

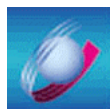
Lenze

Manual

IEC 61131-3

inside

***Global Drive
PLC Developer Studio***



Global Drive

Function library

Lenze9300Servo.lib

The function library **Lenze9300Servo.lib** can be used for the following Lenze PLCs:

	Type	from hardware version	from software version
9300 Servo PLC	EVS93XX-xI	2K	10
9300 Servo PLC	EVS93XX-xT	2K	10
ECSxA	ECSxAxxx	1C	7.0

Important Note:

The software is made available to the user in the currently existing form. All risks with regard to the quality and the results arising from its use remain the responsibility of the user. The user must implement the appropriate security precautions against possible erroneous application.

We do not accept any responsibility for direct or consequential damages, such as loss of profits, loss of orders, or effects on the course of business of any kind.

© 2000 Lenze GmbH & Co KG

No part of this documentation may be copied or made available to third parties without the express written permission of Lenze GmbH & Co KG.

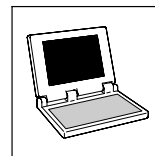
We have taken great care in assembling the information in this documentation, and checked that it corresponds to the hardware and software that is described. Nevertheless, we cannot guarantee that there are no discrepancies. We do not accept any legal responsibility or liability for damage that may thereby ensue. Any necessary corrections will be implemented in subsequent versions.

Windows, Windows NT and MS-DOS are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

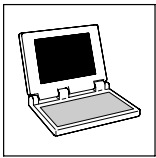
IBM and VGA are registered trademarks of International Business Machines, Inc.

All other designations are trade names of their owners.

Version 1.1 08/2000 - TD22

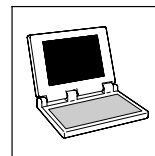


1	Preface and general information	1-1
1.1	About this Manual	1-1
1.1.1	Conventions in this Manual	1-1
1.1.2	Pictograms in this Manual	1-1
1.1.3	Terminology used	1-2
1.2	Lenze software guidelines for variable names	1-3
1.2.1	Hungarian Notation	1-3
1.2.1.1	Recommendation for designating variable types	1-4
1.2.1.2	Designation of the signal type in the variable name	1-5
1.2.1.3	Special handling of system variables	1-5
1.3	Version identifiers of the function library	1-6
2	Function blocks	2-1
2.1	Special functions	2-1
2.1.1	Holding brake (L_BRK)	2-1
2.1.2	Supply-failure control (L_MFAIL)	2-6
3	Appendix	3-1
3.1	Code table	3-1
3.1.1	L_BRK	3-1
3.1.2	L_MFAIL	3-1
4	Index	4-1



Function block library Lenze9300Servo.lib

Contents



1 Preface and general information

1.1 About this Manual

This Manual contains information on the function blocks that are included in the function block library **Lenze9300Servo.lib** for the **Drive PLC Developer Studio**.

- These function blocks can be used, for instance, in the **9300 Servo PLC** automation system.
- The function blocks are based on the functions that are available in the 9300 servo controller (V2.0).

In **Drive PLC Developer Studio** (DDS) you make the basic settings for your drive application offline, by using variables (in accordance with the IEC1131-3 standard) as aids for parameterizing the appropriate function blocks.

Using **Global Drive Control** (GDC) or **keypad** you can then *Online* set the parameters for the required functionality of your drive application, by accessing the code positions for the various instances of the function blocks.

1.1.1 Conventions in this Manual

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

Variable names

are shown in the explanatory texts in italics:

- “The signal at *nIn_a* ...”

can be recognized by the names. They always begin with “L_”:

- “The FB L_ARIT can ...”

Instances

For function blocks that have one or more first instances, there are tables that describe the corresponding codes:

Variable name	L_ARIT1	L_ARIT2		Setting range	Lenze
byFunction	C0338	C0600		0 ... 5	1

You can access these codes *Online* is linked to **Global Drive Control** (GDC) or **keypad**.

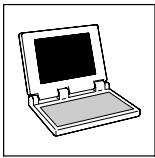


Tip!

You can use the Parameter Manager to assign the same codes to these instances that are assigned in the 9300 servo controller (V2.0).

1.1.2 Pictograms in this Manual

	Use of Pictographs	Signal words	
Warning of material damage		Stop!	Warns of potential damage to material . Possible consequences if disregarded: Damage of the controller/drive system or its environment.
Other notes		Tip!	This note designates general, useful notes. If you observe it, handling of the controller/drive system is made easier.

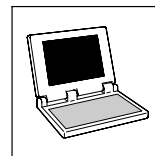


Function block library Lenze9300Servo.lib

Preface and general information

1.1.3 Terminology used

Term	In the following text used for
FB	Function block
SB	System block
Parameter codes	Codes for setting the functionality of a function block
GDC	Global Drive Control (parameterization program from Lenze)



1.2 Lenze software guidelines for variable names

The previous concepts for Lenze controllers were based on codes that represented the input and output signals, and the parameters of function blocks.

- For the sake of clarity, names were defined for the codes in the documentation.
- In addition, the signal types were defined by graphical symbols.

The user could see at a glance which kind of signal (analog, phase-angle etc.) had to be present at the particular interface.

The concept for the new automation system does not use direct codes in the programming. The IEC 61131-3 standard is used instead.

- This standard is based on a structure of variable names.
- If the user applies variables in his project, then he can name the variables as he chooses.

In order to avoid the growth of a multitude of different conventions for naming variables in existing and future projects and function libraries that are programmed by Lenze personnel, we have set up software guidelines that must be followed by all Lenze staff.

In this convention for creating variable names, Lenze keeps to the Hungarian Notation that has been specifically expanded by Lenze.

If you make use of Lenze-specific functions or function blocks, you will immediately be able to see, for instance, which data type you must transfer to a function block, and which type of data you will receive as an output value.

1.2.1 Hungarian Notation

These conventions are used so that the most significant characteristics of a program variable can instantly be recognized from its name.

