

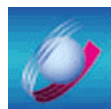
# Lenze

*Manual*

IEC 61131-3

*inside*

***Global Drive  
PLC Developer Studio***



***Global Drive***

*Function library*

*LenzeAifParMapDrv.lib*

The function library **LenzeAifParMapDrv.lib** can be used for the following Lenze PLCs:

	Type	from hardware version	from software version
<b>9300 Servo PLC</b>	EVS93XX-xT	6A	6.0
<b>Drive PLC</b>	EPL10200-ET	1A	6.0
<b>ECSxA</b>	ECSxAxxx	1C	7.0

### **Important note :**

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

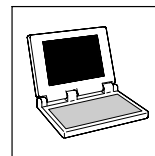
We do not take any liability for direct or indirect damage, e.g. profit loss, order loss or any loss regarding business.

© 2002 Lenze Drive Systems GmbH

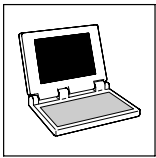
No part of this documentation may be copied or made available to third parties without the explicit written approval of Lenze Drive Systems GmbH.

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

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

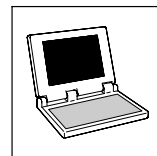
**Function library LenzeAifParMapDrv.lib****Contents**

<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 .....	1-1
1.1.2 Layout of the descriptions .....	1-2
1.1.3 Pictographs used in this Manual .....	1-2
1.1.4 Terminology used .....	1-2
1.2 Version identifiers of the function library .....	1-3
<b>2 Introduction</b> .....	<b>2-1</b>
2.1 General .....	2-1
2.2 AifParMap driver .....	2-1
2.2.1 Operating principle - example .....	2-2
<b>3 Functions</b> .....	<b>3-1</b>
3.1 L_AifParMapInit - configuration of code mapping .....	3-2
3.2 L_AifParMapOpen - activation of code mapping .....	3-3
3.3 L_AifParMapClose - deactivation of code mapping .....	3-4



# ***Function library LenzeAifParMapDrv.lib***

## ***Contents***



# 1 Preface and general information

## 1.1 About this Manual

This Manual contains information about the function library **LenzeAifParMapDrv.lib** for the **Drive PLC Developer Studio**.

- The function library **LenzeAifParMapDrv.lib** includes functions which can be used to transfer ("map") codes via the AIF interface to other PLC codes for parameter access from a higher-level control system.
- It is possible to map up to 256 codes to other codes within the PLC.

### 1.1.1 Conventions used in this Manual

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

#### Variable identifiers

are written in italics in the explanation:

- "Use *bReset*..."



#### Tip!

Information about the conventions used for the variables of the Lenze system blocks, function blocks and functions can be found in the appendix of the DDS online documentation "Introduction into IEC61131-3 programming". The conventions ensure universal and uniform labelling and make reading the PLC program easier.

#### Lenze functions/function blocks

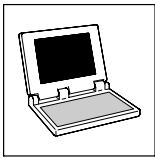
can be recognized by their names. They always begin with an "L\_":

- "With the function **L\_TBCnvBitsToByte**..."

#### Program listings

are written in "Courier", keywords are printed in bold:

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



# Function library LenzeAifParMapDrv.lib

## Preface and general information

### 1.1.2 Layout of the descriptions

All function/function block and system block descriptions contained in this Manual have the same structure:

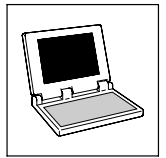
	Function	Function block (FB)/ System block (SB)	
	①	Headline stating the function and the function identifier	
	②	Declaration of the function: <ul style="list-style-type: none"> <li>Data type of the return value</li> <li>Function identifier</li> <li>List of transfer parameters</li> </ul>	-
	③	Short description of the most important properties	
	④	Function chart including all associated variables <ul style="list-style-type: none"> <li>Transfer parameters</li> <li>Return value</li> </ul>	FB/SB chart including all associated variables <ul style="list-style-type: none"> <li>Input variables</li> <li>Output variables</li> </ul>
	⑤	Table giving information about the transfer parameters: <ul style="list-style-type: none"> <li>Identifier</li> <li>Data type</li> <li>Possible settings</li> <li>Info</li> </ul>	Table giving information about the input and output variables: <ul style="list-style-type: none"> <li>Identifier</li> <li>Data type</li> <li>Type of variable</li> <li>Possible settings</li> <li>Info</li> </ul>
	⑥	Table giving information about the return value: <ul style="list-style-type: none"> <li>Data type of the return value</li> <li>Possible return values and their meaning:</li> </ul>	-
	⑦	Additional information (Notes, tips, application examples, etc.)	

