

# ACCU ELECTRIC MOTORS INC

USA: (888) 932-9183

CANADA: (905) 829-2505

- ✓ Over 100 years cumulative experience
- ✓ 24 hour rush turnaround / technical support service
- ✓ Established in 1993



The leading independent repairer of servo motors and drives in North America.

Visit us on the web:

[www.servo-repair.com](http://www.servo-repair.com)

[www.servorepair.ca](http://www.servorepair.ca)

[www.ferrocontrol.com](http://www.ferrocontrol.com)

[www.sandvikrepair.com](http://www.sandvikrepair.com)

[www.accuelectric.com](http://www.accuelectric.com)

**Scroll down to view your document!**

For 24/7 repair services :

USA: 1 (888) 932 - 9183

Canada: 1 (905) 829 -2505

Emergency After hours: 1 (416) 624 0386

Servicing USA and Canada

# Lenze

*Manual*



***Global Drive***

*System bus (CAN)*

*for Lenze PLC devices*

This documentation is valid for the following Lenze PLC devices:

Automation system	Type designation	As of hardware version	As of software version
9300 Servo PLC	EVS93XX-xl	2K	2.0
9300 Servo PLC	EVS93XX-xT	2K	2.0
Drive PLC	EPL10200	Px	2.0
ECSxA	ECSxAxxx	1A	6.0

**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.

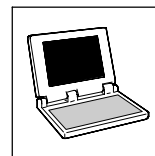
© 2006 Lenze Drive Systems GmbH

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

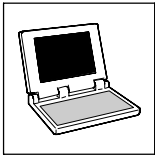
All information given in these Operating Instructions has been selected carefully and comply with the hardware and software described. Nevertheless, deviations cannot be ruled out. We do not take any responsibility or liability for damages which might possibly occur. Required corrections will be made in the following editions.

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

Version 2.0 07/2006 - TD31



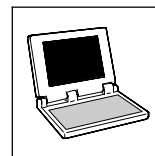
<b>1 Preface and general information</b>	<b>1-1</b>
1.1 About this Manual	1-1
1.1.1 Conventions used in this Manual	1-2
1.1.2 Structure of the description	1-3
1.1.3 Pictographs used in this Manual	1-4
1.1.4 Terminology used	1-4
<b>2 General information on the system bus (CAN)</b>	<b>2-1</b>
2.1 Introduction	2-1
2.2 Interfaces of the Lenze PLCs for system bus connection	2-2
2.3 Identification of the nodes	2-3
2.4 Structure of the CAN telegram	2-3
2.4.1 Identifier	2-3
2.4.2 User data	2-5
2.5 Network management (NMT)	2-6
2.6 Transmission of process data	2-7
2.6.1 Process data channels	2-7
2.6.2 Sync telegram for cyclic process data	2-9
2.6.3 Process data telegram	2-10
2.7 Transmitting parameter data	2-11
2.7.1 Parameter data telegram	2-11
2.7.2 Writing parameters (example)	2-15
2.7.3 Reading a parameter (example)	2-17
2.8 Free CAN objects	2-19
2.9 Application recommendations for the different CAN objects	2-20
2.10 Monitoring mechanisms	2-21
2.10.1 "Heartbeat"	2-21
2.10.2 "Node Guarding"	2-22
<b>3 Configuration (system bus - CAN interface)</b>	<b>3-1</b>
3.1 CAN baud rate	3-1
3.2 CAN boot-up	3-2
3.3 Node address (node ID)	3-3
3.4 Identifiers of the process data objects	3-4
3.4.1 Allocation of individual identifiers	3-4
3.4.2 Display of the identifier set	3-5
3.5 Cycle time (CAN2_OUT/CAN3_OUT)	3-6
3.6 Delay time (CAN2_OUT/CAN3_OUT)	3-6
3.7 Synchronisation	3-7
3.7.1 CAN sync response	3-7
3.7.2 CAN sync identifiers	3-7
3.7.3 CAN sync Tx transmission cycle	3-7
3.8 Reset node	3-8
3.9 System bus management	3-8
3.10 Mapping indexes to codes	3-8
3.10.1 Functional principle considering as example	3-9



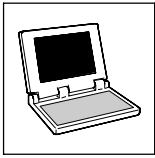
# System bus (CAN) for Lenze PLC devices

## Contents

3.11	Remote parameterisation (gateway function)	3-10
3.12	Monitoring processes	3-11
3.12.1	Time monitoring for CAN1_IN ... CAN3_IN	3-11
3.12.2	Bus-off	3-11
3.12.3	Time-out when remote parameterisation is activated	3-12
3.12.4	Response in the case of system bus fault messages	3-12
3.13	Diagnostics	3-13
3.13.1	Operating status of the CAN interface	3-13
3.13.2	Telegram counter	3-14
3.13.3	Bus load by the PLC	3-15
<b>4</b>	<b>Configuration (AIF interface)</b>	<b>4-1</b>
4.1	CAN baud rate	4-1
4.2	CAN boot-up	4-2
4.3	Node address (node ID)	4-3
4.4	Identifiers of the process data objects	4-4
4.4.1	Allocation of individual identifiers	4-4
4.4.2	Display of the identifier set	4-5
4.5	Cycle time (XCAN1_OUT ... XCAN3_OUT)	4-6
4.6	Synchronisation	4-7
4.6.1	XCAN sync response	4-7
4.6.2	XCAN sync identifier	4-7
4.6.3	XCAN sync Tx transmission cycle	4-7
4.7	Reset node	4-8
4.8	Monitoring processes	4-8
4.8.1	Time monitoring for XCAN1_IN ... XCAN3_IN	4-8
4.8.2	Bus off	4-9
4.8.3	Response for system bus fault messages	4-9
4.9	Diagnostics	4-10
4.9.1	Automation interface (AIF) operating status	4-10
<b>5</b>	<b>Configuration (FIF interface)</b>	<b>5-1</b>
5.1	CAN baud rate	5-1
5.2	CAN boot-up	5-2
5.3	Node address (node ID)	5-3
5.4	Identifiers of the process data objects	5-4
5.4.1	Allocation of individual identifiers	5-4
5.4.2	Display of the identifiers set	5-5
5.5	Cycle time (FIF_CAN2_OUT/FIF_CAN3_OUT)	5-6
5.6	Delay time (FIF_CAN2_OUT/FIF_CAN3_OUT)	5-6
5.7	Synchronisation	5-7
5.7.1	FIF-CAN sync response	5-7
5.7.2	FIF-CAN sync identifier	5-7
5.7.3	FIF-CAN sync Tx transmission cycle	5-7
5.8	Reset node	5-8



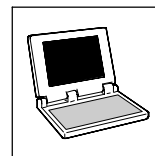
5.9	System bus management	5-8
5.10	Monitoring processes	5-9
5.10.1	Time monitoring for FIF-CAN1_IN ... FIF-CAN3_IN	5-9
5.10.2	Bus-off	5-9
5.10.3	Response in the case of system bus fault messages	5-10
5.11	Diagnostics	5-11
5.11.1	Function interface (FIF) operating status	5-11
5.11.2	Telegram counter	5-12
5.11.3	Bus load by FIF-CAN	5-13
<b>6</b>	<b>Configuration (CAN-AUX system bus interface)</b>	<b>6-1</b>
6.1	CAN baud rate	6-1
6.2	CAN boot-up	6-2
6.3	Node address (Node ID)	6-3
6.4	Identifiers of the process data objects	6-4
6.4.1	Allocation of individual identifiers	6-4
6.4.2	Display of the identifiers set	6-5
6.5	Cycle time (CANaux2_OUT/CANaux3_OUT)	6-6
6.6	Delay time (CANaux2_OUT/CANaux3_OUT)	6-6
6.7	Synchronisation	6-7
6.7.1	CANaux sync response	6-7
6.7.2	CANaux sync identifiers	6-7
6.7.3	CANaux sync Tx transmission cycle	6-7
6.8	Reset node	6-8
6.9	System bus management	6-8
6.10	Monitoring processes	6-9
6.10.1	Time monitoring for CANaux1_IN ... CANaux3_IN	6-9
6.10.2	Bus-off	6-9
6.10.3	Response in the case of system bus fault messages	6-10
6.11	Diagnostics	6-11
6.11.1	Operating status of the CAN-AUX interface	6-11
6.11.2	Telegram counter	6-12
6.11.3	Bus load by CAN-AUX	6-13
<b>7</b>	<b>CAN system blocks</b>	<b>7-1</b>
7.1	CAN1_IO (node number: 31) - 9300 Servo PLC	7-1
7.1.1	Inputs_CAN1	7-2
7.1.2	Outputs_CAN1	7-2
7.1.3	Process data telegram	7-3
7.1.4	Assignment of the user data to variables	7-3
7.1.5	Transferring status and control information of the device control	7-5
7.2	CAN1_IO (node number: 31) - Drive PLC	7-6
7.2.1	Inputs_CAN1	7-7
7.2.2	Outputs_CAN1	7-7
7.2.3	Process data telegram	7-8
7.2.4	Assignment of the user data to variables	7-8



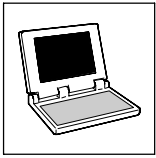
# System bus (CAN) for Lenze PLC devices

## Contents

7.3	CAN1_IO (node number: 31) - ECSxA .....	7-10
7.3.1	Inputs_CAN1 .....	7-11
7.3.2	Outputs_CAN1 .....	7-11
7.3.3	Process data telegram .....	7-12
7.3.4	Assignment of the user data to variables .....	7-12
7.4	CAN2_IO (node number: 32) .....	7-14
7.4.1	Inputs_CAN2 .....	7-15
7.4.2	Outputs_CAN2 .....	7-15
7.4.3	Process data telegram .....	7-15
7.4.4	Assignment of the user data to variables .....	7-16
7.5	CAN3_IO (node number: 33) .....	7-17
7.5.1	Inputs_CAN3 .....	7-18
7.5.2	Outputs_CAN3 .....	7-18
7.5.3	Process data telegram .....	7-18
7.5.4	Assignment of the user data to variables .....	7-19
7.6	CAN_Management (node number: 101) .....	7-20
7.6.1	Inputs_CAN_Management .....	7-20
7.6.2	Outputs_CAN_Management .....	7-21
7.6.3	Activating a reset node .....	7-21
7.6.4	Defining the instant of transmission for CAN2_OUT/CAN3_OUT .....	7-21
7.6.5	Status messages .....	7-22
7.7	CAN_Synchronization (node number: 102) .....	7-23
<b>8</b>	<b>FIF-CAN system blocks (only Drive PLC) .....</b>	<b>8-1</b>
8.1	FIF_CAN1_IO (node number: 34) .....	8-1
8.1.1	FIF_Inputs_CAN1 .....	8-2
8.1.2	FIF_Outputs_CAN1 .....	8-2
8.1.3	Process data telegram .....	8-3
8.1.4	Assignment of the user data to variables .....	8-3
8.2	FIF_CAN2_IO (node number: 35) .....	8-1
8.2.1	FIF_Inputs_CAN2 .....	8-2
8.2.2	FIF_Outputs_CAN2 .....	8-2
8.2.3	Process data telegram .....	8-2
8.2.4	Assignment of the user data to variables .....	8-3
8.3	FIF_CAN3_IO (node number: 36) .....	8-1
8.3.1	FIF_Inputs_CAN3 .....	8-2
8.3.2	FIF_Outputs_CAN3 .....	8-2
8.3.3	Process data telegram .....	8-2
8.3.4	Assignment of the user data to variables .....	8-3
8.4	FIF_CAN_Management (node number: 111) .....	8-4
8.4.1	FIF_Inputs_CAN_Management .....	8-4
8.4.2	FIF_Outputs_CAN_Management .....	8-5
8.4.3	Activating a reset node .....	8-5
8.4.4	Defining the instant of transmission for FIF-CAN2_OUT/FIF-CAN3_OUT .....	8-5
8.4.5	Status messages .....	8-6



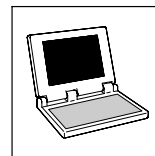
<b>9 CAN-AUX system blocks (only ECSxA)</b> .....	<b>9-1</b>
9.1 CANaux1_IO (node number: 34) .....	9-1
9.1.1 Inputs_CANaux1 .....	9-2
9.1.2 Outputs_CANaux1 .....	9-2
9.1.3 Process data telegram .....	9-3
9.1.4 Assignment of the user data to variables .....	9-3
9.2 CANaux2_IO (node number: 35) .....	9-1
9.2.1 Inputs_CANaux2 .....	9-2
9.2.2 Outputs_CANaux2 .....	9-2
9.2.3 Process data telegram .....	9-2
9.2.4 Assignment of the user data to variables .....	9-3
9.3 CANaux3_IO (node number: 36) .....	9-1
9.3.1 Inputs_CANaux3 .....	9-2
9.3.2 Outputs_CANaux3 .....	9-2
9.3.3 Process data telegram .....	9-2
9.3.4 Assignment of the user data to variables .....	9-3
9.4 CANaux_Management (node number: 111) .....	9-4
9.4.1 Inputs_CANaux_Management .....	9-4
9.4.2 Outputs_CANaux_Management .....	9-5
9.4.3 Activating a reset node .....	9-5
9.4.4 Defining the instant of transmission for CANaux2_OUT/CANaux3_OUT .....	9-5
9.4.5 Status messages .....	9-6
<b>10 LenzeCanDrv.lib function library</b> .....	<b>10-1</b>
10.1 Overview .....	10-1
10.2 Version identifiers of the function library .....	10-1
10.3 L_CanInit - initialising the CAN driver .....	10-2
10.4 L_CanClose - deactivating the CAN driver .....	10-5
10.5 L_CanGetStatus - querying the driver status .....	10-6
10.6 L_CanGetRelocCobId - querying the COB-ID range .....	10-7
10.7 L_CanPdoTransmit - transmitting a CAN object .....	10-8
10.8 L_CanPdoReceive - receiving a CAN object .....	10-12
<b>11 LenzeCanDSxDrv.libfunction library</b> .....	<b>11-1</b>
11.1 Overview .....	11-1
11.2 Version identifiers of the function library .....	11-2
11.3 L_CanDSxInitIndexCode - Configuration of index mapping .....	11-3
11.4 L_CanDSxOpen - initialising the CanDSx driver .....	11-5
11.5 L_CanDSxClose - deactivating the index mapping .....	11-6
11.6 L_CanDSxOpenHeartBeat - initialising a "Heartbeat" .....	11-7
11.7 L_CanDSxHeartBeat - carrying out a "Heartbeat" .....	11-8
11.8 L_CanDSxCloseHeartBeat - deactivating the "Heartbeat" .....	11-10
11.9 L_CanDSxOpenNodeGuarding - initialising the "Node Guarding" .....	11-11
11.10 L_CanDSxNodeGuarding - carrying out a "Node guarding" .....	11-12
11.11 L_CanDSxCloseNodeGuarding - deactivating the "Node Guarding" .....	11-15



# ***System bus (CAN) for Lenze PLC devices***

## ***Contents***

<b>12 Index</b> .....	<b>12-1</b>
-----------------------	-------------

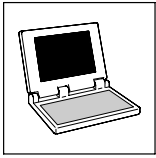


## 1 Preface and general information

### 1.1 About this Manual

This Manual contains information on the system bus interfaces of the Lenze PLC devices  
**9300 Servo PLC, Drive PLC and ECSxA.**

Chapter	Content	
2	<b>General information on the system bus (CAN)</b>	📖 2-1
	<b>Configuration</b>	
3	<b>Integrated "CAN" system bus interface</b>	📖 3-1
4	<b>Optional system bus interface via automation interface (AIF)</b> and corresponding fieldbus module (e. g. 2175)	📖 4-1
5	<b>Optional system bus interface via function interface (FIF)</b> and corresponding function module (e. g. CAN-I/O system bus) • Only for Drive PLC!	📖 5-1
6	<b>Integrated "CAN-AUX" system bus interface</b> • Only for ECSxA!	📖 6-1
7	<b>CAN system blocks</b>	📖 7-1
	<b>CAN objects</b> (CAN1_IO ... CAN3_IO)	
	<b>CAN synchronisation</b> (CAN_Synchronization) <b>CAN management</b> (CAN_Management)	
8	<b>FIF-CAN system blocks (only Drive PLC)</b>	📖 8-1
	<b>CAN objects</b> (FIF_CAN1_IO ... FIF_CAN3_IO)	
	<b>CAN management</b> (FIF_CAN_Management)	
9	<b>CAN-AUX system blocks (only ECSxA)</b>	📖 9-1
	<b>CAN objects</b> (CANaux1_IO ... CANaux3_IO)	
	<b>CAN management</b> (CANaux_Management)	
10	<b>LenzeCanDrv.lib function library</b> • Free CAN objects	📖 10-1
11	<b>LenzeCanDSxDrv.libfunction library</b> • Mapping indexes to codes • "Heartbeat" and "Node Guarding" monitoring mechanisms	📖 11-1



# System bus (CAN) for Lenze PLC devices

## Preface and general information

### About this Manual

#### 1.1.1 Conventions used in this Manual

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

##### Variable identifier

... are presented in italics in the explanatory text:

- "Via *wDrvNr...*"



##### 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

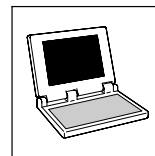
... can be identified by their designation. They always start with an "L\_":

- "The function **L\_CanInit** ..."
- "The **L\_CanPdoTransmit** FB..."

##### Program listings

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

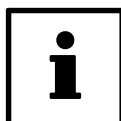
- "**IF** (ReturnValue < 0) **THEN**..."



## 1.1.2 Structure of the description

The descriptions of the individual functions/function blocks as well as of system blocks contained in this Manual have the same structure:

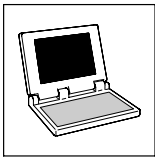
<p>① Headline with SB identifier</p> <p>② SB function and node number</p> <p>③ Short description of the SB and its most important features</p> <p>④ System block chart including all corresponding variables</p> <ul style="list-style-type: none"> <li>• Input variables</li> <li>• Output variables</li> </ul> <p>⑤ Table giving information about input and output variables:</p> <ul style="list-style-type: none"> <li>• Identifier</li> <li>• Data type</li> <li>• Signal type</li> <li>• Address</li> <li>• Display code</li> <li>• Display format</li> <li>• Information</li> </ul> <p>⑥ Detailed functional description of the SB</p>	①	Headline with SB identifier
	②	SB function and node number
	③	Short description of the SB and its most important features
	④	System block chart including all corresponding variables
	⑤	Table giving information about input and output variables:
	⑥	Detailed functional description of the SB



### Information on return values for a function

If it was not possible to carry out a function faultlessly, a **negative return value** is sent back, representing an error number.

- Each error number is assigned to a corresponding error cause in the **Meaning** column.
- If different error numbers (-1, -2, ...) may apply, a specific digit (1, 2, ...) in the **Priority** column additionally is assigned to the error number.
  - The *smaller* this digit, the *higher* is the priority of the associated error number.
  - If several error causes are available at the same time when a function is carried out, always the error number with the **highest priority** is returned by the function.





# System bus (CAN) for Lenze PLC devices

## Preface and general information

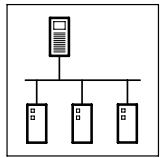
### About this Manual

#### 1.1.3 Pictographs used in this Manual

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

#### 1.1.4 Terminology used

Term	In the following text used for
AIF	Automation interface
DDS	Drive PLC Developer Studio
FB	Function block
FIF	Function interface
GDC	Global Drive Control (parameterisation program from Lenze)
Parameter codes	Codes for setting the function of a function block
PLC	<ul style="list-style-type: none"> <li>• 9300 Servo PLC</li> <li>• Drive PLC</li> <li>• ECSxA</li> </ul>
SB	System block
System bus	System bus (CAN): Lenze standard bus system similar to <i>CANopen</i>



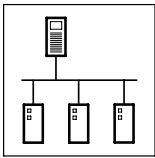
## 2 General information on the system bus (CAN)

### 2.1 Introduction

All Lenze drive and automation systems are provided with an integrated system bus interface for the networking of control components on a field level.

Via the system bus interface, among other things process data and parameter data can be exchanged between the nodes. Furthermore the interface enables the connection of further modules, like for example decentralised terminals, operator and input devices, as well as external controls and host systems.

The system bus interface transfers CAN objects following the CANopen communication profile (CiA DS301, version 4.01), which was developed under the umbrella association of the **CiA (CAN in Automation)**, complying with the **CAL (CAN Application Layer)**.



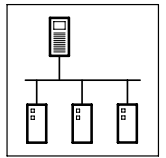
# System bus (CAN) for Lenze PLC devices

## General information

### 2.2 Interfaces of the Lenze PLCs for system bus connection

The following table provides an overview of the Lenze PLC system bus interfaces **9300 Servo PLC**, **Drive PLC** and **ECSxA**:

	CAN objects available		Information
<b>System bus interface CAN</b>	PD0s	CAN1_IN/CAN1_OUT CAN2_IN/CAN2_OUT CAN3_IN/CAN3_OUT	See chapter 3, "Configuration (CAN system bus interface)". ☰ 3-1
	SD0s	SD01 (parameter data channel 1) SD02 (parameter data channel 2) L_ParRead/L_ParWrite functionality	Reading/writing of codes. See documentation on the <b>LenzeDrive.lib</b> function library
	Sync telegram		See chapter 3, "Configuration (CAN system bus interface)". ☰ 3-1
	Synchronisation of the internal time basis by receiving sync telegrams		☰ 3-1
	Free CAN objects		
<b>Automation interface (AIF)</b> with corresponding fieldbus module (e.g. 2175)	PD0s	XCAN1_IN/XCAN1_OUT XCAN2_IN/XCAN2_OUT XCAN3_IN/XCAN3_OUT	See chapter 4, "Configuration (AIF interface)". ☰ 4-1
	SD0s	XSD01 (parameter data channel 1) XSD02 (parameter data channel 2)	
	XSync telegram		
<b>Function interface (FIF)</b> with corresponding function module (e.g. CAN-I/O system bus)	PD0s	FIF-CAN1_IN/FIF-CAN1_OUT FIF-CAN2_IN/FIF-CAN2_OUT FIF-CAN3_IN/FIF-CAN3_OUT	<b>For Drive PLC only!</b> See chapter 5, "Configuration (FIF interface)". ☰ 5-1
	SD0s	FIF-SD01 (parameter data channel 1) FIF-SD02 (parameter data channel 2) L_ParRead/L_ParWrite functionality	Reading/writing of codes. See documentation on the <b>LenzeDrive.lib</b> function library
	Sync telegram		See chapter 5, "Configuration (FIF interface)". ☰ 5-1
<b>System bus interface CAN-AUX</b>	PD0s	CANaux1_IN/CANaux1_OUT CANaux2_IN/CANaux2_OUT CANaux3_IN/CANaux3_OUT	<b>For ECSxA only!</b> See chapter 6, "Configuration (CAN-AUX system bus interface)". ☰ 6-1
	SD0s	CAN-AUX-SD0 (parameter data channel) L_ParRead/L_ParWrite functionality	Reading/writing of codes. See documentation on the function library <b>LenzeDrive.lib</b>
	Sync telegram		See chapter 6, "Configuration (CAN-AUX system bus interface)". ☰ 6-1



## 2.3 Identification of the nodes

Assign a node address - also called *Node ID* - in the range of 1 to 63 to each node within the system bus network as a definite identification.

- The same node address may not be assigned more than once within the network.

## 2.4 Structure of the CAN telegram

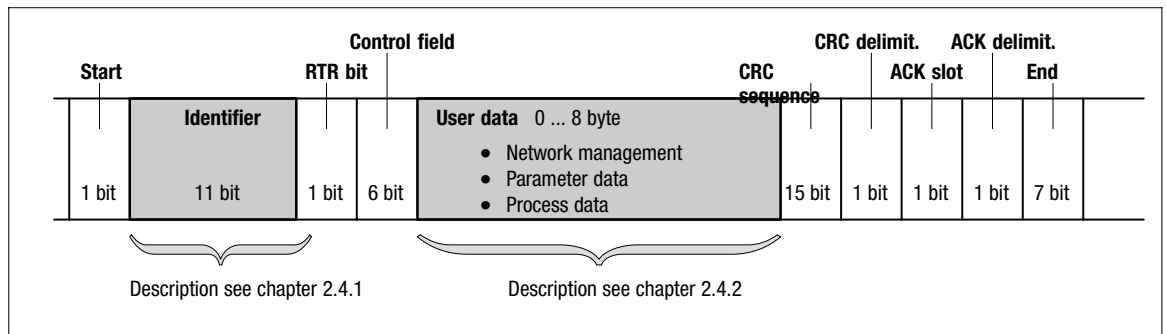


Fig. 2-1

Basic structure of a CAN telegram



### Tip!

For the user only the identifier and the user data are relevant. All further data of the CAN telegram are processed by the system.

### 2.4.1 Identifier

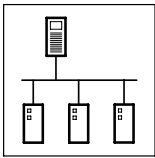
The principle of the CAN communication is based on a message-oriented data exchange between a transmitter and many receivers. Thereby all nodes practically are able to transmit and receive at the same time.

The control with regard to the node which is to receive a transmitted message is effected via the so-called *Identifier* in the CAN telegram, also called *COB-ID (Communication Object Identifier)*. For purposes of addressing, the identifier additionally contains information on the priority of the message, as well as on the type of the user data.

The identifier is composed of a so-called basic identifier and the node address of the node to be activated:

$$\text{Identifier} = \text{basic identifier} + \text{node address}$$

- For Lenze devices, the node address is defined via code C0350. (3-3)
- For the network management and the sync telegram only the basic identifier is required.



# System bus (CAN) for Lenze PLC devices

## General information

The following table contains the preset basic identifiers of the Lenze devices:

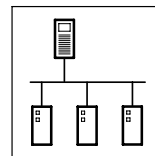
Identifier =	basic identifier		+ node address of the node
	dec	hex	
<b>Network management</b>			
Tx (transmission)	0	0	
Rx (reception)	0	0	
<b>Sync telegram</b>			
Tx (transmission)	128	80	
Rx (reception)	128	80	
<b>SDOs</b>			
Parameter data channel 1			
Output (transmission)	1536	600	+ C0350 (CAN) + C2350 (XCAN) + C2450 (FIF-CAN/CAN-AUX)
Input (reception)	1408	580	
Parameter data channel 2			
Output (transmission)	1600	640	+ C0350 (CAN) + C2350 (XCAN) + C2450 (FIF-CAN/CAN-AUX)
Input (reception)	1472	5C0	
<b>PDOs</b>			
CAN1_IO (cyclic process data)			
CAN1_IN	512	200	+ C0350 (CAN) + C2350 (XCAN) + C2450 (FIF-CAN/CAN-AUX)
CAN1_OUT	384	180	
CAN2_IO (event- or time-controlled process data)			
CAN2_IN	640	280	+ C0350 (CAN) + C2350 (XCAN) + C2450 (FIF-CAN/CAN-AUX)
CAN2_OUT	641	281	
CAN3_IO (event- or time-controlled process data)			
CAN3_IN	768	300	+ C0350 (CAN) + C2350 (XCAN) + C2450 (FIF-CAN/CAN-AUX)
CAN3_OUT	769	301	



### Tip!

For the process data objects you can also set an individual identifier via the following codes, which is independent of the node address:

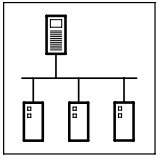
Code	Interface	Information
C0353 / C0354	CAN (system bus interface)	3-4
C2353 / C2354	XCAN (AIF interface)	4-4
C2453 / C2454	FIF-CAN (FIF interface)	5-4
	CAN-AUX (system bus interface)	6-4



### 2.4.2 User data

Via the user data area of the CAN telegram, three different types of data are transported:

Data type	Information		
<b>Network management data</b>	Information on the structure of communication via the CAN network.	Chapter 2.5 (📖 2-6)	
<b>Process data</b>	Process data are data for control-oriented concerns, e. g. setpoints and actual values. <ul style="list-style-type: none"> <li>• Process data are transmitted as so-called PDOs (<i>Process Data Objects</i>) with a high priority.</li> <li>• Process data are processed more quickly by the PLC as parameter data.</li> <li>• The transmission and reception of the process data is effected by the use of specific system blocks or the free CAN objects:</li> </ul>	Chapter 2.6 (📖 2-7)	
	CAN (integrated system bus interface)	<b>SB CAN1_IO</b> for cyclic process data (sync-controlled) 9300 Servo PLC: Drive PLC: ECSxA:	Chapter 7.1 (📖 7-1) Chapter 7.2 (📖 7-6) Chapter 7.2 (📖 7-6)
		<b>SB CAN2_IO</b> for event- or time-controlled process data	Chapter 7.4 (📖 7-14)
		<b>SB CAN3_IO</b> for event- or time-controlled process data	Chapter 7.5 (📖 7-17)
	XCAN (automation interface)	<b>SB AIF1_IO_AutomationInterface</b> for cyclic process data (sync-controlled)	See Manual <ul style="list-style-type: none"> <li>• 9300 Servo PLC</li> <li>• Drive PLC</li> <li>• ECSxA</li> </ul>
		<b>SB AIF2_IO_AutomationInterface</b> for event- or time-controlled process data	
		<b>SB AIF3_IO_AutomationInterface</b> for event- or time-controlled process data	
	FIF-CAN (function interface, Drive PLC only!)	<b>SB FIF_CAN1_IO</b> for cyclic process data (sync-controlled)	Chapter 8.1 (📖 8-1)
		<b>SB FIF_CAN2_IO</b> for event- or time-controlled process data	Chapter 8.2 (📖 8-1)
		<b>SB FIF_CAN3_IO</b> for event- or time-controlled process data	Chapter 8.3 (📖 8-1)
	CAN-AUX (integrated system bus interface, ECSxA only)	<b>SB CANaux1_IO</b> for cyclic process data (sync-controlled)	Chapter 9.1 (📖 9-1)
		<b>SB CANaux2_IO</b> for event- or time-controlled process data	Chapter 9.2 (📖 9-1)
		<b>SB CANaux3_IO</b> for event- or time-controlled process data	Chapter 9.3 (📖 9-1)
	Free CAN objects	By using the functions/function blocks of the <b>LenzeCanDrv.lib</b> function library, so-called "free CAN objects" additionally can be added to the fixedly integrated CAN objects.	Chapter 10 (📖 10-1)
<b>Parameter data</b>	For Lenze devices, parameter data are the so-called codes. <ul style="list-style-type: none"> <li>• Parameter settings for instance are carried out in the case of a one-time setting of the system during commissioning, or in the case of a material change of the production machine.</li> <li>• Parameter data are transferred as so-called SDOs (<i>Service Data Objects</i>) via the CAN network and are acknowledged by the receiver, i. e. the transmitter receives a feedback on whether the transmission was successful.</li> </ul>	Chapter 2.7 (📖 2-11)	



# System bus (CAN) for Lenze PLC devices

## General information

### 2.5 Network management (NMT)

The CAN telegram for the network management is structured as follows:

11bit	2 bytes user data	
Identifier	Command	Device address
000000000000		

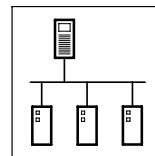
- By means of this telegram the master can carry out state changes for the entire CAN network.

#### Byte 1: command

Command (hex)	Network status after change	Information
01	Operational	The PLC can receive parameter and process data.
02	Stopped	The PLC can receive network management telegrams, parameter and process data, however, cannot be received.
80	Pre-operational	The PLC can receive parameter data. Process data, however, are ignored.
81	Initialisation	Reset mode: Changes with regard to the communication-relevant parameters of the system bus (e. g. CAN address, CAN baud rate, etc.) only are accepted after a reset node.
82		

#### Byte 2: device address

Device address	Information
0	All nodes on the bus are addressed. By this, a state change can be carried out simultaneously for all devices.
1...63	Node address of the node for which a state change is to be effected.



## 2.6 Transmission of process data

Process data are data for control-oriented concerns, e. g. setpoints and actual values.

- Process data are transferred as so-called PDOs (*Process Data Objects*) with a high priority via the system bus.
- Transmitting and receiving the process data is effected by the use of specific system blocks:

CAN (integrated)	XCAN (AIF interface)	FIF-CAN (FIF interface) For Drive PLC only!	CANaux For ECSxA only!	Information
CAN1_IO	AIF1_IO_AutomationInterface	FIF_CAN1_IO	CANaux1_IO	Cyclic process data (sync-controlled)
CAN2_IO	AIF2_IO_AutomationInterface	FIF_CAN2_IO	CANaux2_IO	Event- or time-controlled process data
CAN3_IO	AIF3_IO_AutomationInterface	FIF_CAN3_IO	CANaux3_IO	Event- or time-controlled process data



### Tip!

In the following subchapters you'll receive further information on the CAN1\_IO ... CAN3\_IO process data objects of the CAN interface. This information also applies to the process data objects of the AIF-, FIF- and CAN-AUX interface!

### 2.6.1 Process data channels

#### Process data channel 1: CAN1\_IO

The **CAN1\_IO** SB can be used for the data exchange of cyclic process data (e. g. setpoints and actual values) with a higher-level host system.

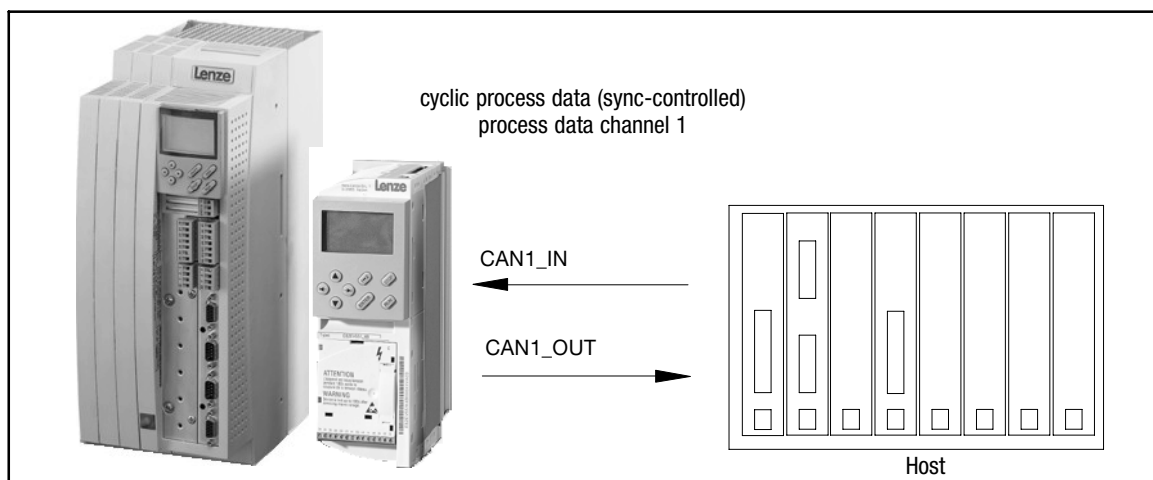
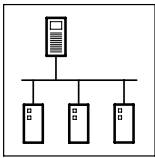


Fig. 2-2

Process data channel 1 (CAN1\_IO) for the cyclic data exchange



# System bus (CAN) for Lenze PLC devices

## General information

### Process data channel 2/3: CAN2\_IO/CAN3\_IO

The SBs **CAN2\_IO** and **CAN3\_IO** are designed for the data exchange of event- or time-controlled process data among the devices. These SBs can also be used for the data exchange with decentralised input/output terminals and higher-level host systems.

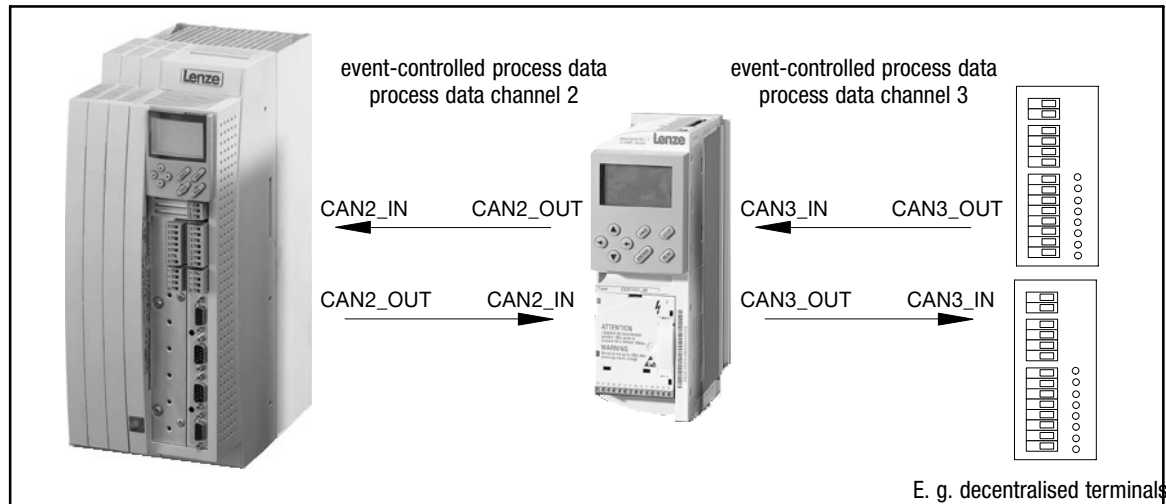


Fig. 2-3

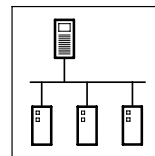
Process data channels 2 and 3 (CAN2\_IO/CAN3\_IO) for the event- or time-controlled data exchange



### Tip!

Detailed information on the CAN1\_IO ... CAN3\_IO CAN objects integrated in the PLC can be found in the chapter 7, "CAN system blocks":

<b>CAN1_IO</b> for cyclic process data (sync-controlled)	9300 Servo PLC: Drive PLC: ECSxA:	Chapter 7.1 (7-1) Chapter 7.2 (7-6) Chapter 7.3 (7-10)
<b>CAN2_IO</b> for event- or time-controlled process data		Chapter 7.4 (7-14)
<b>CAN3_IO</b> for event- or time-controlled process data		Chapter 7.5 (7-17)



### 2.6.2 Sync telegram for cyclic process data

For the transmission of cyclic process data, a specific telegram - the sync telegram - is required for the synchronisation.

The sync telegram which has to be generated by a **different** node initiates the transmission process for the cyclic process data of the PLC and at the same time is the trigger point for the data acceptance of the cyclic process data received in the PLC:

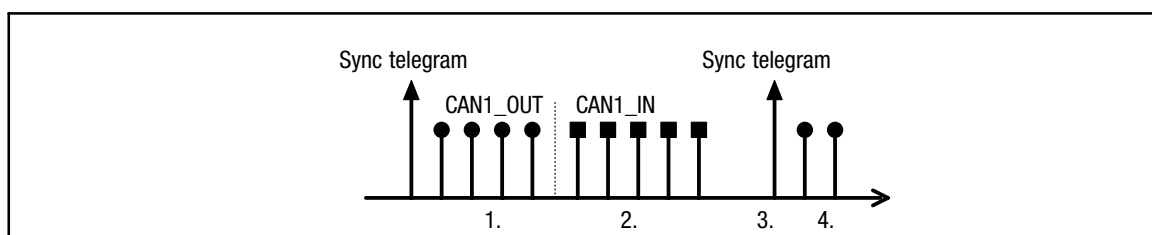


Fig. 2-4

Synchronisation of the cyclic process data by a sync telegram (without considering the asynchronous data)

1. After a sync telegram has been received, the cyclic process output data (CAN1\_OUT) are sent by the PLC if "respond to sync" has been activated.
2. When the transmission process has been completed, the cyclic process input data (CAN1\_IN) are received by the PLC.
3. The data acceptance in the PLC is effected with the next sync telegram.
4. All further telegrams (e. g. for parameters or event-controlled process data) are accepted in an asynchronous manner by the PLC after transmission has been completed.

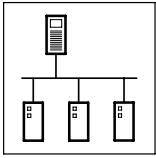


#### Tip!

The response to a sync telegram is configured via the following codes:

- for CAN1\_OUT via C0366. (□ 3-7)
- for XCAN1\_OUT ... XCAN3\_OUT via G2375. (□ 4-7)
- for FIF-CAN1\_OUT via C2466. (□ 5-7)
- for CANaux1\_OUT via C2466. (□ 5-7)

Also the telegrams of CAN2\_OUT and CAN3\_OUT can be transferred after a sync telegram, the parameterisation of this function is carried out via the **CAN\_Management SB**. (□ 7-20)



# System bus (CAN) for Lenze PLC devices

## General information

### 2.6.3 Process data telegram

The process data telegram is structured as follows:

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

#### Identifier

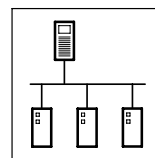
Information on the identifier can be found in chapter 2.4.1. (☞ 2-3)

#### User data

The 8 bytes user data received or to be transmitted respectively can be read or written simultaneously by several variables of different data types.

Detailed information on the user data can be found in the description to the respective system block:

- CAN system blocks (☞ 7-1 ff.)
- FIF-CAN system blocks (☞ 8-1 ff.)
- CAN-AUX system blocks (☞ 9-1 ff.)



## 2.7 Transmitting parameter data

For Lenze devices, parameter data are the so-called codes.

- Parameter settings for instance are carried out in the case of a one-time setting of the system during commissioning, or in the case of a material change of the production machine.
- Parameter data are transferred as so-called SDOs (*Service Data Objects*) via the system bus and are acknowledged by the receiver, i. e. the transmitter receives a feedback on whether the transmission was successful.

### 2.7.1 Parameter data telegram

The telegram for parameter data is structured as follows:

11bit	8 bytes user data							
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					

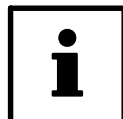
- In the following subchapters the different components of the telegram are explained in detail.
- An example for writing a parameter can be found in chapter 2.7.2. (☞ 2-15)
- An example for reading a parameter can be found in chapter 2.7.3. (☞ 2-17)

#### 2.7.1.1 Identifier

Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					

For the transmission of parameter data, two parameter data channels are provided, which are addressed via the identifier:

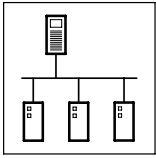
		Identifier =		basic identifier		+ node address of the node	
		dec	hex	dec	hex		
SDOs	Parameter data channel 1						
		Output (transmission)	1536	600	+ C0350	(CAN)	
		Input (reception)	1408	580	+ C2350	(XCAN)	
					+ C2450	(FIF-CAN/CANaux)	
SDOs	Parameter data channel 2						
		Output (transmission)	1600	640	+ C0350	(CAN)	
		Input (reception)	1472	5C0	+ C2350	(XCAN)	
					+ C2450	(FIF-CAN/CANaux)	



#### Tip!

Between the identifiers for parameter data channels 1 and 2 there respectively is an offset of 64:

- Output of parameter data channel 1 = 1536
- Output of parameter data channel 2 = 1536 + 64 = 1600



# System bus (CAN) for Lenze PLC devices

## General information

### 2.7.1.2 Command code

Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					

Among other things, the command code contains the command to be carried out as well as information on the parameter data length, and is structured as follows:

Command	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Command Specifier (cs)			Length		e	s	
Write request	0	0	1	0	00 = 4 bytes 01 = 3 bytes 10 = 2 bytes 11 = 1 byte		1	1
Write response	0	1	1	0			0	0
Read request	0	1	0	0			0	0
Read response	0	1	0	0			1	1
Error Response	1	0	0	0	0	0	0	0

Command code for parameters with 1, 2, or 4 bytes data length:

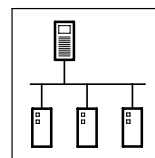
Command	4 byte data (32 bit)		2 byte data (16 bit)		1 byte data (8 bit)		Information
	hex	dec	hex	dec	hex	dec	
Write Request	23	35	2B	43	2F	47	Send parameter to a node
Write Response	60	96	60	64	60	96	Node response to "write request" (acknowledgement)
Read Request	40	64	40	64	40	64	Request for reading a parameter of a node
Read Response	43	67	4B	75	4F	79	Response to the read request with an actual value
Error Response	80	128	80	128	80	128	Node reports an error with regard to communication

#### "Error Response" command

In the case of this error, an "Error Response" is generated by the node that is addressed.