Variable names

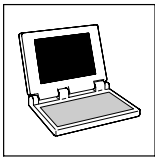
consist of

- a **prefix** (optional)
- a **data-type entry**
- and an **identifier**

The prefix and data-type entry are usually formed by one or two characters. The identifier (the "proper" name) should indicate the application, and is therefore usually somewhat longer.

Prefix examples

prefix	Meaning
a	Array (combined type), field
p	Pointer



Function block library *Lenze9300Servo.lib*

Preface and general information

Examples of the data-type entry

Examples of a data-type	Meaning
b	Bool
by	Byte
n	Integer
w	Word
dn	Double integer
dw	Double word
s	String
f	Real (float)
sn	Short integer
t	Time
un	Unsigned integer
udn	Unsigned double integer
usn	Unsigned short integer

Identifier (the proper variable name)

- An identifier begins with a capital letter.
- If an identifier is assembled from several "words", then each "word" must start with a capital letter.
- All other letters are written in lower case.

Examples:

Array of integers	<i>anJogValue[10] ;</i>
Bool	<i>blsEmpty ;</i>
Word	<i>wNumberOfValues ;</i>
Integer	<i>nLoop ;</i>
Byte	<i>byCurrentSelectedJogValue ;</i>

1.2.1.1 Recommendation for designating variable types

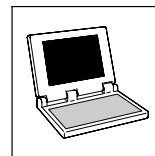
In order to be able to recognize the type of variable in a program according to the name, it makes sense to use the following designations, which are placed in front of the proper variable name and separated from it by an underline stroke:

I_<Variablename>	VAR_INPUT
Q_<Variablename>	VAR_OUTPUT
IQ_<Variablename>	VAR_IN_OUT
R_<Variablename>	VAR_RETAIN
C_<Variablename>	VAR_CONSTANT
CR_<Variablename>	VAR_CONSTANT_RETAIN
g_<Variablename>	VAR_GLOBAL
gR_<Variablename>	VAR_GLOBAL_RETAIN
gC_<Variablename>	VAR_GLOBAL_CONSTANT
gCR_<Variablename>	VAR_GLOBAL_CONSTANT_RETAIN

Example

for a global array of type integer that includes fixed setpoints (analog) for a speed setting:

g_anFixSetSpeedValue_a



1.2.1.2 Designation of the signal type in the variable name

The inputs and outputs of the Lenze function blocks each have a specific signal type assigned. These may be: digital, analog, position, or speed signals.

For this reason, each variable name has an ending attached that provides information on the type of signal.

Signal type	Ending	Previous designation
analog	_a (analog)	○
digital	_b (binary)	□
Phase-angle difference or speed	_v (velocity)	△
Phase-angle or position	_p (position)	▲



Tip!

Normalizing to signal type phase-angle difference/speed: $16384 \text{ (INT)} \triangleq 15000 \text{ rpm}$

Normalizing to signal type analog: $16384 \triangleq 100 \% \triangleq \text{value under [C0011]} = n_{\max}$

Normalizing to signal type angle or position: $65536 \triangleq 1 \text{ motor revolution}$

Examples:

Variable name	Signal type	Type of variable
nIn_a	Analog input value	Integer
dnPhiSet_p	Phase signals	Double integer
bLoad_b	Binary value (TRUE/FALSE)	Bool
nDigitalFrequencyIn_v	Speed input value	Integer

1.2.1.3 Special handling of system variables

System variables require special handling, since the system functions are only available for the user as I/O connections in the control configuration.

In order to be able to access a system variable quickly during programming, the variable name must include a label for the system function.

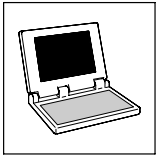
For this reason, the name of the corresponding system block is placed before the name of the variable.

Examples:

AIN1_nIn_a

CAN1_bCtrlTripSet_b

DIGIN_bIn3_b



Function block library Lenze9300Servo.lib

Preface and general information

1.3 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`.

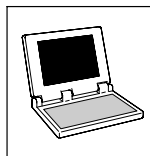
Version identifiers as of PLC software version 7.x:

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".



2 Function blocks

2.1 Special functions

2.1.1 Holding brake (L_BRK)

This FB controls a holding brake. You can use it, for instance, for hoists and traversing drives, as well as for active loads.

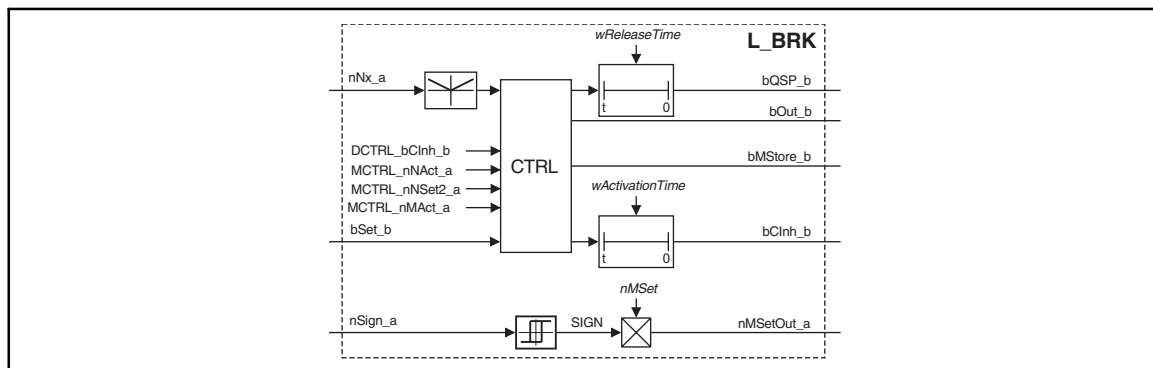


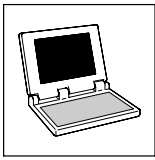
Abb. 2-1

Holding brake (L_BRK)

