 16384 \cdot \frac{I - 4}{16}$ $U = \text{dec} \cdot \frac{16}{16384} + 1$
	+12 mA	+13824	3600		+8192	2000	
	+20 mA	+27648	6C00		+16384	4000	
±20 mA	-20 mA	-27648	9400	$\text{dec} = 27648 \cdot \frac{I}{20}$ $U = \text{dec} \cdot \frac{20}{27648}$	-16384	C000	$\text{dec} = 16384 \cdot \frac{I}{20}$ $U = \text{dec} \cdot \frac{20}{16384}$
	-10 mA	-13824	CA00		-8192	E000	
	0 mA	0	0000		0	0000	
	+10 mA	+13824	3600		+8192	2000	
	+20 mA	+27648	6C00		+16384	4000	



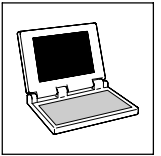
Function library LenzeIOSystem.lib

Appendix

Selecting the counter modes EPM-T410

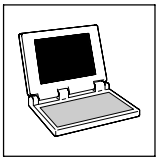
6.5 Selecting the counter modes EPM-T410

Mode		Function	IN1	IN2	IN3	IN4	IN5	IN6	OUT0	OUT1	Auto Reload	Compare Load
[hex]	[dec]											
2 counters			0			1						
00 _h	0	32 bit counter	RES	CLK	DIR	RES	CLK	DIR	•	•	–	–
01 _h	1	Encoder 1 edge	RES	A	B	RES	A	B	•	•	–	–
03 _h	3	Encoder 2 edges	RES	A	B	RES	A	B	•	•	–	–
05 _h	5	Encoder 4 edges	RES	A	B	RES	A	B	•	•	–	–
4 counters				0.1	0.2		1.1	1.2				
08 _h	8	2 × 16 bit counters (counting direction upwards/upwards)	–	CLK	CLK	–	CLK	CLK	–	–	–	–
09 _h	9	2 × 16 bit counters (counting direction downwards/upwards)	–	CLK	CLK	–	CLK	CLK	–	–	–	–
0A _h	10	2 × 16 bit counters (counting direction downwards/upwards)	–	CLK	CLK	–	CLK	CLK	–	–	–	–
0B _h	11	2 × 16 bit counters (counting direction downwards/upwards)	–	CLK	CLK	–	CLK	CLK	–	–	–	–
2 counters			0			1						
0C _h	12	2 × 32 bit counters (counting direction upwards)	RES	CLK	GATE	RES	CLK	GATE	•	•	–	✓
0D _h	13	2 × 32 bit counters (counting direction downwards)	RES	CLK	GATE	RES	CLK	GATE	•	•	–	✓
0E _h	14	2 × 32 bit counters (counting direction upwards)	RES	CLK	GATE	RES	CLK	GATE	•	•	✓	✓
0F _h	15	2 × 32 bit counters (counting direction downwards)	RES	CLK	GATE	RES	CLK	GATE	•	•	✓	✓
1 counter			0/1									
10 _h	16	Frequency measuring	RES	CLK	START	STOP	–	–	•	•	–	✓
11 _h	17	Measuring the period	RES	CLK	START	STOP	–	–	•	•	–	✓
12 _h	18	Frequency measuring (output counter runs/stops)	RES	CLK	START	STOP	–	–	•	•	–	✓
13 _h	19	Measuring the period (output counter runs/stops)	RES	CLK	START	STOP	–	–	•	•	–	✓
2 counters			0			1						
06 _h	6	Measuring the pulse width (f _{ref} 50 kHz, counting direction selectable)	RES	PULSE	DIR	RES	PULSE	DIR	–	–	–	–
14 _h	20	Measuring the pulse width (f _{ref} programmable, counting direction selectable)	RES	PULSE	DIR	RES	PULSE	DIR	–	–	–	–
15 _h	21	Measuring the pulse width (f _{ref} programmable, counting direction upwards)	RES	PULSE	DIR	RES	PULSE	GATE	–	–	–	–
16 _h	22	Measuring the pulse width (f _{ref} programmable, counting direction downwards)	RES	PULSE	DIR	RES	PULSE	GATE	–	–	–	–
2 counters			0			1						
17 _h	23	2 × 32 bit counters (counting direction upwards, set function)	RES	CLK	GATE	RES	CLK	GATE	–	–	–	✓
18 _h	24	2 × 32 bit counters (counting direction upwards, set function)	RES	CLK	GATE	RES	CLK	GATE	–	–	–	✓
19 _h	25	2 × 32 bit counters (counting direction upwards, reset function)	RES	CLK	GATE	RES	CLK	GATE	–	–	–	✓
1A _h	26	2 × 32 bit counters (counting direction upwards, reset function)	RES	CLK	GATE	RES	CLK	GATE	–	–	–	✓
2 counters			0			1						
1B _h	27	32 bit counter	G/RES.]	CLK	DIR	G/RES.]	CLK	DIR	•	•	–	–
1C _h	28	Encoder 1 edge	G/RES.]	A	B	G/RES.]	A	B	•	•	–	–