- This telegram in data 4 always contains the value "6", and in data 3 an error code:

Error Response command code	Data 3	Data 4	Error message
80 <sub>hex</sub>	3	6	Access denied
	5		Incorrect subindex
	6		Incorrect index
C0 <sub>hex</sub>	8		Job was not edited (for 8200 vector + FIF module)



### 2.7.1.3 Addressing the parameter (index/subindex)

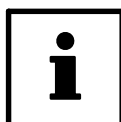
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					

The addressing of the parameter or of the Lenze code which is to be read or written is effected via the index of the telegram:

$$Index = 24575 - Lenze\ code\ number$$

- The value for the index is to be entered divided into a low and a high byte in the left-justified Intel format (see example).
- If a subcode is to be addressed, enter the number of the respective subcode in the subindex of the telegram.
- For codes without subcodes, the subindex always receives the value "0".
- The index for Lenze codes is between  $40C0_{hex}$  (16576) and  $5FFF_{hex}$  (24575).

Lenze code	Index	
	dec	hex
C0000	24575	5FFF
...	...	...
C7999	16576	40C0



#### Tip!

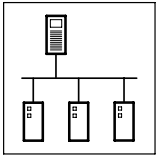
For converting a code number to the corresponding index, the function **L\_FUNCodeIndexConv** in the **LenzeDrive.lib** function library is provided to you.

#### Example:

Subcode 1 of code C0168 (fault messages) is to be addressed:

$$Index = 24575 - 168 = 24407 = 5F57_{hex}$$

11bit	8 bytes user data							
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					
		$57_{hex}$	$5F_{hex}$	$1_{hex}$				



# System bus (CAN) for Lenze PLC devices

## General information

### 2.7.1.4 Data of the parameter (data 1 ... data 4)

Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					

For the data of the parameter up to 4 bytes (data 1 ... data 4) are provided.

- The data are presented in the left-justified Intel format with data 1 as LSB and data 4 as MSB (see example).

#### Example:

For a code in the "Fixed32" data format, the value "20" is to be transmitted.

- "Fixed32" is a fixed point format with 4 decimal positions. Therefore the value has to be multiplied by 10000:

$$Data_{1...4} = 20 \cdot 10000 = 200000 = 00\ 03\ 0D\ 40_{hex}$$

11bit	8 bytes user data							
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					
					40 <sub>hex</sub>	0D <sub>hex</sub>	03 <sub>hex</sub>	00 <sub>hex</sub>

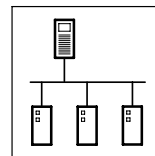
(LSB) (MSB)



#### Tip!

The parameters of the Lenze devices are stored in different formats.

Detailed information on this subject can be found in the Manual for the respective PLC in the chapter "Appendix - Table of attributes".



### 2.7.2 Writing parameters (example)

#### Task

The acceleration time (C0012) of the controller with the node address 1 is to be set to 20 s via the parameter data channel 1.

#### Telegram to the controller

	Formula	Information
<b>Identifier</b>	= basic identifier + node address = 1536 + 1 = <b>1537</b>	<ul style="list-style-type: none"> <li>Basic identifier for parameter data channel 1 (output) = 1536</li> <li>Node address of the controller = 1</li> </ul>
<b>Command code</b>	= <b>23<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>Command "Write Request" (send parameter to controller)</li> </ul>
<b>Index</b>	= 24575 - number of the Lenze code = 24575 - 12 = 24563 = <b>5F F3<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>Code = C0012 (acceleration time)</li> </ul>
<b>Subindex</b>	= <b>0</b>	<ul style="list-style-type: none"> <li>Subcode = 0 (no subcode)</li> </ul>
<b>Data 1 ... 4</b>	= 20 x 10000 = 200000 = <b>00 03 0D 40<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>Value = 20 s</li> <li>Fixed32 data format (4 fixed decimal positions); multiply value by 10000</li> </ul>

11bit	8 bytes user data							
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					
1537	23 <sub>hex</sub>	F3 <sub>hex</sub>	5F <sub>hex</sub>	0	40 <sub>hex</sub> (LSB)	0D <sub>hex</sub>	03 <sub>hex</sub>	00 <sub>hex</sub> (MSB)

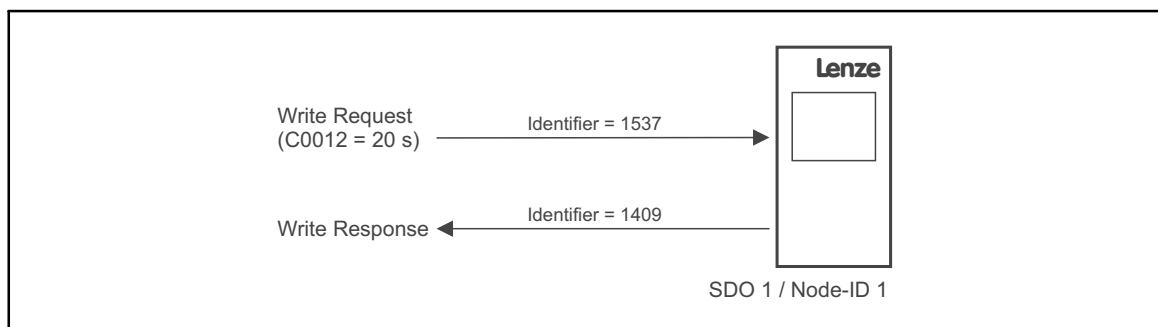
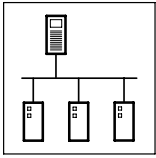


Fig. 2-5

Writing parameters



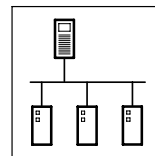
# System bus (CAN) for Lenze PLC devices

## General information

### Telegram from controller (acknowledgement if carried out correctly)

	Formula	Information
<b>Identifier</b>	= basic identifier + node address = 1408 + 1 = <b>1409</b>	<ul style="list-style-type: none"> <li>• Basic identifier for parameter data channel 1 (input) = 1408</li> <li>• Node address of the controller = 1</li> </ul>
<b>Command code</b>	= <b>60<sub>hex</sub></b>	• "Write Response" command (acknowledgement by the controller)
<b>Index</b>	= index of the read request	
<b>Subindex</b>	= subindex of the read request	
<b>Data 1 ... 4</b>	= <b>0</b>	• Acknowledgement only

11bit	8 bytes user data							
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					
1409	60 <sub>hex</sub>	F3 <sub>hex</sub>	5F <sub>hex</sub>	0	0	0	0	0



### 2.7.3 Reading a parameter (example)

#### Task

The heatsink temperature (C0061) of the controller with the node address 5 is to be read via the parameter data channel 1.

#### Telegram to the controller

	Formula	Information
<b>Identifier</b>	= basic identifier + node address = 1536 + 5 = <b>1541</b>	<ul style="list-style-type: none"> <li>Basic identifier for parameter data channel 1 (output) = 1536</li> <li>Node address of the controller = 5</li> </ul>
<b>Command code</b>	= <b>40<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>Command "Read Request" (request to read a parameter from the controller)</li> </ul>
<b>Index</b>	= 24575 - number of the Lenze code = 24575 - 61 = 24514 = <b>5F C2<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>Code = C0061 (heatsink temperature)</li> </ul>
<b>Subindex</b>	= <b>0</b>	<ul style="list-style-type: none"> <li>Subcode = 0 (no subcode)</li> </ul>
<b>Data 1 ... 4</b>	= <b>0</b>	<ul style="list-style-type: none"> <li>Read request only</li> </ul>

11bit	8 bytes user data							
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
		Low byte	High byte					
1541	40 <sub>hex</sub>	C2 <sub>hex</sub>	5F <sub>hex</sub>	0	0	0	0	0

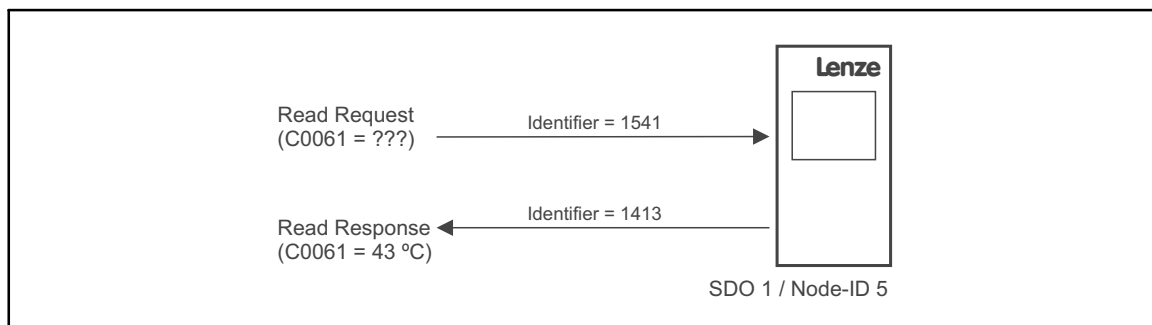
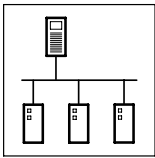


Fig. 2-6

Reading parameters



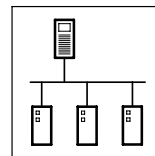
# System bus (CAN) for Lenze PLC devices

## General information

### Telegram from the controller (value of the parameter requested)

	Formula	Information
<b>Identifier</b>	= basic identifier + node address = 1408 + 5 = <b>1413</b>	<ul style="list-style-type: none"> <li>Basic identifier for parameter data channel 1 (input) = 1408</li> <li>Node address of the controller = 5</li> </ul>
<b>Command code</b>	= <b>43<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>"Read Response" command (response to the read request with the current value)</li> </ul>
<b>Index</b>	= index of the read request	
<b>Subindex</b>	= subindex of the read request	
<b>Data 1 ... 4</b>	= 43 x 10000 = 430000 = <b>00 06 8F B0<sub>hex</sub></b>	<ul style="list-style-type: none"> <li>Assumption: The current heatsink temperature of the controller is 43 °C, therefore the value of the parameter to be read is 43</li> <li>Fixed32 data format (4 fixed decimal positions); multiply value by 10000</li> </ul>

11bit	8 bytes user data								
Identifier	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4	
		Low byte	High byte						
1413	43 <sub>hex</sub>	C2 <sub>hex</sub>	5F <sub>hex</sub>	0	B0 <sub>hex</sub>	8F <sub>hex</sub>	06 <sub>hex</sub>	00 <sub>hex</sub>	
					(LSB)				(MSB)



## 2.8 Free CAN objects

If many nodes are connected to the system bus (CAN), it may occur that the CAN objects (CAN1\_IO ... CAN3\_IO) which are fixedly integrated in the PLC are not sufficient for the communication intended, and further CAN objects are required.

By using the functions/function blocks of the **LenzeCanDrv.lib** function library, so-called "free CAN objects" can be added to the fixedly integrated CAN objects. (☞ 10-2)

### Characteristics of the free CAN objects

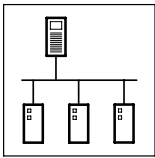
	Free CAN objects
User data per object	1 ... 8 byte
Intended use	Transmission of parameter and process data
Transmission modes	<ul style="list-style-type: none"><li>• Event-controlled transmission</li><li>• Time-controlled transmission</li><li>• Time-controlled transmission with superimposed event control</li><li>• Forced transmission</li></ul>
Range for identifiers	Transmission and reception identifiers can be allocated in the range of 0 ... 2047, but they have to differ from the identifiers of the integrated CAN objects.



### Note!

The free CAN objects are processed via a so-called transmit request memory by the operating system, i. e. the transmission process is not carried out immediately when the **L\_CanPdoTransmit** FB is called, but is effected in a delayed manner. (☞ 10-8)

The transmission and reception jobs of the free CAN objects are not linked to the process image.



# System bus (CAN) for Lenze PLC devices

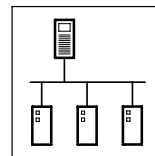
## General information

### 2.9 Application recommendations for the different CAN objects

The following table provides a comparison of the different CAN objects and their specific characteristics:

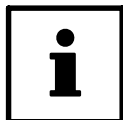
CAN object	Process data transmission	Linked to the process image	Sync telegram required <sup>1)</sup>	Application recommendation
CAN1_IO	☺	☺	☺	Data exchange <ul style="list-style-type: none"> <li>• of position setpoints/actual values</li> <li>• of speed setpoints for servo applications</li> </ul>
CAN2_IO	☺	☺		
CAN3_IO	☺	☺		
XCAN1_IO	☺	☺	☺	
XCAN2_IO	☺	☺		
XCAN3_IO	☺	☺		
FIF_CAN1_IO	☺	☺	☺	
FIF_CAN2_IO	☺	☺		
FIF_CAN3_IO	☺	☺		
CANaux1_IO	☺	☺	☺	
CANaux2_IO	☺	☺		
CANaux3_IO	☺	☺		
Free CAN objects	☺			

<sup>1)</sup> Sync telegram required for the function of the respective CAN1 object



## 2.10 Monitoring mechanisms

In the CANopen communication profile (CiA DS301, version 4.01) two optional monitoring mechanisms for ensuring the function of system bus nodes are specified, "Heartbeat" and "Node Guarding".



### Note!

The "Heartbeat" and "Node Guarding" monitoring mechanisms are supported by the 9300 Servo PLC, Drive, PLC and by the ECSxA axis module as of V6.0.

The initialisation and execution of the monitoring mechanisms is carried out by means of the functions/function blocks of the **LenzeCanDSxDrv.lib** function library. (□ 11-1)

### 2.10.1 "Heartbeat"

The "Heartbeat" monitoring mechanism is a *Producer-Consumer*-oriented method which does not require an enquiry message and where each node is able to monitor the state of all other nodes.

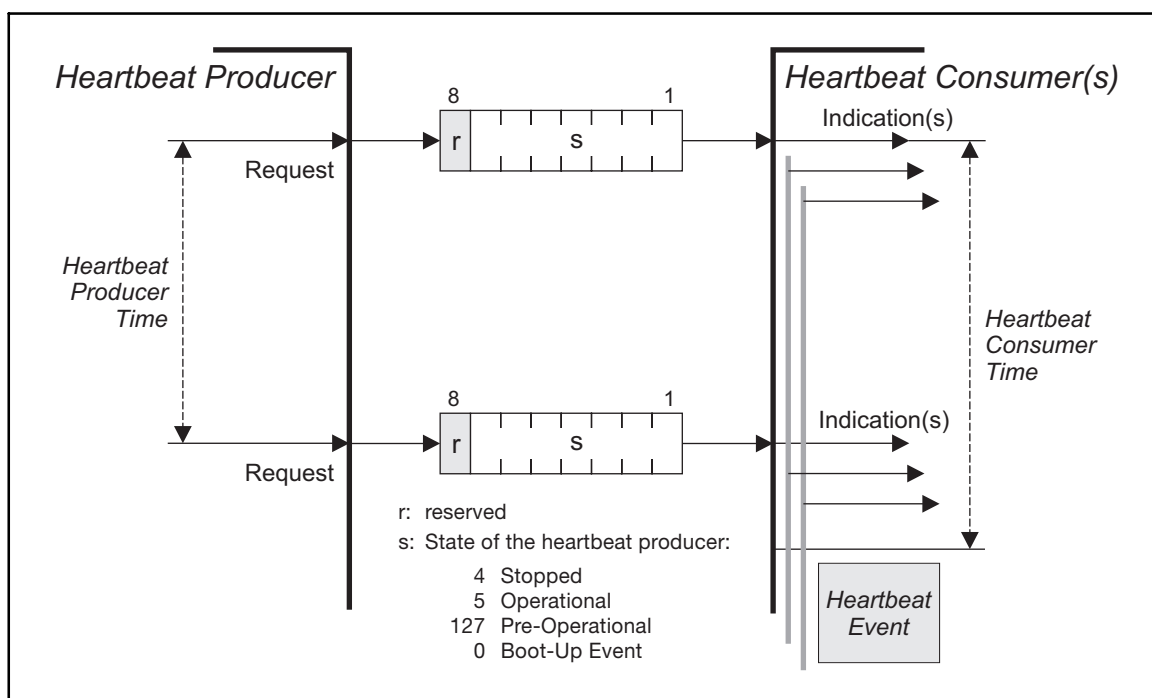
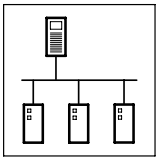


Fig. 2-7 "Heartbeat" monitoring mechanism

- A node (*Producer*) signals its communication status by cyclically transmitting a so-called "Heartbeat" message.
- This "Heartbeat" message can be received by one, several, or by all the other nodes (*Consumer*), and thus they can monitor the respective node.
- If the responsible, monitoring node (*Consumer*) does not receive the "Heartbeat" message from the node to be monitored (*Producer*) within the set monitoring time (*HeartBeatConsumerTime*), a "Heartbeat" event is displayed in its application.



# System bus (CAN) for Lenze PLC devices

## General information

### 2.10.2 "Node Guarding"

In contrast to the "Heartbeat" monitoring mechanism, for the "Node Guarding" an enquiry message from the monitoring node (*NMT Master*) is required.

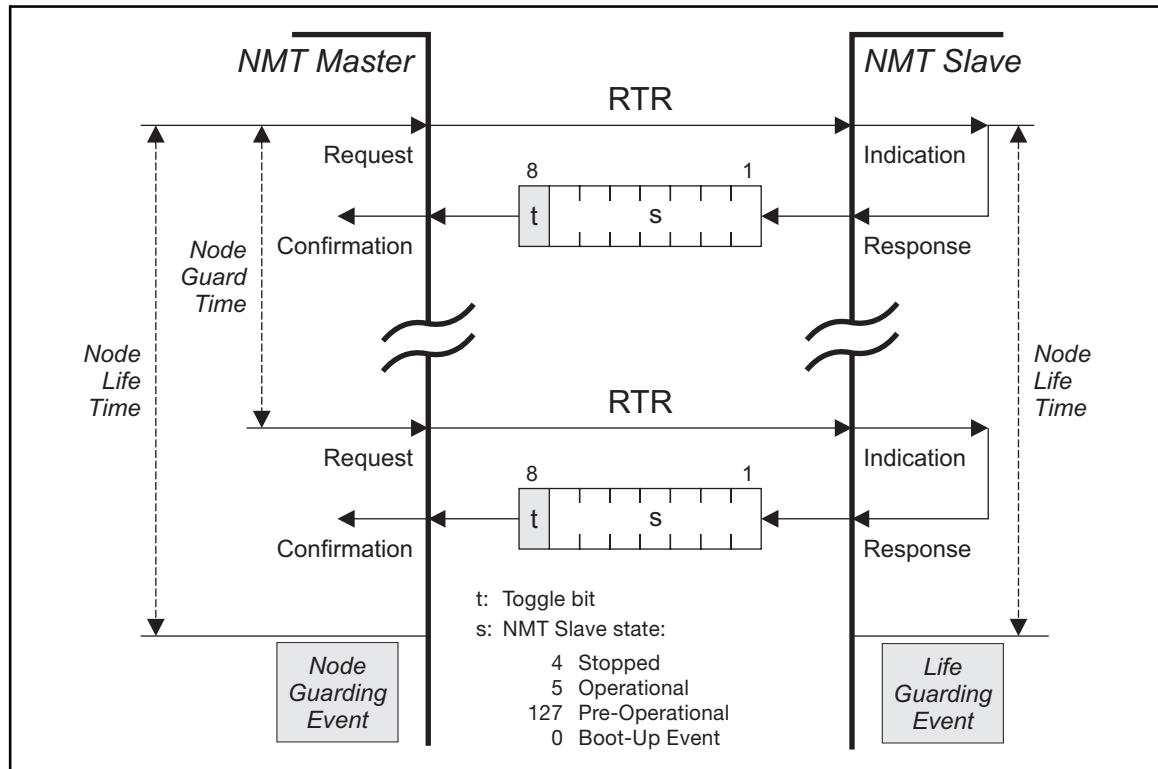
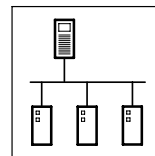


Fig. 2-8 "Node guarding" monitoring mechanism

- The *NMT master* monitors each of the nodes to be monitored (*NMT slave*) cyclically using a node-specific "Remote Transmission Request" telegram.
- The *NMT slave* to be monitored returns its communication status as a response to this request.
- If the *NMT master* does not receive the message from the *NMT Slave* to be monitored within the set monitoring time (*NodeLifeTime*), a "Node Guarding" event is displayed in its application.
- For the *NMT slave* to be monitored, however, a "Life Guarding" event is activated if its status has not been enquired by the monitoring *NMT master* for longer than its "Node Life Time".



### 3 Configuration (system bus - CAN interface)



#### Tip!

Changes with regard to the CAN baud rate, the CAN addresses, and the identifiers for PDOs only are accepted after a reset node.

A reset node can be effected by

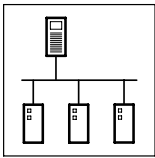
- Reconnection of the mains
- Reset node command via NMT command. (▢ 2-6)
- Reset node via C0358 (▢ 3-8)

#### 3.1 CAN baud rate

In order to accomplish communication via the system bus, all nodes have to use the same baud rate for data transmission.

- The configuration of the baud rate is effected via code C0351:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C0351	CAN baud rate	0	0	500 kbit/s	System bus - baud rate <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>
			1	250 kbit/s	
			2	125 kbit/s	
			3	50 kbit/s	
			4	1000 kbit/s	



# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration

### 3.2 CAN boot-up

If the initialisation of the system bus and the associated state change from *Pre-operational* to *Operational* is not taken over by a higher-level master system, the PLC or a controller can be designated as a "quasi" master to accomplish this task.

- The configuration is effected via code C0352:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C0352	CAN mst	0	0	Boot-up not active	Device sends system bus boot-up and thus is the "quasi" master.
			1	Boot-up active	

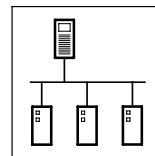
#### Delay time for system bus initialisation (boot-up)

Some nodes (e. g. HMIs) require a specific starting time after mains connection before they can be transferred to the *Operational* state via NMT commands by the master.

In order to ensure that even the node with the greatest starting time really is ready to receive NMT commands, you can set a delay time. When it has expired, NMT commands can only be transmitted after mains power-up.

- The configuration of this delay time is effected via code C0356/1:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C0356 1	CAN boot-up		0	{1 ms}	Delay time after power-on for initialisation via the "quasi" master
		3000		65000	



### 3.3 Node address (node ID)

Assign a node address - also called *Node ID* - within the range of 1 to 63 to each node within the system bus network as a definite identification.

- The same node address may not be assigned more than once within the network.
- The configuration of the node address for the PLC is carried out via code C0350:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0350	CAN address	1	1 {1}	63 System bus - node address <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>

#### Allocation of the node address for the data exchange among Lenze devices

If Lenze devices are provided with node addresses in a consistent ascending order, the identifiers of the event-controlled data objects (CAN2\_IO/CAN3\_IO) are set in way by the factory which enables a communication from one device to the other:

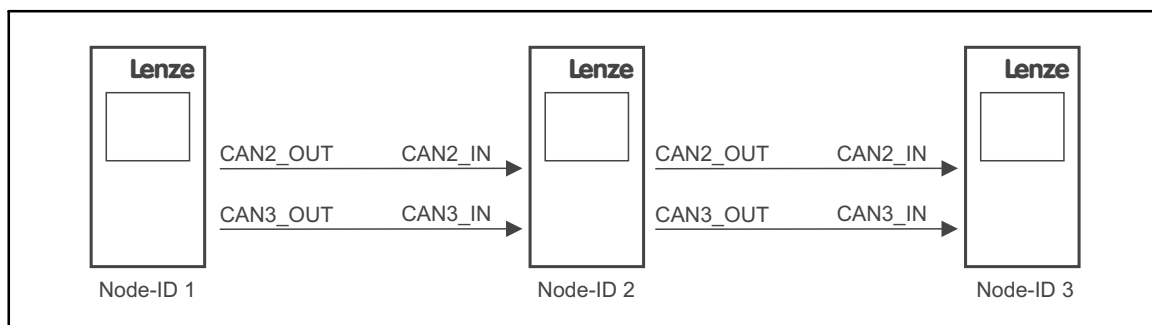
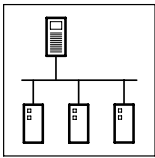


Fig. 3-1

Data exchange among Lenze devices



# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration

### 3.4 Identifiers of the process data objects

The identifiers for the CAN1\_IO ... CAN3\_IO process data objects are generated by the so-called basic identifier and the node address set in C0350:

$$\text{Identifier} = \text{basic identifier} + \text{node address}$$

		Basic identifiers			
		dec	hex		
PDOs	CAN1_IO (cyclic process data)	CAN1_IN	512	200	
		CAN1_OUT	384	180	
		CAN2_IO (event- or time-controlled process data)			
			CAN2_IN	640	280
			CAN2_OUT	641	281
	CAN3_IO (event- or time-controlled process data)				
			CAN3_IN	768	300
			CAN3_OUT	769	301

#### 3.4.1 Allocation of individual identifiers

For greater system bus networks with many nodes it may be reasonable to set individual identifiers for the CAN1\_IO ... CAN3\_IO process data objects via C0353/C0354, which are independent of the node address set in C0350:

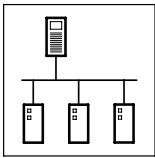
- Set C0353/x to "1".  
– (x = subcode of the corresponding process data object):

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0353	CAN addr sel		0 Identifier assignment under C0350 + basic identifier 1 Identifier assignment under C0354/x	Source for the identifiers of the process data objects • Save changes with C0003 = 1. • Changes are only valid after reset node!
1	CAN addr sel1	0		CAN1_IN/OUT
2	CAN addr sel2	0		CAN2_IN/OUT
3	CAN addr sel3	0		CAN3_IN/OUT

- Set the value which added to "384" makes the desired identifier in C0354/x.  
– (x = subcode of the corresponding process data object):

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0354	CAN addr		1 {1} 512	Specification of individual identifiers for the process data objects
1	IN1 addr2	129		CAN1_IN
2	OUT1 addr2	1		CAN1_OUT
3	IN2 addr2	257		CAN2_IN
4	OUT2 addr2	258		CAN2_OUT
5	IN3 addr2	385		CAN3_IN
6	OUT3 addr2	386		CAN3_OUT





# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration

### 3.5 Cycle time (CAN2\_OUT/CAN3\_OUT)

The transmission of the output data of CAN2\_OUT and CAN3\_OUT can be carried out in an event-controlled or time-controlled manner.

- The configuration of the transmission mode is effected via code C0356/x:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0356			0 {1}	65000
2	CAN2_OUT T	0	0 = event-controlled transmission	Factor to the task time for transmitting the process data object
3	CAN3_OUT T	0		

#### Event-controlled transmission

C0356/x = 0

- The transmission of the output data is always effected if a value has changed within the 8 bytes of user data (Lenze setting).

#### Time-controlled transmission

C0356/x = 1 ... 65000

- The transmission of the output data is effected within the cycle time set in C0356/x (relating to the task cycle time).

Example:

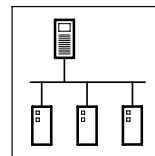
- The CAN object is used in a 10 ms task.
  - Factor set via C0356/2 = 5
- ⇒ The CAN object is transmitted after every fifth cycle of the task, i. e. every 50 ms (10 ms x 5).

### 3.6 Delay time (CAN2\_OUT/CAN3\_OUT)

For the transmission of the output data of CAN2\_OUT and CAN3\_OUT a delay time can be configured via code C0356/4:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0356			0 {1 ms}	65000
4	CAN delay	20		Delay time for sending the process data object.

If the NMT state *Operational* (to *Pre-operational* or *Stopped*) is reached, the "CANdelay" delay time is started. After the delay time has expired, the PDOs CAN-OUT2 and CAN-OUT3 are transmitted for the first time.



### 3.7 Synchronisation



#### Tip!

By means of the **CAN\_Synchronization** SB, the internal time base of the PLC can be synchronised with the arrival of the sync telegram.

Thus, the internal calculating processes (e. g. control-oriented processes) of the PLC can be synchronised with the calculating processes of other nodes which can also process the sync telegram.

Detailed information on the **CAN\_Synchronization** SB can be found in chapter 7.7. (□ 7-23)

#### 3.7.1 CAN sync response

The response to the reception of a sync telegram can be configured via C0366:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C0366	Sync response	1		<b>CAN sync response</b> No response PLC responds to a sync telegram by sending the CAN1_OUT object.
		0	No response	
		1	Response to sync	

#### 3.7.2 CAN sync identifiers

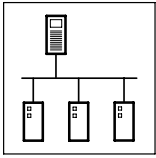
The transmission or reception identifiers of the sync telegram can be configured via C0367/C0368:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C0367	Sync Rx Id	128	1 {1} 256	<b>CAN sync Rx identifier</b> Receive identifier of the sync telegram
C0368	Sync Tx Id	128	1 {1} 256	<b>CAN sync Tx identifier</b> Transmit identifier of the sync telegram

#### 3.7.3 CAN sync Tx transmission cycle

The cycle time within which a sync telegram with the identifier set in C0368 is transmitted can be configured via C0369:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C0369	Sync Tx time	0	0 {1} 65000 0 = Off	<b>CAN sync transmission telegram cycle</b> A sync telegram with the identifier of C0368 is sent with the set cycle time.



# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration

### 3.8 Reset node

Changes with regard to the CAN baud rate, the node addresses and the identifiers only are accepted after a reset node.

A reset node can be effected by

- Reconnection of the mains
- Reset node command via NMT command. (☞ 2-6)
- Reset node command via the **CAN\_Management SB** (☞ 7-20)
- Reset node via C0358:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0358	Reset node	0	No function	Reset node
		1	CAN reset node	

### 3.9 System bus management

Via the **CAN\_Management SB**,

- a reset node can be activated.
- "Communication error" and "Bus off state" can be processed in the PLC program.
- the instant of transmission of CAN2\_OUT and CAN3\_OUT can be influenced.



#### Tip!

Detailed information on the **CAN\_Management SB** can be found in chapter 7.6. (☞ 7-20)

### 3.10 Mapping indexes to codes

The operating system as of V6.0 of the Lenze PLCs contains a specific CanDSx driver which can be activated using the functions of the **LenzeCanDSxDrv.lib** function library. (☞ 11-1)

By means of this driver indexes within the PLC can be assigned to another code as the one which is automatically allocated.



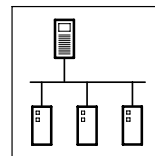
#### Notes!

- Each Lenze code is fixedly assigned to an index via the following formula:

$$\text{Index} = 5FFF_{\text{hex}} - \text{code}$$

$$\text{Index} = 24575_{\text{dec}} - \text{code}$$

- The function of the CanDSx driver only is limited to the system bus (CAN).
  - For the second FIF-CAN channel provided for the drive PLC, the CanDSx driver cannot be used!



### 3.10.1 Functional principle considering as example

#### Task

Within the PLC, a function has been realised from the user side, which can be parameterised via the user code C3200/5. The index 21375<sub>dec</sub> is automatically assigned to code C3200:

$$\text{Index} = 24575_{\text{dec}} - \text{code} = 24575_{\text{dec}} - 3200 = 21375_{\text{dec}}$$

On the basis of the communication profile used, this function, however, is to be parameterisable via the index 4101<sub>dec</sub>/subindex 2 instead.

#### Solution

Via the functions of the **LenzeCanDSxDrv.lib** function library, index 4101<sub>dec</sub>/subindex 2 is simply diverted to code C3200/5 within the PLC, so that the communication profile can be used unchanged.

#### Functional principle

The operating system (as of V6.0) of the Lenze PLCs contains a so-called "mapping table". With this table up to 256 indexes within the PLC can be "mapped" to other codes than to those which are automatically allocated.

If a CAN telegram arrives and the index is in the valid range, it is checked whether this index is listed in the mapping table.

- If the index is listed in the mapping table, the code which has been newly assigned to this index in the mapping table is accessed. ①
- If the index is **not** listed in the mapping table, the code which is automatically allocated is accessed, resulting from the above-mentioned formula. ②

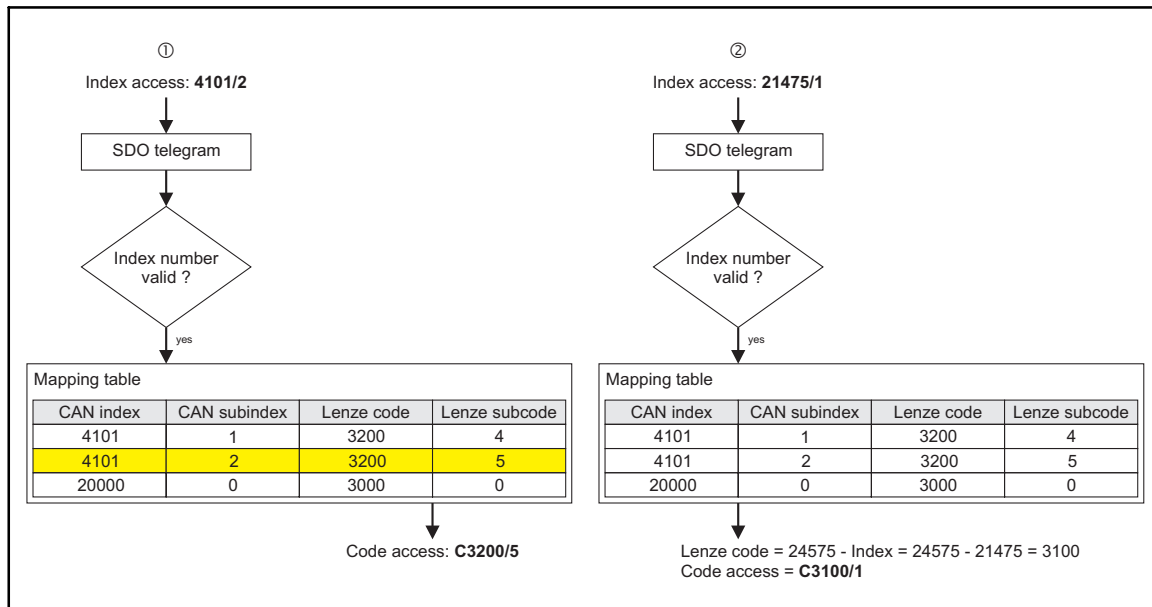
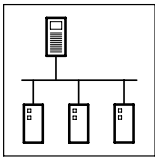


Fig. 3-2 Process of diverting indexes to codes



# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration

### 3.11 Remote parameterisation (gateway function)

The drive PLC, 9300 Servo PLC, and the ECSxA axis module as of the V6.x operating system support the remote parameterisation of other system bus nodes. All write/read accesses to parameters then are no more carried out in the PLC, but are diverted to the node selected for remote maintenance.

- The diversion is effected via the SDO1 parameter data channel of the node selected.
- The node which the diversion of the write/read accesses is to be effected to is defined via C0370 by setting the node address of the corresponding node here:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0370	Gateway addr.	0		System bus: Activate remote parameter setting
			0 {1} 63 0 = remote parameter setting deactivated	

- A timeout during remote parameterisation actuates the system error message CE5; the response to this can be configured via C0603. (☞ 3-12)

#### Example

- The system bus node with the node address 5 has been selected for the remote parameterisation (C0370 = 5).
- A write access to code C0011 is carried out, it is diverted to the selected system bus node via the system bus:

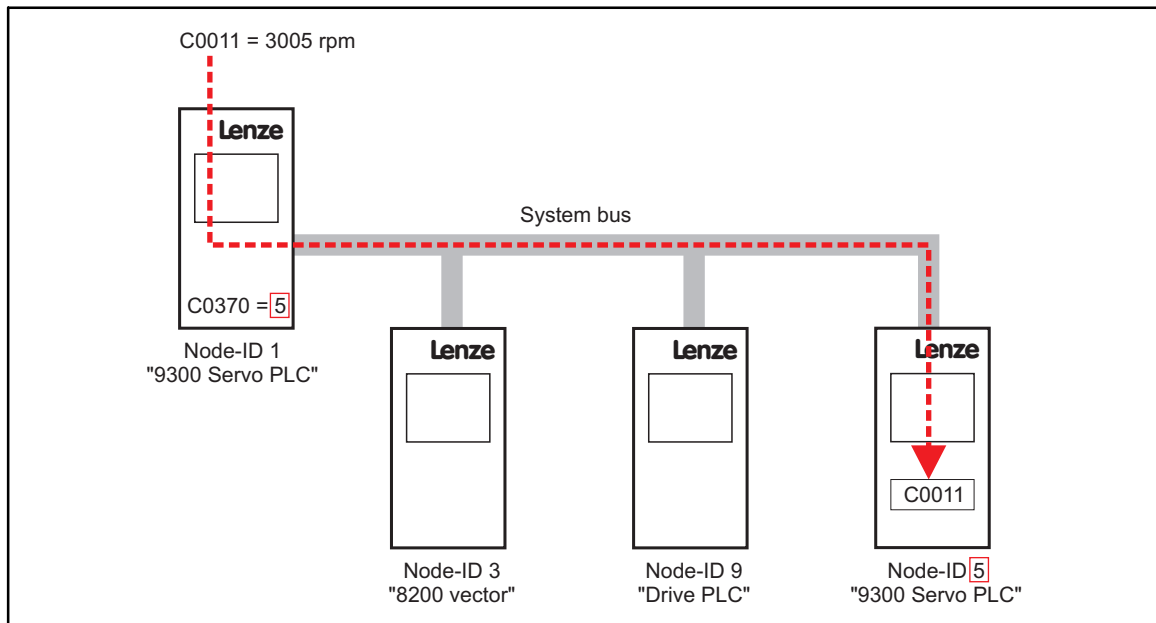
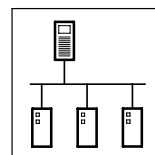


Fig. 3-3 Remote parameterisation (gateway function)



### 3.12 Monitoring processes

#### 3.12.1 Time monitoring for CAN1\_IN ... CAN3\_IN

For the inputs of the CAN1\_IN ... CAN3\_IN process data objects a time monitoring can be configured via C0357:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0357			0 {1 ms} 65000	Monitoring time for process data input objects
1	CE1monit time	3000		
2	CE2monit time	3000		
3	CE3monit time	3000		

The response for the case that no telegram has been received within the defined monitoring time can be configured via codes C0591 ... C0593:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0591	MONIT CE1	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for CAN1_IN error "CommErrCANIN1" (CAN1 CE1)
C0592	MONIT CE2	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for CAN2_IN error "CommErrCANIN2" (CAN2 CE2)
C0593	MONIT CE3	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for CAN3_IN error "CommErrCANIN3" (CAN3 CE3)

#### 3.12.2 Bus-off

If the PLC has disconnected from the system bus due to too many incorrectly received telegrams, the signal "BusOffState" (CE4) is set.

The response to this can be configured via C0595:

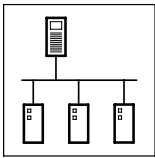
Code	LCD	Possible settings		Information
		Lenze	Selection	
C0595	MONIT CE4	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for "BusOffState" (CE4)



#### Tip!

Possible causes for incorrectly received telegrams can be:

- Missing bus termination
- Non-sufficient shielding
- Differences in potential with regard to the earth connection of the control electronics
- Bus load too high



# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration

### 3.12.3 Time-out when remote parameterisation is activated

If a time-out occurs during the remote parameterisation (gateway function) activated via C0370, the system error message CE5 is output.

The response to this can be configured via C0603:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0603	MONIT CE5	3		System bus: Monitoring configuration Time-out when remote parameterisation is activated (C0370)
			0 TRIP	
			2 Warning	
			3 Off	

### 3.12.4 Response in the case of system bus fault messages

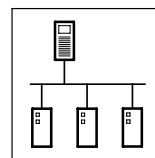
Overview of the system bus error sources registered by the PLC and of the possible settings for the corresponding response:

Fault message			Possible settings/response					
Display	Error No.	Meaning	TRIP	Message	Warning	Fault/QSP	Off	Code
CE1	62	CAN1_IN communication error (time monitoring can be set via C0357/1)	✓	-	✓	-	•	C0591
CE2	63	CAN2_IN communication error (time monitoring can be set via C0357/2)	✓	-	✓	-	•	C0592
CE3	64	CAN3_IN communication error (time monitoring can be set via C0357/3)	✓	-	✓	-	•	C0593
CE4	65	BUS-OFF state (too many faulty telegrams were received)	✓	-	✓	-	•	C0595
CE5	66	CAN time-out (gateway function C0370)	✓	-	✓	-	•	C0603

• Lenze setting  
 ✓ Possible  
 - Not possible

# System bus (CAN) for Lenze PLC devices

## "CAN" system bus interface configuration



### 3.13 Diagnostics

The following codes can be used for diagnostics purposes:

Code	Information displayed	Information
C0359	Operating status of the system bus	Chapter 3.13.1  3-13
C0360	Number of the telegrams sent and received	Chapter 3.13.2  3-14
C0361	Bus load (in %)	Chapter 3.13.3  3-15

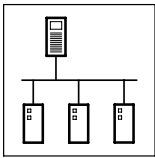
- Settings cannot be carried out via these codes.