VariableName	DataType	SignalType	VariableType	Note
bSet_b	Bool	binary	VAR_INPUT	
nNx_a	Integer	analog	VAR_INPUT	Speed threshold from which the drive can output the signal "close brake". The signal source for this input can be a control code, a fixed value or any other analog output of a FB.
nSign_a	Integer	analog	VAR_INPUT	Direction of the torque which the drive has to build up against the brake. The signal source for this input can be a control code, a fixed value or any other analog output of a FB.
bQSP_b	Bool	binary	VAR_OUTPUT	QSP can be triggered in conjunction with MCTRL.
bOut_b	Bool	binary	VAR_OUTPUT	Set the brake
bMStore_b	Bool	binary	VAR_OUTPUT	provide a defined holding torque
bClnh_b	Bool	binary	VAR_OUTPUT	A controller inhibit can be set in conjunction with DCTRL.
nMSetOut_a	Integer	analog	VAR_OUTPUT	Holding torque of the DC injection brake 16384 = value of C0057 (max. possible torque for the drive configuration)
wReleaseTime	Word	-	VAR CONSTANT RETAIN	Brake disengaging time
wActivationTime	Word	-	VAR CONSTANT RETAIN	Brake engaging time
nMSet	Integer	-	VAR CONSTANT RETAIN	Holding torque
DCTRL_bClnh_b	Bool	-	-	These signals are processed as quantities within the FB.
MCTRL_nNAct_a	Integer	-	-	
MCTRL_nNSet2_a	Integer	-	-	
MCTRL_nMAct_a	Integer	-	-	

Parameter codes of the instances

VariableName	L_BRK1			SettingRange	Lenze
wReleaseTime	C0196			0.0 ... 60.0 s	0.0
wActivationTime	C0195			0.0 ... 99.9 s	99.9
nMSet	C0244			-199.99 ... 199.99 %	0.00



Function block library *Lenze9300Servo.lib*

Special functions

Holding brake (*L_BRK*)

Range of functions

- Close brake
- Open the brake (release)
- Set controller inhibit

2.1.1.1 Close brake

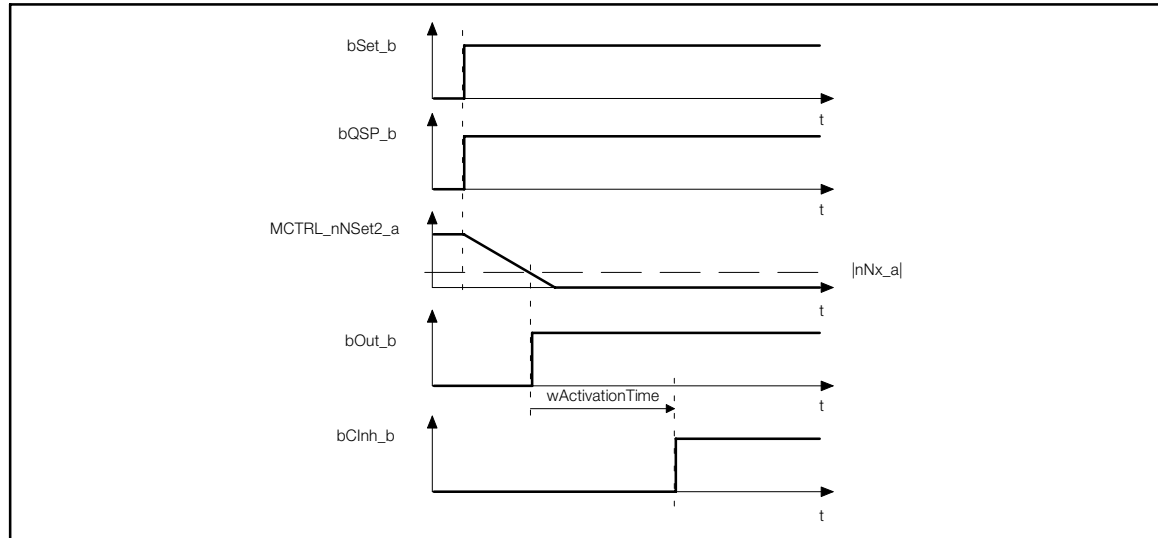
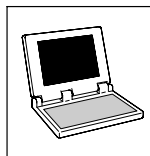


Abb. 2-2

Signal sequence when the brake is closed

Functional sequence

1. Select the constant FIXED0% (selection number 1000) using $bSet_b = \text{TRUE}$, the function "close brake" is activated.
 - At the same time, $bQSP_b$ switches immediately = TRUE. You can use this signal to steer the drive down to zero speed along a deceleration ramp.
2. If the speed setpoint goes below the value at nNx_a , then $bOut_b$ switches = TRUE (operation of the brake by a digital output).
 - Invert the signal at the digital output, if you require a version that is safe against cable breakage (e.g. through C0118).
3. Select the constant FIXED0% (selection number 1000) using $bOut_b = \text{TRUE}$, a timer is started. After the time defined by $wActivationTime$ has elapsed, then $bClnh_b$ switches immediately = TRUE.
 - With this signal you can, for instance, switch the controller inhibit (device-internal). In general, the brake-closing time is set here. This is necessary, because the brake does not engage immediately with $bOut_b = \text{TRUE}$, and so the drive must provide a holding moment during this preset period.