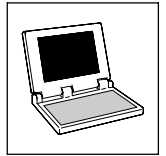
Mode		Function	IN1	IN2	IN3	IN4	IN5	IN6	OUT0	OUT1	Auto Reload	Compare Load
[hex]	[dec]											
1D _h	29	Encoder 2 edges	G/RES _↓	A	B	G/RES _↓	A	B	•	•	–	–
1E _h	30	Encoder 4 edges	G/RES _↓	A	B	G/RES _↓	A	B	•	•	–	–
		2 counters	0		1							
1F _h	31	2 × 32 bit counters (counting direction downwards)	RES _↓	CLK	GATE	RES _↓	CLK	GATE	•	•	–	✓
20 _h	32	2 × 32 bit counters (counting direction downwards)	RES _↓	CLK	GATE	RES _↓	CLK	GATE	•	•	–	✓
21 _h	33	2 × 32 bit counters (counting direction downwards)	RES _↓	CLK	GATE	RES _↓	CLK	GATE	•	•	✓	✓
22 _h	34	2 × 32 bit counters (counting direction downwards)	RES _↓	CLK	GATE	RES _↓	CLK	GATE	•	•	✓	✓

- Digital output may signal an event.
- ✓ Function available.
- No function / function not available.
- A Encoder signal A
- Auto Reload Auto Reload causes the counter to accept a preset value as soon as the counter reading matches the Compare register content.
- B Encoder signal B
- Compare Load You may use Compare Load to specify a counter limit value to trigger an output or to restart the counters via Auto Reload when it is reached.
- CLK Clock signal of a linked encoder
- DIR Indicates counting direction depending on signal level.
LOW: Upcounter
HIGH: Downcounter
- GATE Gate signal level-triggered
- G/RES_↓ Gate signal level-triggered and reset signal edge-triggered
HIGH: Pulses are measured
LOW-HIGH edge: Resets one or both counter(s)
- PULSE The pulse width of the supplied signal is measured with an internal time base
- RES Reset signal level-triggered
HIGH: Resets one or both counter(s)
- RES_↓ Reset signal edge-triggered
LOW-HIGH edge: Resets one or both counter(s)
- START Start signal edge-triggered
- STOP Stop signal edge-triggered



Function library LenzeIOSystem.lib

Appendix



7 Index

A

Analog inputs and outputs EPM-T330, 6-6

Analog inputs EPM-T310, 6-1

Analog outputs EPM-T320, 6-4

B

Bit conversion, 5-20 , 5-21

C

C_wLenzeIOSystemVersion, 1-1

Conversion function, 5-18 , 5-19

Coordinate data, 5-2 , 5-5 , 5-8

Counter function, 4-14 , 4-16 , 5-11 , 5-13 , 5-14 , 5-15 , 5-16 , 5-17

Counter modes, 6-10

I

Initialising, 3-5

Initialising the CAN driver, CAN driver, Initialising, 3-1

L

L_IOAInModule, 5-22

L_IOAOutModule, 5-23

L_IOConvByteArrayToByte, 5-19

L_IOConvByteToByteArray, 5-18

L_IOCOUNTERDataFromIO, 5-13

L_IOCOUNTERDataToIO, 5-11

L_IOCOUNTERDIModuleDataFromIO, 5-15

L_IOCOUNTERDIModuleDataToIO, 5-14

L_IOData15, 5-2

L_IOData610, 5-5

L_IODataCompactModule, 5-8

L_IODInModule, 5-20

L_IODOutModule, 5-21

L_IOParAIAOModule, 4-9

L_IOParAlnModule, 4-4

L_IOParAOutModule, 4-7

L_IOParComGuarding, 4-2

L_IOParCompactModule, 4-26

L_IOParCounterDIModule, 4-16

L_IOParCounterModule, 4-14

L_IOParPDO15, 4-18

L_IOParPDO610, 4-22

L_IOParSSIModule, 4-12

L_IOSSIDataFromIO, 5-17

L_IOSSIDataToIO, 5-16

M

Minimum configuration, 2-2

Monitoring function, 4-2

P

Parameter data blocks, 2-1

Parameterisation function, 4-4 , 4-7 , 4-9 , 4-12 , 4-18 , 4-22 , 4-26

Process data blocks, 2-1

S

Safety information, layout

more notes, 1-2

warning of material damage, 1-2

Scaling, 6-9

Signal conversion, 5-22 , 5-23

Signal functions, 6-1 , 6-4 , 6-6

T

Term definitions, 1-2

V

Version identifiers of the function library, 1-3