#### 3.13.1 Operating status of the CAN interface

Via C0359 you can have the operating status of the system bus displayed:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0359	CAN state	0	Operational	System bus status
			1 Pre-operational	
			2 Warning	
			3 Bus off	

C0359	Operating status	Information
0	Operational	The system bus is fully functional. The PLC can transmit and receive parameter and process data.
1	Pre-operational	The PLC can transmit and receive parameter data. Process data, however, are ignored. A status change from <i>Pre-operational</i> to <i>Operational</i> can be effected by: <ul style="list-style-type: none"> <li>• the CAN master  3-2</li> <li>• a reset node <ul style="list-style-type: none"> <li>– via C0358, if the PLC has been configured as a "quasi" master.  3-8</li> <li>– via the binary input signal "reset node" at the <b>CAN_Management SB</b>  7-20</li> </ul> </li> </ul>
2	Warning	The PLC has received faulty telegrams and is only involved in the system bus passively, i. e. no data can be sent from the PLC anymore. Possible causes: <ul style="list-style-type: none"> <li>• Missing bus termination</li> <li>• Non-sufficient shielding</li> <li>• Differences in potential with regard to the earth connection of the control electronics</li> <li>• Bus load too high</li> <li>• PLC is not connected to the system bus.</li> </ul>
3	Bus-off	The PLC has disconnected from the system bus due to too many faultily received telegrams. <ul style="list-style-type: none"> <li>• The response to this status can be configured via C0595.  3-11</li> </ul>



# System bus (CAN) for Lenze PLC devices

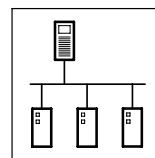
## "CAN" system bus interface configuration

### 3.13.2 Telegram counter

Via C0360 you can have the number of the telegrams sent and received by the PLC via CAN1\_IO ... CAN3\_IO displayed.

- Only the telegrams which are valid for the PLC are counted.
- Each transmit and receive channel is evaluated separately.
- The max. counter content is 65535 (16 bit); if this value is exceeded, the corresponding counter starts with 0 again.

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0360	CAN message	<input type="checkbox"/> Disp	0 {1} 65535	System bus telegram counter (number of telegrams) <ul style="list-style-type: none"> <li>• For values &gt; 65535 counting restarts with 0.</li> </ul>
1	Message OUT			All sent
2	Message IN			All received
3	Message OUT1			Sent on CAN_OUT1
4	Message OUT2			Sent on CAN_OUT2
5	Message OUT3			Sent on CAN_OUT3
6	Message POUT1			Sent on parameter data channel 1
7	Message POUT2			Sent on parameter data channel 2
8	Message IN1			Received from CAN_IN1
9	Message IN2			Received from CAN_IN2
10	Message IN3			Received from CAN_IN3
11	Message PIN1			Received from parameter data channel 1
12	Message PIN2			Received from parameter data channel 2



### 3.13.3 Bus load by the PLC

Via C0361 you can receive a percentage display of the extent to which the system bus is loaded by the telegrams of the PLC.

- Only valid telegrams are considered.
- Each transmit and receive channel is evaluated separately.

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C0361	Load IN/OUT	<input type="text" value="Disp"/>	0 {1 %}	100	System bus - bus load <ul style="list-style-type: none"> <li>• Trouble-free operation demands that the total bus load (all connected devices) does not exceed 80 %.</li> </ul> All sent All received Sent on CAN_OUT1 Sent on CAN_OUT2 Sent on CAN_OUT3 Sent on parameter data channel 1 Sent on parameter data channel 2 Received from CAN_IN1 Received from CAN_IN2 Received from CAN_IN3 Received from parameter data channel 1 Received from parameter data channel 2
	1 Load OUT				
	2 Load IN				
	3 Load OUT1				
	4 Load OUT2				
	5 Load OUT3				
	6 Load POUT1				
	7 Load POUT2				
	8 Load IN1				
	9 Load IN2				
	10 Load IN3				
	11 Load PIN1				
	12 Load PIN2				

#### Limits of the data transmission

Data transmission is limited by the number of telegrams per time unit and by the data transmission speed.

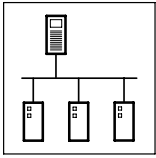
- For a data exchange within a drive system only consisting of Lenze controllers/PLCs, these limits can be determined by adding code C0361/1 of all devices involved.
- **Example:** Three controllers are connected to each other via the system bus:

Value of C0361/1 on controller 1:	23.5 % bus load
Value of C0361/1 on controller 2:	12.6 % bus load
Value of C0361/1 on controller 3:	16.0 % bus load
	<b>52.1 % total bus load</b>



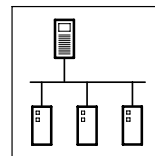
#### Tip!

- The bus load of all devices involved should not exceed 80 %.
- If other devices, like for example decentralised inputs and outputs are connected, these telegrams are to be considered also.
- A bus overload for instance can be effected by an event-controlled transmission of continually changing signals.
  - **Remedy:** Accordingly alter the setting of the cycle time for CAN2\_OUT or CAN3\_OUT, so that the total of all bus loads does not exceed the value of 80 %. (📖 3-6)



## ***System bus (CAN) for Lenze PLC devices***

***"CAN" system bus interface configuration***



## 4 Configuration (AIF interface)

By means of an according fieldbus module (e. g. 2175) you can use the AIF interface of the 9300 Servo PLC, drive PLC and of the ECSxA axis module as an additional system bus interface.



### Note!

If the fieldbus module attached to the AIF interface and the integrated system bus interface are connected to the **same** system bus network, please be absolutely sure that **different** CAN addresses as well as **different** identifiers have been set for the interfaces!



### Tip!

Changes with regard to the CAN baud rate, the CAN addresses, and the identifiers for PDOs only are accepted after a reset node.

A reset node can be effected by

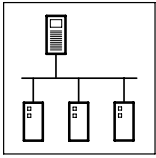
- Reconnection of the mains
- Reset node command via NMT command (▢ 2-6)
- Reset node via C0358 (▢ 3-8)

### 4.1 CAN baud rate

In order to accomplish communication via the system bus, all nodes have to use the same baud rate for data transmission.

- The configuration of the baud rate is effected via code C2351:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C2351	XCAN baud rate	0	0	500 kbit/s	System bus - baud rate <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>
			1	250 kbit/s	
			2	125 kbit/s	
			3	50 kbit/s	
			4	1000 kbit/s	
			5	20 kbit/s (not for ECSxA!)	
			6	10 kbit/s (not for ECSxA!)	



# System bus (CAN) for Lenze PLC devices

## Configuration (AIF interface)

### 4.2 CAN boot-up

If the initialisation of the system bus and the associated state change of *Pre-operational* to *Operational* is not taken over by a higher-level master system, the PLC or a controller can be designated as a "quasi" master to accomplish this task.

- The configuration is effected via code C2352:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2352	XCAN mst	0	0	Boot-up not active
			1	Boot-up active
				Device sends system bus boot-up and thus is the "quasi" master.

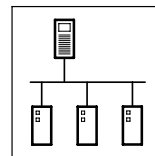
#### Delay time for system bus initialisation (boot-up)

Some nodes (e. g. HMIs) require a specific starting time after mains connection before they can be transferred to the *Operational* state via NMT commands by the master.

In order to ensure that even the node with the greatest starting time really is ready to receive NMT commands, you can set a delay time. When it has expired, NMT commands can only be transmitted after mains power-up.

- The configuration of this delay time is effected via code C2356/1:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2356/1	XCAN boot-up	3000	0	{1 ms} 65000
				Delay time after power-on for initialisation via the "quasi" master



### 4.3 Node address (node ID)

Assign a node address - also called *Node ID* - within the range of 1 to 63 to each node within the system bus network as a definite identification.

- The same node address may not be allocated more than once within the network.
- The configuration of the node address for the AIF interface of the PLC is effected via code C2350:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2350	XCAN address	1	1 {1}	63 System bus - node address <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>

#### Allocation of the node address for the data exchange among Lenze devices

If Lenze devices are provided with node addresses in a consistent ascending order, the identifiers of the event-controlled data objects (XCAN2\_IO/XCAN3\_IO) are set in way by the factory which enables a communication from one device to the other:

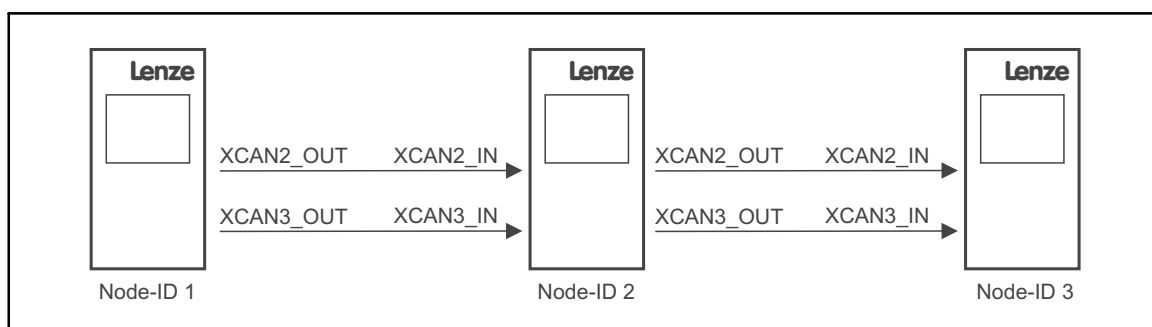
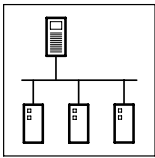


Fig. 4-1

Data exchange among Lenze devices



# System bus (CAN) for Lenze PLC devices

## Configuration (AIF interface)

### 4.4 Identifiers of the process data objects

The identifiers for the XCAN1\_IO ... XCAN3\_IO process data objects are generated by the so-called basic identifier and the node address set in C2350:

$$\text{Identifier} = \text{basic identifier} + \text{node address}$$

		Basic identifiers		
		dec	hex	
PDOs	XCAN1_IO (cyclic process data)	XCAN1_IN	512	200
		XCAN1_OUT	384	180
	XCAN2_IO (event- or time-controlled process data)	XCAN2_IN	640	280
		XCAN2_OUT	641	281
	XCAN3_IO (event- or time-controlled process data)	XCAN3_IN	768	300
		XCAN3_OUT	769	301

#### 4.4.1 Allocation of individual identifiers

For greater system bus networks with many nodes it can be reasonable to set individual identifiers for the XCAN1\_IO ... XCAN3\_IO process data objects via C2353/C2354, which are independent of the node address set in C2350:

- Set C2353/x to "1".  
– (x = subcode of the corresponding process data object):

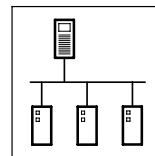
Code	LCD	Possible settings		Information
		Lenze	Selection	
C2353	XCAN addr sel		0 Identifier assignment via C2350 + basic identifier 1 Identifier assignment via C2354/x	Source for the identifiers of the process data objects • Save changes with C0003 = 1. • Changes are only valid after reset node!
1	XCAN addr sel1	0		XCAN1_IN/OUT
2	XCAN addr sel2	0		XCAN2_IN/OUT
3	XCAN addr sel3	0		XCAN3_IN/OUT

- Set the value which added to "384" makes the desired identifier in C2354/x.  
– (x = subcode of the corresponding process data object):

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2354	XCAN addr		0 {1}	1663 Specification of individual identifiers for the process data objects
1	IN1 addr2	129		XCAN1_IN
2	OUT1 addr2	1		XCAN1_OUT
3	IN2 addr2	257		XCAN2_IN
4	OUT2 addr2	258		XCAN2_OUT
5	IN3 addr2	385		XCAN3_IN
6	OUT3 addr2	386		XCAN3_OUT

# System bus (CAN) for Lenze PLC devices

## Configuration (AIF interface)



- Please note that the identifier of the telegram to be transmitted has to comply with the identifier of the process data input object to be activated.
- In the case of an individual address allocation, the identifier for the process data objects is composed as follows:

$\text{Identifier} = 384 + \text{value of C2354}/x$	<i>x</i> = subcode
---	--------------------

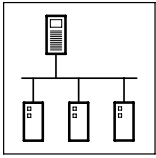
- Thus, identifiers in the range of 384 ... 2047 can be allocated for the process data objects.

### 4.4.2 Display of the identifier set

Via C2355 you can have the identifier displayed which is set for the process data objects.

- C2355 is a display code, settings cannot be carried out via C2355.

Code	LCD	Possible settings			Information	
		Lenze	Selection			
C2355	XCAN Id	<input type="checkbox"/> Disp	384	{1}	2047	System bus identifier for the process data objects XCAN1_IN XCAN1_OUT XCAN2_IN XCAN2_OUT XCAN3_IN XCAN3_OUT
1	XCAN1_IN Id					
2	XCAN1_OUT Id					
3	XCAN2_IN Id					
4	XCAN2_OUT Id					
5	XCAN3_IN Id					
6	XCAN3_OUT Id					



# System bus (CAN) for Lenze PLC devices

## Configuration (AIF interface)

### 4.5 Cycle time (XCAN1\_OUT ... XCAN3\_OUT)

The transmission of the output data of XCAN1\_OUT ... XCAN3\_OUT can be carried out in an event-controlled or time-controlled manner.

- The configuration of the transmission mode is effected via code C2356/x:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2356			0 {1 ms} 65000	Cycle time for transmitting the process data object
2	XCAN1_OUT T	0	0 = event-controlled transmission	
3	XCAN2_OUT T	0		
4	XCAN3_OUT T	0		

#### Event-controlled transmission

C2356/x = 0

- The transmission of the output data is always effected if a value has changed within the 8 bytes of user data (Lenze setting).

#### Time-controlled transmission

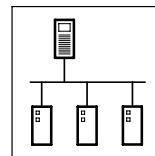
C2356/x = 1 ... 65000

- The transmission of the output data is effected within the cycle time set in C2356/x (relating to the task cycle time).

*Example:*

- The CAN object is used in a 10 ms task.
- Factor set via C2356/2 = 5

⇒ The CAN object is transmitted after every fifth cycle of the task, i. e. every 50 ms (10 ms x 5).



## 4.6 Synchronisation



### Tip!

By means of the **CAN\_Synchronization** SB the internal time base of the PLC can be synchronised with the arrival of the sync telegram.

Thus, the internal calculating processes (e. g. control-oriented processes) of the PLC can be synchronised with the calculating processes of other nodes which can also process the sync telegram.

Detailed information on the **CAN\_Synchronization** SB can be found in chapter 7.7. (□ 7-23)

### 4.6.1 XCAN sync response

The response to the reception of a sync telegram can be configured via C2375:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C2375	XCAN Tx mode		0 Sync with response 1 Sync without response 2 Event-controlled (with mask)/cyclically 3 Event-controlled (with mask) with cyclic overlay	<b>AIF-CAN sync response</b> <ul style="list-style-type: none"> <li>Selection of cycle time under C2356</li> </ul>
1		0		XCAN1_OUT
2		0		XCAN2_OUT
3		0		XCAN3_OUT

### 4.6.2 XCAN sync identifier

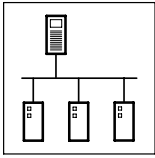
The transmission or reception identifiers of the sync telegram can be configured via C2367/C2368:

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C2367	Sync Rx Id	128	1 {1}	256	<b>XCAN sync Rx identifier</b> Receive identifier for the sync telegram
C2368	Sync Tx Id	128	1 {1}	256	<b>XCAN sync Tx identifier</b> Transmit identifier for the sync telegram

### 4.6.3 XCAN sync Tx transmission cycle

The cycle time within which a sync telegram with the identifier set in C2368 is transmitted can be configured via C2356/5:

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C2356 5	Sync Tx time	0	0 0 = Off	{1 ms} 65535	<b>XCAN sync time</b> Sync transmission telegram cycle



# System bus (CAN) for Lenze PLC devices

## Configuration (AIF interface)

### 4.7 Reset node

Changes with regard to the CAN baud rate, the node addresses, and the identifiers only are accepted after a reset node.

A reset node can be effected by

- Reconnection of the mains
- Reset node command via NMT command. (☞ 2-6)
- Reset node command via the **CAN\_Management SB** (☞ 7-20)
- Reset node via C0358:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0358	Reset node	0	0 No function 1 CAN reset node	Reset node

### 4.8 Monitoring processes

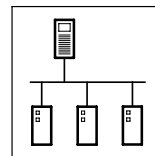
#### 4.8.1 Time monitoring for XCAN1\_IN ... XCAN3\_IN

For the inputs of the XCAN1\_IN ... XCAN3\_IN process data objects a time monitoring can be configured via C2357:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2357	CE monit time		0 {1 ms} 65000	Monitoring time for XCAN process data input objects
1		3000		XCAN1_IN
2		3000		XCAN2_IN
3		3000		XCAN3_IN
4		1		Bus-off

The response for the case that no telegram has been received within the defined monitoring time can be configured via code C2382/x:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2382	XCAN Conf. CE	0	0 Off 1 Controller inhibit 2 Quick stop (QSP)	Configuration of XCAN monitoring "BusOffState" • Controller inhibit or quick stop (QSP) can only be carried out for 9300 Servo PLC and ECSxA!
1		0		XCAN1_IN
2		0		XCAN2_IN
3		0		XCAN3_IN
4		0		Bus-off
5		0		Live guarding event

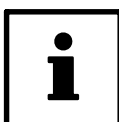


### 4.8.2 Bus off

If the PLC has disconnected from the system bus due to too many incorrectly received telegrams, the signal "BusOffState" (CE14) is set.

The response to this can be configured via C2382/4:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2382 4	XCAN Conf. CE	0	0 Off 1 Controller inhibit 2 Quick stop (QSP)	Configuration of XCAN monitoring "BusOffState" • Controller inhibit or quick stop (QSP) can only be carried out for 9300 Servo PLC and ECSxA!



#### Tip!

Possible causes for incorrectly received telegrams can be:

- Missing bus termination
- Non-sufficient shielding
- Differences in potential with regard to the earth connection of the control electronics
- Bus load too high. See chapter 3.13.3, "Bus load by the PLC". ( 3-15)

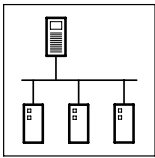
### 4.8.3 Response for system bus fault messages

Overview of the system bus error sources registered by the PLC and of the possible settings for the corresponding response:

Fault message			Possible settings/response			
Display	Error No.	Meaning	CINH	Fault/QSP	Off	Code
XCAN Conf. CE	1)	XCAN1_IN communication error (time monitoring can be set via C2357/1)	✓	✓	⊙	C2382/1
XCAN Conf. CE	1)	XCAN2_IN communication error (time monitoring can be set via C2357/2)	✓	✓	•	C2382/2
XCAN Conf. CE	1)	XCAN3_IN communication error (time monitoring can be set via C2357/3)	✓	✓	•	C2382/3
XCAN Conf. CE	1)	BUS-OFF state (too many faulty telegrams were received)	✓	✓	•	C2382/4

1) Not available; status can be determined via C2121. ( 4-10

• Lenze setting  
✓ Possible



# System bus (CAN) for Lenze PLC devices

## Configuration (AIF interface)

### 4.9 Diagnostics

The following codes can be used for diagnostics purposes:

Code	Information displayed	Information
C2121	AIF-CAN operating status	Chapter 4.9.1  4-10

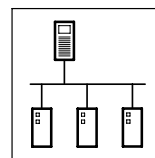
- Settings cannot be carried out via these codes.

#### 4.9.1 Automation interface (AIF) operating status

Via C2121 you can have the operating status of the automation interface displayed:

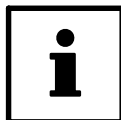
Code	LCD	Possible settings		Information
		Lenze	Selection	
C2121	AIF state		Bit 0 XCAN1_IN monitoring time Bit 1 XCAN2_IN monitoring time Bit 2 XCAN3_IN monitoring time Bit 3 XCAN bus-off Bit 4 XCAN operational Bit 5 XCAN pre-operational Bit 6 XCAN warning Bit 7 Internally assigned	AIF status, bit-coded <ul style="list-style-type: none"> <li>• Detailed information on this can be found in the description for the corresponding fieldbus module.</li> </ul>

C2121	Operating status	Information
Bit4 = 1	Operational	The system bus is fully functional. The PLC can transmit and receive parameter and process data.
Bit5 = 1	Pre-operational	The PLC can transmit and receive parameter data. Process data, however, are ignored. A status change from <i>Pre-Operational</i> to <i>Operational</i> can be effected by: <ul style="list-style-type: none"> <li>• the CAN master  4-2</li> <li>• a reset node               <ul style="list-style-type: none"> <li>– via C0358, if the PLC has been configured as a "quasi" master.  3-8</li> <li>– via the binary input signal "Reset node" at the <b>CAN_Management SB</b>  7-20</li> </ul> </li> </ul>
Bit6 = 1	Warning	The PLC has received faulty telegrams and is only involved in the system bus passively, i. e. no data can be sent from the PLC anymore. Possible causes: <ul style="list-style-type: none"> <li>• Missing bus termination</li> <li>• Non-sufficient shielding</li> <li>• Differences in potential with regard to the earth connection of the control electronics</li> <li>• Bus load too high</li> <li>• PLC is not connected to the system bus.</li> </ul>
Bit3 = 1	Bus-off	The PLC has disconnected from the system bus due to too many faultily received telegrams. <ul style="list-style-type: none"> <li>• The response to this status can be configured via C2382/4.  4-9</li> </ul>



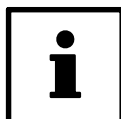
## 5 Configuration (FIF interface)

By means of an appropriate function module (e. g. CAN-I/O system bus) you can use the FIF interface of the Drive PLC as an additional system bus interface.



### Note!

If the fieldbus module attached to the FIF interface and the integrated system bus interface are connected to the **same** system bus network, please be absolutely sure that **different** CAN addresses as well as **different** identifiers have been set for the interfaces!



### Tip!

Changes with regard to the CAN baud rate, the CAN addresses, and the identifiers for PDOs only are accepted after a reset node.

A reset node can be effected by

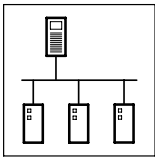
- Reconnection of the mains
- Reset node command via NMT command. (▣ 2-6)
- Reset node via C0358 (▣ 5-8)

### 5.1 CAN baud rate

In order to accomplish communication via the system bus, all nodes have to use the same baud rate for data transmission.

- The configuration of the baud rate is effected via code C2451:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C2451	CAN1 baud rate	0	0	500 kbit/s	System bus - baud rate <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>
			1	250 kbit/s	
			2	125 kbit/s	
			3	50 kbit/s	
			4	1000 kbit/s	



# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

### 5.2 CAN boot-up

If the initialisation of the system bus and the associated state change from *Pre-operational* to *Operational* is not taken over by a higher-level master system, the PLC or a controller can be designated as a "quasi" master to accomplish this task.

- The configuration is effected via code C2452:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2452	CAN1 mst	0	0	Boot-up not active
			1	Boot-up active
				Device sends system bus boot-up and thus is the "quasi" master.

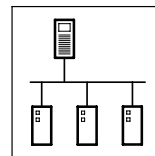
#### Delay time for system bus initialisation (boot-up)

Some nodes (e. g. HMIs) require a specific starting time after mains connection before they can be transferred to the *Operational* state via NMT commands by the master.

In order to ensure that even the node with the greatest starting time really is ready to receive NMT commands, you can set a delay time. When it has expired, NMT commands can only be transmitted after mains power-up.

- The configuration of this delay time is effected via code C2456/1:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2456 1	CAN1 boot-up		0	{1 ms} 65000
		3000		
				Delay time after power-on for initialisation via the "quasi" master



### 5.3 Node address (node ID)

Assign a node address - also called *Node ID* - within the range of 1 to 63 to each node within the system bus network as a definite identification.

- The same node address may not be assigned more than once within the network.
- The configuration of the node address for the FIF interface of the PLC is effected via code C2450:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2450	CAN1 address	1	1 {1}	63 System bus node address <ul style="list-style-type: none"> <li>• Save modifications with C0003 = 1.</li> <li>• Modifications are only valid after reset node!</li> </ul>

#### Allocation of the node address for the data exchange among Lenze devices

If Lenze devices are provided with node addresses in a consistent ascending order, the identifiers of the event-controlled data objects (FIF\_CAN2\_IO/FIF\_CAN3\_IO) are set in way by the factory which enables a communication from one device to the other:

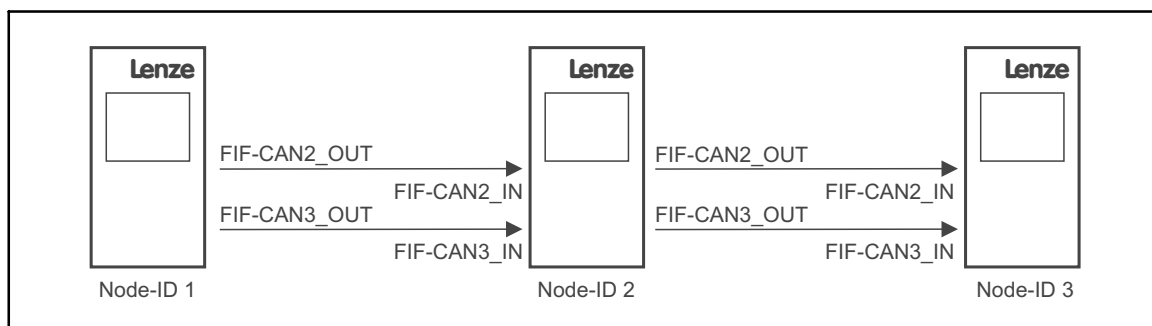
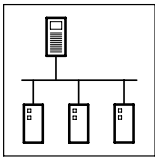


Fig. 5-1

Data exchange among Lenze devices



# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

### 5.4 Identifiers of the process data objects

The identifiers for the FIF\_CAN1\_IO ... FIF\_CAN3\_IO process data objects are generated by the so-called basic identifier and the node address set in C2450:

$$\text{Identifier} = \text{basic identifier} + \text{node address}$$

		Basic identifier		
		dec	hex	
PDOs	FIF_CAN1_IO (cyclic process data)	FIF-CAN1_IN	512	200
		FIF-CAN1_OUT	384	180
	FIF_CAN2_IO (event- or time-controlled process data)	FIF-CAN2_IN	640	280
		FIF-CAN2_OUT	641	281
	FIF_CAN3_IO (event- or time-controlled process data)	FIF-CAN3_IN	768	300
		FIF-CAN3_OUT	769	301

#### 5.4.1 Allocation of individual identifiers

For greater system bus networks with many nodes it can be reasonable to set individual identifiers for the process data objects FIF\_CAN1\_IO ... FIF\_CAN3\_IO via C2453/C2454, which are independent of the node address set in C2350:

- Set C2453/x to "1".  
– (x = subcode of the corresponding process data object):

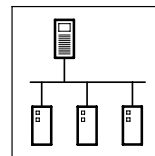
Code	LCD	Possible settings		Information
		Lenze	Selection	
C2453	CAN addr sel		0 Identifier assignment under C2450 + basic identifier 1 Identifier assignment under C2454/x	Source for the identifiers of the process data objects • Save changes with C0003 = 1. • Changes are only valid after reset node!
1	CAN addr sel1	0		FIF-CAN1_IN/OUT
2	CAN addr sel2	0		FIF-CAN2_IN/OUT
3	CAN addr sel3	0		FIF-CAN3_IN/OUT

- Set the value which, added to "384", makes the desired identifier in C2453/x.  
– (x = subcode of the corresponding process data object):

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2454	CAN addr		1 {1} 512	Specification of individual identifiers of the process data objects
1	IN1 addr2	129		FIF-CAN1_IN
2	OUT1 addr2	1		FIF-CAN1_OUT
3	IN2 addr2	257		FIF-CAN2_IN
4	OUT2 addr2	258		FIF-CAN2_OUT
5	IN3 addr2	385		FIF-CAN3_IN
6	OUT3 addr2	386		FIF-CAN3_OUT

# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)



- Please note that the identifier of the telegram to be transmitted has to comply with the identifier of the process data input object to be activated.
- In the case of an individual address allocation, the identifier for the process data objects is composed as follows:

$\text{Identifier} = 384 + \text{value of C2454/x}$	<i>x</i> = subcode
---	--------------------

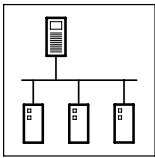
- Thus, for the process data objects identifiers in the range of 385 ... 896 can be allocated.

### 5.4.2 Display of the identifiers set

Via C2455 you can have the identifier displayed which is set for the process data objects.

- C2455 is a display code, settings cannot be carried out via C2455.

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2455	CAN Id	<input type="checkbox"/> Disp	385 {1} 896	System bus identifier for the process data objects
1	CAN1_IN Id			FIF-CAN1_IN
2	CAN1_OUT Id			FIF-CAN1_OUT
3	CAN2_IN Id			FIF-CAN2_IN
4	CAN2_OUT Id			FIF-CAN2_OUT
5	CAN3_IN Id			FIF-CAN3_IN
6	CAN3_OUT Id			FIF-CAN3_OUT



# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

### 5.5 Cycle time (FIF\_CAN2\_OUT/FIF\_CAN3\_OUT)

The transmission of the output data of FIF-CAN2\_OUT and FIF-CAN3\_OUT can be carried out in an event-controlled or time-controlled manner.

- The configuration of the transmission mode is effected via code C2456/x:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2456			0 {1} 0 = event-controlled transmission	65000 Factor to the task time for transmitting the process data object
2	CAN2_OUT T	0		FIF-CAN2_OUT
3	CAN3_OUT T	0		FIF-CAN3_OUT

#### Event-controlled transmission

C2456/x = 0

- The transmission of the output data is always effected if a value has changed within the 8 bytes of user data (Lenze setting).

#### Time-controlled transmission

C2456/x = 1 ... 65000

- The transmission of the output data is effected within the cycle time set in C2456/x (relating to the task cycle time).

Example:

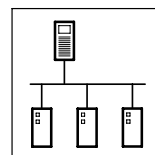
- The CAN object is used in a 10 ms task.
- Factor set via C2456/2 = 5

⇒ The CAN object is transmitted after every fifth cycle of the task, i. e. every 50 ms (10 ms x 5).

### 5.6 Delay time (FIF\_CAN2\_OUT/FIF\_CAN3\_OUT)

For the transmission of the output data of FIF-CAN2\_OUT and FIF-CAN3\_OUT a delay time can be configured via code C2456/4:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2456 4	CAN delay	20	0 {1 ms}	65000 Delay time for sending the process data object



## 5.7 Synchronisation



### Tip!

By means of the **CAN\_Synchronization** SB, the internal time base of the PLC can be synchronised with the arrival of the sync telegram.

Thus the internal calculating processes (e. g. control oriented processes) of the PLC can be synchronised with the calculating processes of other nodes which can also process the sync telegram.

Detailed information on the **CAN\_Synchronization** SB can be found in chapter 7.7. (□ 7-23)

### 5.7.1 FIF-CAN sync response

The response to the reception of a sync telegram can be configured via C2466:

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C2466	Sync response	1	0	No response	<b>FIF-CAN sync response</b> No response PLC responds to a sync telegram by sending the FIF-CAN1_OUT object.
			1	Response to sync	

### 5.7.2 FIF-CAN sync identifier

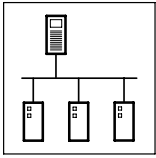
The transmission or reception identifiers of the sync telegram can be configured via C2467/C2468:

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C2467	Sync Rx Id	128	1 {1}	256	<b>FIF-CAN sync Rx identifier</b> Receive identifier for the sync telegram
C2468	Sync Tx Id	128	1 {1}	256	<b>FIF-CAN sync Tx identifier</b> Send identifier for the sync telegram

### 5.7.3 FIF-CAN sync Tx transmission cycle

The cycle time within which a sync telegram with the identifier set in C2468 is transmitted can be configured via C2469:

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C2469	Sync Tx time	0	0 {1 ms} 0 = Off	65000	<b>FIF-CAN sync time</b> Sync transmission telegram cycle



# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

### 5.8 Reset node

Changes with regard to the CAN baud rate, the node addresses, and the identifiers only are accepted after a reset node.

A reset node can be effected by

- Reconnection of the mains
- Reset node command via NMT command. (☞ 2-6)
- Reset node command via the **FIF\_CAN\_Management** SB (☞ 8-4)
- Reset node via C2458:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2458	Reset node	0	0	No function
			1	FIF-CAN reset node

### 5.9 System bus management

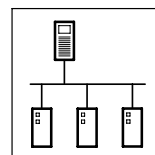
Via the **FIF\_CAN\_Management** SB,

- a reset node can be activated.
- "Communication error" and "Bus off state" can be processed in the PLC program.
- the instant of transmission of FIF-CAN2\_OUT and FIF-CAN3\_OUT can be influenced.



#### Tip!

Detailed information on the **FIF\_CAN\_Management** SB can be found in chapter 8.4. (☞ 8-4)



## 5.10 Monitoring processes

### 5.10.1 Time monitoring for FIF-CAN1\_IN ... FIF-CAN3\_IN

For the inputs of the process data objects FIF-CAN1\_IN ... FIF-CAN3\_IN a time monitoring can be configured via C2457:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2457			0 {1 ms} 65000	Monitoring time for process data input objects
1	CE11monit time	3000		
2	CE12monit time	3000		
3	CE13monit time	3000		

The response for the case that no telegram has been received within the defined monitoring time can be configured via codes C2481 ... C2483:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2481	MONIT CE11	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for FIF-CAN1_IN error (CE11)
C2482	MONIT CE12	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for FIF-CAN2_IN error (CE12)
C2483	MONIT CE13	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for FIF-CAN3_IN error (CE13)

### 5.10.2 Bus-off

If the PLC has disconnected from the system bus due to too many incorrectly received telegrams, the signal "BusOffState" (CE14) is set.

The response to this can be configured via C2484:

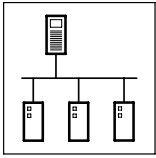
Code	LCD	Possible settings		Information
		Lenze	Selection	
C2484	MONIT CE14	0	0 Off 1 Controller inhibit 2 Quick stop (QSP)	Configuration of the monitoring for "BusOffState" (CE14)



#### Tip!

Possible causes for incorrectly received telegrams can be:

- Missing bus termination
- Non-sufficient shielding
- Differences in potential with regard to the earth connection of the control electronics
- Bus load too high. See chapter 3.13.3, "Bus load by the PLC". ( 3-15)



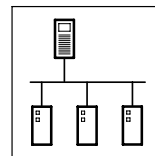
# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

### 5.10.3 Response in the case of system bus fault messages

Overview of the system bus error sources registered by the PLC as well as of the possible settings for the corresponding response:

Fault message			Possible settings/response					
Display	Error No.	Meaning	TRIP	Message	Warning	Fault/QSP	Off	Code
CE11	122	Communication error FIF-CAN1_IN (Time monitoring can be set via C2457/1)	✓	-	✓	-	•	C2481
CE12	123	Communication error FIF-CAN2_IN (Time monitoring can be set via C2457/2)	✓	-	✓	-	•	C2482
CE13	124	Communication error FIF-CAN3_IN (Time monitoring can be set via C2457/3)	✓	-	✓	-	•	C2483
CE14	125	BUS-OFF state (too many faulty telegrams were received)	✓	-	✓	-	•	C2484
			<ul style="list-style-type: none"> <li>• Lenze setting</li> <li>✓ Possible</li> <li>- Not possible</li> </ul>					



## 5.11 Diagnostics

The following codes can be used for diagnostics purposes:

Code	Information displayed	Information
C2459	FIF-CAN operating status	Chapter 5.11.1  5-11
C2460	Number of the telegrams sent and received	Chapter 5.11.2  5-12
C2461	Bus load (in %)	Chapter 5.11.3  5-13

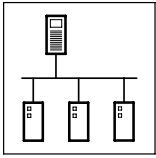
- Settings cannot be carried out via these codes.

### 5.11.1 Function interface (FIF) operating status

Via C2459 you can have the operating status of the function interface displayed:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2459	CAN1 state	0	Operational	System bus status (FIF-CAN)
			1 Pre-operational	
			2 Warning	
			3 Bus off	

C2459	Operating status	Information
0	Operational	The system bus is fully functional. The PLC can transmit and receive parameter and process data.
1	Pre-operational	The PLC can transmit and receive parameter data. Process data, however, are ignored. A status change from <i>Pre-Operational</i> to <i>Operational</i> can be effected by: <ul style="list-style-type: none"> <li>• the CAN master  3-2</li> <li>• a reset node <ul style="list-style-type: none"> <li>– via C2458, if the PLC has been configured as a "quasi" master.  5-8</li> <li>– via the binary input signal "Reset node" at the <b>FIF_CAN_Management SB</b>  7-20</li> </ul> </li> </ul>
2	Warning	The PLC has received faulty telegrams and is only involved in the system bus passively, i. e. no data can be sent from the PLC anymore. Possible causes: <ul style="list-style-type: none"> <li>• Missing bus termination</li> <li>• Non-sufficient shielding</li> <li>• Differences in potential with regard to the earth connection of the control electronics</li> <li>• Bus load too high</li> <li>• PLC is not connected to the system bus.</li> </ul>
3	Bus-off	The PLC has disconnected from the system bus due to too many faultily received telegrams. <ul style="list-style-type: none"> <li>• The response to this status can be configured via C2484.  5-9</li> </ul>



# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

### 5.11.2 Telegram counter

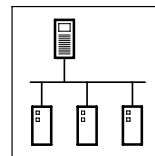
Via C2460 you can have the number of the telegrams sent and received by the PLC via FIF\_CAN1\_IO ... FIF\_CAN3\_IO.

- Only the telegrams which are valid for the PLC are counted.
- Each transmit and receive channel is evaluated separately.
- The max. counter content is 65535 (16 bit); if this value is exceeded, the corresponding counter starts with 0 again.

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2460	CAN message	<input type="checkbox"/> Disp	0 {1} 65535	System bus telegram counter (number of telegrams) <ul style="list-style-type: none"> <li>• For values &gt; 65535 counting restarts with 0.</li> </ul>
1	Message OUT			All sent
2	Message IN			All received
3	Message OUT1			Sent on FIF-CAN_OUT1
4	Message OUT2			Sent on FIF-CAN_OUT2
5	Message OUT3			Sent on FIF-CAN_OUT3
6	Message POUT1			Sent on parameter data channel 1
7	Message POUT2			Sent on parameter data channel 2
8	Message IN1			Received from FIF-CAN_IN1
9	Message IN2			Received from FIF-CAN_IN2
10	Message IN3			Received from FIF-CAN_IN3
11	Message PIN1			Received from parameter data channel 1
12	Message PIN2			Received from parameter data channel 2

# System bus (CAN) for Lenze PLC devices

## Configuration (FIF interface)

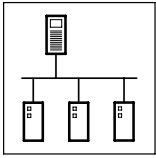


### 5.11.3 Bus load by FIF-CAN

Via C2461 you can receive a percentage display of the extent to which the system bus is loaded by the telegrams of the function interface.

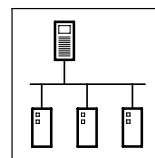
- Only valid telegrams are considered.
- Each transmit and receive channel is evaluated separately.

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C2461	Load IN/OUT	<input type="checkbox"/> Disp	0 {1 %}	100	System bus load (FIF-CAN) <ul style="list-style-type: none"> <li>• Trouble-free operation demands that the total bus load (all connected devices) does not exceed 80 %.</li> </ul> All sent All received Sent on FIF-CAN_OUT1 Sent on FIF-CAN_OUT2 Sent on FIF-CAN_OUT3 Sent on parameter data channel 1 Sent on parameter data channel 2 Received from FIF-CAN_IN1 Received from FIF-CAN_IN2 Received from FIF-CAN_IN3 Received from parameter data channel 1 Received from parameter data channel 2
	1 Message OUT				
	2 Message IN				
	3 Message OUT1				
	4 Message OUT2				
	5 Message OUT3				
	6 Message POUT1				
	7 Message POUT2				
	8 Message IN1				
	9 Message IN2				
	10 Message IN3				
	11 Message PIN1				
	12 Message PIN2				



## ***System bus (CAN) for Lenze PLC devices***

***Configuration (FIF interface)***



## 6 Configuration (CAN-AUX system bus interface)

By means of a corresponding function module (e. g. CAN-I/O system bus) you can use the CAN-AUX interface of the ECSxA axis module as an additional system bus interface.



### Note!

If the function module attached to the CAN-AUX interface and the integrated system bus interface are connected to the **same** system bus network, please be absolutely sure that **different** CAN addresses and **different** identifiers have been set for the interfaces!



### Tip!

Changes with regard to the CAN baud rate, the CAN addresses, and the identifiers for PDOs only are accepted after a reset node.

A reset node can be effected by

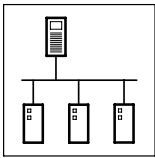
- Reconnection of the mains
- Reset node command via NMT command. (▣ 2-6)
- Reset node via C0358 (▣ 6-8)

### 6.1 CAN baud rate

In order to accomplish a communication via the system bus, all nodes have to use the same baud rate for data transmission.

- The configuration of the baud rate is effected via code C2451:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C2451	CAN1 baud rate	0	0	500 kbit/s	System bus - baud rate <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>
			1	250 kbit/s	
			2	125 kbit/s	
			3	50 kbit/s	
			4	1000 kbit/s	



# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)

### 6.2 CAN boot-up

If the initialisation of the system bus and the associated state change from *Pre-operational* to *Operational* is not taken over by a higher-level master system, the PLC or a controller can be designated as a "quasi" master to accomplish this task instead.

- The configuration is effected via code C2452:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C2452	CAN1 mst	0	0	Boot-up not active	Device sends system bus boot-up and thus is the "quasi" master.
			1	Boot-up active	

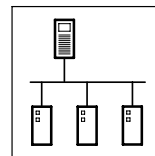
#### Delay time for system bus initialisation (boot-up)

Some nodes (e. g. HMIs) require a specific starting time after mains connection before they can be transferred to the *Operational* state via NMT commands by the master.

In order to ensure that even the node with the greatest starting time really is ready to receive NMT commands, you can set a delay time. When it has expired, NMT commands can only be transmitted after mains power-up.

- The configuration of this delay time is effected via code C2456/1:

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C2456 1	CAN1 boot-up		0	{1 ms}	Delay time after power-on for initialisation via the "quasi" master
		3000		65000	



### 6.3 Node address (Node ID)

Assign a node address - also called *Node ID* - within the range of 1 to 63 to each node within the system bus network as a definite identification.