### 1.1.3 Pictographs used in this Manual

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

### 1.1.4 Terminology used

Term	In the following text used for
DDS	Drive PLC Developer Studio
FB	Function block
GDC	Global Drive Control (parameterization program from Lenze)
Parameter codes	Codes for setting the functionality of a function block
PLC	<ul style="list-style-type: none"> <li>9300 Servo PLC</li> <li>Drive PLC</li> <li>ECSxA "Application" axis module</li> </ul>
SB	System block



## 1.2 Version identifiers of the function library

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

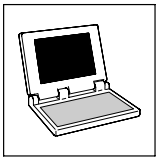
Version identifiers as of PLC software version 7.x:

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

Version: 01 05 00 00

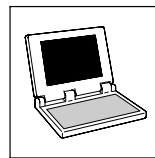
The value of this constant is a hexadecimal code.

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



# ***Function library LenzeAifParMapDrv.lib***

## ***Preface and general information***



## 2 Introduction

### 2.1 General

Lenze drive controllers and automation systems are parameterized via codes.

If a drive system is replaced by a PLC it might be necessary to adapt parameter access from a higher-level control system to the new IEC61131 application, e.g. because some functions might have to be parameterized via other codes than before.

The function library **LenzeAifParMapDrv.lib** includes functions which can be used to redirect ("map") codes via the AIF interface to other PLC codes for parameter access from a higher-level control system.

### 2.2 AifParMap driver

The operating system (as of V6.0) of Lenze PLCs includes a specific AifParMap driver which can be activated by using the functions of the function library **LenzeAifParMapDrv.lib**.

The driver can be used to redirect access to individual codes via the AIF interface to other PLC codes.

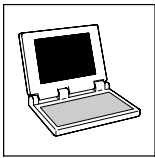


---

#### **Note!**

The function of the AifParMap driver is only available for the AIF interface.

---



## Function library *LenzeAifParMapDrv.lib*

### Introduction

#### *AifParMap driver*

## 2.2.1 Operating principle - example

### Task

A drive controller function is parameterized under code C0011/0. The drive controller is replaced by a PLC. From now on this function has to be parameterized under code C3011/3.

- The functionality is accessed from a higher-level control via the AIF interface.

The new PLC has to be adapted to the existing system environment so that the higher-level control program need not be changed.

### Solution

By using the functions of the function library **LenzeAifParMapDrv.lib** code C0011/0 can be easily mapped to PLC code C3011/3. In this way, the higher-level control system can access the PLC in the same way as before via the AIF interface.

### Operating principle

The operating system (as of V6.0) of Lenze PLCs includes a "mapping table" which allows to map up to 256 codes to other codes within the PLC.

If an AIF telegram is received and the code number is within the valid range, the system checks whether the code is listed in the mapping table.

- If the code is listed in the mapping table the system accesses the code which has been configured as corresponding mapping target code in the mapping table. ①
- If the code is **not** listed in the mapping table the code will be accessed without mapping. ②