2.1.1.2 Open the brake (release)

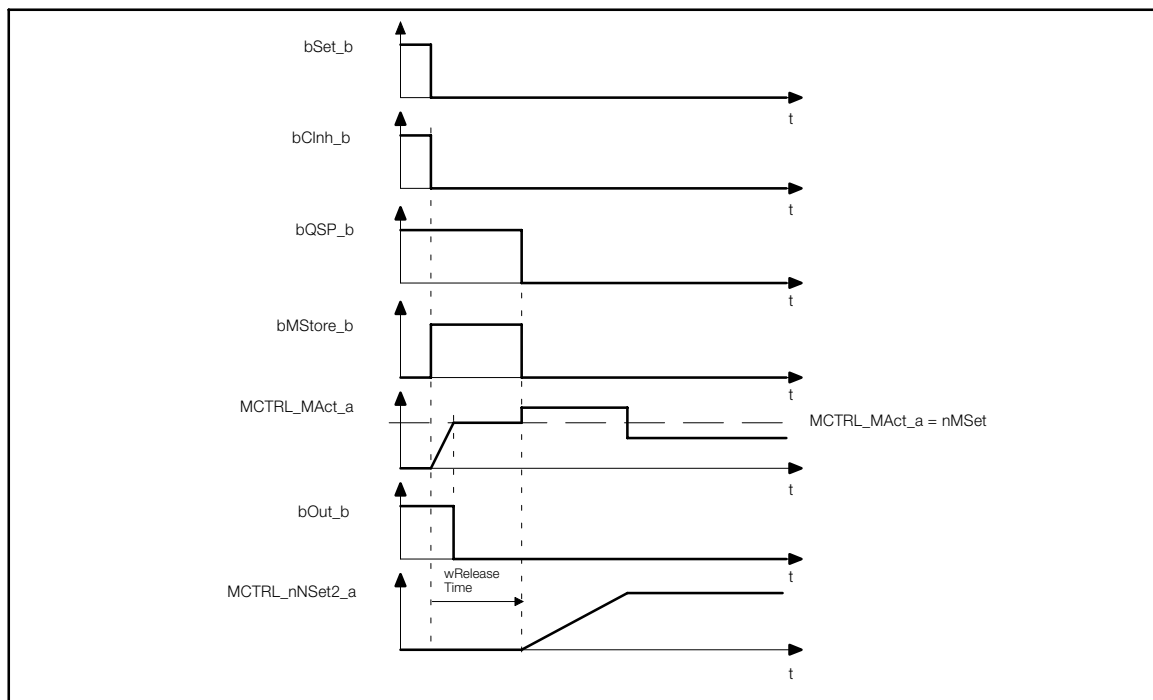


Abb. 2-3 Signal sequence when the brake is opened (released)

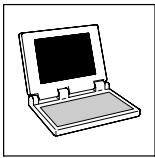
Functional sequence

1. Select the constant FIXED0% (selection number 1000) using $bSet_b = FALSE$, then $bClnh_b$ switches = FALSE immediately. At the same time, $bMStore_b$ switches immediately = TRUE.
 - You can use this signal to make the drive build up a defined torque against the brake. The drive thus takes up the torque while the brake opens. The signal is reset only after the time set by $wReleaseTime$ has elapsed.
2. As soon as the torque set by $nMSet$ has been reached (holding torque), then $bOut_b = FALSE$, immediately.
3. When the input is reset, a time element is triggered. After the time defined by $wReleaseTime$ has elapsed, then $bQSP_b = FALSE$, immediately.
 - With this signal you can, for instance, enable the setpoint integrator after the brake-opening period.



Note!

If, before the end of the brake-opening period, ($wReleaseTime$) an actual speed is detected, that is greater than the value at bNx_a , then $bQSP_b$ switches = FALSE and $bMStore_b = FALSE$, immediately. The drive can immediately operate speed- or phase controlled. If $bQSP_b$ has an influence on the control word QSP, then the drive synchronises itself to the momentary speed and follows its setpoint.



Function block library Lenze9300Servo.lib

Special functions

Holding brake (L_BRK)

2.1.1.3 Set controller inhibit

The controller inhibit can, for instance, be set in the event of a fault (LU, OU).

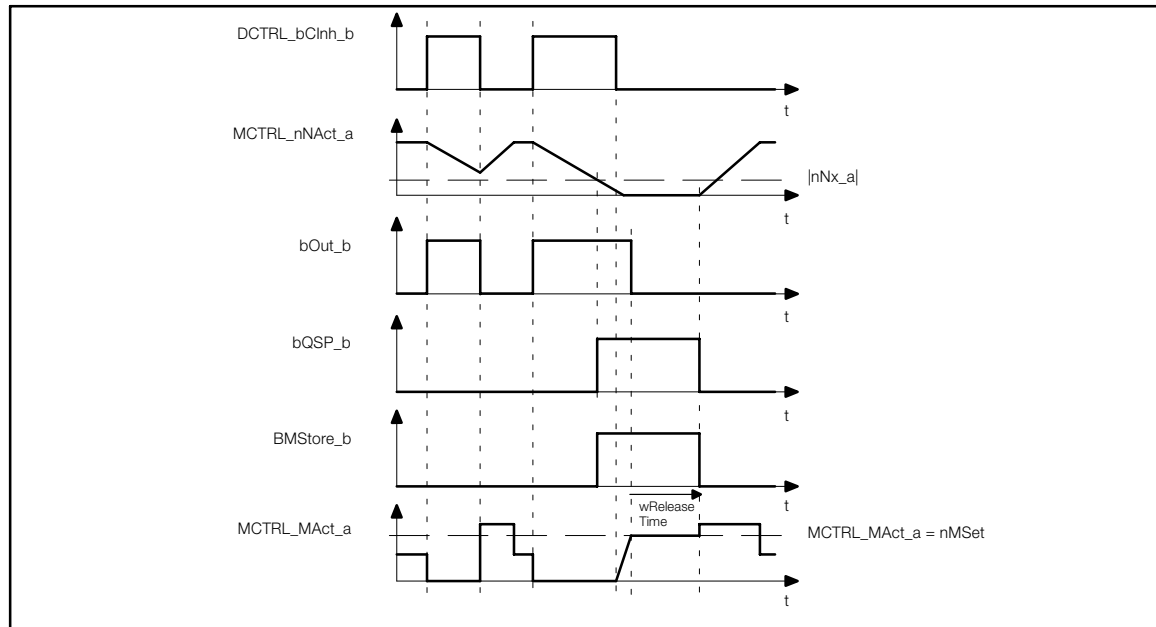


Abb. 2-4

Operate brake by controller inhibit

Functional sequence

1. By setting the controller inhibit ($DCTRL_bCInh_b = TRUE$) $bOut_b$ switches immediately = TRUE.
 - The drive is then braked by its mechanical brake.
2. If the controller inhibit ($DCTRL_bCInh_b = FALSE$) before the actual speed falls below the threshold at nNx_a , then $bOut_b = FALSE$, immediately.
 - The drive synchronizes itself to the momentary speed and follows its setpoint.
 - If the value falls below the threshold, the drive starts. (2-3)