- The same node address may not be assigned more than once within the network.
- The configuration of the node address for the CAN-AUX interface of the PLC is effected via code C2450:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2450	CAN1 address	1	1 {1}	63 System bus node address <ul style="list-style-type: none"> <li>• Save changes with C0003 = 1.</li> <li>• Changes are only valid after reset node!</li> </ul>

#### Allocation of the node address for the data exchange among Lenze devices

If Lenze devices are provided with node addresses in a consistent ascending order, the identifiers of the event-controlled data objects (CANaux2\_IO/CANaux3\_IO) are set in way by the factory which enables a communication from one device to the other:

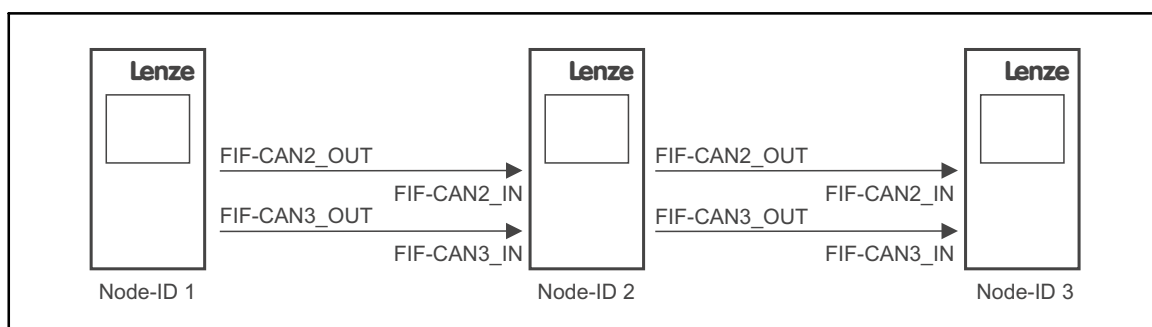
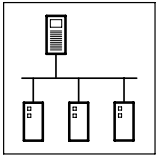


Fig. 6-1

Data exchange among Lenze devices



# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)

### 6.4 Identifiers of the process data objects

The identifiers for the CANaux1\_IO ... CANaux3\_IO process data objects are generated by the so-called basic identifier and the node address set in C2450:

$$\text{Identifier} = \text{basic identifier} + \text{node address}$$

		Basic identifiers		
		dec	hex	
PDOs	CANaux1_IO (cyclic process data)	CANaux1_IN	512	200
		CANaux1_OUT	384	180
	CANaux2_IO (event- or time-controlled process data)	CANaux2_IN	640	280
		CANaux2_OUT	641	281
	CANaux3_IO (event- or time-controlled process data)	CANaux3_IN	768	300
		CANaux3_OUT	769	301

#### 6.4.1 Allocation of individual identifiers

For greater system bus networks with many nodes it can be reasonable to set individual identifiers for the CANaux1\_IO ... CANaux3\_IO process data objects via C2453/C2454, which are independent of the node address set in C2350:

- Set C2453/x to "1".  
– (x = subcode of the corresponding process data object):

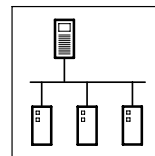
Code	LCD	Possible settings		Information
		Lenze	Selection	
C2453	CAN addr sel		0 Identifier assignment under C2450 + basic identifier 1 Identifier assignment under C2454/x	Source for the identifiers of the process data objects • Save changes with C0003 = 1. • Changes are only valid after reset node!
1	CAN addr sel1	0		CANaux1_IN/OUT
2	CAN addr sel2	0		CANaux2_IN/OUT
3	CAN addr sel3	0		CANaux3_IN/OUT

- Set the value which, added to "384", makes the desired identifier in C2453/x.  
– (x = subcode of the corresponding process data object):

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2454	CAN addr		1 {1} 512	Specification of individual identifiers of the process data objects
1	IN1 addr2	129		CANaux1_IN
2	OUT1 addr2	1		CANaux1_OUT
3	IN2 addr2	257		CANaux2_IN
4	OUT2 addr2	258		CANaux2_OUT
5	IN3 addr2	385		CANaux3_IN
6	OUT3 addr2	386		CANaux3_OUT

# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)



- Please note that the identifier of the telegram to be transmitted has to comply with the identifier of the process data input object to be activated.
- In the case of an individual address allocation, the identifier for the process data objects is composed as follows:

$\text{Identifier} = 384 + \text{value of C2454/x}$	<i>x</i> = subcode
---	--------------------

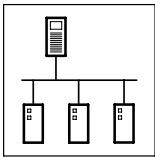
- Thus, for the process data objects identifiers in the range of 385 ... 896 can be allocated.

### 6.4.2 Display of the identifiers set

Via C2455 you can have the identifier displayed which is set for the process data objects.

- C2455 is a display code, settings cannot be carried out via C2455.

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2455	CAN Id	<input type="checkbox"/> Disp	385 {1} 896	System bus identifier for the process data objects CANaux1_IN CANaux1_OUT CANaux2_IN CANaux2_OUT CANaux3_IN CANaux3_OUT
1	CAN1_IN Id			
2	CAN1_OUT Id			
3	CAN2_IN Id			
4	CAN2_OUT Id			
5	CAN3_IN Id			
6	CAN3_OUT Id			



# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)

### 6.5 Cycle time (CANaux2\_OUT/CANaux3\_OUT)

The transmission of the output data of CANaux2\_OUT and CANaux3\_OUT can be carried out in an event-controlled or time-controlled manner.

- The configuration of the transmission mode is effected via code C2456/x:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2456			0 {1} 0 = event-controlled transmission	65000 Factor to the task time for transmitting the process data object
2	CAN2_OUT T	0		CANaux2_OUT
3	CAN3_OUT T	0		CANaux3_OUT

#### Event-controlled transmission

C2456/x = 0

- The transmission of the output data is always effected if a value has changed within the 8 bytes of user data (Lenze setting).

#### Time-controlled transmission

C2456/x = 1 ... 65000

- The transmission of the output data is effected within the cycle time set in C2456/x (relating to the task cycle time).

*Example:*

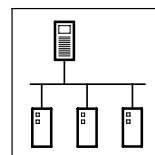
- The CAN object is used in a 10 ms task.
- Factor set via C2456/2 = 5

⇒ The CAN object is transmitted after every fifth cycle of the task, i. e. every 50 ms (10 ms x 5).

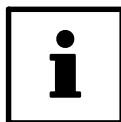
### 6.6 Delay time (CANaux2\_OUT/CANaux3\_OUT)

For the transmission of the output data of CANaux2\_OUT and CANaux3\_OUT a delay time can be configured via code C2456/4:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2456 4	CAN delay	20	0 {1 ms}	65000 Delay time for sending the process data object



## 6.7 Synchronisation



### Tip!

By means of the **CAN\_Synchronization** SB, the internal time base of the PLC can be synchronised with the arrival of the sync telegram.

Thus the internal calculating processes (e. g. control-oriented processes) of the PLC can be synchronised with the calculating processes of other nodes which can also process the sync telegram.

Detailed information on the **CAN\_Synchronization** SB can be found in chapter 7.7. (□ 7-23)

### 6.7.1 CANaux sync response

The response to the reception of a sync telegram can be configured via C2466:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C2466	Sync response	1		<b>CANaux sync response</b> No response PLC responds to a sync telegram by sending the CANaux1_OUT object.
		0	No response	
		1	Response to sync	

### 6.7.2 CANaux sync identifiers

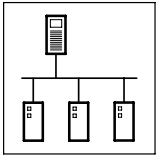
The transmission or reception identifiers of the sync telegram can be configured via C2467/C2468:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C2467	Sync Rx Id	128	1 {1}	256 <b>CANaux sync Rx identifier</b> Receive identifier for the sync telegram
C2468	Sync Tx Id	128	1 {1}	256 <b>CANaux sync Tx identifier</b> Send identifier for the sync telegram

### 6.7.3 CANaux sync Tx transmission cycle

The cycle time within which a sync telegram with the identifier set in C2468 is transmitted can be configured via C2469:

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
C2469	Sync Tx time	0	0 {1 ms} 0 = Off	65000 <b>CANaux sync time</b> Sync transmission telegram cycle



# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)

### 6.8 Reset node

Changes with regard to the CAN baud rate, the node addresses, and the identifiers only are accepted after a reset node.

A reset node can be effected by

- Reconnection of the mains
- Reset node command via NMT command. (☞ 2-6)
- Reset node command via the **CANaux\_Management SB** (☞ 8-4)
- Reset node via C2458:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2458	Reset node	0	0	No function
			1	CANaux reset node

### 6.9 System bus management

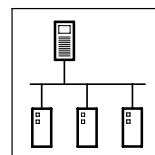
Via the **CANaux\_Management SB**,

- a reset node can be activated.
- "Communication error" and "Bus off state" can be processed in the PLC program.
- the instant of transmission of CANaux2\_OUT and CANaux3\_OUT can be influenced.



#### Tip!

Detailed information on the **CANaux\_Management SB** can be found in chapter 8.4. (☞ 8-4)



## 6.10 Monitoring processes

### 6.10.1 Time monitoring for CANaux1\_IN ... CANaux3\_IN

For the inputs of the process data objects CANaux1\_IN ... CANaux3\_IN a time monitoring can be configured via C2457:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2457			0 {1 ms} 65000	Monitoring time for process data input objects
1	CE11monit time	3000		
2	CE12monit time	3000		
3	CE13monit time	3000		

The response for the case that no telegram has been received within the defined monitoring time can be configured via codes C2481 ... C2483:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2481	MONIT CE11	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for CANaux1_IN error (CE11)
C2482	MONIT CE12	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for CANaux2_IN error (CE12)
C2483	MONIT CE13	3	0 TRIP 2 Warning 3 Off	Configuration of the monitoring for CANaux3_IN error (CE13)

### 6.10.2 Bus-off

If the PLC has disconnected from the system bus due to too many incorrectly received telegrams, the signal "BusOffState" (CE14) is set.

The response to this can be configured via C2484:

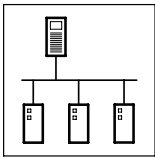
Code	LCD	Possible settings		Information
		Lenze	Selection	
C2484	MONIT CE14	0	0 Off 1 Controller inhibit 2 Quick stop (QSP)	Configuration of the monitoring for "BusOffState" (CE14)



#### Tip!

Possible causes for incorrectly received telegrams can be:

- Missing bus termination
- Non-sufficient shielding
- Differences in potential with regard to the earth connection of the control electronics
- Bus load too high. See chapter 3.13.3, "Bus load by the PLC". ( 3-15)



## System bus (CAN) for Lenze PLC devices

### Configuration (CAN-AUX interface)

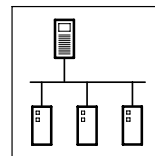
#### 6.10.3 Response in the case of system bus fault messages

Overview of the system bus error sources registered by the PLC as well as of the possible settings for the corresponding response:

Fault message			Possible settings/response					
Display	Error No.	Meaning	TRIP	Message	Warning	Fault/QSP	Off	Code
CE11	122	Communication error CANaux1_IN (Time monitoring can be set via C2457/1)	✓	-	✓	-	•	C2481
CE12	123	CANaux2_IN communication error (Time monitoring can be set via C2457/2)	✓	-	✓	-	•	C2482
CE13	124	CANaux3_IN communication error (Time monitoring can be set via C2457/3)	✓	-	✓	-	•	C2483
CE14	125	BUS-OFF state (too many faulty telegrams were received)	✓	-	✓	-	•	C2484
			<ul style="list-style-type: none"> <li>• Lenze setting</li> <li>✓ Possible</li> <li>- Not possible</li> </ul>					

# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)



## 6.11 Diagnostics

The following codes can be used for diagnostics purposes:

Code	Information displayed	Information
C2459	CAN-AUX operating status	Chapter 6.11.1  6-11
C2460	Number of the telegrams sent and received	Chapter 6.11.2  6-12
C2461	Bus load (in %)	Chapter 6.11.3  6-13

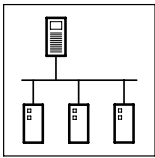
- Settings cannot be carried out via these codes.

### 6.11.1 Operating status of the CAN-AUX interface

Via C2459 you can have the operating status of the system bus displayed:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2459	CAN1 state	0	Operational	System bus status (CAN-AUX)
			1 Pre-operational	
			2 Warning	
			3 Bus off	

C2459	Operating status	Information
0	Operational	The system bus is fully functional. The PLC can transmit and receive parameter and process data.
1	Pre-operational	The PLC can transmit and receive parameter data. Process data, however, are ignored. A status change from <i>Pre-operational</i> to <i>Operational</i> can be effected by: <ul style="list-style-type: none"> <li>• the CAN master  3-2</li> <li>• a reset node <ul style="list-style-type: none"> <li>– via C2458, if the PLC has been configured as a "quasi" master.  6-8</li> <li>– via the binary input signal "Reset node" at the <b>CANaux_Management SB</b>  7-20</li> </ul> </li> </ul>
2	Warning	The PLC has received faulty telegrams and is only involved in the system bus passively, i. e. no data can be sent from the PLC anymore. Possible causes: <ul style="list-style-type: none"> <li>• Missing bus termination</li> <li>• Non-sufficient shielding</li> <li>• Differences in potential with regard to the earth connection of the control electronics</li> <li>• Bus load too high</li> <li>• PLC is not connected to the system bus.</li> </ul>
3	Bus-off	The PLC has disconnected from the system bus due to too many faultily received telegrams. <ul style="list-style-type: none"> <li>• The response to this status can be configured via C2484.  6-9</li> </ul>



# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)

### 6.11.2 Telegram counter

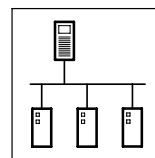
Via C2460 you can have the number of the telegrams sent and received by the PLC via CANaux1\_IO ... CANaux3\_IO.

- Only the telegrams which are valid for the PLC are counted.
- Each transmit and receive channel is evaluated separately.
- The max. counter content is 65535 (16 bit); if this value is exceeded, the corresponding counter starts with 0 again.

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2460	CAN message	<input type="checkbox"/> Disp	0 {1} 65535	System bus telegram counter (number of telegrams) <ul style="list-style-type: none"> <li>• For values &gt; 65535 counting restarts with 0.</li> </ul>
1	Message OUT			All sent
2	Message IN			All received
3	Message OUT1			Sent on CANaux_OUT1
4	Message OUT2			Sent on CANaux_OUT2
5	Message OUT3			Sent on CANaux_OUT3
6	Message POUT1			Sent on parameter data channel 1
7	Message POUT2			Sent on parameter data channel 2
8	Message IN1			Received from CANaux_IN1
9	Message IN2			Received from CANaux_IN2
10	Message IN3			Received from CANaux_IN3
11	Message PIN1			Received from parameter data channel 1
12	Message PIN2			Received from parameter data channel 2

# System bus (CAN) for Lenze PLC devices

## Configuration (CAN-AUX interface)

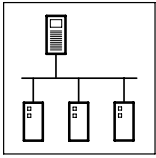


### 6.11.3 Bus load by CAN-AUX

Via C2461 you can receive a percentage display of the extent to which the system bus is loaded by the telegrams of the CAN-AUX interface.

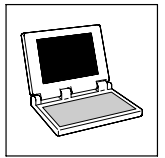
- Only valid telegrams are considered.
- Each transmit and receive channel is evaluated separately.

Code	LCD	Possible settings		Information
		Lenze	Selection	
C2461	Load IN/OUT	<input type="checkbox"/> Disp	0 {1 %} 100	System bus - bus load (CAN-AUX) <ul style="list-style-type: none"> <li>• Trouble-free operation demands that the total bus load (all connected devices) does not exceed 80 %.</li> </ul> All sent All received Sent on CANaux_OUT1 Sent on CANaux_OUT2 Sent on CANaux_OUT3 Sent on parameter data channel 1 Sent on parameter data channel 2 Received from CANaux_IN1 Received from CANaux_IN2 Received from CANaux_IN3 Received from parameter data channel 1 Received from parameter data channel 2
	1 Message OUT			
	2 Message IN			
	3 Message OUT1			
	4 Message OUT2			
	5 Message OUT3			
	6 Message POUT1			
	7 Message POUT2			
	8 Message IN1			
	9 Message IN2			
	10 Message IN3			
	11 Message PIN1			
	12 Message PIN2			



## ***System bus (CAN) for Lenze PLC devices***

***Configuration (CAN-AUX interface)***



## 7 CAN system blocks

### 7.1 CAN1\_IO (node number: 31) - 9300 Servo PLC

This SB serves to transmit cyclic process data via the system bus.

- For the transmission a sync telegram is required, which has to be generated by a **different** node. (📖 2-9)

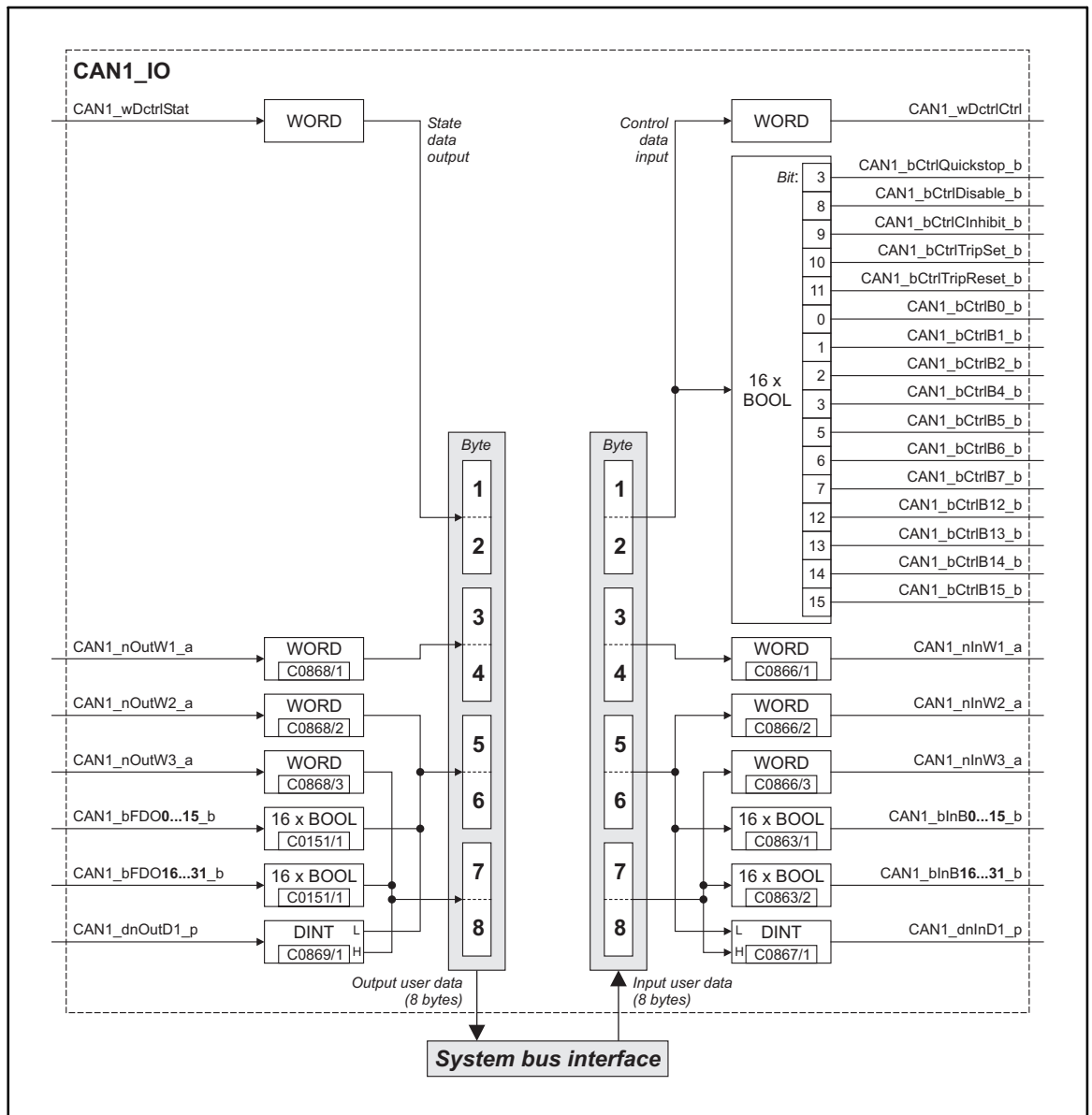
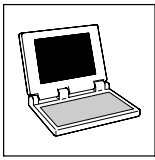


Fig. 7-1

CAN1\_IO system block



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

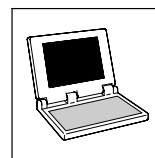
CAN1\_IO (node number: 31) - 9300 Servo PLC

### 7.1.1 Inputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN1_wDctrlCtrl	Word	-	%IW31.0	C0136/2	hex	
CAN1_bCtrlQuickstop_b	Bool	Binary	%IX31.0.3	C0136/2	bin	
CAN1_bCtrlDisable_b			%IX31.0.8			
CAN1_bCtrlClnhibit_b			%IX31.0.9			
CAN1_bCtrlTripSet_b			%IX31.0.10			
CAN1_bCtrlTripReset_b			%IX31.0.11			
CAN1_bCtrlB0_b			%IX31.0.0			
CAN1_bCtrlB1_b			%IX31.0.1			
CAN1_bCtrlB2_b			%IX31.0.2			
CAN1_bCtrlB4_b			%IX31.0.4			
CAN1_bCtrlB5_b			%IX31.0.5			
CAN1_bCtrlB6_b			%IX31.0.6			
CAN1_bCtrlB7_b			%IX31.0.7			
CAN1_bCtrlB12_b			%IX31.0.12			
CAN1_bCtrlB13_b			%IX31.0.13			
CAN1_bCtrlB14_b			%IX31.0.14			
CAN1_bCtrlB15_b	%IX31.0.15					
CAN1_nlnW1_a	Integer	Analog	%IW31.1	C0866/1	dec [%]	
CAN1_nlnW2_a			%IW31.2	C0866/2		
CAN1_nlnW3_a			%IW31.3	C0866/3		
CAN1_blnB0_b	Bool	Binary	%IX31.2.0	C0863/1	hex	
...			...			
CAN1_blnB15_b			%IX31.2.15			
CAN1_blnB16_b			%IX31.3.0	C0863/2		
...			...			
CAN1_blnB31_b	%IX31.3.15					
CAN1_dnlnD1_p	Double integer	Position	%ID31.1	C0867/1	dec [inc]	

### 7.1.2 Outputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN1_wDctrlStat	Word	-	%QW31.0	-	-	
CAN1_nOutW1_a	Integer	Analog	%QW31.1	C0868/1	dec [%]	
CAN1_nOutW2_a			%QW31.2	C0868/2		
CAN1_nOutW3_a			%QW31.3	C0868/3		
CAN1_bFD00_b	Bool	Binary	%QX31.2.0	C0151/1	hex	Display code in hex as double word
...			...			
CAN1_bFD015_b			%QX31.2.15			
CAN1_bFD016_b			%QX31.3.0			
...			...			
CAN1_bFD031_b	%QX31.3.15					
CAN1_dnOutD1_p	Double integer	Position	%QD31.1	C0869/1	dec [inc]	



### 7.1.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)

### 7.1.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

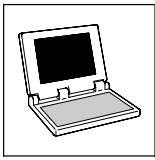
User data		Assigned variables				
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)		
1	0...7					
2	0...7		CAN1_wDctrlStat			
3	0...7					
4	0...7		CAN1_nOutW1_a			
5	0	CAN1_bFDO0_b	CAN1_nOutW2_a	CAN1_dnOutD1_p		
	...	...				
	7	CAN1_bFDO7_b				
6	0	CAN1_bFDO8_b	CAN1_nOutW2_a		CAN1_dnOutD1_p	
	...	...				
	7	CAN1_bFDO15_b				
7	0	CAN1_bFDO16_b	CAN1_nOutW3_a			CAN1_dnOutD1_p
	...	...				
	7	CAN1_bFDO23_b				
8	0	CAN1_bFDO24_b	CAN1_nOutW3_a	CAN1_dnOutD1_p		
	...	...				
	7	CAN1_bFDO31_b				



#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 5 and 6, only use the variable *CAN1\_dnOutD1\_p*, *CAN1\_nOutW2\_a*, or only the variables *CAN1\_bFDO0\_b* ... *CAN1\_bFDO15\_b* for this purpose!



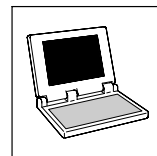
# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN1\_IO (node number: 31) - 9300 Servo PLC

### Variables for received user data

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CAN1_blnB0_b	CAN1_wDctrlCtrl	
	1	CAN1_blnB1_b		
	2	CAN1_blnB2_b		
	3	CAN1_bCtrlQuickstop_b		
	4	CAN1_blnB4_b		
	5	CAN1_blnB5_b		
	6	CAN1_blnB6_b		
7	CAN1_blnB7_b			
2	0	CAN1_bCtrlDisable_b		
	1	CAN1_bCtrlClnhibit_b		
	2	CAN1_bCtrlTripSet_b		
	3	CAN1_bCtrlTripReset_b		
	4	CAN1_blnB12_b		
	5	CAN1_blnB13_b		
	6	CAN1_blnB14_b		
7	CAN1_blnB15_b			
3	0...7		CAN1_nlnW1_a	
4	0...7			
5	0	CAN1_blnB0_b	CAN1_nlnW2_a	CAN1_dnlnD1_p
	...	...		
7	CAN1_blnB7_b			
6	0	CAN1_blnB8_b		
	...	...		
7	CAN1_blnB15_b			
7	0	CAN1_blnB16_b	CAN1_nlnW3_a	
	...	...		
7	CAN1_blnB23_b			
8	0	CAN1_blnB24_b		
	...	...		
	7	CAN1_blnB31_b		



#### 7.1.5 Transferring status and control information of the device control

Via the user data bytes 1 and 2, you can exchange status and control information of the device control (DCTRL) between different 9300 Servo PLCs via the system bus (CAN) in a simple manner.

##### Sending the status word of the DCTRL SB

Connect the variable *DCTRL\_wStat* of the **DCTRL** SB to the variable *CAN1\_wDctrlStat* to transfer the status word of the **DCTRL** SB via the user data bytes 1 and 2.



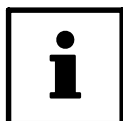
##### Tip!

In addition to signals such as IMP and CINH, the SB status word **DCTRL** contains some freely assignable signals which can be overwritten via the variables *DCTRL\_bStateB...\_b* of the **DCTRL** SB.

Detailed information on the **DCTRL** SB can be found in the "9300 Servo PLC" Online Manual.

##### Transferring the control word to the DCTRL SB

Connect the variable *CAN1\_wDctrlCtrl* to the variable *DCTRL\_wCAN1Ctrl* of the **DCTRL** SB to transfer the control word received via the user data bytes 1 and 2 to the **DCTRL** SB.

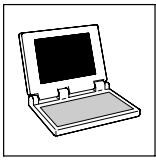


##### Tip!

The control signals for the functions quick stop (QSP), DISABLE, CINH, TRIP-SET, and TRIP-RESET can be additionally read out and processed via the following variables:

- *CAN1\_bCtrlQuickstop\_b*
- *CAN1\_bCtrlDisable\_b*
- *CAN1\_bCtrlInhibit\_b*
- *CAN1\_bCtrlTripSet\_b*
- *CAN1\_bCtrlTripReset\_b*

The remaining 11 bits (*CAN1\_bCtrlB...\_b*) can be used for controlling further functions/function blocks.



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN1\_IO (node number: 31) - Drive PLC

## 7.2 CAN1\_IO (node number: 31) - Drive PLC

This SB serves to transmit cyclic process data via the system bus.

- For the transmission a sync telegram is required, which has to be generated by a **different** node. (☐ 2-9)

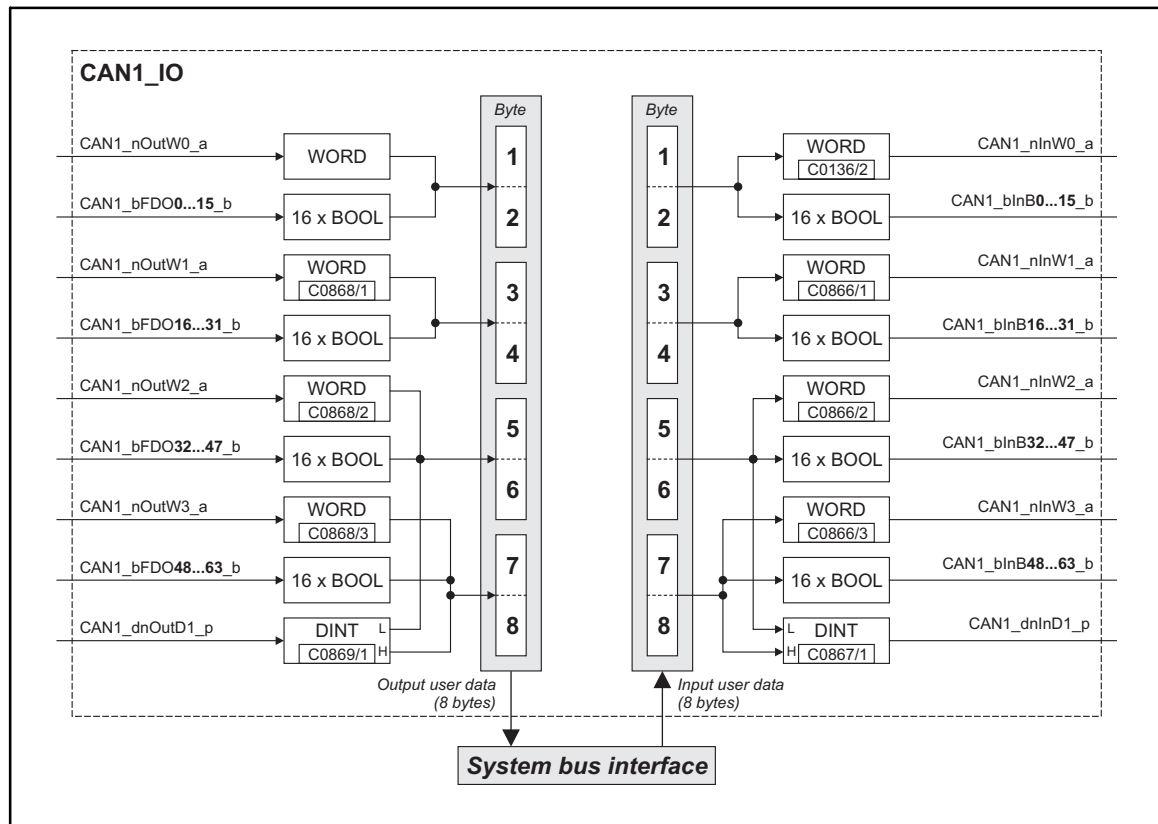
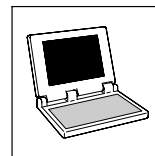


Fig. 7-2 CAN1\_IO system block

# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN1\_IO (node number: 31) - Drive PLC

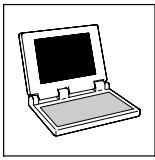


### 7.2.1 Inputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN1_nInW0_a	Integer	Analog	%IW31.0	C0136/2	dec [%]	
CAN1_bInB0_b	Bool	Binary	%IX31.0.0	C0863/1	hex	
...			...			
CAN1_bInB15_b			%IX31.0.15			
CAN1_nInW1_a	Integer	Analog	%IW31.1	C0866/1	dec [%]	
CAN1_bInB16_b	Bool	Binary	%IX31.1.0	C0863/2	hex	
...			...			
CAN1_bInB31_b			%IX31.1.15			
CAN1_nInW2_a	Integer	Analog	%IW31.2	C0866/2	dec [%]	
CAN1_bInB32_b	Bool	Binary	%IX31.2.0			
...			...			
CAN1_bInB47_b			%IX31.2.15			
CAN1_nInW3_a	Integer	Analog	%IW31.3	C0866/3	dec [%]	
CAN1_bInB48_b	Bool	Binary	%IX31.3.0			
...			...			
CAN1_bInB63_b			%IX31.3.15			
CAN1_dnInD1_p	Double integer	Position	%ID31.1	C0867/1	dec [inc]	

### 7.2.2 Outputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN1_nOutW0_a	Integer	Analog	%QW31.0	-	-	
CAN1_bFD00_b	Bool	Binary	%QX31.0.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD015_b			%QX31.0.15			
CAN1_nOutW1_a	Integer	Analog	%QW31.1	C0868/1	dec [%]	
CAN1_bFD016_b	Bool	Binary	%QX31.1.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD031_b			%QX31.1.15			
CAN1_nOutW2_a	Integer	Analog	%QW31.2	C0868/2	dec [%]	
CAN1_bFD032_b	Bool	Binary	%QX31.2.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD047_b			%QX31.2.15			
CAN1_nOutW3_a	Integer	Analog	%QW31.3	C0868/3	dec [%]	
CAN1_bFD048_b	Bool	Binary	%QX31.3.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD063_b			%QX31.3.15			
CAN1_dnOutD1_p	Double integer	Position	%QD31.1	C0869/1	dec [inc]	



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN1\_IO (node number: 31) - Drive PLC

### 7.2.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (☞ 2-3)

### 7.2.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CAN1_bFD00_b	CAN1_nOutW0_a	CAN1_dnOutD1_p
	...	...		
2	7	CAN1_bFD07_b	CAN1_nOutW1_a	
	...	...		
3	7	CAN1_bFD015_b	CAN1_nOutW2_a	
	...	...		
4	0	CAN1_bFD016_b	CAN1_nOutW3_a	
	...	...		
5	7	CAN1_bFD023_b	CAN1_nOutW2_a	
	...	...		
6	0	CAN1_bFD024_b	CAN1_nOutW3_a	
	...	...		
7	7	CAN1_bFD031_b	CAN1_nOutW2_a	
	...	...		
8	0	CAN1_bFD032_b	CAN1_nOutW3_a	
	...	...		
8	7	CAN1_bFD039_b	CAN1_nOutW2_a	
	...	...		
8	7	CAN1_bFD040_b	CAN1_nOutW3_a	
	...	...		
8	7	CAN1_bFD047_b	CAN1_nOutW2_a	
	...	...		
8	7	CAN1_bFD048_b	CAN1_nOutW3_a	
	...	...		
8	7	CAN1_bFD055_b	CAN1_nOutW2_a	
	...	...		
8	7	CAN1_bFD056_b	CAN1_nOutW3_a	
	...	...		
8	7	CAN1_bFD063_b	CAN1_nOutW2_a	
	...	...		



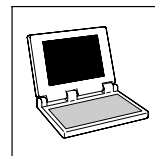
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 5 and 6, only use the variable *CAN1\_dnOutD1\_p*, *CAN1\_nOutW2\_a*, or only the variables *CAN1\_bFD032\_b* ... *CAN1\_bFD047\_b* for this purpose!

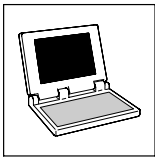
# System bus (CAN) for Lenze PLC devices

**CAN system blocks**  
**CAN1\_IO (node number: 31) - Drive PLC**



## Variables for received user data

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CAN1_blnB0_b	CAN1_nlnW0_a	
	...	...		
7	CAN1_blnB7_b			
2	0	CAN1_blnB8_b		
	...	...		
7	CAN1_blnB15_b			
3	0	CAN1_blnB16_b	CAN1_nlnW1_a	
	...	...		
7	CAN1_blnB23_b			
4	0	CAN1_blnB24_b		CAN1_nlnW2_a
	...	...		
7	CAN1_blnB31_b			
5	0	CAN1_blnB32_b	CAN1_nlnW3_a	
	...	...		
7	CAN1_blnB39_b			
6	0	CAN1_blnB40_b		
	...	...		
7	CAN1_blnB47_b			
7	0	CAN1_blnB48_b		
	...	...		
7	CAN1_blnB55_b			
8	0	CAN1_blnB56_b		
	...	...		
7	CAN1_blnB63_b			



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN1\_IO (node number: 31) - ECSxA

### 7.3 CAN1\_IO (node number: 31) - ECSxA

This SB serves to transmit cyclic process data via the system bus.

- For the transmission a sync telegram is required, which has to be generated by a **different** node. (☐ 2-9)

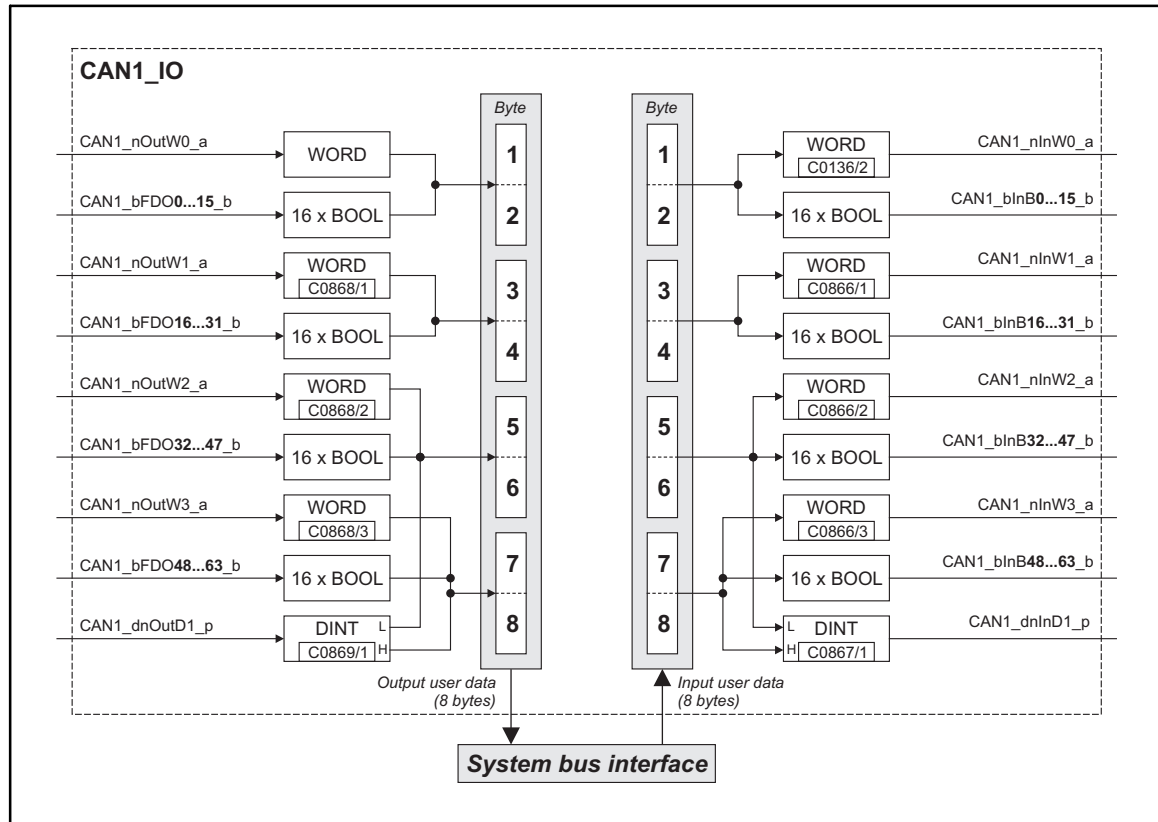
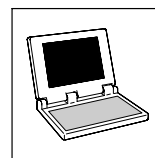


Fig. 7-3 CAN1\_IO system block

# System bus (CAN) for Lenze PLC devices

## CAN system blocks CAN1\_IO (node number: 31) - ECSxA

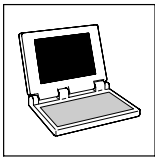


### 7.3.1 Inputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN1_nInW0_a	Integer	Analog	%IW31.0	C0136/2	dec [%]	
CAN1_bInB0_b	Bool	Binary	%IX31.0.0	C0863/1	hex	
...			...			
CAN1_bInB15_b			%IX31.0.15			
CAN1_nInW1_a	Integer	Analog	%IW31.1	C0866/1	dec [%]	
CAN1_bInB16_b	Bool	Binary	%IX31.1.0	C0863/2	hex	
...			...			
CAN1_bInB31_b			%IX31.1.15			
CAN1_nInW2_a	Integer	Analog	%IW31.2	C0866/2	dec [%]	
CAN1_bInB32_b	Bool	Binary	%IX31.2.0			
...			...			
CAN1_bInB47_b			%IX31.2.15			
CAN1_nInW3_a	Integer	Analog	%IW31.3	C0866/3	dec [%]	
CAN1_bInB48_b	Bool	Binary	%IX31.3.0			
...			...			
CAN1_bInB63_b			%IX31.3.15			
CAN1_dnInD1_p	Double integer	Position	%ID31.1	C0867/1	dec [inc]	

### 7.3.2 Outputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN1_nOutW0_a	Integer	Analog	%QW31.0	-	-	
CAN1_bFD00_b	Bool	Binary	%QX31.0.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD015_b			%QX31.0.15			
CAN1_nOutW1_a	Integer	Analog	%QW31.1	C0868/1	dec [%]	
CAN1_bFD016_b	Bool	Binary	%QX31.1.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD031_b			%QX31.1.15			
CAN1_nOutW2_a	Integer	Analog	%QW31.2	C0868/2	dec [%]	
CAN1_bFD032_b	Bool	Binary	%QX31.2.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD047_b			%QX31.2.15			
CAN1_nOutW3_a	Integer	Analog	%QW31.3	C0868/3	dec [%]	
CAN1_bFD048_b	Bool	Binary	%QX31.3.0	-	hex	Display code in hex as double word
...			..			
CAN1_bFD063_b			%QX31.3.15			
CAN1_dnOutD1_p	Double integer	Position	%QD31.1	C0869/1	dec [inc]	



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN1\_IO (node number: 31) - ECSxA

### 7.3.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (☞ 2-3)

### 7.3.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- Binary information (1 bit)
- Status word/quasi-analog value (16 bit)
- Angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CAN1_bFD00_b	CAN1_nOutW0_a	CAN1_dnOutD1_p
	...	...		
2	7	CAN1_bFD07_b		
	0	CAN1_bFD08_b	CAN1_nOutW1_a	
3	...	...		
	7	CAN1_bFD015_b		
4	0	CAN1_bFD016_b	CAN1_nOutW2_a	
	...	...		
5	7	CAN1_bFD023_b		
	0	CAN1_bFD024_b	CAN1_nOutW3_a	
6	...	...		
	7	CAN1_bFD031_b		
7	0	CAN1_bFD032_b	CAN1_nOutW2_a	
	...	...		
8	7	CAN1_bFD039_b		
	0	CAN1_bFD040_b	CAN1_nOutW3_a	
...	...	...		
	7	CAN1_bFD047_b		
...	0	CAN1_bFD048_b	CAN1_nOutW3_a	
	...	...		
7	CAN1_bFD055_b			
8	0	CAN1_bFD056_b	CAN1_nOutW3_a	
	...	...		
7	CAN1_bFD063_b			



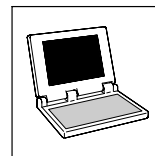
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write the bytes 5 and 6, only use the variable *CAN1\_dnOutD1\_p*, *CAN1\_nOutW2\_a*, or only the variables *CAN1\_bFD032\_b* ... *CAN1\_bFD047\_b* for this purpose!

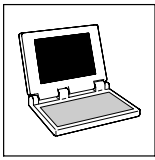
# System bus (CAN) for Lenze PLC devices

CAN system blocks  
CAN1\_IO (node number: 31) - ECSxA



## Variables for received user data

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CAN1_blnB0_b	CAN1_nlnW0_a	CAN1_dnlnD1_p
	...	...		
7	CAN1_blnB7_b			
2	0	CAN1_blnB8_b		
	...	...		
7	CAN1_blnB15_b			
3	0	CAN1_blnB16_b	CAN1_nlnW1_a	
	...	...		
7	CAN1_blnB23_b			
4	0	CAN1_blnB24_b		CAN1_nlnW2_a
	...	...		
7	CAN1_blnB31_b			
5	0	CAN1_blnB32_b	CAN1_nlnW3_a	
	...	...		
7	CAN1_blnB39_b			
6	0	CAN1_blnB40_b		CAN1_nlnW0_a
	...	...		
7	CAN1_blnB47_b			
7	0	CAN1_blnB48_b	CAN1_nlnW1_a	
	...	...		
7	CAN1_blnB55_b			
8	0	CAN1_blnB56_b		
	...	...		
7	CAN1_blnB63_b			



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

### CAN2\_IO (node number: 32)

## 7.4 CAN2\_IO (node number: 32)

This SB serves to transmit event-controlled or time-controlled process data via the system bus.