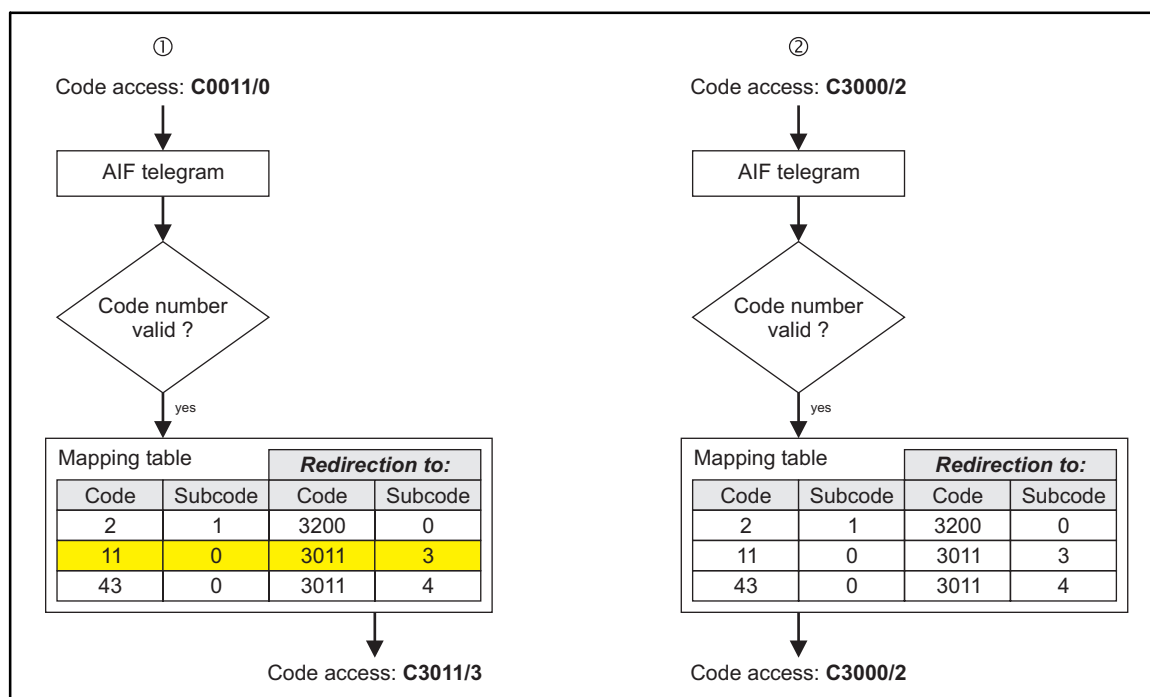
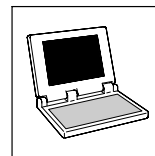


Fig. 2-1

Redirection of codes to other codes



### 3 Functions

The function library includes the following functions for code mapping:

- **L\_AifParMapInit**: Configuration of code mapping. (☞ 3-2)
- **L\_AifParMapOpen**: Activation of code mapping. (☞ 3-3)
- **L\_AifParMapClose**: Deactivation of code mapping. (☞ 3-4)



#### Note!

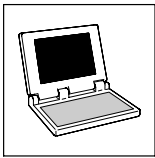
The functions are only available for code access via AIF interface and the corresponding fieldbus module.

- These functions do not reconfigure internal access to codes via the FBs **L\_ParRead** or **L\_ParWrite**.
- Codes which are not created in the PLC operating system but in the AIF fieldbus module cannot be mapped to other codes.



#### Tip!

For mapping CAN indices to codes please use the functions provided in the function library **LenzeCanDSxDrv.lib**.



## Function library *LenzeAifParMapDrv.lib*

### Functions

#### *L\_AifParMapInit* - configuration of code mapping

## 3.1 *L\_AifParMapInit* - configuration of code mapping

```
INT L_AifParMapInit (byIndex, wCode, bySubCode,
                    wCorrCode, byCorrSubCode)
```

This function is used to configure the mapping table and the redirection of codes.

- With every function call one code and the corresponding correction code can be entered in the mapping table.
- A maximum of 256 entries can be entered in the mapping table:

byIndex	Code to be redirected		Redirection target	
	wCode	bySubCode	wCorrCode	byCorrSubCode
0				
1	11	0	3011	3
2				
...				
255				

### Transfer parameters

Identifiers	Data type	Possible settings	Info
byIndex	Byte	0 ... 255	Number of the configuration entry in the mapping table.
Code to be redirected:			
wCode	Word	1 ... 7999	Code number
bySubCode	Byte	0 ... 255	Subcode number
Redirection target:			
wCorrCode	Word	1 ... 7999	Code number
byCorrSubCode	Byte	0 ... 255	Subcode number

### Return value

Data type	Value	Meaning
Integer	0	Entry in the mapping table has been successful.
	-20	Error: Transfer parameter <i>wCode</i> is invalid.
	-30	Error: Transfer parameter <i>wCorrCode</i> is invalid.