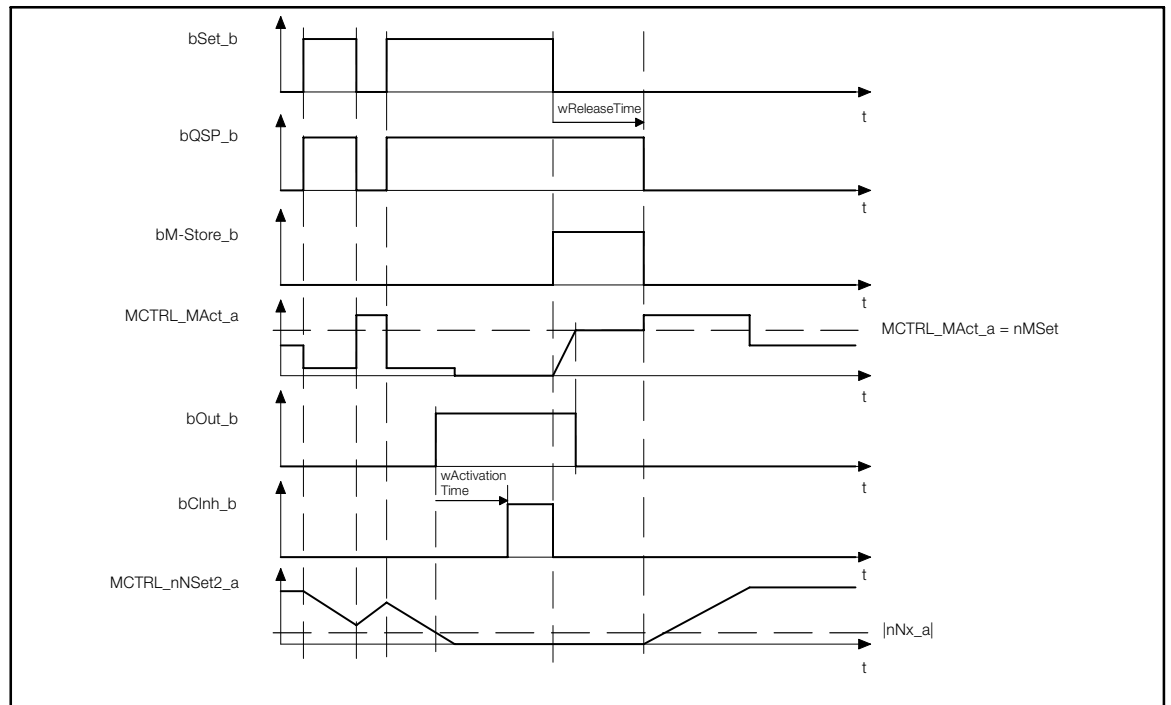
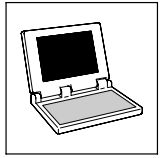
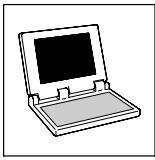


Abb. 2-5

Switching cycle when braking



Function block library Lenze9300Servo.lib

Special functions

Supply-failure control (L_MFAIL)

2.1.2 Supply-failure control (L_MFAIL)

If the supply voltage via L1, L2, L3 or +UG, -UG fails, then the drive (drive network) can be decelerated (braked) in a controlled manner.

Without this function, the drive (drive network) would coast down.

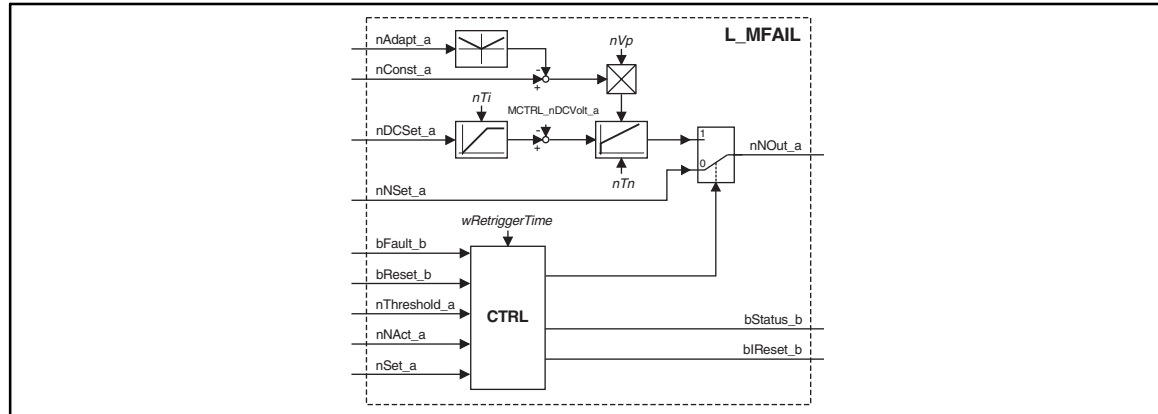
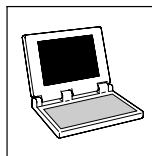


Abb. 2-6 Supply-failure control (L_MFAIL)