- The setting of the transmission mode (event- or time-controlled) is effected via C0356. (☞ 3-6)
- A sync telegram is not required.

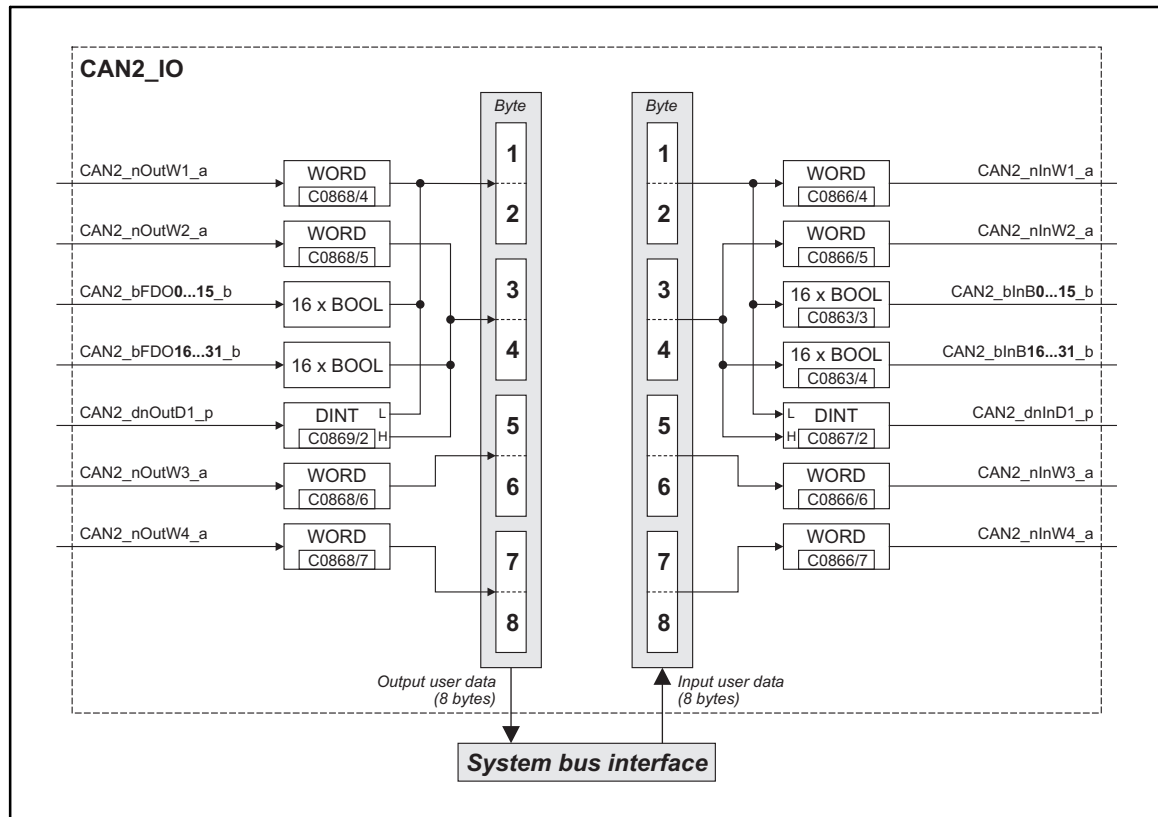


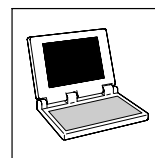
Fig. 7-4 CAN2\_IOsystem block



### Tip!

Via C0357/2 you can set the monitoring time for the data reception. (Lenze setting: 3000 ms)

- Further information on this subject can be found in chapter 3.12.1. (☞ 3-11)



### 7.4.1 Inputs\_CAN2

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN2_nInW1_a	Integer	Analog	%IW32.0	C0866/4	dec [%]	
CAN2_nInW2_a			%IW32.1	C0866/5		
CAN2_bInB0_b	Bool	Binary	%IX32.0.0	C0863/3	hex	
...			...			
CAN2_bInB15_b			%IX32.0.15			
CAN2_bInB16_b			%IX32.1.0	C0863/4		
...	...					
CAN2_bInB31_b	%IX32.1.15					
CAN2_dnInD1_p	Double integer	Position	%ID32.0	C0867/2	dec [inc]	
CAN2_nInW3_a	Integer	Analog	%IW32.2	C0866/6	dec [%]	
CAN2_nInW4_a			%IW32.3	C0866/7		

### 7.4.2 Outputs\_CAN2

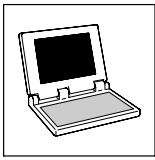
Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN2_nOutW1_a	Integer	Analog	%QW32.0	C0868/4	dec [%]	
CAN2_nOutW2_a			%QW32.1	C0868/5		
CAN2_bFD00_b	Bool	Binary	%QX32.0.0	C0151/2	hex	Display code in hex as double word
...			...			
CAN2_bFD015_b			%QX32.0.15			
CAN2_bFD016_b			%QX32.1.0	C0151/2		
...	...					
CAN2_bFD031_b	%QX32.1.15					
CAN2_dnOutD1_p	Double integer	Position	%QD32.0	C0869/2	dec [inc]	
CAN2_nOutW3_a	Integer	Analog	%QW32.2	C0868/6	dec [%]	
CAN2_nOutW4_a			%QW32.3	C0868/7		

### 7.4.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11 bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN2\_IO (node number: 32)

### 7.4.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables			
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)	
1	0	CAN2_bFD00_b	CAN2_nOutW1_a	CAN2_dnOutD1_p	
	...	...			
2	7	CAN2_bFD07_b			
	0	CAN2_bFD08_b			
3	...	...			CAN2_nOutW2_a
	7	CAN2_bFD015_b			
4	0	CAN2_bFD016_b			
	...	...			
5	7	CAN2_bFD023_b	CAN2_nOutW3_a		
	0	CAN2_bFD024_b			
6	...	...	CAN2_nOutW4_a		
	7	CAN2_bFD031_b			
7	0...7				
	0...7				
8	0...7				
	0...7				



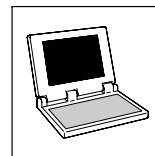
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 1 and 2, only use the variable *CAN2\_dnOutD1\_p*, *CAN2\_nOutW2\_a*, or only the variables *CAN2\_bFD00\_b* ... *CAN2\_bFD015\_b* for this purpose!

#### Variables for received user data

User data		Assigned variables			
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)	
1	0	CAN2_bInB0_b	CAN2_nInW1_a	CAN2_dnInD1_p	
	...	...			
2	7	CAN2_bInB7_b			
	0	CAN2_bInB8_b			
3	...	...			CAN2_nInW2_a
	7	CAN2_bInB15_b			
4	0	CAN2_bInB16_b			
	...	...			
5	7	CAN2_bInB23_b	CAN2_nInW3_a		
	0	CAN2_bInB24_b			
6	...	...	CAN2_nInW4_a		
	7	CAN2_bInB31_b			
7	0...7				
	0...7				
8	0...7				
	0...7				



## 7.5 CAN3\_IO (node number: 33)

This SB serves to transmit event-controlled or time-controlled process data via the system bus.

- The setting of the transmission mode (event- or time-controlled) is effected via C0356. (☞ 3-6)
- A sync telegram is not required.

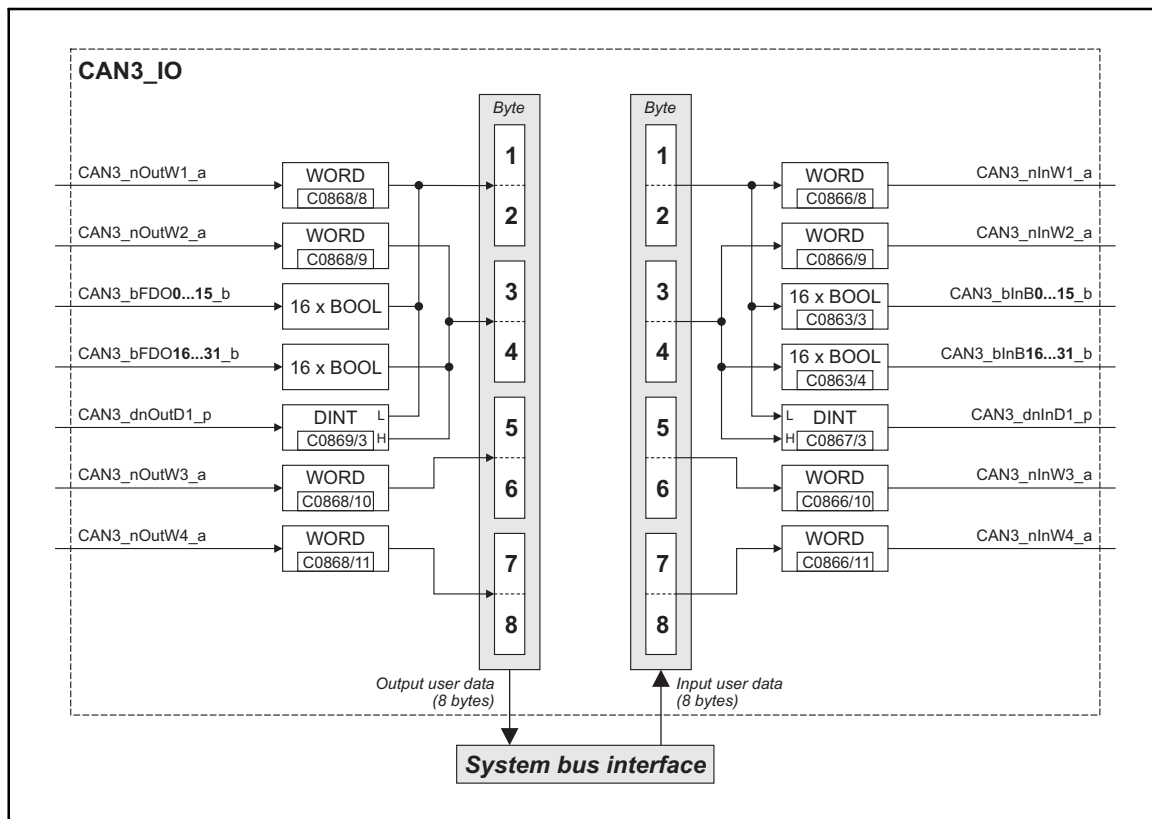


Fig. 7-5

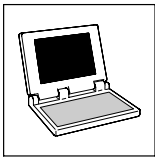
CAN3\_IOsystem block



### Tip!

Via C0357/3 you can set the monitoring time for the data reception. (Lenze setting: 3000 ms)

- Further information on this subject can be found in chapter 3.12.1. (☞ 3-11)



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

CAN3\_IO (node number: 33)

### 7.5.1 Inputs\_CAN3

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN3_nInW1_a	Integer	Analog	%IW33.0	C0866/8	dec [%]	
CAN3_nInW2_a			%IW33.1	C0866/9		
CAN3_bInB0_b	Bool	Binary	%IX33.0.0	C0863/5	hex	
...			...			
CAN3_bInB15_b			%IX33.0.15			
CAN3_bInB16_b			%IX33.1.0	C0863/6		
...	...					
CAN3_bInB31_b	%IX33.1.15					
CAN3_dnInD1_p	Double integer	Position	%ID33.0	C0867/3	dec [inc]	
CAN3_nInW3_a	Integer	Analog	%IW33.2	C0866/10	dec [%]	
CAN3_nInW4_a			%IW33.3	C0866/11		

### 7.5.2 Outputs\_CAN3

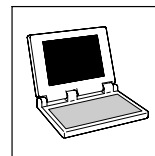
Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN3_nOutW1_a	Integer	Analog	%QW33.0	C0868/8	dec [%]	
CAN3_nOutW2_a			%QW33.1	C0868/9		
CAN3_bFD00_b	Bool	Binary	%QX33.0.0	C0151/3	hex	Display code in hex as double word
...			...			
CAN3_bFD015_b			%QX33.0.15			
CAN3_bFD016_b			%QX33.1.0	C0151/3		
...	...					
CAN3_bFD031_b	%QX33.1.15					
CAN3_dnOutD1_p	Double integer	Position	%QD33.0	C0869/3	dec [inc]	
CAN3_nOutW3_a	Integer	Analog	%QW33.2	C0868/10	dec [%]	
CAN3_nOutW4_a			%QW33.3	C0868/11		

### 7.5.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)



### 7.5.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables			
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)	
1	0	CAN3_bFDO0_b	CAN3_nOutW1_a	CAN3_dnOutD1_p	
	...	...			
2	7	CAN3_bFDO7_b			
	0	CAN3_bFDO8_b			
3	...	...			CAN3_nOutW2_a
	7	CAN3_bFDO15_b			
4	0	CAN3_bFDO16_b			
	...	...			
5	7	CAN3_bFDO23_b			
	0	CAN3_bFDO24_b	CAN3_nOutW3_a		
6	...	...			
	7	7	CAN3_bFDO31_b	CAN3_nOutW4_a	
0...7					
8	0...7				



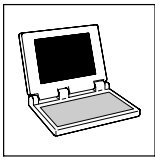
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 1 and 2, only use the variable *CAN3\_dnOutD1\_p*, *CAN3\_nOutW1\_a*, or only the variables *CAN3\_bFDO0\_b* ... *CAN3\_bFDO15\_b* for this purpose!

#### Variables for received user data

User data		Assigned variables			
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)	
1	0	CAN3_bInB0_b	CAN3_nInW1_a	CAN3_dnInD1_p	
	...	...			
2	7	CAN3_bInB7_b			
	0	CAN3_bInB8_b			CAN3_nInW2_a
3	...	...			
	4	7			
0		CAN3_bInB16_b	CAN3_nInW3_a		
5	...	...			
	6	7	CAN3_bInB23_b	CAN3_nInW4_a	
0		CAN3_bInB24_b			
7	...	...			
	8	7	CAN3_bInB31_b		
0...7					
8	0...7				



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

### CAN\_Management (node number: 101)

## 7.6 CAN\_Management (node number: 101)

By using this SB

- a **reset node** can be activated, e. g. to accept changes with regard to the baud rate and addressings.
- **Communication error, Bus-off state**, and further states can be processed in the PLC program.
- the instant of transmission of CAN2\_OUT and CAN3\_OUT can be influenced.

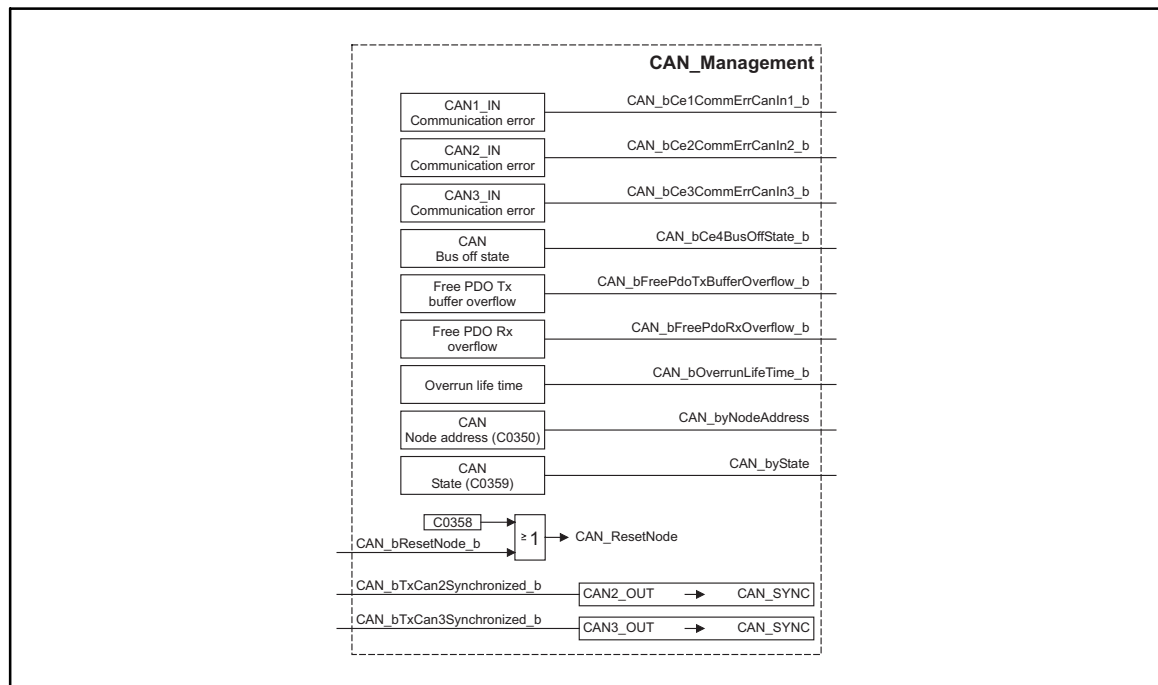


Fig. 7-6 CAN\_Management system block

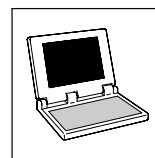


### Note!

The process image for this SB is generated in a fixed system task (interval: 1 ms).

### 7.6.1 Inputs\_CAN\_Management

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN_bCe1CommErrCanIn1_b	Bool	Binary	%IX101.0.0	-	-	CAN1_IN communication error
CAN_bCe2CommErrCanIn2_b			%IX101.0.1			CAN2_IN communication error
CAN_bCe3CommErrCanIn3_b			%IX101.0.2			CAN3_IN communication error
CAN_bCe4BusOffState_b			%IX101.0.3			CAN bus "off state" recognised
CAN_bFreePdoTxBufferOverflow_b			%IX101.0.4			Transmit request memory overflow
CAN_bFreePdoRxOverflow_b			%IX101.0.5			Receive memory overflow
CAN_bOverrunLifeTime_b			%IX101.0.6			"Node life time" exceeded
CAN_byNodeAddress	Byte	-	%IB101.2	C0350	-	CAN mode address
CAN_byState			%IB101.3	C0359	-	CAN status



### 7.6.2 Outputs\_CAN\_Management

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN_bResetNode_b	Bool	Binary	%QX101.0.0	-	-	Carry out reset node of the PLC
CAN_bTxCan2Synchronized_b			%QX101.0.1			Transmit CAN2_OUT with sync telegram.
CAN_bTxCan3Synchronized_b			%QX101.0.2			Transmit CAN3_OUT with sync telegram.



#### Note!

If *CAN\_bTxCan2Synchronized\_b* and/or *CAN\_bTxCan3Synchronized\_b* are integrated, it is required to configure the device as sync transmitter. The data are transmitted immediately after the sync signal is sent.

### 7.6.3 Activating a reset node

A reset node is activated by setting *CAN\_bResetNode\_b* to TRUE or C0358 = 1.



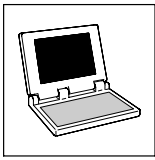
#### Tip!

Even if the **CAN\_Management** SB has not been assigned to the control configuration, a reset node can be activated via C0358. (☞ 3-8)

### 7.6.4 Defining the instant of transmission for CAN2\_OUT/CAN3\_OUT

Via *CAN\_bTxCan2Synchronized\_b* and *CAN\_bTxCan3Synchronized\_b* you define the instant of transmission for the CAN objects CAN2\_OUT and CAN3\_OUT:

- **FALSE:** Data from CAN2\_OUT/CAN3\_OUT are sent at the end of the process image.
- **TRUE:** Data from CAN2\_OUT/CAN3\_OUT are sent to sync.
  - The identifiers for the sync transmission C0369 and reception telegram can be set via C0367/C0368.
  - The *sync Tx time* can be set via C0369.
  - Detailed information on this can be found in the description of the **CAN\_Synchronization** SB. (☞ 7-23)



# System bus (CAN) for Lenze PLC devices

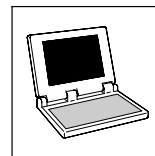
## CAN system blocks

CAN\_Management (node number: 101)

### 7.6.5 Status messages

The **CAN\_Management** SB provides different status messages which can be processed in the PLC program:

Variable	Description
CAN_bCe1CommErrCanIn1_b	TRUE CAN1_IN communication error
CAN_bCe2CommErrCanIn1_b	TRUE CAN2_IN communication error
CAN_bCe3CommErrCanIn1_b	TRUE CAN3_IN communication error
CAN_bCe4BusOffState_b	TRUE CAN bus "off state" recognised
CAN_bFreePdoTxBufferOverflow_b	Free CAN objects
	TRUE Overflow of send order memory • See <b>L_CanPdoTransmit</b> FB. (☞ 10-8)
CAN_bFreePdoRxOverflow_b	Free CAN objects
	TRUE Overflow of the receive memory • See <b>L_CanPdoReceive</b> FB. (☞ 10-12)
CAN_bOverrunLifeTime_b	CAN "Node guarding" monitoring mechanism
	TRUE The "Node life time" has been exceeded • See chapter 2.10, "monitoring mechanisms". (☞ 2-21)
CAN_byNodeAddress	1...63 CAN node address (☞ 3-3)
CAN_byState	Operating status of the system bus
	1 Operational
	2 Pre-operational
	3 Warning
	4 Bus-off



## 7.7 CAN\_Synchronization (node number: 102)

This SB can be used to synchronise the internal time base of the controller to the instant of reception of the sync telegram or of a terminal signal. Thus, the start of cyclic and time-controlled internal processes (e. g. data transfer from tasks to the DCTRL function block) is effected in a synchronous manner for all controllers involved in the synchronisation.

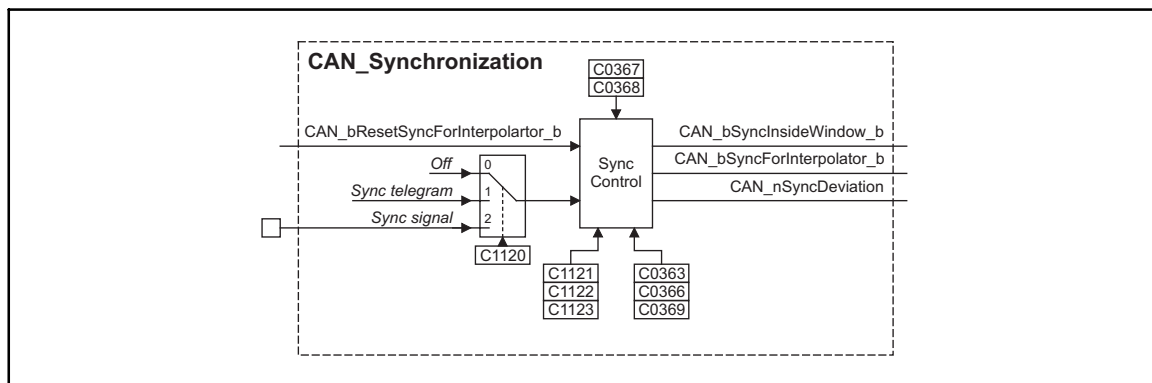


Fig. 7-7 CAN\_Synchronizationsystem block

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CAN_bSyncInsideWindow_b	Bool	Binary	%IX102.0.0	-	-	TRUE: Sync telegram/signal within the time slot (C1123) FALSE: <ul style="list-style-type: none"> <li>Exit synchronicity</li> <li>No sync telegram/signal</li> <li>Time slot (C1123) too small</li> </ul>
CAN_bSyncForInterpolator_b			%IX102.0.1			TRUE: Sync telegram/signal recognised
CAN_nSyncDeviation	Integer	-	%IX102.1			Current sync deviation
CAN_bResetSyncForInterpolator_b	Bool	Binary	%QX102.0.0			TRUE: Reset CAN_bSyncForInterpolator_b

### Operating mode

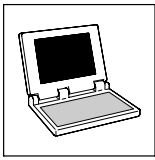
Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
[C1120]	Sync mode	0	0 Off 1 Synchronisation via system bus (CAN) 2 Synchronisation via terminal	<b>CAN sync. source</b> Synchronisation deactivated. Synchronisation by sync telegram via system bus. Synchronisation by sync signal via terminal: <ul style="list-style-type: none"> <li>9300 Servo PLC: X5/E5</li> <li>Drive PLC: X3/I1</li> <li>ECSxA: X6/DI1</li> </ul>

### Axis synchronisation via system bus (CAN)

The system bus transmits both the sync telegram and the process signals.

Example for applications:

- Specification of cyclic, synchronised position setpoint information for multi-axis applications via the system bus.



## System bus (CAN) for Lenze PLC devices

### CAN system blocks

CAN\_Synchronization (node number: 102)

#### Axis synchronisation via terminal

The transmission paths for the sync signal and the process signals are separated.

- The process signals are applied via a freely selectable input channel (e. g. AIF interface, DF input).
- The sync signal is injected via terminal:

PLC	Terminal for sync signal
9300 Servo PLC	X5/E5
Drive PLC	X3/I1
ECSxA	X6/DI1



#### Note!

For the synchronisation via terminal, in addition to the SB **CAN\_Synchronization** the SB **DIGITAL\_IO** has to be included in the control configuration of the Drive PLC Developer Studio.

Examples for applications:

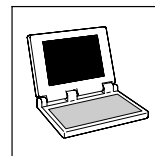
- Specification of cyclic, synchronised position setpoint information for multi-axis applications via other bus systems (e. g. INTERBUS).
- Synchronisation of the internal process cycles to higher-level process controls.

#### Synchronisation time

After mains power-up and the initialisation time of the PLC, an additional period is required for synchronisation.

The synchronisation time depends on

- the baud rate of the system bus
- the starting time (arrival of the first sync telegram/signal)
- the interval of the sync telegrams/signals
- the sync correction factor (C0363)
- the operating mode (C1120)



### Synchronisation cycle

The controllers/PLCs receive the sync telegram/signal and compare the time between two LOW-HIGH edges of the signal to the specified cycle time (C1121).

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
[C1121]	Sync cycle	2	1 {1 ms}	13 <b>Synchronisation cycle</b> Definition of the cycle time of the sync telegram/signal. • Parameterisation is only required for the slave!

- The value set in C1121 is the time between two sync telegrams/signals of the master.

### Phase displacement

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
[C1122]	Sync phase	0.460	0 {0.001 ms}	6.5 <b>Synchronisation phase</b> Phase shift between the sync telegram/signal and the start of the internal control program.

### Monitoring the synchronisation (time slot)

The variable `CAN_bSyncInsideWindow_b` can be used for monitoring the synchronisation.

Code	LCD	Possible settings		IMPORTANT
		Lenze	Selection	
[C1123]	Sync window	0	0 {0.001 ms}	6.5 <b>Synchronisation window</b> If the sync telegram/signal sent by the master is in this "time slot", <code>CAN_bSyncInsideWindow_b</code> switches to TRUE.

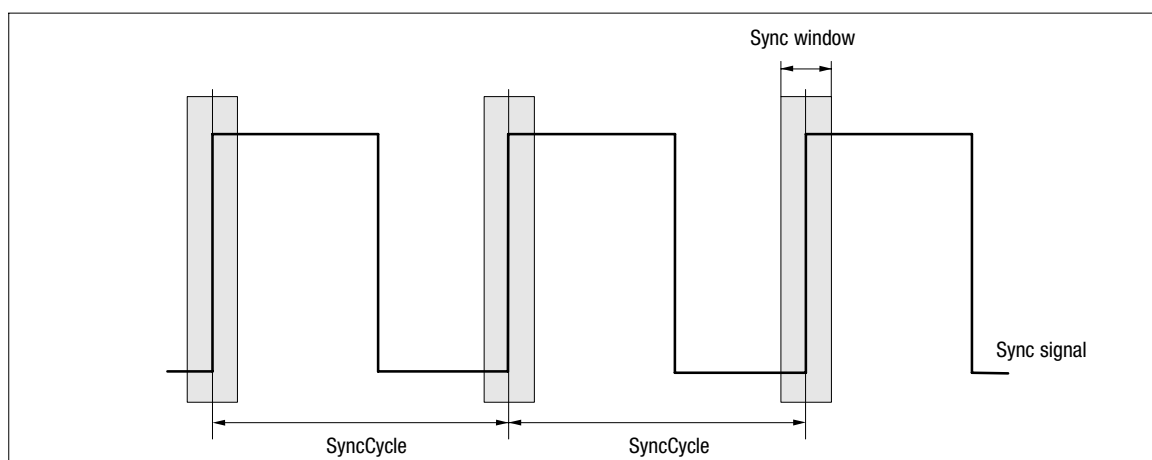
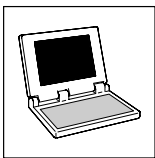


Fig. 7-8

"Time slot" for the LOW-HIGH edges of the sync signal



# System bus (CAN) for Lenze PLC devices

## CAN system blocks

### CAN\_Synchronization (node number: 102)



#### Tip!

A jitter\* up to  $\pm 200 \mu\text{s}$  on the LOW-HIGH edges of the sync signal is permissible. The extent of the jitter has an effect on the parameter setting of the "time slot".

\* Jitter refers to the phase shiftings and therefore to the periodic changes of signal frequencies. They are variations of fixed points in time (e. g. the time of the transition from one signal amplitude to another) of a digital signal. Jitter particularly occurs with high frequencies and can result in data losses.

#### Correction value of the phase controller

Code	LCD	Possible settings		IMPORTANT		
		Lenze	Selection			
C0363	Sync corr	1	9300 Servo PLC:	Drive PLC:	<b>CAN sync correction increment</b> Alter the correction value until <i>CAN_nSyncDeviation</i> has reached a minimum.	
			1	0.2 $\mu\text{s}/\text{ms}$		0.4 $\mu\text{s}/\text{ms}$
			2	0.4 $\mu\text{s}/\text{ms}$		0.8 $\mu\text{s}/\text{ms}$
			3	0.6 $\mu\text{s}/\text{ms}$		1.2 $\mu\text{s}/\text{ms}$
			4	0.8 $\mu\text{s}/\text{ms}$		1.6 $\mu\text{s}/\text{ms}$
			5	1.0 $\mu\text{s}/\text{ms}$		2.0 $\mu\text{s}/\text{ms}$

#### CAN sync response

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C0366	Sync response	1	0	No response	<b>CAN sync response</b> No response PLC responds to a sync telegram by sending the CAN1_OUT object.
			1	Response to sync	

#### CAN sync identifier

Transmit or receive identifier of the sync telegram

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C0367	Sync Rx Id	128	1 {1}	256	<b>CAN sync Rx identifier</b> Receive identifier of the sync telegram.
C0368	Sync Tx Id	128	1 {1}	256	<b>CAN sync Tx identifier</b> Send identifier of the sync telegram.

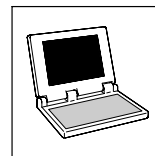
#### CAN sync Tx transmission cycle

Code	LCD	Possible settings		IMPORTANT	
		Lenze	Selection		
C0369	Sync Tx time	0	0 {1} 0 = Off	65000	<b>CAN sync transmission telegram cycle</b> A sync telegram with the identifier of C0368 is sent with the set cycle time.

# System bus (CAN) for Lenze PLC devices

## CAN system blocks

### CAN\_Synchronization (node number: 102)



#### Configuration example: synchronisation via system bus (CAN)

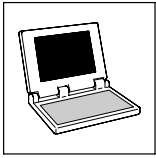
When carrying out the commissioning, observe the following sequence:

Location	Step	Information	
All devices	1.	Commission controller/PLC and system bus.	
	2.	Inhibit controller/PLC.	
DDS	3.	Integrate <b>CAN_Synchronization</b> SB into the control configuration.	
Slave devices	4.	Connect <i>CAN_bSynclnsideWindow_b</i> to a digital output.	
	5.	C1120 = 1	Synchronisation by sync telegram via system bus active.
	6.	C0366 = 1 (Lenze setting)	CAN sync response: Slave devices respond to sync telegram.
Master	7.	Define order of the telegrams (identifiers): 1. Send new setpoint to all slaves 2. Send sync telegram 3. Receive response of all slaves	
	8.	Start communication / send sync telegrams.	
Slave devices	9.	Read C0362 from the master.	Query cycle time of the sync telegram from the master.
	10.	Set C1121 in accordance with C0362 from the master.	Adjust interval of the sync telegrams to be received to the cycle time of the master.
	11.	Set C1123.	Set optimum size for the "time slot". • If the sync signal "jitters" heavily, increase "time slot".
	12.	Enable controller/PLC via the signal <i>CAN_bSynclnsideWindow_b</i> applied to the digital output.	Monitor the synchronisation. • If <i>CAN_bSynclnsideWindow_b</i> = TRUE, enable controller/PLC.

#### Configuration example: synchronisation via terminal

When carrying out the commissioning, observe the following sequence:

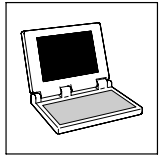
Location	Step	Information	
All devices	1.	Commission controller/PLC and system bus.	
	2.	Inhibit controller/PLC.	
DDS	3.	Integrate <b>CAN_Synchronization</b> SB into the control configuration.	
Slave devices	4.	Connect <i>CAN_bSynclnsideWindow_b</i> to a digital output.	
	5.	Apply sync signal of the master to terminal.	9300 Servo PLC: terminal X5/E5 Drive PLC: terminal X3/I1 ECSxA: X6/DI1
Slave devices	6.	C1120 = 2	Synchronisation by sync signal via terminal active.
Slave devices	7.	C0366 = 1 (Lenze setting)	CAN sync response: Slave devices respond to sync telegram.
Master	8.	Start communication, send sync telegrams.	
Slave devices	9.	Read C0362 from the master.	Query cycle time of the sync signal from the master.
	10.	Set C1121 in accordance with C0362 from the master.	Adjust interval of the sync signals to be received to the cycle time of the master.
	11.	Set C1123.	Set optimum size for the "time slot". • If the sync signal "jitters" heavily, increase "time slot".
	12.	Enable controller/PLC via the signal <i>CAN_bSynclnsideWindow_b</i> applied to the digital output.	Monitor the synchronisation. • If <i>CAN_bSynclnsideWindow_b</i> = TRUE, enable controller/PLC.



## ***System bus (CAN) for Lenze PLC devices***

***CAN system blocks***

***CAN\_Synchronization (node number: 102)***



## 8 FIF-CAN system blocks (only Drive PLC)

### 8.1 FIF\_CAN1\_IO (node number: 34)

This SB serves to the transmission of cyclic process data via the function interface of the Drive PLC.

- For the transmission a sync telegram is required, which has to be generated by a **different** node. (📖 2-9)

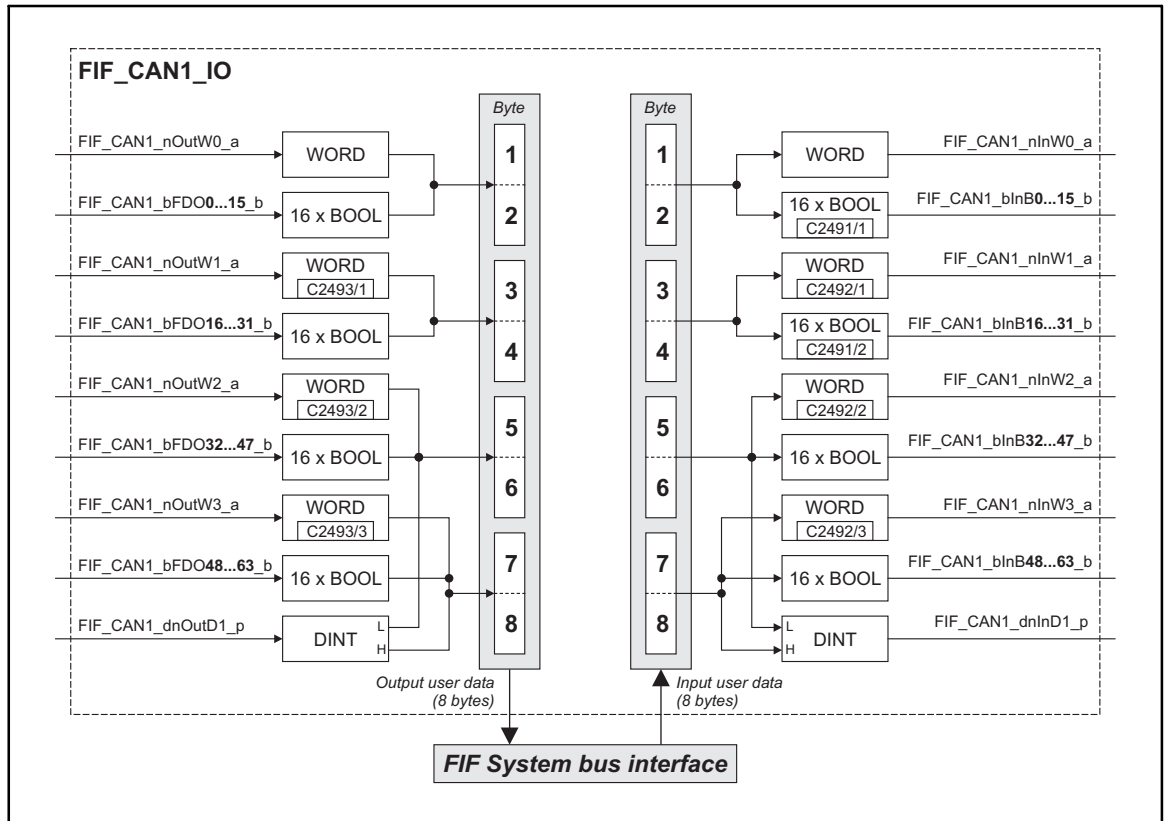
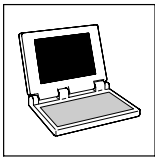


Fig. 8-1

FIF\_CAN1\_IO system block



# System bus (CAN) for Lenze PLC devices

## FIF-CAN system blocks

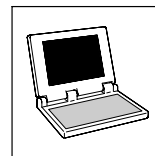
FIF\_CAN1\_IO (node number: 34)

### 8.1.1 FIF\_Inputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN1_nInW0_a	Integer	Analog	%IW34.0			
FIF_CAN1_bInB0_b	Bool	Binary	%IX34.0.0	C2491/1	hex	
...			...			
FIF_CAN1_bInB15_b			%IX34.0.15			
FIF_CAN1_nInW1_a	Integer	Analog	%IW34.1	C2492/1	dec [%]	
FIF_CAN1_bInB16_b	Bool	Binary	%IX34.1.0	C2491/2	hex	
...			...			
FIF_CAN1_bInB31_b			%IX34.1.15			
FIF_CAN1_nInW2_a	Integer	Analog	%IW34.2	C2492/2	dec [%]	
FIF_CAN1_bInB32_b	Bool	Binary	%IX34.2.0			
...			...			
FIF_CAN1_bInB47_b			%IX34.2.15			
FIF_CAN1_nInW3_a	Integer	Analog	%IW34.3	C2492/3	dec [%]	
FIF_CAN1_bInB48_b	Bool	Binary	%IX34.3.0			
...			...			
FIF_CAN1_bInB63_b			%IX34.3.15			
FIF_CAN1_dnInD1_p	Double integer	Position	%ID34.1			

### 8.1.2 FIF\_Outputs\_CAN1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN1_nOutW0_a	Integer	Analog	%QW34.0	-	-	
FIF_CAN1_bFD00_b	Bool	Binary	%QX34.0.0	-	-	
...			..			
FIF_CAN1_bFD015_b			%QX34.0.15			
FIF_CAN1_nOutW1_a	Integer	Analog	%QW34.1	C2493/1	dec [%]	
FIF_CAN1_bFD016_b	Bool	Binary	%QX34.1.0	-	-	
...			..			
FIF_CAN1_bFD031_b			%QX34.1.15			
FIF_CAN1_nOutW2_a	Integer	Analog	%QW34.2	C2493/2	dec [%]	
FIF_CAN1_bFD032_b	Bool	Binary	%QX34.2.0	-	-	
...			..			
FIF_CAN1_bFD047_b			%QX34.2.15			
FIF_CAN1_nOutW3_a	Integer	Analog	%QW34.3	C2493/3	dec [%]	
FIF_CAN1_bFD048_b	Bool	Binary	%QX34.3.0	-	-	
...			..			
FIF_CAN1_bFD063_b			%QX34.3.15			
FIF_CAN1_dnOutD1_p	Double integer	Position	%QD34.1	-	-	



## 8.1.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (☞ 2-3)

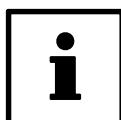
## 8.1.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

### Variables for user data to be transmitted

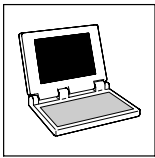
User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	FIF_CAN1_bFDO0_b	FIF_CAN1_nOutW0_a	
	...	...		
2	7	FIF_CAN1_bFDO7_b		
	...	...		
3	0	FIF_CAN1_bFDO8_b	FIF_CAN1_nOutW1_a	
	...	...		
4	7	FIF_CAN1_bFDO15_b		
	...	...		
5	0	FIF_CAN1_bFDO16_b	FIF_CAN1_nOutW2_a	
	...	...		
6	7	FIF_CAN1_bFDO23_b		
	...	...		
7	0	FIF_CAN1_bFDO24_b	FIF_CAN1_nOutW3_a	
	...	...		
8	7	FIF_CAN1_bFDO31_b		
	...	...		
5	0	FIF_CAN1_bFDO32_b	FIF_CAN1_nOutW2_a	FIF_CAN1_dnOutD1_p
	...	...		
6	7	FIF_CAN1_bFDO39_b		
	...	...		
7	0	FIF_CAN1_bFDO40_b	FIF_CAN1_nOutW3_a	
	...	...		
8	7	FIF_CAN1_bFDO47_b		
	...	...		
5	0	FIF_CAN1_bFDO48_b	FIF_CAN1_nOutW3_a	
	...	...		
6	7	FIF_CAN1_bFDO55_b		
	...	...		
7	0	FIF_CAN1_bFDO56_b	FIF_CAN1_nOutW3_a	
	...	...		
8	7	FIF_CAN1_bFDO63_b		
	...	...		



### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write the bytes 5 and 6, only use the variable *FIF\_CAN1\_dnOutD1\_p*, *FIF\_CAN1\_nOutW2\_a*, or only the variables *FIF\_CAN1\_bFDO32\_b ... FIF\_CAN1\_bFDO47\_b* for this purpose!