### Note!

If the function *L\_AifParMapInit* is called while read or write requests for codes are active an error may occur!

This is why all actions with code access should be completed before this function is called.

### Example

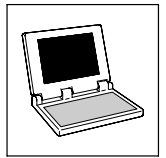
Calling the function in ST:

```
nReturnInitIndexCode := L_AifParMapInit (byIndex:=1,
                                         wCode:=11,
                                         bySubCode:=0,
                                         wCorrCode:=3011,
                                         byCorrSubCode:=3);
```

# Function library LenzeAifParMapDrv.lib

## Functions

### L\_AifParMapOpen - activation of code mapping



## 3.2 L\_AifParMapOpen - activation of code mapping

DWORD **L\_AifParMapOpen** (bOpen)

This function is used to activate the mapping table and the redirection of codes.

- The transfer parameter *bOpen* must be set to TRUE.
- After this function has been executed, the codes accessed will be redirected via the AIF interface to the corresponding codes listed in the mapping table.

#### Transfer parameters

Identifiers	Data type	Possible settings	Info
bOpen	Bool	TRUE	Activation of code redirections according to the mapping table. <ul style="list-style-type: none"> <li>• The AifParMap driver in the operating system is initialised.</li> </ul>

#### Return value

Data type	Bit	Value	Meaning
Double word	0	0	Driver is initialised.
		1	Driver is not initialised. • Remedy: Calling the function with transfer parameter <i>bOpen</i> = TRUE.
	1-15		Reserved for future extensions (bits are set to 0). • Invalid with bit 0 = 1
	16-31		Version of the function library LenzeAifParMapDrv.lib • Format: Main version/subversion (e. g. 0103hex = version 1.03) • Invalid with bit 0 = 1



#### Note!

If the function **L\_AifParMapOpen** is called while read or write requests for codes are active an error may occur!

This is why all actions with code access should be completed before this function is called.

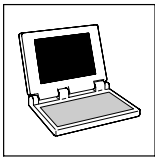
#### Example

Calling the function in ST:

```

IF bOpenParMapDriver AND NOT bOpen THEN
  bOpen := TRUE;
  dwReturnOpen := L_AifParMapOpen (bOpen := TRUE);
END_IF

```



## Function library *LenzeAifParMapDrv.lib*

### Functions

#### *L\_AifParMapClose* - deactivation of code mapping

### 3.3 *L\_AifParMapClose* - deactivation of code mapping

```
BOOL L_AifParMapClose (bClose)
```

This function is used to deactivate the mapping table and the redirection of codes.

- The transfer parameter *bClose* must be set to TRUE.
- After this function has been executed, the codes accessed via the AIF interface are **no longer** redirected to the corresponding codes of the mapping table.

#### Transfer parameters

Identifiers	Data type	Possible settings	Info
<i>bClose</i>	Bool	TRUE	Deactivation of code redirections according to the mapping table. <ul style="list-style-type: none"> <li>• The AifParMap driver in the operating system is deactivated.</li> </ul>

#### Return value

Data type	Value	Meaning
Bool	TRUE	The AifParMap driver in the operating system has been deactivated.
	FALSE	The AifParMap driver in the operating system has not been deactivated. <ul style="list-style-type: none"> <li>• Remedy: Calling the function with transfer parameter <i>bClose</i> = TRUE.</li> </ul>



#### Note!

If the function *L\_AifParMapClose* is called while read or write requests for codes are active an error may occur!

This is why all actions with code access should be completed before this function is called.

#### Example

Calling the function in ST:

```
IF bCloseParMapDriver AND NOT bClose THEN
  bClose := TRUE;
  dwReturnClose := L_AifParMapClose(bClose:= TRUE);
END_IF
```