VariableName	DataType	SignalType	VariableType	Note
nAdapt_a	Integer	analog	VAR_INPUT	Dynamic adjustment of the proportional gain of the U_{Gsetp} -control in [%] of nVp . (16384 = 100 %)
nConst_a	Integer	analog	VAR_INPUT	Proportional gain of the U_{Gsetp} controller in [%] from nVp . (16384 = 100 %)
nDcSet_a	Integer	analog	VAR_INPUT	Voltage setpoint at which the DC bus voltage is to be maintained. (1000 V = 100 %)
nNSet_a	Integer	analog	VAR_INPUT	Speed setpoint in [%] of C0011 (C0011 = 100 % = 16384)
bFault_b	Bool	binary	VAR_INPUT	TRUE = activates the supply-failure control
bReset_b	Bool	binary	VAR_INPUT	TRUE = reset
nThreshold_a	Integer	analog	VAR_INPUT	Restart threshold in [%] of C0011 (C0011 = 100 % = 16384)
nNAct_a	Integer	analog	VAR_INPUT	Comparison value for the restart threshold in [%] of C0011
nSet_a	Integer	analog	VAR_INPUT	Speed starting point for run-down/deceleration in [%] of C0011 (C0011 = 100 % = 16384)
nNOut_a	Integer	analog	VAR_OUTPUT	Speed setpoint in [%] of C0011 (C0011 = 100 % = 16384)
bStatus_b	Bool	binary	VAR_OUTPUT	TRUE = supply-failure control is active
blReset_b	Bool	binary	VAR_OUTPUT	TRUE = supply-failure control active, the drive is braking
nVp	Integer	-	VAR CONSTANT RETAIN	Gain
nTn	Integer	-	VAR CONSTANT RETAIN	Integral-action time T
nTi	Integer	-	VAR CONSTANT RETAIN	Acceleration time
wRetriggerTime	Unsigned Integer	-	VAR CONSTANT RETAIN	Retrigger time

Parameter codes of the instances

VariableName	L_MFAIL1		SettingRange	Lenze
nVp	C0980		0.001 ... 31.000	0.500
nTn	C0981		20 ... 2000 ms	100
nTi	C0982		0.001 ... 16.000 s	2.000
wRetriggerTime	C0983		0.001 ... 60.000 s	1.000



Range of functions

- Supply-failure detection
- Supply-failure control
- Restart protection
- Reset of the supply failure control
- Dynamic adaptation of the control parameters
- Fast supply recovery (KU)
- Application example

2.1.2.1 Supply failure detection

The type of the supply-failure detection to be used depends on the drive system used.

A failure of the voltage supply of the power stage is detected:

- by the level of the DC-bus voltage
- by an external system (e.g. supply module 934X or voltage-detection relay).
- Different systems can be combined.

Supply-failure detected by the level of the DC bus voltage

Use with single drives or multi-axis drives, which do not use external monitoring systems. For this, you can use a comparator (e.g. L_CMP).

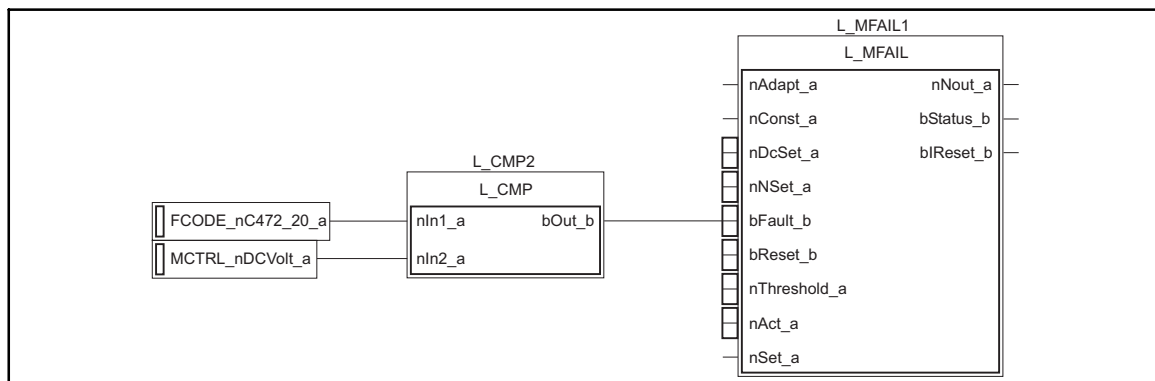
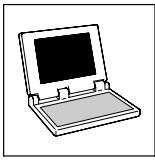


Abb. 2-7

Example of a supply-failure detection with internal function blocks (section)

Programming the example in Abb. 2-7:

1. Set the signal links according to Abb. 2-7
2. In FB L_CMP, set *byFunction* = 3 ($nln1 < nln2$)



Function block library *Lenze9300Servo.lib*

Special functions

Supply-failure control (*L_MFAIL*)

Supply-failure detection of the supply module

- A digital output of the supply module 934x is switched to the function block *L_MFAIL* via the digital inputs DIGIN of the 93XX controller. In the example, input X5/E4 is used.

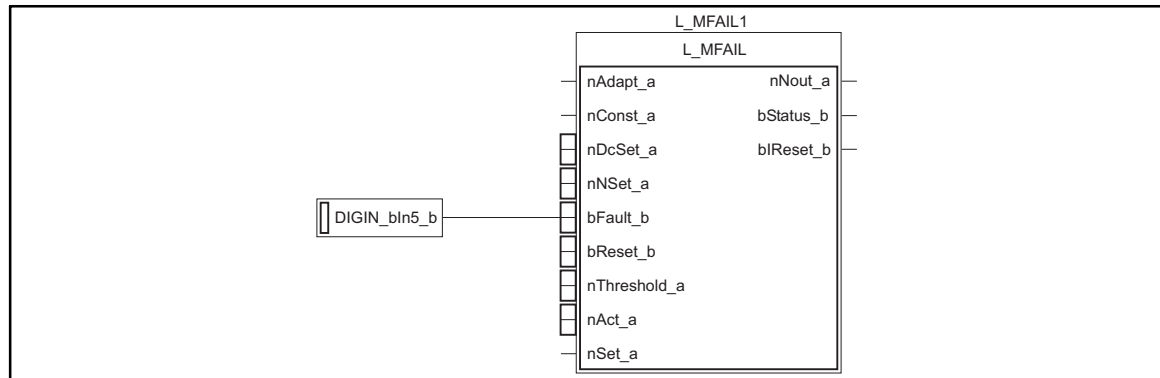


Abb. 2-8

Example of a supply-failure detection by an external monitoring system

Programming the example in Abb. 2-8:

- Set the signal links according to Abb. 2-8
- Select the input level (TRUE- or FALSE-active) for X5/E4 with C0114/4

Combination of these methods

These methods are combined via an OR link.