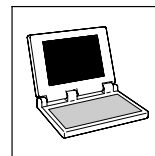
# System bus (CAN) for Lenze PLC devices

## FIF-CAN system blocks

FIF\_CAN1\_IO (node number: 34)

### Variables for received user data

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	FIF_CAN1_blnB0_b	FIF_CAN1_nlnW0_a	FIF_CAN1_dlnD1_p
	...	...		
7	FIF_CAN1_blnB7_b			
2	0	FIF_CAN1_blnB8_b		
	...	...		
7	FIF_CAN1_blnB15_b			
3	0	FIF_CAN1_blnB16_b	FIF_CAN1_nlnW1_a	
	...	...		
7	FIF_CAN1_blnB23_b			
4	0	FIF_CAN1_blnB24_b		FIF_CAN1_nlnW2_a
	...	...		
7	FIF_CAN1_blnB31_b			
5	0	FIF_CAN1_blnB32_b	FIF_CAN1_nlnW3_a	
	...	...		
7	FIF_CAN1_blnB39_b			
6	0	FIF_CAN1_blnB40_b		FIF_CAN1_nlnW3_a
	...	...		
7	FIF_CAN1_blnB47_b			
7	0	FIF_CAN1_blnB48_b	FIF_CAN1_nlnW3_a	
	...	...		
7	FIF_CAN1_blnB55_b			
8	0	FIF_CAN1_blnB56_b		FIF_CAN1_nlnW3_a
	...	...		
7	FIF_CAN1_blnB63_b			



### 8.2 FIF\_CAN2\_IO (node number: 35)

This SB serves to the transmission of event- or time-controlled process data via the function interface of the Drive PLC.

- The setting of the transmission mode (event- or time-controlled) is effected via C2456. (📖 5-6)
- A sync telegram is not required.

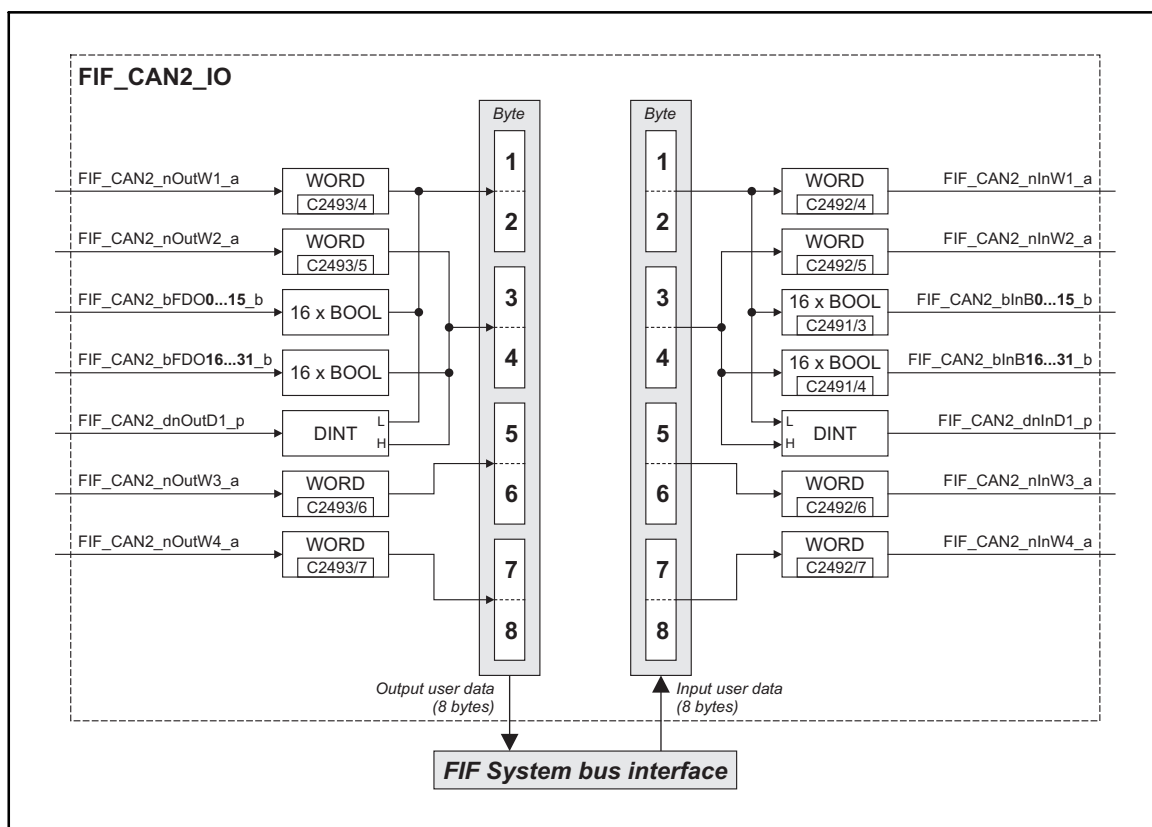


Fig. 8-2

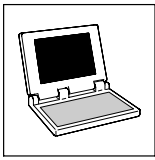
FIF\_CAN2\_IO system block



#### Tip!

Via code C2457/2 you can set the monitoring time for the data reception. (Lenze setting: 3000 ms)

- Further information on this subject can be found in chapter 5.10.1. (📖 5-9)



# System bus (CAN) for Lenze PLC devices

## FIF-CAN system blocks

FIF\_CAN2\_IO (node number: 35)

### 8.2.1 FIF\_Inputs\_CAN2

Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN2_nInW1_a	Integer	Analog	%IW35.0	C2492/4	dec [%]	
FIF_CAN2_nInW2_a			%IW35.1	C2492/5		
FIF_CAN2_bInB0_b	Bool	Binary	%IX35.0.0	C2491/3	hex	
...			...			
FIF_CAN2_bInB15_b			%IX35.0.15			
FIF_CAN2_bInB16_b			%IX35.1.0			
...			...	C2491/4		
FIF_CAN2_bInB31_b			%IX35.1.15			
FIF_CAN2_dnInD1_p	Double integer	Position	%ID35.0	-	-	
FIF_CAN2_nInW3_a	Integer	Analog	%IW35.2	C2492/6	dec [%]	
FIF_CAN2_nInW4_a			%IW35.3	C2492/7		

### 8.2.2 FIF\_Outputs\_CAN2

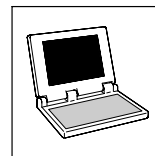
Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN2_nOutW1_a	Integer	Analog	%QW35.0	C2493/4	dec [%]	
FIF_CAN2_nOutW2_a			%QW35.1	C2493/5		
FIF_CAN2_bFD00_b	Bool	Binary	%QX35.0.0	-	-	
...			...			
FIF_CAN2_bFD015_b			%QX35.0.15			
FIF_CAN2_bFD016_b			%QX35.1.0			
...			...			
FIF_CAN2_bFD031_b			%QX35.1.15			
FIF_CAN2_dnOutD1_p	Double integer	Position	%QD35.0	-	-	
FIF_CAN2_nOutW3_a	Integer	Analog	%QW35.2	C2493/6	dec [%]	
FIF_CAN2_nOutW4_a			%QW35.3	C2493/7		

### 8.2.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)



### 8.2.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	FIF_CAN2_bFDO0_b	FIF_CAN2_nOutW1_a	FIF_CAN2_dnOutD1_p
	...	...		
	7	FIF_CAN2_bFDO7_b		
2	0	FIF_CAN2_bFDO8_b		
	...	...		
	7	FIF_CAN2_bFDO15_b		
3	0	FIF_CAN2_bFDO16_b	FIF_CAN2_nOutW2_a	
	...	...		
4	7	FIF_CAN2_bFDO23_b		
	0	FIF_CAN2_bFDO24_b		
5	...	...		
	7	FIF_CAN2_bFDO31_b		
6	0...7		FIF_CAN2_nOutW3_a	
7	0...7			
8	0...7		FIF_CAN2_nOutW4_a	



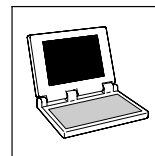
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write the bytes 1 and 2, only use the variable *FIF\_CAN2\_dnOutD1\_p*, *FIF\_CAN2\_nOutW1\_a*, or only the variables *FIF\_CAN2\_bFDO0\_b* ... *FIF\_CAN2\_bFDO15\_b* for this purpose!

#### Variables for received user data

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	FIF_CAN2_bInB0_b	FIF_CAN2_nInW1_a	FIF_CAN2_dnInD1_p
	...	...		
	7	FIF_CAN2_bInB7_b		
2	0	FIF_CAN2_bInB8_b		
	...	...		
	7	FIF_CAN2_bInB15_b		
3	0	FIF_CAN2_bInB16_b	FIF_CAN2_nInW2_a	
	...	...		
4	7	FIF_CAN2_bInB23_b		
	0	FIF_CAN2_bInB24_b		
5	...	...		
	7	FIF_CAN2_bInB31_b		
6	0...7		FIF_CAN2_nInW3_a	
7	0...7			
8	0...7		FIF_CAN2_nInW4_a	



### 8.3 FIF\_CAN3\_IO (node number: 36)

This SB serves to the transmission of event- or time-controlled process data via the function interface of the Drive PLC.

- The setting of the transmission mode (event- or time-controlled) is effected via C2456. (☞ 5-6)
- A sync telegram is not required.

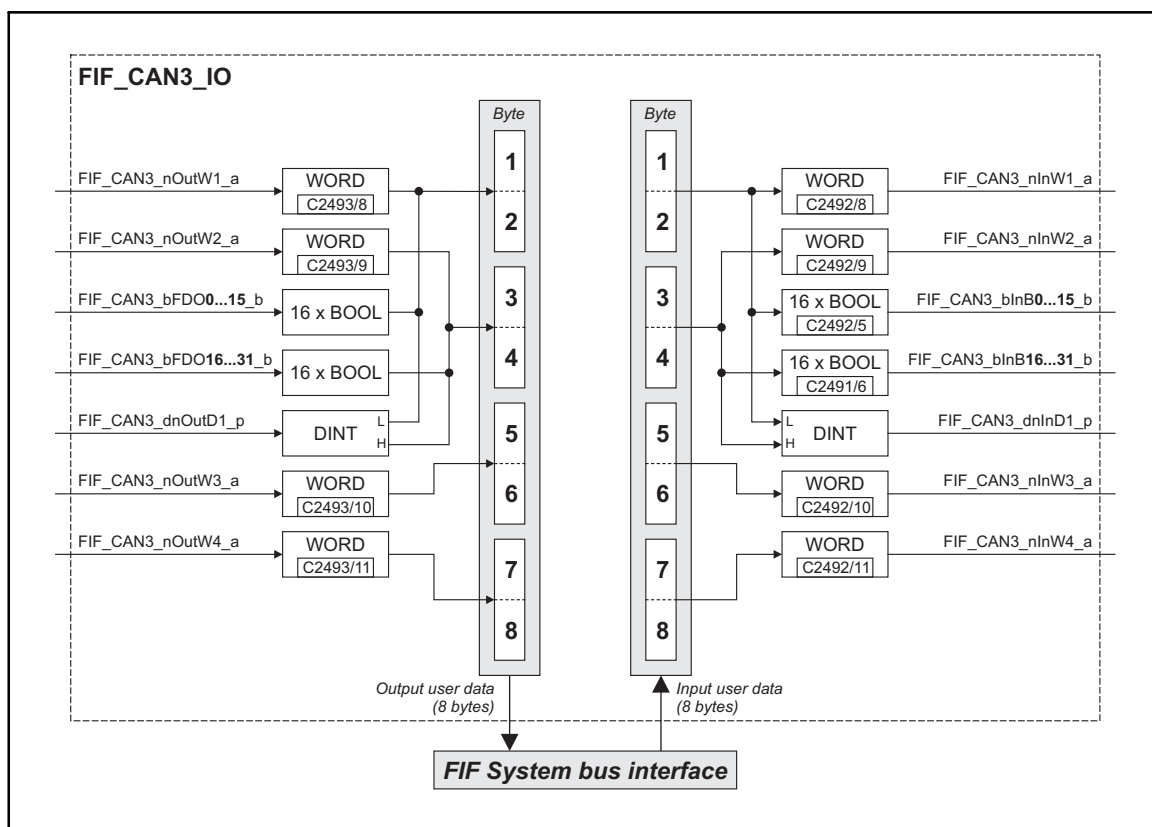


Fig. 8-3

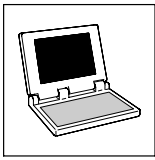
FIF\_CAN3\_IO system block



#### Tip!

Via code C2457/3 you can set the monitoring time for the data reception. (Lenze setting: 3000 ms)

- Further information on this subject can be found in chapter 5.10.1. (☞ 5-9)



# System bus (CAN) for Lenze PLC devices

## FIF-CAN system blocks

FIF\_CAN3\_IO (node number: 36)

### 8.3.1 FIF\_Inputs\_CAN3

Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN3_nInW1_a	Integer	Analog	%IW36.0	C2492/8	dec [%]	
FIF_CAN3_nInW2_a			%IW36.1	C2492/9		
FIF_CAN3_bInB0_b	Bool	Binary	%IX36.0.0	C2491/5	hex	
...			...			
FIF_CAN3_bInB15_b			%IX36.0.15			
FIF_CAN3_bInB16_b			%IX36.1.0			
...			...	C2491/6		
FIF_CAN3_bInB31_b			%IX36.1.15			
FIF_CAN3_dnInD1_p	Double integer	Position	%ID36.0	-	-	
FIF_CAN3_nInW3_a	Integer	Analog	%IW36.2	C2492/10	dec [%]	
FIF_CAN3_nInW4_a			%IW36.3	C2492/11		

### 8.3.2 FIF\_Outputs\_CAN3

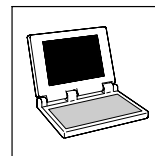
Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN3_nOutW1_a	Integer	Analog	%QW36.0	C2493/8	dec [%]	
FIF_CAN3_nOutW2_a			%QW36.1	C2493/9		
FIF_CAN3_bFD00_b	Bool	Binary	%QX36.0.0	-	-	
...			...			
FIF_CAN3_bFD015_b			%QX36.0.15			
FIF_CAN3_bFD016_b			%QX36.1.0			
...			...			
FIF_CAN3_bFD031_b			%QX36.1.15			
FIF_CAN3_dnOutD1_p	Double integer	Position	%QD36.0	-	-	
FIF_CAN3_nOutW3_a	Integer	Analog	%QW36.2	C2493/10	dec [%]	
FIF_CAN3_nOutW4_a			%QW36.3	C2493/11		

### 8.3.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)



### 8.3.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables				
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)		
1	0	FIF_CAN3_bFDO0_b	FIF_CAN3_nOutW1_a	FIF_CAN3_dnOutD1_p		
	...	...				
	7	FIF_CAN3_bFDO7_b				
2	0	FIF_CAN3_bFDO8_b			FIF_CAN3_nOutW2_a	FIF_CAN3_dnOutD1_p
	...	...				
	7	FIF_CAN3_bFDO15_b				
3	0	FIF_CAN3_bFDO16_b	FIF_CAN3_nOutW3_a	FIF_CAN3_dnOutD1_p		
	...	...				
	7	FIF_CAN3_bFDO23_b				
4	0	FIF_CAN3_bFDO24_b			FIF_CAN3_nOutW4_a	FIF_CAN3_dnOutD1_p
	...	...				
	7	FIF_CAN3_bFDO31_b				
5	0...7					
6	0...7					
7	0...7					
8	0...7					



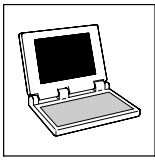
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write the bytes 1 and 2, only use the variable *FIF\_CAN3\_dnOutD1\_p*, *FIF\_CAN3\_nOutW1\_a*, or only the variables *FIF\_CAN3\_bFDO0\_b* ... *FIF\_CAN3\_bFDO15\_b* for this purpose!

#### Variables for received user data

User data		Assigned variables				
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)		
1	0	FIF_CAN3_bInB0_b	FIF_CAN3_nInW1_a	FIF_CAN3_dnInD1_p		
	...	...				
	7	FIF_CAN3_bInB7_b				
2	0	FIF_CAN3_bInB8_b			FIF_CAN3_nInW2_a	FIF_CAN3_dnInD1_p
	...	...				
	7	FIF_CAN3_bInB15_b				
3	0	FIF_CAN3_bInB16_b	FIF_CAN3_nInW3_a	FIF_CAN3_dnInD1_p		
	...	...				
	7	FIF_CAN3_bInB23_b				
4	0	FIF_CAN3_bInB24_b			FIF_CAN3_nInW4_a	FIF_CAN3_dnInD1_p
	...	...				
	7	FIF_CAN3_bInB31_b				
5	0...7					
6	0...7					
7	0...7					
8	0...7					



# System bus (CAN) for Lenze PLC devices

## FIF-CAN system blocks

### FIF\_CAN\_Management (node number: 111)

## 8.4 FIF\_CAN\_Management (node number: 111)

By means of this SB

- a **reset node** can be activated, e. g. to accept changes with regard to the baud rate and addressings.
- **Communication error, Bus-off state**, and further states can be processed in the PLC program.
- the instant of transmission of FIF-CAN2\_OUT and FIF-CAN3\_OUT can be influenced.

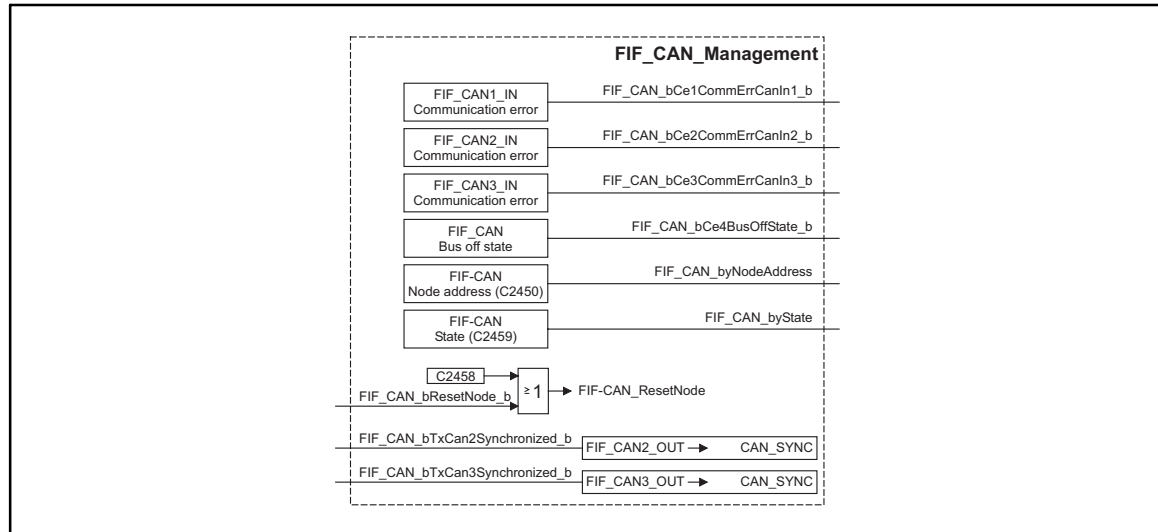


Fig. 8-4 FIF\_CAN\_Managementsystem block

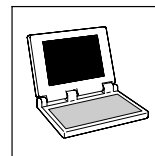


### Note!

The process image for this SB is generated in a fixed system task (interval: 1 ms).

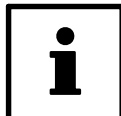
### 8.4.1 FIF\_Inputs\_CAN\_Management

Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN_bCe1CommErrCanIn1_b	Bool	Binary	%IX111.0.0	-	-	FIF-CAN1_IN communication error
FIF_CAN_bCe2CommErrCanIn2_b			%IX111.0.1			FIF-CAN2_IN communication error
FIF_CAN_bCe3CommErrCanIn3_b			%IX111.0.2			FIF-CAN3_IN communication error
FIF_CAN_bCe4BusOffState_b			%IX111.0.3			FIF-CAN-bus "off state" recognised
FIF_CAN_byNodeAddress	Byte	-	%IB111.2	C2450	-	FIF-CAN node address
FIF_CAN_byState			%IB111.3	C2459	-	FIF-CAN status



#### 8.4.2 FIF\_Outputs\_CAN\_Management

Variable	Data type	Signal type	Address	Display code	Display format	Notes
FIF_CAN_bResetNode_b	Bool	Binary	%QX111.0.0	-	-	Carry out reset node of the FIF-CAN
FIF_CAN_bTxCan2Synchronized_b			%QX111.0.1			Transmit FIF-CAN2_OUT with sync telegram.
FIF_CAN_bTxCan3Synchronized_b			%QX111.0.2			Transmit FIF-CAN_OUT with sync telegram.

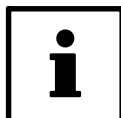


#### Note!

If *FIF\_CAN\_bTxCan2Synchronized\_b* and/or *FIF\_CAN\_bTxCan3Synchronized\_b* are integrated, it is required to configure the device as sync transmitter. The data are transmitted immediately after the sync telegram is sent.

#### 8.4.3 Activating a reset node

A reset node can be activated by setting *FIF\_CAN\_bResetNode\_b* to TRUE or C2458 = 1.



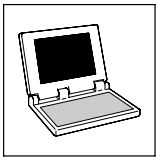
#### Tip!

Even if the **FIF\_CAN\_Management** SB has not been assigned to the control configuration, a reset node can be activated via C2458. (☞ 5-8)

#### 8.4.4 Defining the instant of transmission for FIF-CAN2\_OUT/FIF-CAN3\_OUT

Via *FIF\_CAN\_bTxCan2Synchronized\_b* and *FIF\_CAN\_bTxCan3Synchronized\_b* you define the instant of transmission for the CAN objects FIF-CAN2\_OUT and FIF-CAN3\_OUT:

- **FALSE:** Data from FIF-CAN2\_OUT/FIF-CAN3\_OUT are sent at the end of the process image.
- **TRUE:** Data from FIF-CAN2\_OUT/FIF-CAN3\_OUT are sent to sync.
  - The identifiers for the sync transmission and reception telegram can be set via C2467/C2468.
  - The *sync Tx time* can be set via C2469.



## System bus (CAN) for Lenze PLC devices

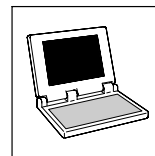
### FIF-CAN system blocks

FIF\_CAN\_Management (node number: 111)

#### 8.4.5 Status messages

The **FIF\_CAN\_Management** SB provides different status messages which can be processed in the PLC program:

Variable	Description
FIF_CAN_bCe1CommErrCanIn1_b	TRUE FIF-CAN1_IN communication error
FIF_CAN_bCe2CommErrCanIn1_b	TRUE FIF-CAN2_IN communication error
FIF_CAN_bCe3CommErrCanIn1_b	TRUE FIF-CAN3_IN communication error
FIF_CAN_bCe4BusOffState_b	TRUE FIF-CAN bus "off state" recognised
FIF_CAN_byNodeAddress	1...63 FIF-CAN node address (□ 3-3)
FIF_CAN_byState	FIFsystem busoperating status
	1 Operational
	2 Pre-operational
	3 Warning
	4 Bus-off



## 9 CAN-AUX system blocks (only ECSxA)

### 9.1 CANaux1\_IO (node number: 34)

This SB serves to transmit cyclic process data via the system bus.

- For the transmission a sync telegram is required, which has to be generated by a **different** node. (☒ 2-9)

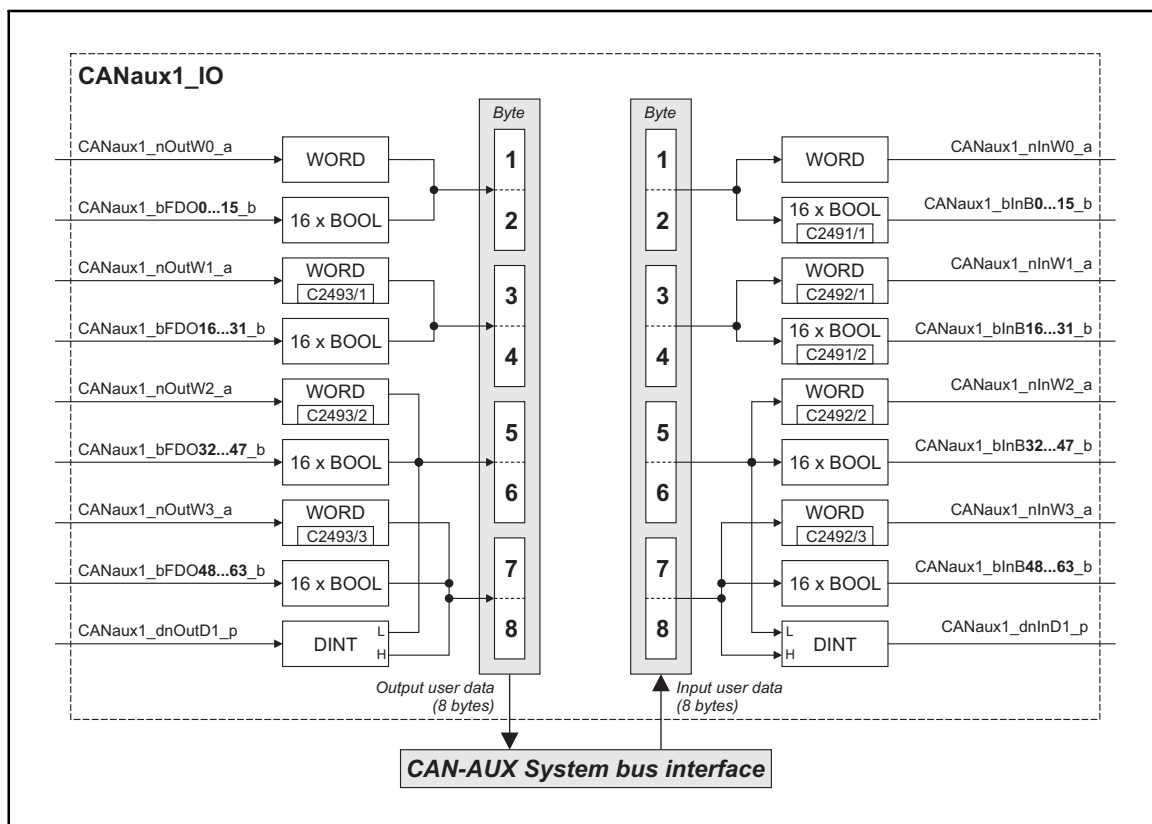
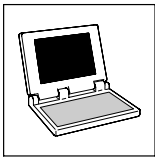


Fig. 9-1

CANaux1\_IOsystem block



# System bus (CAN) for Lenze PLC devices

## CAN-AUX system blocks

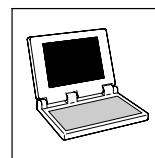
CANaux1\_IO (node number: 34)

### 9.1.1 Inputs\_CANaux1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CANaux1_nInW0_a	Integer	Analog	%IW34.0			
CANaux1_bInB0_b	Bool	Binary	%IX34.0.0	C2491/1	hex	
...			...			
CANaux1_bInB15_b			%IX34.0.15			
CANaux1_nInW1_a	Integer	Analog	%IW34.1	C2492/1	dec [%]	
CANaux1_bInB16_b	Bool	Binary	%IX34.1.0	C2491/2	hex	
...			...			
CANaux1_bInB31_b			%IX34.1.15			
CANaux1_nInW2_a	Integer	Analog	%IW34.2	C2492/2	dec [%]	
CANaux1_bInB32_b	Bool	Binary	%IX34.2.0			
...			...			
CANaux1_bInB47_b			%IX34.2.15			
CANaux1_nInW3_a	Integer	Analog	%IW34.3	C2492/3	dec [%]	
CANaux1_bInB48_b	Bool	Binary	%IX34.3.0			
...			...			
CANaux1_bInB63_b			%IX34.3.15			
CANaux1_dnInD1_p	Double integer	Position	%ID34.1			

### 9.1.2 Outputs\_CANaux1

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CANaux1_nOutW0_a	Integer	Analog	%QW34.0	-	-	
CANaux1_bFD00_b	Bool	Binary	%QX34.0.0	-	-	
...			..			
CANaux1_bFD015_b			%QX34.0.15			
CANaux1_nOutW1_a	Integer	Analog	%QW34.1	C2493/1	dec [%]	
CANaux1_bFD016_b	Bool	Binary	%QX34.1.0	-	-	
...			..			
CANaux1_bFD031_b			%QX34.1.15			
CANaux1_nOutW2_a	Integer	Analog	%QW34.2	C2493/2	dec [%]	
CANaux1_bFD032_b	Bool	Binary	%QX34.2.0	-	-	
...			..			
CANaux1_bFD047_b			%QX34.2.15			
CANaux1_nOutW3_a	Integer	Analog	%QW34.3	C2493/3	dec [%]	
CANaux1_bFD048_b	Bool	Binary	%QX34.3.0	-	-	
...			..			
CANaux1_bFD063_b			%QX34.3.15			
CANaux1_dnOutD1_p	Double integer	Position	%QD34.1	-	-	



### 9.1.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (☞ 2-3)

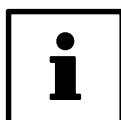
### 9.1.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

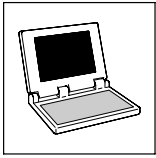
User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CANaux1_bFD00_b	CANaux1_nOutW0_a	
	...	...		
7	CANaux1_bFD07_b			
2	0	CANaux1_bFD08_b		
	...	...		
7	CANaux1_bFD015_b			
3	0	CANaux1_bFD016_b	CANaux1_nOutW1_a	
	...	...		
7	CANaux1_bFD023_b			
4	0	CANaux1_bFD024_b		
	...	...		
7	CANaux1_bFD031_b			
5	0	CANaux1_bFD032_b	CANaux1_nOutW2_a	
	...	...		
7	CANaux1_bFD039_b			
6	0	CANaux1_bFD040_b		CANaux1_dnOutD1_p
	...	...		
7	CANaux1_bFD047_b			
7	0	CANaux1_bFD048_b	CANaux1_nOutW3_a	
	...	...		
7	CANaux1_bFD055_b			
8	0	CANaux1_bFD056_b		
	...	...		
7	CANaux1_bFD063_b			



#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 5 and 6, only use the variable *CANaux1\_dnOutD1\_p*, *CANaux1\_nOutW2\_a*, or only the variables *CANaux1\_bFD032\_b ... CANaux1\_bFD047\_b* for this purpose!



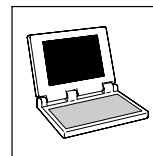
# System bus (CAN) for Lenze PLC devices

## CAN-AUX system blocks

CANaux1\_IO (node number: 34)

### Variables for received user data

User data		Assigned variables		
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)
1	0	CANaux1_blnB0_b	CANaux1_nlnW0_a	
	...	...		
7	CANaux1_blnB7_b			
2	0	CANaux1_blnB8_b		
	...	...		
7	CANaux1_blnB15_b			
3	0	CANaux1_blnB16_b	CANaux1_nlnW1_a	
	...	...		
7	CANaux1_blnB23_b			
4	0	CANaux1_blnB24_b		CANaux1_nlnW2_a
	...	...		
7	CANaux1_blnB31_b			
5	0	CANaux1_blnB32_b	CANaux1_nlnW3_a	
	...	...		
7	CANaux1_blnB39_b			
6	0	CANaux1_blnB40_b		CANaux1_dlnD1_p
	...	...		
7	CANaux1_blnB47_b			
7	0	CANaux1_blnB48_b	CANaux1_nlnW3_a	
	...	...		
7	CANaux1_blnB55_b			
8	0	CANaux1_blnB56_b		
	...	...		
7	CANaux1_blnB63_b			



### 9.2 CANaux2\_IO (node number: 35)

This SB serves to transmit event-controlled or time-controlled process data via the system bus.

- The setting of the transmission mode (event- or time-controlled) is effected via C2456. (☞ 3-6)
- A sync telegram is not required.

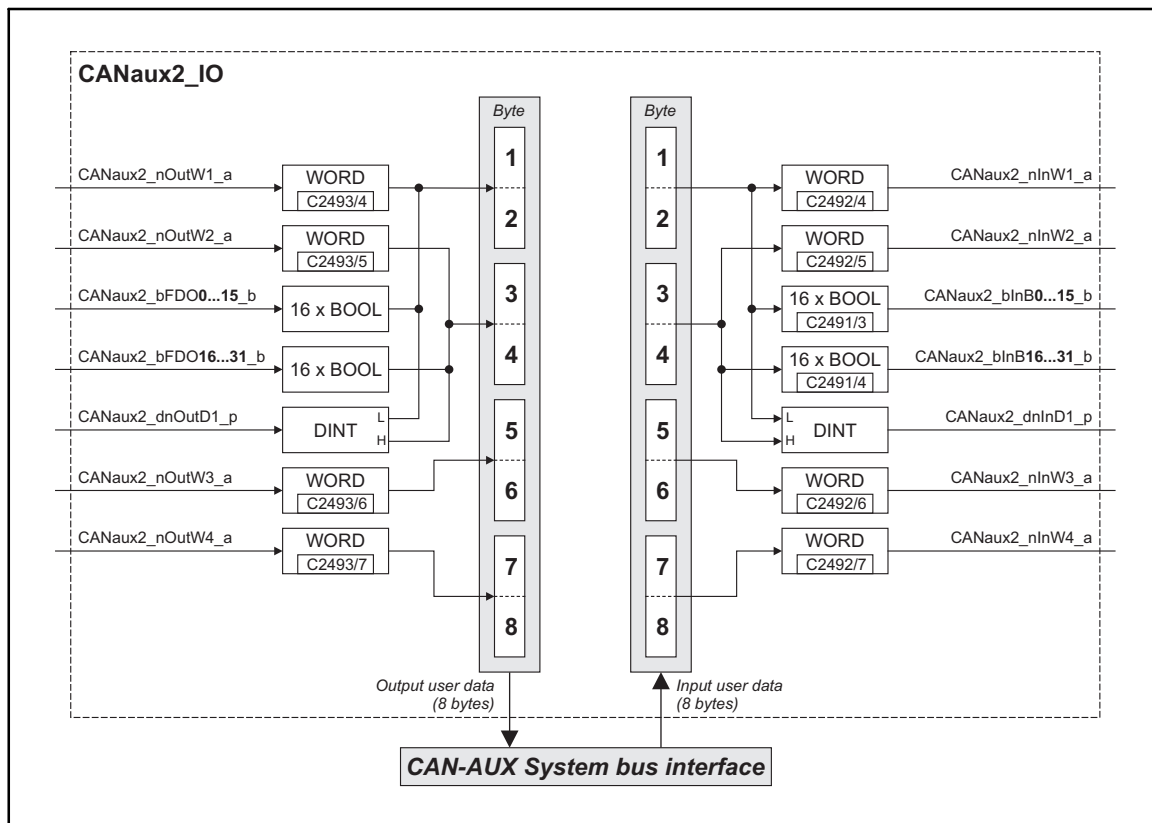


Fig. 9-2

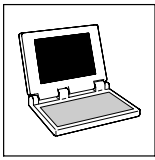
CANaux2\_IOsystem block



#### Tip!

Via C2457/2 you can set the monitoring time for the data reception. (Lenze setting: 3000 ms)

- Further information on this subject can be found in chapter 3.12.1. (☞ 3-11)



# System bus (CAN) for Lenze PLC devices

## CAN-AUX system blocks

CANaux2\_IO (node number: 35)

### 9.2.1 Inputs\_CANaux2

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CANaux2_nInW1_a	Integer	Analog	%IW35.0	C2492/4	dec [%]	
CANaux2_nInW2_a			%IW35.1	C2492/5		
CANaux2_bInB0_b	Bool	Binary	%IX35.0.0	C2491/3	hex	
...			...			
CANaux2_bInB15_b			%IX35.0.15	C2491/4		
CANaux2_bInB16_b			%IX35.1.0			
...	...					
CANaux2_bInB31_b			%IX35.1.15			
CANaux2_dnInD1_p	Double integer	Position	%ID35.0	-	-	
CANaux2_nInW3_a	Integer	Analog	%IW35.2	C2492/6	dec [%]	
CANaux2_nInW4_a			%IW35.3	C2492/7		

### 9.2.2 Outputs\_CANaux2

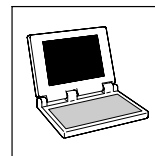
Variable	Data type	Signal type	Address	Display code	Display format	Notes
CANaux2_nOutW1_a	Integer	Analog	%QW35.0	C2493/4	dec [%]	
CANaux2_nOutW2_a			%QW35.1	C2493/5		
CANaux2_bFDO0_b	Bool	Binary	%QX35.0.0	-	-	
...			...			
CANaux2_bFDO15_b			%QX35.0.15	-		
CANaux2_bFDO16_b			%QX35.1.0			
...	...					
CANaux2_bFDO31_b			%QX35.1.15			
CANaux2_dnOutD1_p	Double integer	Position	%QD35.0	-	-	
CANaux2_nOutW3_a	Integer	Analog	%QW35.2	C2493/6	dec [%]	
CANaux2_nOutW4_a			%QW35.3	C2493/7		

### 9.2.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)



### 9.2.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables				
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)		
1	0	CANaux2_bFD00_b	CANaux2_nOutW1_a	CANaux2_dnOutD1_p		
	...	...				
	7	CANaux2_bFD07_b				
2	0	CANaux2_bFD08_b			CANaux2_nOutW2_a	CANaux2_dnOutD1_p
	...	...				
	7	CANaux2_bFD015_b				
3	0	CANaux2_bFD016_b	CANaux2_nOutW3_a	CANaux2_dnOutD1_p		
	...	...				
	7	CANaux2_bFD023_b				
4	0	CANaux2_bFD024_b			CANaux2_nOutW4_a	CANaux2_dnOutD1_p
	...	...				
	7	CANaux2_bFD031_b				
5	0...7					
6	0...7					
7	0...7					
8	0...7					



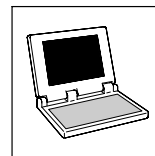
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 1 and 2, only use the variable *CANaux2\_dnOutD1\_p*, *CANaux2\_nOutW1\_a*, or only the variables *CANaux2\_bFD00\_b ... CANaux2\_bFD015\_b* for this purpose!

#### Variables for received user data

User data		Assigned variables				
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)		
1	0	CANaux2_blnB0_b	CANaux2_nInW1_a	CANaux2_dnInD1_p		
	...	...				
	7	CANaux2_blnB7_b				
2	0	CANaux2_blnB8_b			CANaux2_nInW2_a	CANaux2_dnInD1_p
	...	...				
	7	CANaux2_blnB15_b				
3	0	CANaux2_blnB16_b	CANaux2_nInW3_a	CANaux2_dnInD1_p		
	...	...				
	7	CANaux2_blnB23_b				
4	0	CANaux2_blnB24_b			CANaux2_nInW4_a	CANaux2_dnInD1_p
	...	...				
	7	CANaux2_blnB31_b				
5	0...7					
6	0...7					
7	0...7					
8	0...7					



### 9.3 CANaux3\_IO (node number: 36)

This SB serves to transmit event-controlled or time-controlled process data via the system bus.

- The setting of the transmission mode (event- or time-controlled) is effected via C2456. (☞ 3-6)
- A sync telegram is not required.

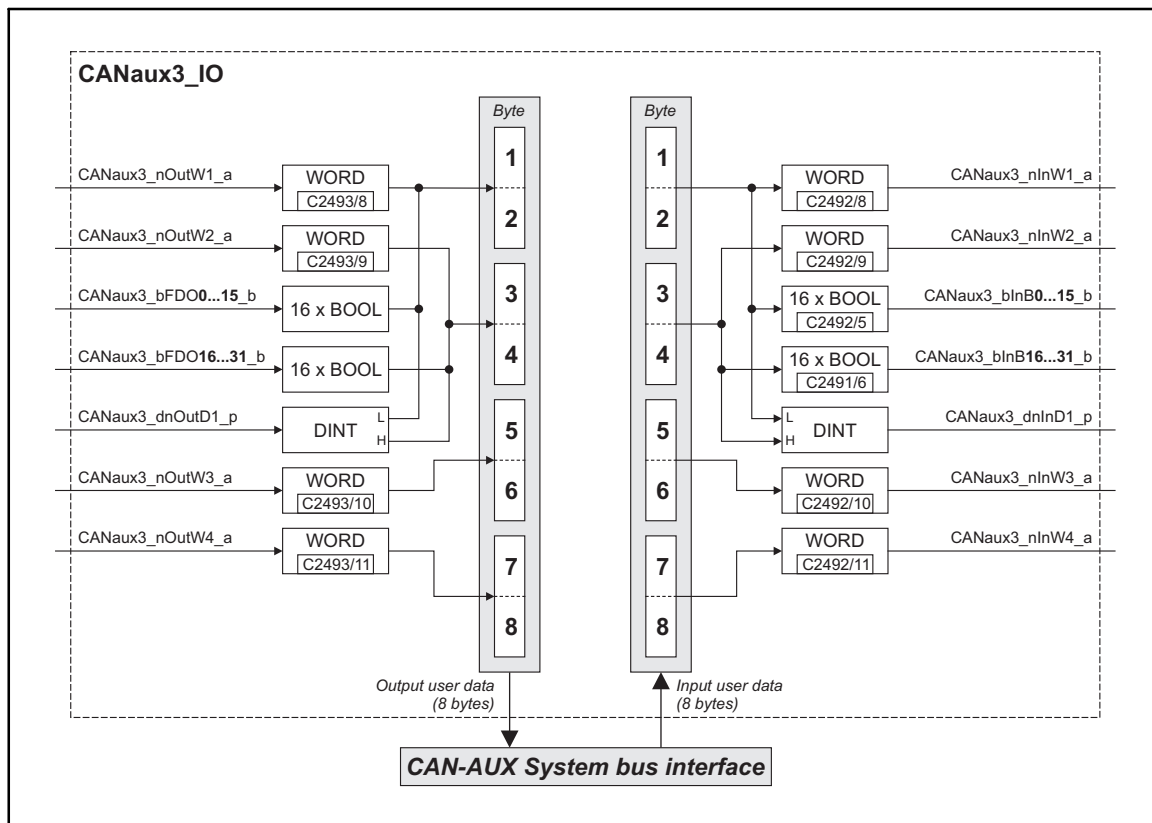


Fig. 9-3

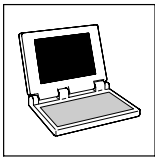
CANaux3\_IOsystem block



#### Tip!

Via C2457/3 you can set the monitoring time for the data reception. (Lenze setting: 3000 ms)

- Further information on this subject can be found in chapter 3.12.1. (☞ 3-11)



# System bus (CAN) for Lenze PLC devices

## CAN-AUX system blocks

CANaux3\_IO (node number: 36)

### 9.3.1 Inputs\_CANaux3

Variable	Data type	Signal type	Address	Display code	Display format	Notes
CANaux3_nInW1_a	Integer	Analog	%IW36.0	C2492/8	dec [%]	
CANaux3_nInW2_a			%IW36.1	C2492/9		
CANaux3_bInB0_b	Bool	Binary	%IX36.0.0	C2491/5	hex	
...			...			
CANaux3_bInB15_b			%IX36.0.15			
CANaux3_bInB16_b			%IX36.1.0	C2491/6		
...	...					
CANaux3_bInB31_b	%IX36.1.15					
CANaux3_dnInD1_p	Double integer	Position	%ID36.0	-	-	
CANaux3_nInW3_a	Integer	Analog	%IW36.2	C2492/10	dec [%]	
CANaux3_nInW4_a			%IW36.3	C2492/11		

### 9.3.2 Outputs\_CANaux3

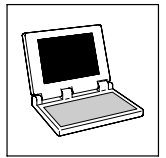
Variable	Data type	Signal type	Address	Display code	Display format	Notes
CANaux3_nOutW1_a	Integer	Analog	%QW36.0	C2493/8	dec [%]	
CANaux3_nOutW2_a			%QW36.1	C2493/9		
CANaux3_bFDO0_b	Bool	Binary	%QX36.0.0	-	-	
...			...			
CANaux3_bFDO15_b			%QX36.0.15			
CANaux3_bFDO16_b			%QX36.1.0	-		
...	...					
CANaux3_bFDO31_b	%QX36.1.15					
CANaux3_dnOutD1_p	Double integer	Position	%QD36.0	-	-	
CANaux3_nOutW3_a	Integer	Analog	%QW36.2	C2493/10	dec [%]	
CANaux3_nOutW4_a			%QW36.3	C2493/11		

### 9.3.3 Process data telegram

The process data telegram consists of an *identifier* and 8 bytes of user data.

11bit	8 bytes user data							
Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Information on the identifier can be found in chapter 2.4.1. (□ 2-3)



### 9.3.4 Assignment of the user data to variables

Several variables of different data types are assigned to the user data to be transmitted and received. Thus, the data in the PLC program can be optionally interpreted as:

- binary information (1 bit)
- status word/quasi-analog value (16 bit)
- angle information (32 bit)

#### Variables for user data to be transmitted

User data		Assigned variables			
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)	
1	0	CANaux3_bFD00_b	CANaux3_nOutW1_a	CANaux3_dnOutD1_p	
	...	...			
	7	CANaux3_bFD07_b			
2	0	CANaux3_bFD08_b			
	...	...			
	7	CANaux3_bFD015_b			
3	0	CANaux3_bFD016_b			CANaux3_nOutW2_a
	...	...			
	7	CANaux3_bFD023_b			
4	0	CANaux3_bFD024_b			
	...	...			
	7	CANaux3_bFD031_b			
5	0...7		CANaux3_nOutW3_a		
6	0...7				
7	0...7				
8	0...7		CANaux3_nOutW4_a		



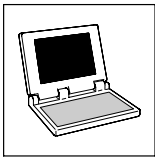
#### Note!

Avoid simultaneous overwriting via different variable types to ensure data consistency.

For instance, if you want to write bytes 1 and 2, only use the variable *CANaux3\_dnOutD1\_p*, *CANaux3\_nOutW1\_a*, or only the variables *CANaux3\_bFD00\_b ... CANaux3\_bFD015\_b* for this purpose!

#### Variables for received user data

User data		Assigned variables			
Byte	Bit	Variable (1 bit)	Variable (16 bit)	Variable (32 bit)	
1	0	CANaux3_blnB0_b	CANaux3_nlnW1_a	CANaux3_dnlnD1_p	
	...	...			
	7	CANaux3_blnB7_b			
2	0	CANaux3_blnB8_b			
	...	...			
	7	CANaux3_blnB15_b			
3	0	CANaux3_blnB16_b			CANaux3_nlnW2_a
	...	...			
	7	CANaux3_blnB23_b			
4	0	CANaux3_blnB24_b			
	...	...			
	7	CANaux3_blnB31_b			
5	0...7		CANaux3_nlnW3_a		
6	0...7				
7	0...7				
8	0...7		CANaux3_nlnW4_a		



# System bus (CAN) for Lenze PLC devices

## CAN-AUX system blocks

### CANaux\_Management (node number: 111)

## 9.4 CANaux\_Management (node number: 111)

By means of this SB

- a **reset node** can be activated, e. g. to accept changes with regard to the baud rate and addressings.
- **Communication error, Bus-off state**, and further states can be processed in the PLC program.
- the instant of transmission of CANaux2\_OUT and CANaux3\_OUT can be influenced.

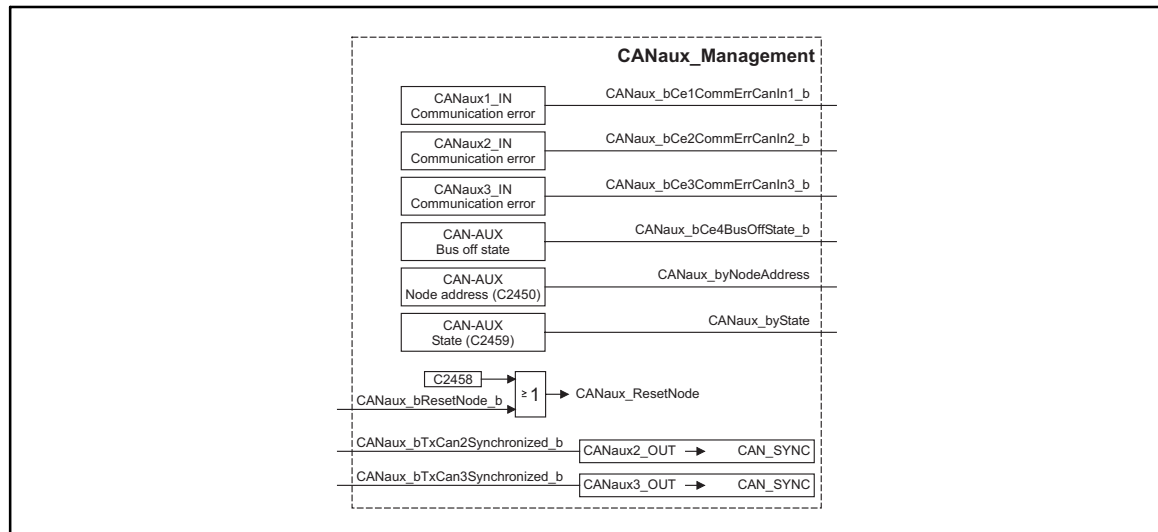


Fig. 9-4 CANaux\_Managementsystem block

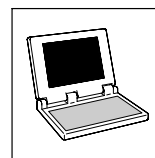


### Note!

The process image for this SB is generated in a fixed system task (interval: 1 ms).

### 9.4.1 Inputs\_CANaux\_Management

Identifier	Data type	Signal type	Address	DIS	DIS format	Information
CANaux_bCe1CommErrCanIn1_b	Bool	Binary	%IX111.0.0	-	-	CANaux1_IN communication error
CANaux_bCe2CommErrCanIn1_b			%IX111.0.1			CANaux2_IN communication error
CANaux_bCe3CommErrCanIn1_b			%IX111.0.2			CANaux3_IN communication error
CANaux_bCe4BusOffState_b			%IX111.0.3			CAN-AUX bus "off state" recognised
CANaux_byNodeAddress	Byte	-	%IB111.2	C2450	-	CAN-AUX node address
CANaux_byState			%IB111.3	C2459	-	CAN-AUX status



### 9.4.2 Outputs\_CANaux\_Management

Identifier	Data type	Signal type	Address	DIS	DIS format	Information
CANaux_bResetNode_b	Bool	Binary	%QX111.0.0	-	-	Carry out reset node of the CANaux
CANaux_bTxCan2Synchronized_b			%QX111.0.1			Transfer CANaux2_OUT with sync telegram.
CANaux_bTxCan3Synchronized_b			%QX111.0.2			Transfer CANaux3_OUT with sync telegram.

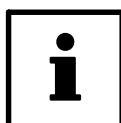


#### Note!

If *CANaux\_bTxCan2Synchronized\_b* and/or *CANaux\_bTxCan3Synchronized\_b* are integrated, it is required to configure the device as sync transmitter. The data are transmitted immediately after the sync telegram is sent.

### 9.4.3 Activating a reset node

A reset node can be activated by setting *CANaux\_bResetNode\_b* to TRUE or C2458 = .



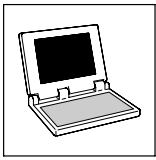
#### Tip!

Even if the **CANaux\_Management** SB has not been assigned to the control configuration, a reset node can be activated via C2458. (□ 3-8)

### 9.4.4 Defining the instant of transmission for CANaux2\_OUT/CANaux3\_OUT

Via *CANaux\_bTxCan2Synchronized\_b* and *CANaux\_bTxCan3Synchronized\_b* you define the instant of transmission for the CAN objects CANaux2\_OUT and CANaux3\_OUT:

- **FALSE:** Data from the CANaux2\_OUT/CANaux3\_OUT are transmitted at the end of the process image.
- **TRUE:** Data from the CANaux2\_OUT/CANaux3\_OUT are sent to sync.
  - The identifiers for the sync transmission and reception telegram can be set via C2467/C2468.
  - The *sync Tx time* can be set via C2469.



## System bus (CAN) for Lenze PLC devices

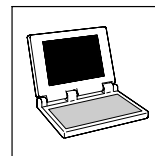
### CAN-AUX system blocks

CANaux\_Management (node number: 111)

#### 9.4.5 Status messages

The **CANaux\_Management** SB provides different status messages which can be processed in the PLC program:

Identifier	Information
CANaux_bCe1CommErrCanIn1_b	TRUE CANaux1_IN communication error
CANaux_bCe2CommErrCanIn1_b	TRUE CANaux2_IN communication error
CANaux_bCe3CommErrCanIn1_b	TRUE CANaux3_IN communication error
CANaux_bCe4BusOffState_b	TRUE CAN-AUX-bus "off state" recognised
CANaux_byNodeAddress	1...63 CAN-AUX node address (☐ 3-3)
CANaux_byState	Operating status of the system bus
	1 Operational
	2 Pre-operational
	3 Warning
	4 Bus-off



## 10 LenzeCanDrv.lib function library

By using the functions/function blocks of the **LenzeCanDrv.lib** function library, so-called "free CAN objects" can be added to the fixedly integrated CAN objects.

### 10.1 Overview

Function/FB	Information
CAN driver	
<b>L_CanInit</b>	Initialise CAN driver  10-2
<b>L_CanClose</b>	Deactivate CAN driver  10-5
<b>L_CanGetStatus</b>	Query driver status  10-6
<b>L_CanGetRelocCobId</b>	Query COB-ID range  10-7
Transmitting/receiving CAN objects	
<b>L_CanPdoTransmit</b>	Send CAN object  10-8
<b>L_CanPdoReceive</b>	Receive CAN object  10-12

### 10.2 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`.

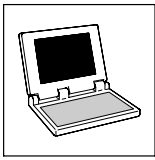
Version identifiers as of PLC software version 7.x:

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".



# System bus (CAN) for Lenze PLC devices

## LenzeCanDrv.lib function library

### L\_CanInit - initialising the CAN driver

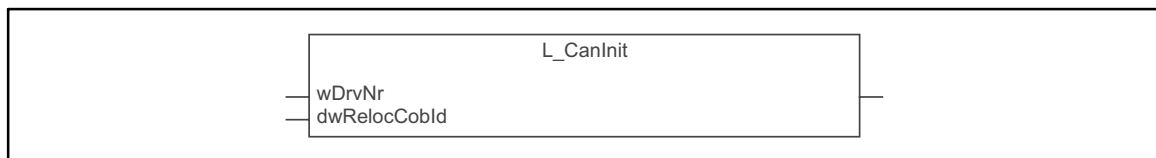
## 10.3 L\_CanInit - initialising the CAN driver

### Function

DWORD **L\_CanInit** (wDrvNr, dwRelocCobIdArea)

Before it is possible to work with the free CAN objects, an initialisation of the CAN driver has to be carried out.

- By repeatedly calling this function, the CAN driver in the initialised state can be switched over to other parameters.
  - The acceptance of the new parameters is effected immediately and therefore can have an impact on transmission and reception jobs which are still pending. Thus, the function may not be called cyclically.

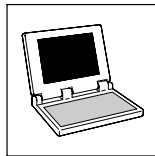


### Transfer parameters

Identifier	Data type	Possible settings	Information
wDrvNr	Word	10	System bus
dwRelocCobIdArea	Double Word	0 192...319 832...1344 1664...1728 1856...1984	Identifier area - <b>Only relevant for 9300 Servo PLC!</b> <i>dwRelocCobIdArea</i> = 832 (Lenze setting) Free range 1 Free range 2 Free range 3 Free range 4

### Return value

Data type	Bit	Value	Meaning	Priority
Double Word	0	0	Driver is initialised.	-
		1	Error during initialisation. • In this case the bits 1 ... 31 are all set to "1" and therefore are invalid.	
	1	0	Setting <i>dwRelocCobIdArea</i> OK.	
		1	Setting <i>dwRelocCobIdArea</i> not OK. • The identifier area featuring 64 objects overlaps with another area or is settled beyond the limits 0 ... 2047.	
	2	0	Free object found.	
		1	No free object found. Remedy: • Do not use one of the CAN objects CAN1_IN ... CAN3_IN or CAN1_OUT ... CAN3_OUT. • Set C2118 to "0" (write parameters via SD02). PLEASE NOTE: at C2118 = 0 the SD02 channel is no longer available! • Switch off generation of the sync object (C0369 = 0).	
	3-15		Reserved for future supplements (bits are set to "0").	
	16-31		Version of the <b>LenzeCanDrv.lib</b> function library Format: main version/subversion (e. g. 0103hex = version 1.03)	



## Reception identifier area

Fig. 10-1 shows the identifier area provided for data reception.

- Basically the entire identifier area of 0 ... 2047 is provided.
- The grayed out fields in Fig. 10-1 are pre-assigned for specific applications, e. g. for network management commands and process data.
- Fields which are not pre-assigned are designated as free area 1 ... 4.



## Note on the 9300 Servo PLC (firmware version 2.6 or lower)!

In contrast to the Drive PLC, in the case of the 9300 Servo PLC due to memory limitations only the following data can be received **at the same time**:

- Data with identifiers within the grayed out fields in Fig. 10-1.
- Data with identifiers within the identifier field **COB-ID field** featuring 64 objects.

NMT commands	0
	127
Tx-sync	128
Alarm objects	129
	191
Free range 1	192
	383
Process data objects (PDO)	384
	831
COB-ID field (shiftable identifier field - Lenze setting)	832
	895
Free range 2	1407
Service data objects (SDO)	1408
	1663
Free range 3	1664
	1791
Node monitoring	1792
	1855
Free range 4	1856
	2047

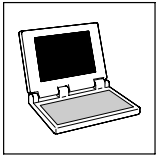
**COB-ID area  
(only relevant for 9300 Servo PLC)**

For the simultaneous data reception concerning the 9300 Servo PLC in addition to the pre-assigned areas, 64 further identifiers are provided to you via the COB-ID area.

- In the Lenze setting these are the identifiers 832 ... 895.
- If you require other identifiers you can shift the COB-ID area within the free areas 1...4 during initialisation using the variable *dwRelocCobIdArea*.
- If the COB-ID area by means of *dwRelocCobIdArea* is shifted in a way that causes an overlapping with a fixedly predefined area, an error in the return value of the function is reported.  
(Bit 0 = 1 / Bit 1 = 1)

Fig. 10-1

Pre-assigned and free identifier fields as well as relocatable COB-ID area for 9300 Servo PLC



# System bus (CAN) for Lenze PLC devices

## LenzeCanDrv.lib function library

### L\_CanInit - initialising the CAN driver

#### Response with regard to errors (Tx/Rx buffer)

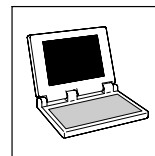
The response in the case of errors in the Tx/Rx buffer can be configured via the codes C0608/C0609:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0608	over Tx-Queue	0	0 TRIP 1 Message 2 Warning 3 Off 4 FAIL QSP (not for Drive PLC!)	Configuration of the monitoring for Tx buffer
C0609	over Rx-Isr	0	0 TRIP 4 Fail-QSP	Configuration of the monitoring for Rx buffer • Not for Drive PLC!

#### Example

Calling the function in ST with decoding of the return value:

```
(* open CAN driver - returns g_dwReturnValue *)  
  
g_dwReturnValue:=L_CanInit(10, 0);  
  
g_bInitOK      := NOT DWORD_TO_BOOL(g_dwReturnValue AND 16#0000_0001);  
g_bDriverFail  := DWORD_TO_BOOL(SHR(g_dwReturnValue,1) AND 16#0000_0001);  
g_wVersion     := DWORD_TO_WORD(SHR(g_dwReturnValue,16) AND 16#FFFF);
```



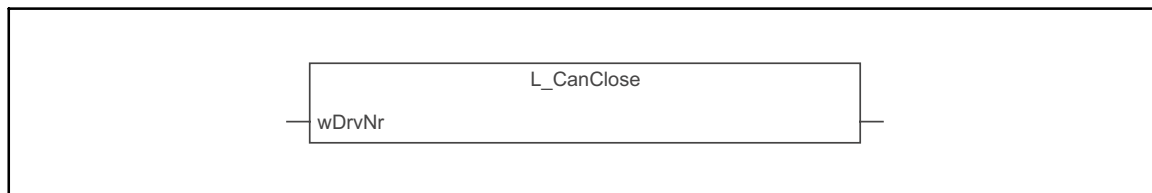
## 10.4 L\_CanClose - deactivating the CAN driver

### Function

BOOL **L\_CanClose** (wDrvNr)

By means of this function the CAN driver is deactivated.

- Transmission and reception tasks which are still available are cancelled when this function is called.



### Transfer parameters

Identifier	Data type	Possible settings	Notes
wDrvNr	Word	10	System bus

### Return value

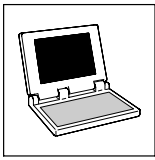
Data type	Value	Meaning	Priority
Bool	0	Error while activating the driver (incorrect driver number, or driver was not initialised.)	-
	1	Driver has been deactivated.	

### Example

Calling the function in ST:

```
(* close CAN driver - returns g_bCanCloseState *)
```

```
g_bCanCloseState := L_CanClose(10)
```



# System bus (CAN) for Lenze PLC devices

## LenzeCanDrv.lib function library

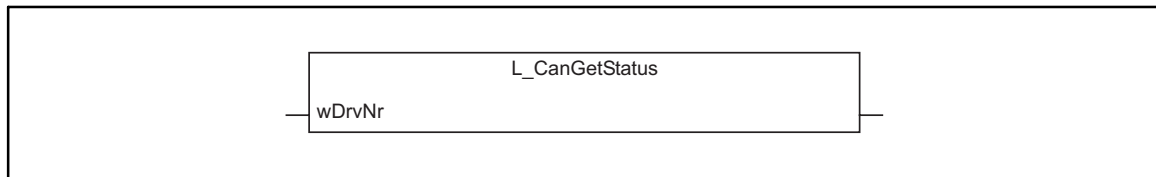
### L\_CanGetStatus - querying the driver status

## 10.5 L\_CanGetStatus - querying the driver status

### Function

DWORD **L\_CanGetStatus** (wDrvNr)

By means of this function the status of the CAN driver can be determined.



### Transfer parameters

Identifier	Data type	Possible settings	Information
wDrvNr	Word	10	System bus

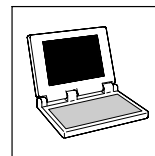
### Return value

Data type	Bit	Value	Meaning	Priority
Double Word	0, 1	0	CAN driver is <i>Operational</i> .	-
		1	CAN driver is not available or not initialised. • In this case the bits 16 ... 31 are all set to "1" and therefore are invalid.	
		2	CAN driver is <i>Pre-operational</i> .	
		3	CAN driver is stopped.	
Double Word	2...15		Reserved for future supplements (bits are set to "0").	
	16...31		Version of the CAN driver Format: main version/subversion (e. g. 0103hex = version 1.03)	

### Example

Calling the function in ST:

```
g_dwCanDrvState:=L_Can_GetStatus(10);
```



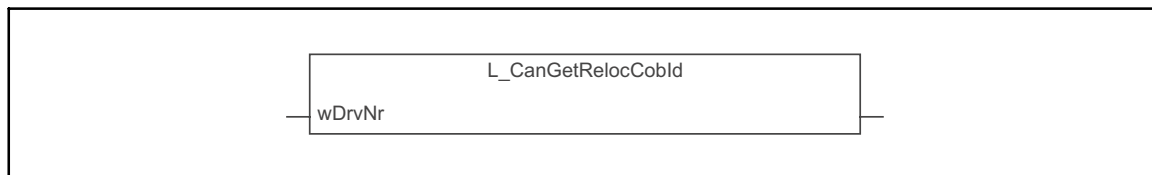
## 10.6 L\_CanGetRelocCobId - querying the COB-ID range

### Function

INT L\_CanGetRelocCobId (wDrvNr)

By means of this function the starting identifier of the identifier range (COB-ID range) consisting of 64 objects can be determined for reception operations.

- **This function only is relevant for the 9300 Servo PLC!**



### Transfer parameters

Identifier	Data type	Possible settings	Information
wDrvNr	Word	10	System bus

### Return value

Data type	Value	Meaning	Priority
Double integer	-121	Incorrect driver number ( <i>wDrvNr</i> )	1 (high)
	-120	Driver not initialised	2 (low)
	* If there are several error causes, always the return value associated with the error cause of the highest priority is returned.		
	0	Return value for use in Drive PLC, as function only is relevant for 9300 Servo PLC!	
	192...319	Start identifier of the free range 1	
	832...1344	Start identifier of the free range 2	
	1664...1728	Start identifier of the free range 3	
	1856...1984	Start identifier of the free range 4	



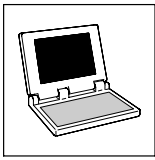
### Note!

From version 6.x of the 9300 Servo PLC, all free identifiers are provided.

### Example

Calling the function in ST:

```
g_nRelocCobIdLocation := L_CanGetRelocCobId(10);
```



# System bus (CAN) for Lenze PLC devices

## LenzeCanDrv.lib function library

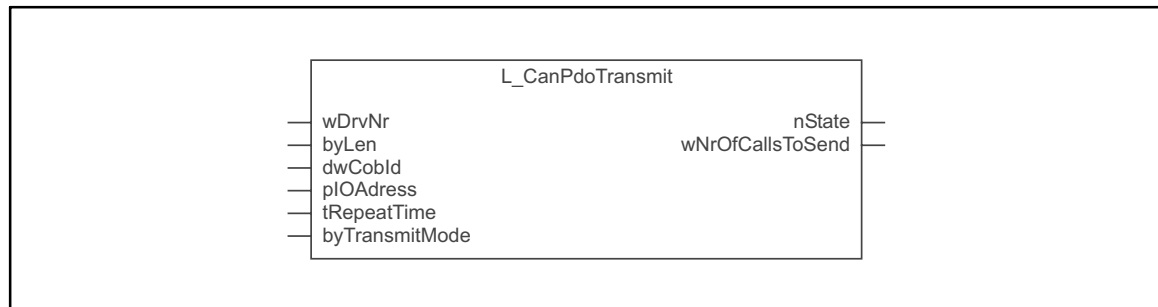
### L\_CanPdoTransmit - transmitting a CAN object

## 10.7 L\_CanPdoTransmit - transmitting a CAN object

### Function block

This FB serves to transmit data via the system bus interface according to CANopen.

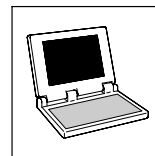
- The data transmission is carried out simultaneously to the process of the PLC program by the operating system of the controller, whereby an object can be max. sent every 250  $\mu$ s.



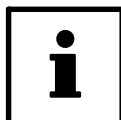
Identifier	Data type	Variable type	Possible settings	Information
wDrvNr	Word	VAR_INPUT	10	System bus
byLen	Byte	VAR_INPUT	0...8	Telegram length (in bytes)
dwCobID	Double Word	VAR_INPUT	0...2047	CAN identifier
pIOAddress	Pointer to Byte	VAR_INPUT	Pointer to the address in the memory from which the data bytes to be transmitted are stored.	The address of a variable can be determined via the address function <b>ADR</b> .
tRepeatTime	Time	VAR_INPUT	T#0s T#xms	Parameter for the time-controlled transmission ( <i>byTransmitMode</i> = 1/2) <ul style="list-style-type: none"> <li>Transmission takes place at each call of the FB.</li> <li>Transmission takes place after the set cycle time x (in ms) has expired.</li> </ul>
byTransmitMode	Byte	VAR_INPUT	0  1  2  3	<p>Event-controlled transmission</p> <ul style="list-style-type: none"> <li>Transmission takes place if the input data have changed.</li> <li>If the bus state changes from <i>Pre-operational</i> to <i>Operational</i>, the telegram on principle is transmitted once.</li> </ul> <p>Time-controlled transmission</p> <ul style="list-style-type: none"> <li>Transmission takes place after the cycle time set via <i>tRepeatTime</i> has expired.</li> </ul> <p>Time-controlled transmission with superimposed event control</p> <ul style="list-style-type: none"> <li>Transmission takes place after the cycle time set via <i>tRepeatTime</i> has expired, and if the transmitted data have changed.</li> </ul> <p>Forced transmission</p> <ul style="list-style-type: none"> <li>Transmission takes place if the action &lt;Instance name&gt;. <b>SendData</b> is called, irrespective of the cycle time set.</li> </ul>

# System bus (CAN) for Lenze PLC devices

LenzeCanDrv.lib function library  
L\_CanPdoTransmit - transmitting a CAN object



Identifier	Data type	Variable type	Possible settings	Information
nState	Integer	VAR_OUTPUT	-	Displays the current transmit status. <ul style="list-style-type: none"> <li>See the following table "Transmit state (nState)"</li> </ul>
wNrOfCallsToSend	Word	VAR_OUTPUT	-	Displays how many FB calls were required to transmit the object. <ul style="list-style-type: none"> <li>The "time measurement" starts with entering the transmit request in the transmit request memory and ends with the actual transmission of the object.</li> </ul>

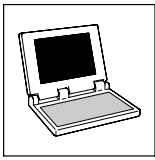


## Note!

With regard to the allocation of the CAN identifier (*dwCobId*) please be sure that it is not already used by one of the other CAN objects CAN1\_IO ... CAN3\_IO, as otherwise bus errors/bus overload may occur!

## Transmit status (nState)

Data type	Value	Meaning	Priority
Integer	-150	CAN bus is not in the <i>Operational</i> state.	1 (high)
	-121	Incorrect driver number ( <i>wDrvNr</i> )	2
	-120	Driver not initialised	3
	-119	The transmit request memory is full. The transmit request could not be entered anymore. Remedy: <ul style="list-style-type: none"> <li>Decrease number of the transmission objects.</li> <li>Increase cycle time of the transmission objects.</li> <li>Increase baud rate.</li> </ul> Basically, an object is transmitted every 250 µs.	4
	-118	No free CAN channel is provided. Remedy: <ul style="list-style-type: none"> <li>Do not use one of the CAN objects CAN1_IN ... CAN3_IN or CAN1_OUT ... CAN3_OUT.</li> <li>Set C2118 to 1" (write parameters via SD02).</li> </ul> PLEASE NOTE: at C2118 = 1 the SD02 channel no longer is available! <ul style="list-style-type: none"> <li>Switch off generation of the sync object (C0369 = 0).</li> </ul>	5
	-12	The set message identifier (COB-ID) is beyond the permissible range (0 ... 2047).	6
	-11	Pointer <i>pIOAddress</i> does not point to PLC-RAM.	7 (low)
		* If there are several error causes, always the return value associated with the error cause of the highest priority is returned.	
	0	The transmit request has been carried out and the data have been transmitted.	
	1	The transmit request has not yet been completed and is still pending in the transmit request memory.	
10	The specification for the telegram length <i>byLen</i> is higher than 8. The telegram length has been limited to 8 bytes.		



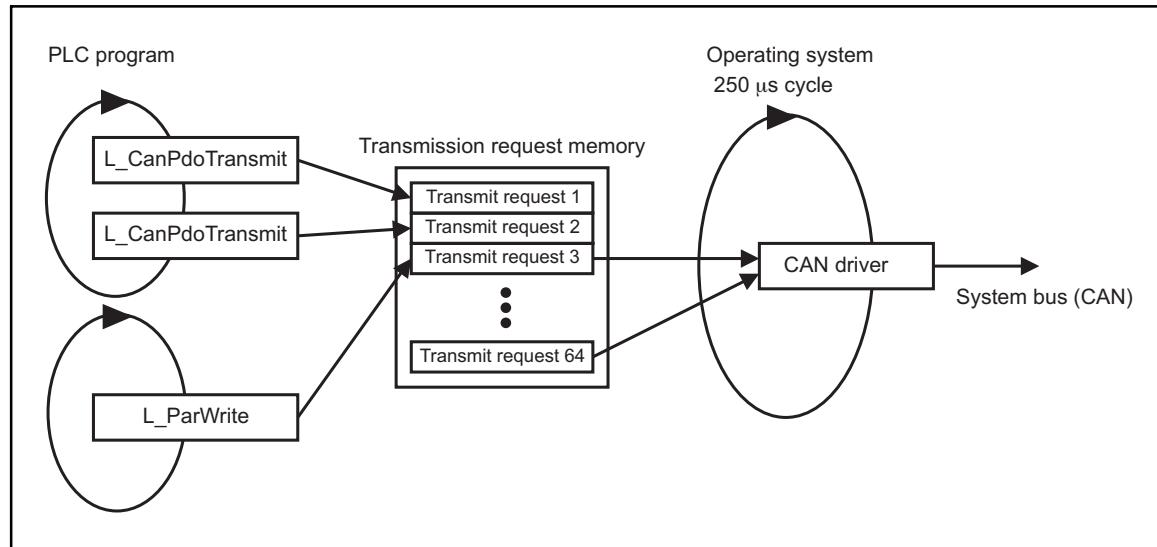
## System bus (CAN) for Lenze PLC devices

### LenzeCanDrv.lib function library

#### L\_CanPdoTransmit - transmitting a CAN object

### Transmit request memory as an interface between the CAN driver and the L\_CanPdoTransmitFB

As the data transmission via the CAN driver is effected simultaneously to the process of the PLC program, a temporary storage for the transmit requests, the so-called transmit request memory, is used between the CAN driver and the **L\_CanPdoTransmit** FB.



- Every time the **L\_CanPdoTransmit** FB is called, a transmit request is stored in the transmit request memory.
- Transmit requests can also be stored in the transmit request memory using the **L\_ParWrite** FB of the **LenzeDrive.lib** function library.
- The transmit request memory can buffer 64 transmit requests of the **L\_CanPdoTransmit** or **L\_ParWrite** FBs at a total.
- Every 250 μs, simultaneously to the process of the PLC program, a transmit request is collected from the transmit request memory and is processed by the operating system.



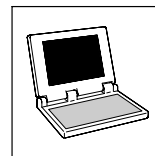
### Note!

If the transmit request memory due to a too frequent call of the **L\_CanPdoTransmit** or **L\_ParWrite** FBs is filled up faster as it is emptied by the operating system which collects the transmit requests, an overflow error will occur.

- In the case of an overflow error, the variable *nState* receives the value "-119".
- Additionally a corresponding error message is output by the operating system. (See chapter "Error messages" in the Manual to the respective target system, e. g. 9300 Servo PLC, Drive PLC, or ECSxA).

# System bus (CAN) for Lenze PLC devices

LenzeCanDrv.lib function library  
L\_CanPdoTransmit - transmitting a CAN object



## Forced transmission

Some cases may require an entry of a transmit request in the transmit request memory **without** considering a data change (event-controlled) or a cycle time (time-controlled).

For this purpose, the **L\_CanPdoTransmit** FB supports the action **SendData** in the "Forced transmission" transmit mode (*byTransmitMode* = 3).

## Example

Declaration of the function block in ST:

```
SendWithID678: L_CanPdoTransmit; (* send data with identifier 678 *)
```

Calling the function block in ST:

```
SendWithID678 (wDrvNr:=10,  
byLen:=8,  
dwCobID:=678,  
pIOAddress:=ADR (abySendData) ,  
tRepeatTime:=T#5ms,  
byTransmitMode:=3);
```

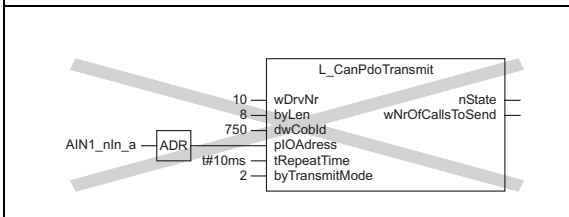
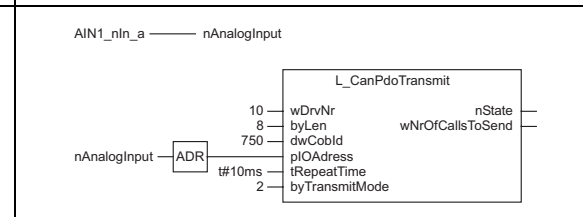
Action "Forced transmission" in ST:

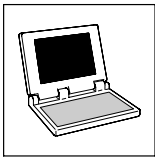
```
SendWithID678.SendData; (* force send procedure *)
```



## Note!

The use of an address which is stored in the memory area of the operating system is not permitted for the **L\_CanPdoTransmit** and **L\_CanPdoReceive** FBs!

Problem:	Remedy:
The addresses of the system variables (inputs/outputs of system blocks) are also placed in the memory area of the operating system, in the following illustration the system variable <i>AIN1_nIn_a</i> .	Instead of the address of the system variable, use the address of a temporary variable, which the value of the corresponding system variable is copied into:
	



# System bus (CAN) for Lenze PLC devices

## LenzeCanDrv.lib function library

### L\_CanPdoReceive - receiving a CAN object

## 10.8 L\_CanPdoReceive - receiving a CAN object

### Function block

This FB serves to receive data via the system bus interface according to CANopen.

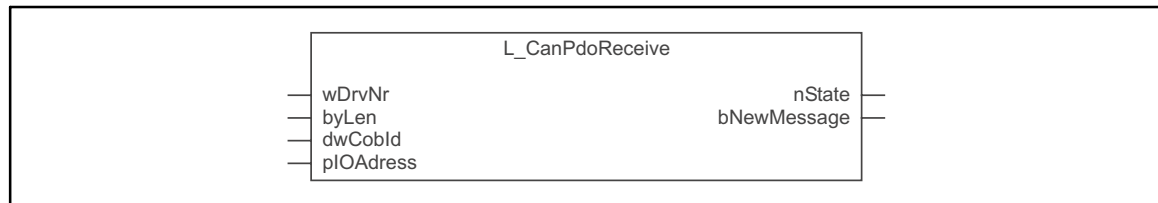


### Tip!

An overload with regard to the receive process can occur if due to a high bus utilisation or very fast transmission activities of the other nodes further receive telegrams already arrive while a receive telegram is processed.

The operating system responds to this with an overflow error:

- A corresponding error message is output (see chapter "Error messages" in the Manual for the respective PLC, e. g. 9300 Servo PLC, Drive PLC, or ECSxA).
- Furthermore the CAN driver is deactivated for the free CAN objects and has to be reinitialised via the function **L\_CanInit**.

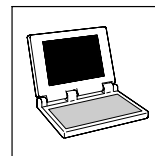


Identifier	Data type	Variable type	Possible settings	Information
wDrvNr	Word	VAR_INPUT	10	System bus
byLen	Byte	VAR_INPUT	0..8	Telegram length (in bytes)
dwCobID	Double Word	VAR_INPUT	0...2047	CAN identifier
pIOAddress	Pointer to Byte	VAR_INPUT	Pointer to the address in the memory from which the data bytes received are stored.	The address of a variable can be determined via the address function <b>ADR</b> .
bNewMessage	Bool	VAR_OUTPUT	-	Is set to <b>TRUE</b> if data have been received, and only is reset to <b>FALSE</b> by calling the following action: <Instance name> . <b>ResetNewMessage</b> <ul style="list-style-type: none"> <li>• If no reset is carried out, data can furthermore be received, but data reception is not displayed.</li> </ul>
nState	Integer	VAR_OUTPUT	-	Displays the current receive status. <ul style="list-style-type: none"> <li>• See the following table "Receive state (nState)"</li> </ul>



### Note!

With regard to the allocation of the CAN identifier (dwCobId) please be sure that it is not already used by one of the other CAN objects CAN1\_IO ... CAN3\_IO, as otherwise incorrect data are received!



## Receive status (nState)

Data type	Value	Meaning	Priority	
Integer	-150	CAN bus is not in the <i>Operational</i> state.	1 (high)	
	-121	Incorrect driver number ( <i>wDrvNr</i> )	2	
	-120	Driver not initialised	3	
	-12	The set message identifier (COB-ID) is beyond the permissible range (0 ... 2047).	4	
	-11	Pointer <i>pIOAddress</i> does not point to PLC-RAM.	5 (low)	
		* If there are several error causes, always the value associated with the error cause of the highest priority is returned.		
	0	Data have been faultlessly received.		
	10	The specification for the telegram length <i>byLen</i> is higher than 8. The telegram length has been limited to 8 bytes.		
	200	Data were received without resetting <i>bNewMessage</i> . Therefore valid data were possibly overwritten in the receive memory.		

## Resetting the variable "bNewMessage"

If data have been received from the CAN bus, the output variable *bNewMessage* is set to **TRUE** and only is reset to **FALSE** again by calling the action **<Instance name>.ResetNewMessage**:

```
(* FB instance *)

ReceiveFromID678: L_CanPdoReceive;    (* receive data      *)
                                      (* from identifier 678 *)

...

(* reset NewMessage information at FB ReceiveFromID678 *)

ReceiveFromID678.ResetNewMessage;    (* reset NewMessage  *)
```

## Example

Calling the function block in ST:

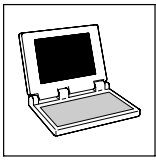
```
ReceiveFromID678 (wDrvNr:=10,
                  byLen:=8,
                  dwCobID:=678,
                  pIOAddress:=ADR(abyReceiveData));
```



## Note!

The use of an address which is stored in the memory area of the operating system is not permitted for the **L\_CanPdoTransmit** and **L\_CanPdoReceive**FBs!

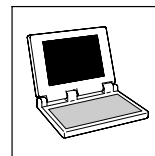
- See also note to the function **L\_CanPdoTransmit**. (10-8)



## ***System bus (CAN) for Lenze PLC devices***

***LenzeCanDrv.lib function library***

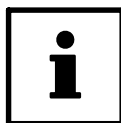
***L\_CanPdoReceive - receiving a CAN object***



## 11 LenzeCanDSxDrv.libfunction library

The **LenzeCanDSxDrv.lib** function library contains functions by means of which CAN indexes received via the system bus interface within the PLC can be "mapped" to other codes than to those which are automatically allocated.

Furthermore functions/FBs are provided, by means of which the "Heartbeat" and "Node guarding" monitoring mechanisms for ensuring the function of system bus nodes can be realised.



### Tip!

General information

- on the mapping of indexes to codes can be found in chapter 3.10. (☞ 3-8)
- on the "Heartbeat" and "Node Guarding" monitoring mechanisms can be found in chapter 2.10. (☞ 2-21)

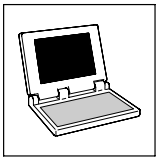
### 11.1 Overview

Function/FB	Information
Mapping indexes to codes	
• The functions are limited to the parameter access via the system bus interface integrated in the PLC.	
<b>L_CanDSxInitIndexCode</b>	Configure index mapping <span style="float: right;">☞ 11-3</span>
<b>L_CanDSxOpen</b>	Activate index mapping <span style="float: right;">☞ 11-5</span>
<b>L_CanDSxClose</b>	Deactivate index mapping again <span style="float: right;">☞ 11-6</span>
"Heartbeat" monitoring mechanism	
<b>L_CanDSxOpenHeartBeat</b>	Initialise "Heartbeat" <span style="float: right;">☞ 11-7</span>
<b>L_CanDSxHeartBeat</b>	Carry out "Heartbeat" <span style="float: right;">☞ 11-8</span>
<b>L_CanDSxCloseHeartBeat</b>	Deactivate "Heartbeat" again <span style="float: right;">☞ 11-10</span>
"Node guarding" monitoring mechanism	
<b>L_CanDSxOpenNodeGuarding</b>	Initialise "Node guarding" <span style="float: right;">☞ 11-11</span>
<b>L_CanDSxNodeGuarding</b>	Carry out "Node guarding" <span style="float: right;">☞ 11-12</span>
<b>L_CanDSxCloseNodeGuarding</b>	Deactivate "Node guarding" again <span style="float: right;">☞ 11-15</span>



### Tip!

For mapping codes which are accessed via the AIF interface, the **LenzeAifParMapDrv.lib** function library with its corresponding functions is provided to you.



# System bus (CAN) for Lenze PLC devices

## LenzeCanDSxDrv.lib function library

### Version identifiers of the function library

## 11.2 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`.

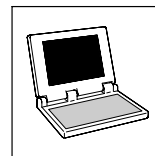
Version identifiers as of PLC software version 7.x:

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".



## 11.3 L\_CanDSxInitIndexCode - Configuration of index mapping

### Function

This function is used to configure the mapping table and the redirection of indices to codes other than the automatically assigned codes.

- With every function call one index and the corresponding Lenze code can be entered in the mapping table.

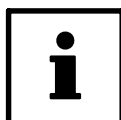
Declaration	
INT	<code>L_CanDSxInitIndexCode (byTabIndex, wCANIndex, byCANSubIndex, wLenzeCodeNumber, byLenzeSubCodeNumber);</code>

Transfer parameters	Data type	Information/possible settings	
byTabIndex	Byte	0 ... 255	Number of the configuration entry in the mapping table.
		Index to be redirected:	
wCANIndex	Word	1000 <sub>hex</sub> ... 8FFF <sub>hex</sub> (4096 <sub>dec</sub> ... 36863 <sub>dec</sub> )	CAN index
byCANSubIndex	Byte	0 ... 255	CAN subindex
		Redirection target (Lenze code):	
wLenzeCodeNumber	Word	1 ... 7999	Code number
byLenzeSubCodeNumber	Byte	0 ... 255	Subcode number

Return value	Data type	Value/meaning
	Integer	Status
		0 Entry in the mapping table has been successful.
		-20 Error: Transfer parameter <i>wCANIndex</i> is invalid.
		-30 Error: Transfer parameter <i>wLenzeCodeNumber</i> is invalid.

- A maximum of 256 entries can be entered in the mapping table:

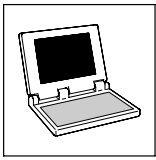
byTabIndex	Code to be redirected		Redirection target	
	wCANIndex	byCANSubIndex	wLenzeCodeNumber	byLenzeSubCodeNumber
0				
1	4104	2	3200	5
2				
...				
255				



### Note!

If the function **L\_CanDSxInitIndexCode** is called while code read or write requests are active an error may occur!

This is why all actions with code access should be completed before this function is called.



## **System bus (CAN) for Lenze PLC devices**

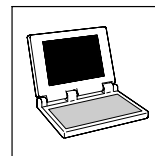
***LenzeCanDSxDrv.lib function library***

***L\_CanDSxInitIndexCode - Configuration of index mapping***

### **Example**

Calling the function in ST:

```
nReturnInitIndexCode := L_CanDSxInitIndexCode (byTabIndex:=1,  
wCANIndex:=4101,  
byCANSubIndex:=2,  
wLenzeCodeNumber:=3200,  
byLenzeSubCodeNumber:=5);
```



## 11.4 L\_CanDSxOpen - initialising the CanDSx driver

### Function

By means of this function the CanDSx driver is initialised in the operating system of the PLC.