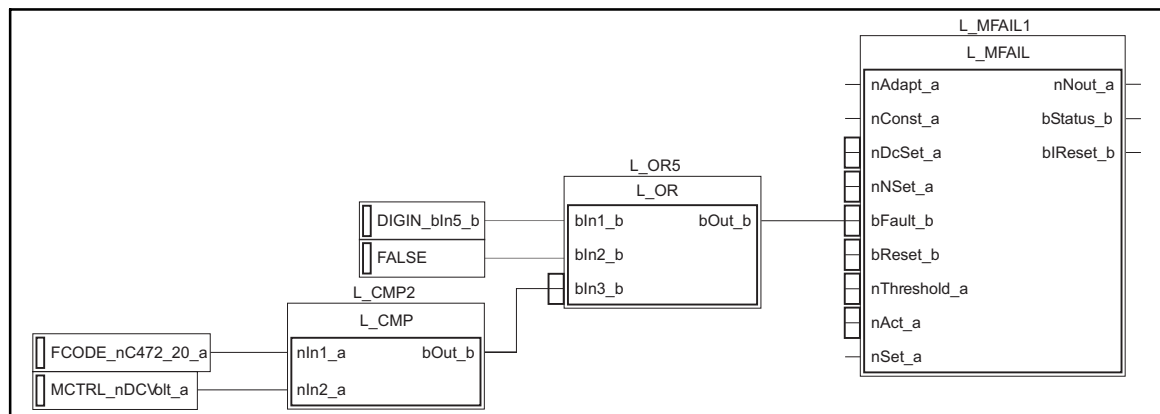
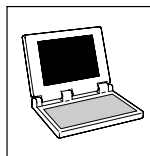


Abb. 2-9

Example of a supply-failure detected by different sources

Programming the example in Abb. 2-9:

- Set the signal links according to Abb. 2-9
- In FB *L_CMP*, set *byFunction* = 3 ($nln1 < nln2$)



2.1.2.2 Supply failure control

Integration of the FB into the signal flow of the controller

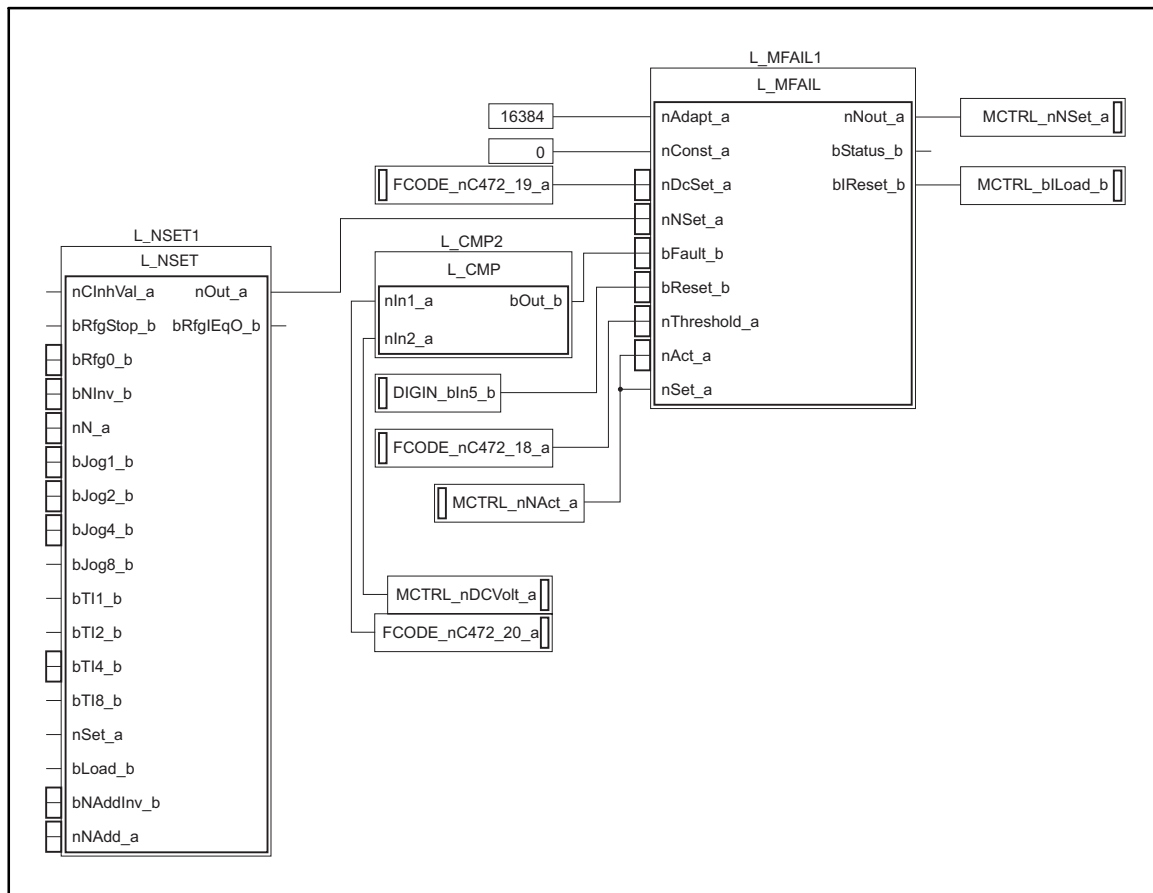


Abb. 2-10 Links for the configuration CFG1000.lpc

Programming the example (SpeedModelInternal24VSupply.lpc) in Abb. 2-10:

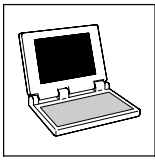
1. Set the signal links according to Abb. 2-10 (see the following table for explanation).