- For the initialisation the transfer parameter *bOpen* has to be set to TRUE.
- After this function has been carried out, index accesses via the system bus interface are accordingly diverted to other codes than to the ones which are automatically allocated, using the mapping table configured by means of the function **L\_CanDSxIndexInitCode**.

Declaration	
DWORD	<b>L_CanDSxOpen</b> (bOpen);

Transfer parameters	Data type	Information/possible settings
bOpen	Bool	Initialising the CanDSx driver in the operating system.
		TRUE The CanDSx driver in the operating system is initialised.

Return value	Data type	Value/meaning														
	Double word	Status														
		<table border="1"> <thead> <tr> <th>Bit</th> <th>Value</th> <th></th> </tr> </thead> <tbody> <tr> <td rowspan="2">0</td> <td>0</td> <td>Driver is initialised.</td> </tr> <tr> <td>1</td> <td>Driver is not initialised. • Remedy: function call via transfer parameter <i>bOpen</i> = TRUE.</td> </tr> <tr> <td>1-15</td> <td></td> <td>Reserved for future supplements (bits are set to 0). • Invalid for bit 0 = 1</td> </tr> <tr> <td>16-31</td> <td></td> <td>Version of the <b>LenzeCanDSxDrv.lib</b> function library • Format: main version/subversion (e. g. 0103hex = version 1.03) • Invalid for bit 0 = 1</td> </tr> </tbody> </table>	Bit	Value		0	0	Driver is initialised.	1	Driver is not initialised. • Remedy: function call via transfer parameter <i>bOpen</i> = TRUE.	1-15		Reserved for future supplements (bits are set to 0). • Invalid for bit 0 = 1	16-31		Version of the <b>LenzeCanDSxDrv.lib</b> function library • Format: main version/subversion (e. g. 0103hex = version 1.03) • Invalid for bit 0 = 1
Bit	Value															
0	0	Driver is initialised.														
	1	Driver is not initialised. • Remedy: function call via transfer parameter <i>bOpen</i> = TRUE.														
1-15		Reserved for future supplements (bits are set to 0). • Invalid for bit 0 = 1														
16-31		Version of the <b>LenzeCanDSxDrv.lib</b> function library • Format: main version/subversion (e. g. 0103hex = version 1.03) • Invalid for bit 0 = 1														



### Note!

If the function **L\_CanDSxOpen** is called up while write or read requests for codes are still active, they are possibly disturbed!

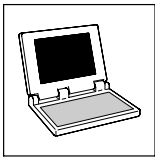
Therefore all actions with a code access should be completed before this function is called.

### Example

Calling the function in ST:

```

IF bOpenCanDSxDriver AND NOT bOpen THEN
  bOpen := TRUE;
  dwReturnOpen := L_CanDSxOpen (bOpen :=TRUE);
END_IF
    
```



# System bus (CAN) for Lenze PLC devices

## LenzeCanDSxDrv.lib function library

### L\_CanDSxClose - deactivating the index mapping

## 11.5 L\_CanDSxClose - deactivating the index mapping

### Function

By means of this function the mapping table and therefore the diversion of indexes is deactivated again.

- For the deactivation the transfer parameter *bClose* has to be set to TRUE.
- After carrying out this function, index accesses via the system bus interface according to the mapping table are **not** diverted to other codes anymore.

Declaration	
<pre>BOOL L_CanDSxClose (bClose);</pre>	

Transfer parameters	Data type	Information/possible settings
bClose	Bool	Deactivating the index diversions according to the mapping table.
		TRUE   The CanDSx driver in the operating system is deactivated.

Return value	Data type	Value/meaning
	Bool	Status
		TRUE   The CanDSx driver in the operating system has been deactivated.
		FALSE   The CanDSx driver in the operating system has not been deactivated. • Remedy: function call via transfer parameter <i>bClose</i> = TRUE.



### Note!

If the function **L\_CanDSxClose** is called up while write or read requests for codes are still active, they are possibly disturbed!

Therefore all actions with a code access should be completed before this function is called.

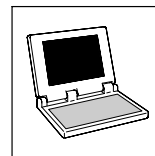
### Example

Calling the function in ST:

```

IF bCloseCanDSxDriver AND NOT bClose THEN
  bClose := TRUE;
  dwReturnClose := L_CanDSxClose (bClose:=TRUE) ;
END_IF

```



## 11.6 L\_CanDSxOpenHeartBeat - initialising a "Heartbeat"

### Function

In the CANopen communication profile (CiA DS301, version 4.01) two optional monitoring mechanisms for ensuring the function of system bus nodes are specified, "Heartbeat" and "Node Guarding".

By means of this function, the "Heartbeat" monitoring mechanism of the CanDSx driver is initialised.

- For the initialisation the transfer parameter *bOpen* has to be set to TRUE.
- The actual monitoring is carried out using the **L\_CanDSxHeartBeat** FB. (□ 11-8)
- By means of the function **L\_CanDSxCloseHeartBeat** you can deactivate the "Heartbeat" monitoring mechanism again. (□ 11-10)



### Note!

Using both monitoring mechanisms at the same is not permitted!

If a non-zero transmission cycle time for the "Heartbeat" message is configured for the node to be monitored, the "Heartbeat" mechanism is used prior to the "Node Guarding" mechanism.

#### Declaration

```
BOOL L_CanDSxOpenHeartBeat (bOpen);
```

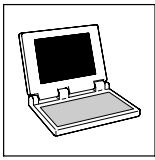
Transfer parameters	Data type	Information/possible settings
bOpen	Bool	Initialising the "Heartbeat" monitoring mechanism.
		TRUE The "Heartbeat" monitoring mechanism of the CanDSx driver is initialised.

Return value	Data type	Value/meaning	
	Bool	Status	
		TRUE	The "Heartbeat" monitoring mechanism has been initialised.
		FALSE	A The "Heartbeat" monitoring mechanism has not been initialised. – Remedy: function call with transfer parameter <i>bOpen</i> = TRUE. <i>or</i> B Beforehand the function <b>L_CanDSxOpenNodeGuarding</b> has been called ("Node Guarding" is activated). – Remedy: function call <b>L_CanDSxCloseNodeGuarding</b> with transfer parameter <i>bClose</i> = TRUE (deactivate "Node Guarding").

### Example

Calling the function in ST:

```
bReturnOpenHeartBeat := L_CanDSxOpenHeartBeat (bOpen := TRUE);
```



# System bus (CAN) for Lenze PLC devices

## LenzeCanDSxDrv.lib function library

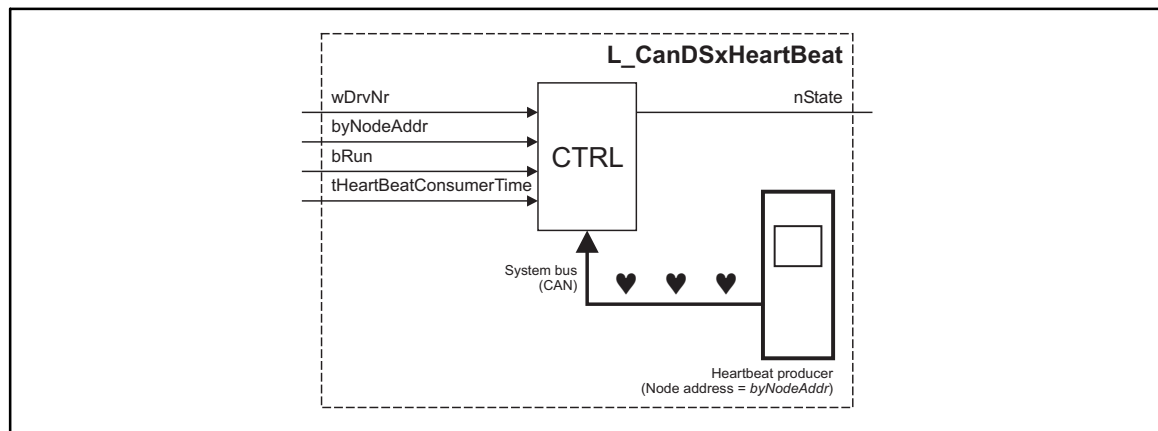
### L\_CanDSxHeartBeat - carrying out a "Heartbeat"

## 11.7 L\_CanDSxHeartBeat - carrying out a "Heartbeat"

### Function block

Use this FB to cyclically monitor the CAN connection between the PLC and other system bus nodes by means of the so-called "Heartbeat" mechanism.

- Here the FB assumes the function of the "heartbeat consumer" and therefore has to be called up on the monitoring PLC.
- The "Heartbeat" monitoring mechanism first has to be initialised in the CanDSx driver using the function **L\_CanDSxOpenHeartBeat**. (11-7)

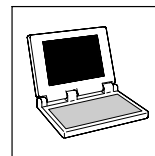


<b>FB call in:</b>	<input type="checkbox"/> Cyclic task (PLC_PRG)	<input checked="" type="checkbox"/> Time-controlled task (INTERVAL)	<input type="checkbox"/> Event-controlled task (EVENT)	<input type="checkbox"/> Interrupt task
--------------------	--	---	--	---

Inputs	Data type	Information/possible settings
wDrv	Word	Driver number for the CAN interface of the PLC that is to be used 10   On board system bus (CAN)
byNodeAddr	Byte	Node address of the node to be monitored. 1...128   Node address
bRun	Bool	Activate "Heartbeat" monitoring. TRUE   Monitoring is activated. • The FB now continuously monitors whether the node with the node address <i>byNodeAddr</i> sends its "Heartbeat" message within the set monitoring time <i>tHeartBeatConsumerTime</i> via the system bus and outputs a corresponding status at output <i>nState</i> .
tHeartBeat ConsumerTime	Time	Monitoring time within which the "Heartbeat" message has to arrive from the node to be monitored, so that no "Heartbeat" event ( <i>nState</i> = -10) is actuated. • Adjust this time in accordance with the "heartbeat producer time", by which the node to be monitored sends the "Heartbeat" message. • Recommended setting: 200 ... 2000 ms • Maximum setting: 65535 ms

# System bus (CAN) for Lenze PLC devices

## LenzeCanDSxDrv.lib function library L\_CanDSxHeartBeat - carrying out a "Heartbeat"



Outputs	Data type	Information/possible settings	
nState	Integer	Status	
		300	FB is deactivated ( <i>bRun</i> = FALSE).
		127	Node to be monitored is in the <i>Pre-operational</i> CAN status.
		5	Node to be monitored is in the <i>Operational</i> CAN status.
		4	Node to be monitored is in the <i>Stopped</i> CAN status.
		0	Node to be monitored is in the <i>Boot-up</i> CAN status, or the FB is not called up.
		-5	The monitoring time <i>tHeartBeatConsumerTime</i> is set to the value "0".
		-10	"Heartbeat" event: No "Heartbeat" message was received from the node to be monitored within the monitoring time <i>tHeartBeatConsumerTime</i> .
		-12	The node address set ( <i>byNodeAddr</i> ) is invalid.
		-120	The monitoring mechanism has not been initialised in the CanDSx driver. <ul style="list-style-type: none"> <li>Initialise the monitoring mechanism using the function <b>L_CanDSxOpenHeartBeat</b>.</li> </ul>
-121	The set driver number ( <i>wDrvNr</i> ) is invalid.		

### Settings required for the PLC to be monitored

(valid for 9300 Servo PLC, Drive PLC, ECSxA as of V6.2)

The following settings are required for the PLC to be monitored, thus causing the PLC to take over the function of the "Heartbeat producer":

1. Set code C0352 in the PLC to be monitored to the value "3" to configure the corresponding PLC as a "slave and heartbeat producer":

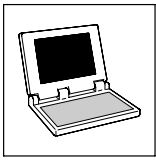
Code	LCD	Possible settings		Information	
		Lenze	Selection		
C0352	CAN mst	0		System bus: master/slave configuration of the PLC	
			0		Slave (boot-up not active)
			1		Master (boot-up active)
			2		Master with node guarding (SyncReceived no longer possible)
			3		<b>Slave and heartbeat producer</b>
4	Slave with node guarding				

2. Set the time interval for sending the "Heartbeat" message in the PLC to be monitored via C0381; according to the bus load, we recommend a setting in the range of 200 ... 2000 ms:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0381	HeartProTime	0		System bus: heartbeat (slave): heartbeat producer time
			0	

3. Set code C0003 in the PLC to be monitored to the value "1" to save the effected changes safe against mains failure.
4. Set code C0358 in the PLC to be monitored to the value "1" to carry out a CAN reset node.
  - Alternatively, the CAN reset node can also be carried out by mains switching.

After carrying out the CAN reset node, the PLC to be monitored as a "heartbeat producer" continuously sends the "Heartbeat" telegram via the system bus within the time interval set under C0381. Then, by means of this FB, the "heartbeat" monitoring can be activated within the monitoring PLC.



# System bus (CAN) for Lenze PLC devices

## LenzeCanDSxDrv.lib function library

### L\_CanDSxCloseHeartBeat - deactivating the "Heartbeat"

## 11.8 L\_CanDSxCloseHeartBeat - deactivating the "Heartbeat"

### Function

By means of this function, the "Heartbeat" monitoring mechanism of the CanDSx driver is deactivated again.

- For the deactivation the transfer parameter *bClose* has to be set to TRUE.

Declaration
<pre>BOOL L_CanDSxCloseHeartBeat (bClose);</pre>

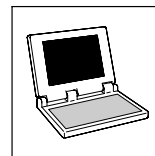
Transfer parameters	Data type	Information/possible settings
bClose	Bool	Deactivating the "Heartbeat" monitoring mechanism.
		<table border="1"> <tr> <td>TRUE</td> <td>The "Heartbeat" monitoring mechanism of the CanDSx driver in the operating system is deactivated.</td> </tr> </table>
TRUE	The "Heartbeat" monitoring mechanism of the CanDSx driver in the operating system is deactivated.	

Return value	Data type	Value/meaning		
	Bool	Status		
		<table border="1"> <tr> <td>TRUE</td> <td>The "Heartbeat" monitoring mechanism of the CanDSx driver has been deactivated.</td> </tr> </table>	TRUE	The "Heartbeat" monitoring mechanism of the CanDSx driver has been deactivated.
		TRUE	The "Heartbeat" monitoring mechanism of the CanDSx driver has been deactivated.	
<table border="1"> <tr> <td>FALSE</td> <td>The "Heartbeat" monitoring mechanism of the CanDSx driver has not been deactivated. <ul style="list-style-type: none"> <li>• Remedy: function call via transfer parameter <i>bClose</i> = TRUE.</li> </ul> </td> </tr> </table>	FALSE	The "Heartbeat" monitoring mechanism of the CanDSx driver has not been deactivated. <ul style="list-style-type: none"> <li>• Remedy: function call via transfer parameter <i>bClose</i> = TRUE.</li> </ul>		
FALSE	The "Heartbeat" monitoring mechanism of the CanDSx driver has not been deactivated. <ul style="list-style-type: none"> <li>• Remedy: function call via transfer parameter <i>bClose</i> = TRUE.</li> </ul>			

### Example

Calling the function in ST:

```
bReturnCloseHeartBeat := L_CanDSxCloseHeartBeat (bClose:=TRUE);
```



## 11.9 L\_CanDSxOpenNodeGuarding - initialising the "Node Guarding"

### Function

In the CANopen communication profile (CiA DS301, version 4.01) two optional monitoring mechanisms for ensuring the function of system bus nodes are specified, "Heartbeat" and "Node Guarding".

By means of this function the "Node Guarding" monitoring mechanism of the CanDSx driver is initialised.

- For the initialisation, the transfer parameter *bOpen* has to be set to TRUE.
- The actual monitoring is carried out using the **L\_CanDSxNodeGuarding** FB. (▣ 11-12)
- By means of the function **L\_CanDSxCloseNodeGuarding** you can deactivate the "Node Guarding" monitoring mechanism again. (▣ 11-15)



### Note!

Using both monitoring mechanisms at the same is not permitted!

If a non-zero transmission cycle time for the "Heartbeat" message is configured for the node to be monitored, the "Heartbeat" mechanism is used prior to the "Node Guarding" mechanism.

Declaration
<pre>BOOL L_CanDSxOpenNodeGuarding (bOpen) ;</pre>

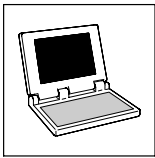
Transfer parameters	Data type	Information/possible settings
bOpen	Bool	Initialising the "Node Guarding" monitoring mechanism
		TRUE The "Node Guarding" monitoring mechanism of the CanDSx driver is initialised.

return value	Data type	Value/meaning	
	Bool	Status	
		TRUE	The "Node Guarding" monitoring mechanism has been initialised.
		FALSE	A The "Node Guarding" monitoring mechanism has not been initialised. – Remedy: function call with transfer parameter <i>bOpen</i> = TRUE. <i>or</i> B The PLC is not configured as "Master with Node Guarding" anymore. – Remedy: set code C352 to the value "2" to configure the PLC as "Master with Node Guarding".

### Example

Calling the function in ST:

```
bReturnOpenNodeGuarding := L_CanDSxOpenNodeGuarding (bOpen:=TRUE) ;
```



# System bus (CAN) for Lenze PLC devices

## LenzeCanDSxDrv.lib function library

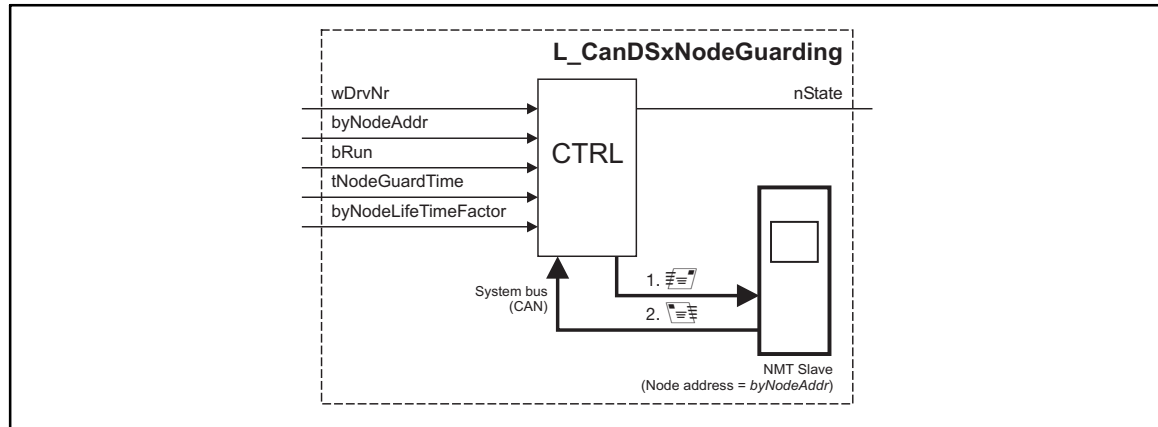
### L\_CanDSxNodeGuarding - carrying out a "Node guarding"

## 11.10 L\_CanDSxNodeGuarding - carrying out a "Node guarding"

### Function block

Use this FB to cyclically monitor the CAN connection between the PLC and other system bus nodes by means of the so-called "Node guarding" mechanism.

- This monitoring mechanism first has to be initialised in the CanDSx driver using the function **L\_CanDSxOpenNodeGuarding**. (📖 11-11)

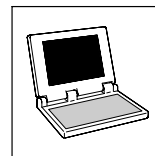


<b>FB call in:</b>	<input type="checkbox"/> Cyclic task (PLC_PRG)	<input checked="" type="checkbox"/> Time-controlled task (INTERVAL)	<input type="checkbox"/> Event-controlled task (EVENT)	<input type="checkbox"/> Interrupt task
--------------------	--	---	--	---

Inputs	Data type	Information/possible settings
wDrv	Word	Driver number for the CAN interface of the PLC that is to be used 10 On board system bus (CAN)
byNodeAddr	Byte	Node address of the node to be monitored. 1...128 Node address
bRun	Bool	Activating the "Node guarding". TRUE Monitoring is activated. <ul style="list-style-type: none"> <li>The FB now sends a "remote transmission request" telegram in the transmission cycle set via <i>tNodeGuardTime</i> to the node with the node address <i>byNodeAddr</i> and waits for a corresponding response. If it is not effected within the "NodeLifeTime" monitoring time, the FB outputs a corresponding status at the output <i>nState</i>.</li> </ul>
tNodeGuardTime	Time	Transmission cycle of the remote transmission request telegram. <ul style="list-style-type: none"> <li>Time interval in which the PLC sends a status request to the node to be monitored (cyclic polling).</li> <li>This setting has to correspond to the "Guard Time" set in the PLC to be monitored (C0382).</li> </ul>
byNodeLifeTimeFactor	Byte	Factor for the so-called "NodeLifeTime". "NodeLifeTime" = <i>byNodeLifeTimeFactor</i> · <i>tNodeGuardTime</i> <ul style="list-style-type: none"> <li>If the node to be monitored does not respond to the status request of the PLC within the "NodeLifeTime", a "Node Guarding" event is actuated (<i>nState</i> = -10).</li> <li>This setting has to correspond to the "NodeLifeTimeFactor" set in the PLC to be monitored (C0383).</li> </ul>

# System bus (CAN) for Lenze PLC devices

**LenzeCanDSxDrv.lib function library**  
**L\_CanDSxNodeGuarding - carrying out a "Node guarding"**



Outputs	Data type	Information/possible settings	
nState	Integer	Status	
		300	FB is deactivated ( <i>bRun</i> = FALSE).
		127	Node to be monitored is in the <i>Pre-operational</i> CAN status.
		5	Node to be monitored is in the <i>Operational</i> CAN status.
		4	Node to be monitored is in the <i>Stopped</i> CAN status.
		0	Node to be monitored is in the <i>Boot-up</i> CAN status, or the FB is not called up.
		-5	The <i>tNodeGuardTime</i> monitoring time or the factor <i>byNodeLifeTimeFactor</i> is set to the value "0".
		-9	Response of the node to be monitored is invalid.
		-10	"Node Guarding" event: No status response from the node to be monitored was received within the "NodeLifeTime" monitoring time.
		-12	The set node address ( <i>byNodeAddr</i> ) is invalid.
-120	The monitoring mechanism has not been initialised in the CanDSx driver. <ul style="list-style-type: none"> <li>Initialise the monitoring mechanism using the function <b>L_CanDSxOpenNodeGuarding</b>.</li> </ul>		
-121	The set driver number ( <i>wDrvNr</i> ) is invalid.		

## Settings required for the PLC to be monitored

(valid for 9300 Servo PLC, Drive PLC, ECSxA as of V6.2)

The following settings are required for the PLC to be monitored, thus causing the PLC to take over the function of the "Node Guarding Slave":

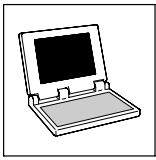
1. Set code C0352 in the PLC to be monitored to the value "4" to configure the corresponding PLC as a "Slave with Node Guarding":

Code	LCD	Possible settings		Information	
		Lenze	Selection		
C0352	CAN mst	0		System bus: master/slave configuration of the PLC	
			0		Slave (boot-up not active)
			1		Master (boot-up active)
			2		Master with Node Guarding (SyncReceived no longer possible)
			3		Slave and heartbeat producer
4	<b>Slave with Node Guarding</b>				

2. Set the time interval for the status enquiry by the master in the PLC to be monitored via C0382. This value has to correspond to the setting at the FB input *tNodeGuardTime* in the monitoring PLC:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0382	GuardTime	0		System bus: Node Guarding (slave): NodeGuardTime
			0	

3. Set the "Node Guarding Master" in the monitoring PLC with C0352/0 = 2.



## System bus (CAN) for Lenze PLC devices

### LenzeCanDSxDrv.lib function library

#### L\_CanDSxNodeGuarding - carrying out a "Node guarding"

- Set the factor for the "NodeLifeTime" monitoring time via C0383 in the PLC to be monitored. This value has to correspond to the setting at the FB input *byNodeLifeTimeFactor* in the monitoring PLC:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0383	LifeTimeFact.	0		System bus: Node Guarding (slave): NodeLifeTimeFactor
		0	{1} 255	



### Tip!

From the settings for the "NodeGuardTime" (C0382) and the "NodeLifeTimeFactor" (C0383) in the PLC to be monitored, the so-called "NodeLifeTime" results:

$$NodeLifeTime = NodeGuardTime(C0382) \cdot NodeLifeTimeFactor (C0383)$$

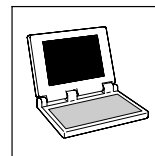
- If the PLC to be monitored does not receive a status enquiry from the monitoring PLC within this "NodeLifeTime", a so-called "Life Guarding" event is actuated in the PLC to be monitored.

- Set the desired response by which the PLC to be monitored is to react to a "Life Guarding Event" via C0384 in the PLC to be monitored:

Code	LCD	Possible settings		Information
		Lenze	Selection	
C0384	Err NodeGuard	3		System bus: Node Guarding (slave): response to a "Life Guard" event.
			0 TRIP	
			1 Message	
			2 Warning	
			3 Off	
			4 Fail-QSP	

- Set code C0003 in the PLC to be monitored to the value "1" to save the effected changes in a manner safe against mains failure.
- Set code C0358 in the PLC to be monitored to the value "1" to carry out a CAN reset node.
  - Alternatively, the CAN reset node can also be carried out by mains switching.

After the CAN reset node has been carried out, the PLC to be monitored is configured as "node guarding slave", and monitoring in the "node guarding master" can be activated using this FB.



## 11.11 L\_CanDSxCloseNodeGuarding - deactivating the "Node Guarding"

### Function

By means of this function, the "Node Guarding" monitoring mechanism of the CanDSx driver is deactivated again.

- For the deactivation the transfer parameter *bClose* has to be set to TRUE.

Declaration	
<pre>BOOL L_CanDSxCloseNodeGuarding (bClose);</pre>	

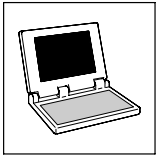
Transfer parameters	Data type	Information/possible settings
bClose	Bool	Deactivating the "Node Guarding" monitoring mechanism
		TRUE   The "Node Guarding" monitoring mechanism of the CanDSx driver is deactivated.

Return value	Data type	Value/meaning
	Bool	Status
		TRUE   The "Node Guarding" monitoring mechanism of the CanDSx driver has been deactivated.
		FALSE   The "Node Guarding" monitoring mechanism of the CanDSx driver has not been deactivated. • Remedy: function call via transfer parameter <i>bClose</i> = TRUE.

### Example

Calling the function in ST:

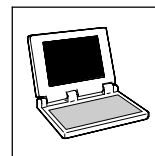
```
bReturnCloseNodeGuarding := L_CanDSxCloseNodeGuarding (bClose:=TRUE);
```



## ***System bus (CAN) for Lenze PLC devices***

***LenzeCanDSxDrv.lib function library***

***L\_CanDSxCloseNodeGuarding - deactivating the "Node Guarding"***



## 12 Index

### A

Addressing parameters, 2-13

Axis synchronisation, 7-23

### B

Baud rate

AIF interface, 4-1

CAN interface, 3-1

CAN-AUX interface, 6-1

FIF interface, 5-1

Boot-up

AIF interface, 4-2

CAN interface, 3-2

CAN-AUX interface, 6-2

FIF interface, 5-2

Bus load

CAN interface, 3-15

CAN-AUX interface, 6-13

FIF interface, 5-13

Bus off, AIF interface, 4-9

Bus-off

CAN interface, 3-11

CAN-AUX interface, 6-9

FIF interface, 5-9

### C

C0350, 3-3

C0351, 3-1

C0352, 3-2

C0353, 3-4

C0354, 3-4

C0355, 3-5

C0356, 3-2, 3-6

C0357, 3-11

C0358, 3-8, 4-8

C0359, 3-13

C0360, 3-14

C0361, 3-15

C0363, 7-26

C0366, 3-7, 7-26

C0367, 3-7, 7-26

C0368, 3-7, 7-26

C0369, 3-7, 7-26

C0591, 3-11

C0592, 3-11

C0593, 3-11

C0595, 3-11

C0608, 10-4

C0609, 10-4

C1120, 7-23

C1121, 7-25

C1122, 7-25

C1123, 7-25

C2121, 4-10

C2350, 4-3

C2351, 4-1

C2352, 4-2

C2353, 4-4

C2354, 4-4

C2355, 4-5

C2356, 4-2, 4-6, 4-7

C2357, 4-8

C2367, 4-7

C2368, 4-7

C2375, 4-7

C2382, 4-8, 4-9

C2450, 5-3, 6-3

C2451, 5-1, 6-1

C2452, 5-2, 6-2

C2453, 5-4, 6-4

C2454, 5-4, 6-4

C2455, 5-5, 6-5

C2456, 5-2, 5-6, 6-2, 6-6

C2457, 5-9, 6-9

C2458, 5-8, 6-8

C2459, 5-11, 6-11

C2460, 5-12, 6-12

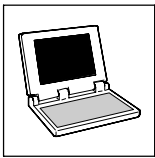
C2461, 5-13, 6-13

C2466, 5-7, 6-7

C2467, 5-7, 6-7

C2468, 5-7, 6-7

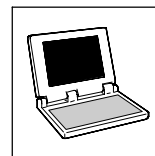
C2469, 5-7, 6-7



# System bus (CAN) for Lenze PLC devices

## Index

- C2481, 5-9 , 6-9
- C2482, 5-9 , 6-9
- C2483, 5-9 , 6-9
- C2484, 5-9 , 6-9
- CAN objects, application recommendations, 2-20
- CAN sync identifier, 7-26
- CAN sync identifiers, 3-7
- CAN sync response, 3-7 , 7-26
- CAN sync Tx transmission cycle, 3-7 , 7-26
- CAN system blocks, 7-1
  - CAN\_Management, 7-20
  - CAN\_Synchronization, 7-23
  - CAN1\_IO (9300 Servo PLC), 7-1
  - CAN1\_IO (Drive PLC), 7-6
  - CAN1\_IO (ECSxA), 7-10
  - CAN2\_IO, 7-14
  - CAN23\_IO, 7-17
- CAN telegram, 2-3
- CAN-AUX system blocks, 9-1
  - CANaux\_Management, 9-4
  - CANaux1\_IO, 9-1
  - CANaux2\_IO, 9-1
  - CANaux3\_IO, 9-1
- CAN\_bResetNode\_b, 7-21
- CAN\_bTxCan2Synchronized\_b, 7-21
- CAN\_bTxCan3Synchronized\_b, 7-21
- CAN\_Management, 7-20
- CAN\_Synchronization, 7-23
- CAN1\_IO (9300 Servo PLC), 7-1
- CAN1\_IO (Drive PLC), 7-6
- CAN1\_IO (ECSxA), 7-10
- CAN2\_IO, 7-14
- CAN3\_IO, 7-17
- CANaux sync identifiers, 6-7
- CANaux sync response, 6-7
- CANaux sync Tx transmission cycle, 6-7
- CANaux\_bResetNode\_b, 9-5
- CANaux\_bTxCan2Synchronized\_b, 9-5
- CANaux\_bTxCan3Synchronized\_b, 9-5
- CANaux\_Management, 9-4
- CANaux1\_IO, 9-1
- CANaux2\_IO, 9-1
- CANaux3\_IO, 9-1
- COB-ID, 2-3
- Command, 2-6
- Command code, 2-12
- Communication Object Identifier, 2-3
- Configuring the AIF interface, 4-1
  - baud rate, 4-1
  - boot-up, 4-2
  - cycle time, 4-6
  - diagnostics, 4-10
    - operating status, 4-10
  - identifiers of the process data objects, 4-4
  - monitoring processes, 4-8
    - bus off, 4-9
    - fault messages, 4-9
    - time monitoring, 4-8
  - node address (node ID), 4-3
  - reset node, 4-8
  - synchronisation, 4-7
    - XCAN sync identifier, 4-7
    - XCAN sync response, 4-7
    - XCAN sync Tx transmission cycle, 4-7
- Configuring the CAN interface, 3-1
  - baud rate, 3-1
  - boot-up, 3-2
  - cycle time, 3-6
  - delay time, 3-6
  - diagnostics, 3-13
    - bus load, 3-15
    - operating status, 3-13
    - telegram counter, 3-14
  - identifiers of the process data objects, 3-4
  - mapping indexes to codes, 3-8
  - monitoring processes, 3-11
    - bus-off, 3-11
    - fault messages, 3-12
    - time monitoring, 3-11
    - time-out during activated remote parameterisation, 3-12
  - node address (node ID), 3-3
  - remote parameterisation (gateway function), 3-10
  - reset node, 3-8
  - synchronisation, 3-7
    - CAN sync identifiers, 3-7
    - CAN sync response, 3-7
    - CAN sync Tx transmission cycle, 3-7
  - system bus management, 3-8
- Configuring the CAN-AUX interface, 6-1
  - baud rate, 6-1
  - boot-up, 6-2
  - cycle time, 6-6
  - delay time, 6-6
  - diagnostics, 6-11
    - bus load, 6-13
    - operating status, 6-11
    - telegram counter, 6-12
  - identifiers of the process data objects, 6-4
  - monitoring processes, 6-9
    - bus-off, 6-9
    - fault messages, 6-10
    - time monitoring, 6-9
  - node address (node ID), 6-3
  - reset node, 6-8
  - synchronisation, 6-7
    - CANaux sync identifiers, 6-7
    - CANaux sync response, 6-7
    - CANaux sync Tx transmission cycle, 6-7
  - system bus management, 6-8



### Configuring the FIF interface, 5-1

- baud rate, 5-1
- boot-up, 5-2
- cycle time, 5-6
- delay time, 5-6
- diagnostics, 5-11
  - bus load, 5-13
  - operating status, 5-11
- diagnostics , telegram counter, 5-12
- identifiers of the process data objects, 5-4
- monitoring processes, 5-9
  - bus-off, 5-9
  - fault messages, 5-10
  - time monitoring, 5-9
- node address (node ID), 5-3
- reset node, 5-8
- synchronisation, 5-7
  - FIF-CAN sync identifier, 5-7
  - FIF-CAN sync response, 5-7
  - FIF-CAN sync Tx transmission cycle, 5-7
- system bus management, 5-8

### Correction value of the phase controller, 7-26

### Cycle time

- AIF interface, 4-6
- CAN interface, 3-6
- CAN-AUX interface, 6-6
- FIF interface, 5-6

## D

### Data of the parameter, 2-14

### Delay time

- CAN interface, 3-6
- CAN-AUX interface, 6-6
- FIF interface, 5-6

### Device address, 2-6

### Diagnostics

- AIF interface, 4-10
  - operating status, 4-10
- CAN interface, 3-13
  - bus load, 3-15
  - operating status, 3-13
  - telegram counter, 3-14
- CAN-AUX interface, 6-11
  - bus load, 6-13
  - operating status, 6-11
  - telegram counter, 6-12
- FIF interface, 5-11 , 5-12
  - bus load, 5-13
  - operating status, 5-11

## E

### Error Response, 2-12

## F

### Fault messages, 6-10

- AIF interface, 4-9
- CAN interface, 3-12
- CAN-AUX interface, 6-10
- FIF interface, 5-10

### FIF-CAN sync identifier, 5-7

### FIF-CAN sync response, 5-7

### FIF-CAN sync Tx transmission cycle, 5-7

### FIF-CAN system blocks, 8-1

- FIF\_CAN\_Management, 8-4
- FIF\_CAN1\_IO, 8-1
- FIF\_CAN2\_IO, 8-1
- FIF\_CAN3\_IO, 8-1

### FIF\_CAN\_bResetNode\_b, 8-5

### FIF\_CAN\_Management, 8-4

### FIF\_CAN1\_IO, 8-1

### FIF\_CAN2\_IO, 8-1

### FIF\_CAN3\_IO, 8-1

### Forced transmission, 10-11

### Free CAN objects, 2-19

## G

### Gateway function, CAN interface, 3-10

## H

### Heartbeat, 2-21

## I

### Identifier, 2-3 , 2-10

### Identifiers, 2-11

- AIF interface, 4-4
- CAN interface, 3-4
- CAN-AUX interface, 6-4
- FIF interface, 5-4

### Index, 2-13

## L

### L\_CanClose, 10-5

### L\_CanGetRelocCobld, 10-7

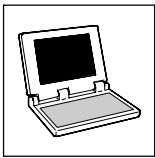
### L\_CanGetStatus, 10-6

### L\_CanInit, 10-2

### L\_CanPdoReceive, 10-12

### L\_CanPdoTransmit, 10-8

### L\_ParWrite, 10-10



# System bus (CAN) for Lenze PLC devices

## Index

### M

Mapping indexes to codes, CAN interface, 3-8

Monitoring mechanisms, 2-21

    "Heartbeat", 2-21

    "Node Guarding", 2-22

Monitoring processes

    AIF interface, 4-8  
        bus off, 4-9  
        fault messages, 4-9  
        time monitoring, 4-8

    CAN interface, 3-11  
        bus-off, 3-11  
        fault messages, 3-12  
        time monitoring, 3-11  
        time-out during activated remote parameterisation, 3-12

    CAN-AUX interface, 6-9  
        bus-off, 6-9  
        fault messages, 6-10  
        time monitoring, 6-9

    FIF interface, 5-9  
        bus-off, 5-9  
        fault messages, 5-10  
        time monitoring, 5-9

### N

Network management (NMT), 2-6

    command, 2-6

    device address, 2-6

Network status, 2-6

Node address (node ID)

    AIF interface, 4-3

    CAN interface, 3-3

    CAN-AUX interface, 6-3

    FIF interface, 5-3

Node Guarding, 2-22

Node ID, 2-3, 6-3

### P

Parameter data, 2-5, 2-11

    addressing (index/subindex), 2-13

    command code, 2-12

    data format, 2-14

    identifiers, 2-11

    reading parameters, 2-17

    telegram structure, 2-11

    transmission, 2-11

    writing parameters, 2-15

PDOs, 2-5

Phase displacement, 7-25

Process data, 2-5

    identifier, 2-10

    sync telegram, 2-9

    telegram structure, 2-10

    transmission, 2-7

    user data, 2-10

Process data channels, 2-7

Process Data Objects, 2-5

### R

Read request, 2-12

Read response, 2-12

Reading parameters, 2-17

Receive status, 10-13

Remote parameterisation (gateway function), CAN interface, 3-10

Reset node

    AIF interface, 4-8

    CAN interface, 3-8

    CAN-AUX interface, 6-8

    FIF interface, 5-8

### S

Safety information, layout, More notes, 1-4

Safety instructions, layout, warning of material damage, 1-4

SDOs, 2-5

Service Data Objects, 2-5

Subindex, 2-13

Sync telegram, 2-9

Synchronisation

    AIF interface, 4-7

    CAN interface, 3-7

    CAN sync identifiers, 3-7

    CAN sync response, 3-7

    CAN sync Tx transmission cycle, 3-7

    CAN-AUX interface, 6-7

    CANaux sync identifiers, 6-7

    CANaux sync response, 6-7

    CANaux sync Tx transmission cycle, 6-7

    FIF interface, 5-7

    FIF-CAN sync identifier, 5-7

    FIF-CAN sync response, 5-7

    FIF\_CAN sync Tx transmission cycle, 5-7

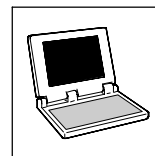
    XCAN sync identifier, 4-7

    XCAN sync response, 4-7

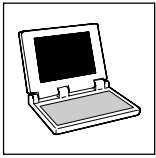
    XCAN sync Tx transmission cycle, 4-7

Synchronisation cycle, 7-25

Synchronisation time, 7-24



- System bus (CAN), 2-1
    - CAN objects, application recommendations, 2-20
    - cycle time
      - AIF interface, 4-6
      - CAN interface, 3-6
      - CAN-AUX interface, 6-6
      - FIF interface, 5-6
    - delay time
      - CAN interface, 3-6
      - CAN-AUX interface, 6-6
      - FIF interface, 5-6
    - diagnostics
      - AIF interface, 4-10
      - CAN interface, 3-13
      - CAN-AUX interface, 6-11
      - FIF interface, 5-11
    - free CAN objects, 2-19
    - identification of the nodes, 2-3
    - identifier, 2-3
    - identifiers
      - AIF interface, 4-4
      - CAN interface, 3-4
      - CAN-AUX interface, 6-4
      - FIF interface, 5-4
    - interfaces for system bus connection, 2-2
      - configuring the AIF interface, 4-1
      - configuring the CAN-AUX interface, 6-1
    - interfaces for system bus connection, Configuring the FIF interface, 5-1
    - interfaces for the system bus connection, configuring the CAN interface, 3-1
    - mapping indexes to codes, CAN interface, 3-8
    - monitoring mechanisms, 2-21
      - "Heartbeat", 2-21
      - "Node Guarding", 2-22
    - monitoring processes
      - AIF interface, 4-8
      - CAN interface, 3-11
      - CAN-AUX interface, 6-9
      - FIF interface, 5-9
    - network management (NMT), 2-6
      - command, 2-6
      - device address, 2-6
    - operating status
      - AIF interface, 4-10
      - CAN interface, 3-13
      - CAN-AUX interface, 6-11
      - FIF interface, 5-11
    - parameter data
      - addressing (index/subindex), 2-13
      - command code, 2-12
      - data format, 2-14
      - identifiers, 2-11
      - reading parameters, 2-17
      - telegram structure, 2-11
      - transmission, 2-11
      - writing parameters, 2-15
    - process data
      - identifier, 2-10
      - sync telegram, 2-9
      - telegram structure, 2-10
      - transmission, 2-7
      - user data, 2-10
    - remote parameterisation (gateway function), CAN interface, 3-10
    - reset node
      - AIF interface, 4-8
      - CAN interface, 3-8
      - CAN-AUX interface, 6-8
      - FIF interface, 5-8
    - synchronisation
      - AIF interface, 4-7
      - CAN interface, 3-7
      - CAN-AUX interface, 6-7
      - FIF interface, 5-7
    - system bus management
      - CAN interface, 3-8
      - CAN-AUX interface, 6-8
      - FIF interface, 5-8
    - telegram structure, 2-3
    - user data, 2-5
  - System bus management
    - CAN, 7-20
    - CAN interface, 3-8
    - CAN-AUX, 8-4 , 9-4
    - CAN-AUX interface, 6-8
    - FIF interface, 5-8
- ## T
- Telegram counter
    - CAN interface, 3-14
    - CAN-AUX interface, 6-12
    - FIF interface, 5-12
  - Term definitions, 1-4
  - Time monitoring
    - AIF interface, 4-8
    - CAN interface, 3-11
    - CAN-AUX interface, 6-9
    - FIF interface, 5-9
  - Time slot, 7-25
  - Time-out during activated remote parameterisation, CAN interface, 3-12
  - Transmission
    - parameter data, 2-11
    - process data, 2-7
  - Transmit request memory, 10-10
  - Transmit status, 10-9
- ## U
- User data, 2-5 , 2-10
- ## V
- Version identifiers of the function library, 10-1 , 11-2
- ## W
- Write request, 2-12
  - Write response, 2-12
  - Writing parameters, 2-15



# ***System bus (CAN) for Lenze PLC devices***

## ***Index***

### **X**

XCAN sync identifier, 4-7

XCAN sync response, 4-7

XCAN sync Tx transmission cycle, 4-7