Function	Inputs/outputs of L_MFAIL	Note
Speed setpoint path	nNSet_a, nNOut_a	
Start value for deceleration	nSet_a	Here, the actual speed value
Source for the setpoint of the DC bus voltage	nDcSet_a	Here, from the freely linkable code FCODE_nC472_19_a
Source for the activation of the supply-failure control:	b_Fault_b	2-7: Supply-failure detection
Proportional gain and adaptation of the DC-bus voltage controller:	n_Adapt_a, nConst_a	
Restart protection	nThreshold_a, nNAct_a	In FCODE_nC472_18_a first, enter approx. 2% (reference: n_{max} , C0011)
Reset input	bReset_b	Here, with terminal DIGIN_bIn5_b



Note!

Use C0003 to save all settings in the parameter set, if they are to be retained on power-off.



Function block library *Lenze9300Servo.lib*

Special functions

Supply-failure control (*L_MFAIL*)

Activation of the supply-failure control

- *bFault_b* TRUE = activates the supply-failure control
- *bFault_b* = FALSE triggers a timing element. After elapse of the preset time in *wRetriggerTime* the supply-failure control is ended/cancelled. (□ 2-15: fast return of supply power)
 - The drive is accelerated to the speed setpoint if the restart protection is not active.
 - The drive is still braked to zero speed, if the restart protection is active. (□ 2-14: Restart protection)
 - When restart protection is active, the drive can only be reset by *bReset_b* = TRUE.

Function of the supply-failure control

The drive controller gains the required energy from the rotational energy of the driven machine. The drive is braked through the power loss of the controller and the motor. The speed deceleration ramp is thus shorter than for an uncontrolled system (coasting drive).

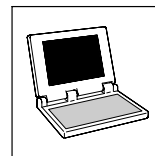
After activation of the supply-failure control:

1. The DC bus voltage is controlled to the value at *nDCSet_a*
2. At *nNOut_a* an internally generated speed setpoint is output. The drive can thus be braked to zero speed (via the speed setpoint).
 - The start value for the controlled deceleration is the value at *nSet_a*. This input is usually connected to *MCTRL_nNAct_a* (actual speed), *MCTRL_nNIn_a* or *L_MFAIL_nNOut_a* (set speed).
 - The speed deceleration ramp (and thus the brake torque) results from the moment of inertia of the driven machine(s), the power loss of the drive (group), and the parameter settings.



Stop!

- If a connected brake unit is activated, the drive is braked with the maximum possible torque ($I_{\max}$). In this case, it may be necessary to adapt the parameterization (see following page).
- If the power stage is not supplied, the drive cannot generate a standstill torque (important for active loads such as hoists).



Parameterization of the supply-failure control

The parameters to be set are strongly dependent on the motor used, the inertia of the driven machine and the drive configuration (single drive, drive network, master-slave operation, etc.). This function must therefore be adapted to the individual application in every case.

The following specifications refer to Chapter 2.1.2.1

Important settings prior to the initial commissioning:



Stop!

With internal voltage supply to the terminals, terminal X6/63 is used as a voltage source for external potentiometers. In this case, measure across terminals $+U_G$, $-U_G$.



Note!

To perform the measurements, a new download that includes the changes must be made.

1. Measure the DC-bus voltage with an oscilloscope (channel 1)
 - with a suitable voltage divider across the terminals $+U_G$, $-U_G$ or
 - by providing the DC bus voltage at terminal X6/62, for instance. To do this, connect the system variable *MCTRL_nDCVolt_a* with the system variable *AOUT2_nOut_a*.
2. Measure the speed with an oscilloscope (channel 2)
 - by supplying the speed on terminal X6/62, for instance, (standard setting). To do this, connect the system variable *MCTRL_nNAct_a* with the system variable *AOUT1_nOut_a*.
3. Enter, in C0472/20 (*FCODE_nC0472_20_a*) the threshold for the supply-failure detection. The entry depends on the setting in C0173 (adjustment of the U_G -threshold).
 - Set the threshold approx. 50 V above the switch-off threshold LU (example for C0173 = 0.1; C0472/20 = 48 % = 480 V).

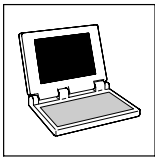
Supply voltage range	C0173 =	Switch-off threshold LU	Switch-on threshold LU	Switch-off threshold OU	Switch-on threshold OU
< 400 V	0	285 V	430 V	770 V	755 V
400 V	1	285 V	430 V	770 V	755 V
400 V ... 460 V	2	328 V	473 V	770 V	755 V
480 V without brake chopper	3	342 V	487 V	770 V	755 V
Operation with brake chopper (up to 480 V)	4	342 V	487 V	800 V	785 V

4. Set the setpoint to which the DC bus voltage is to be controlled:
 - Set the setpoint to approx. 700 V (C0472/18 = 70%).



Stop!

This setpoint must be below the threshold of any brake unit which may be connected. If a connected brake unit is activated, the drive is braked with the maximum possible torque (I_{max}). The desired operating behaviour is lost.



Function block library *Lenze9300Servo.lib*

Special functions

Supply-failure control (*L_MFAIL*)

Commissioning of the supply-failure control

The commissioning should be carried out with motors without any load.

1. Start the drive with a FALSE-TRUE transition at X5/E5 (when *DIGIN_bIn5_b* is connected to *bReset_b*).
2. Setting the acceleration time *nTi* :
 - Set speed setpoint to 100%, operate controller with maximum speed.
 - Inhibit controller via terminal X5/28 (you can also use any other source for the controller inhibit, *CINH*) and measure the deceleration time to standstill.
 - Set approx. 1/10 of the deceleration time in *nTi* .
3. Setting the retrigger time
 - For supply-failure detection by detecting the DC-bus voltage level:
In *wRetriggerTime* set the run-down/deceleration time measured under point 2. .
 - For supply-failure detection via an external system (e.g. supply module 934X):
In *wRetriggerTime* set the time for which the drive continues to be braked in a controlled way in the event of short-term supply recovery.
4. Switch off the supply voltage (supply or DC-bus).

The oscilloscope should display the following sequence

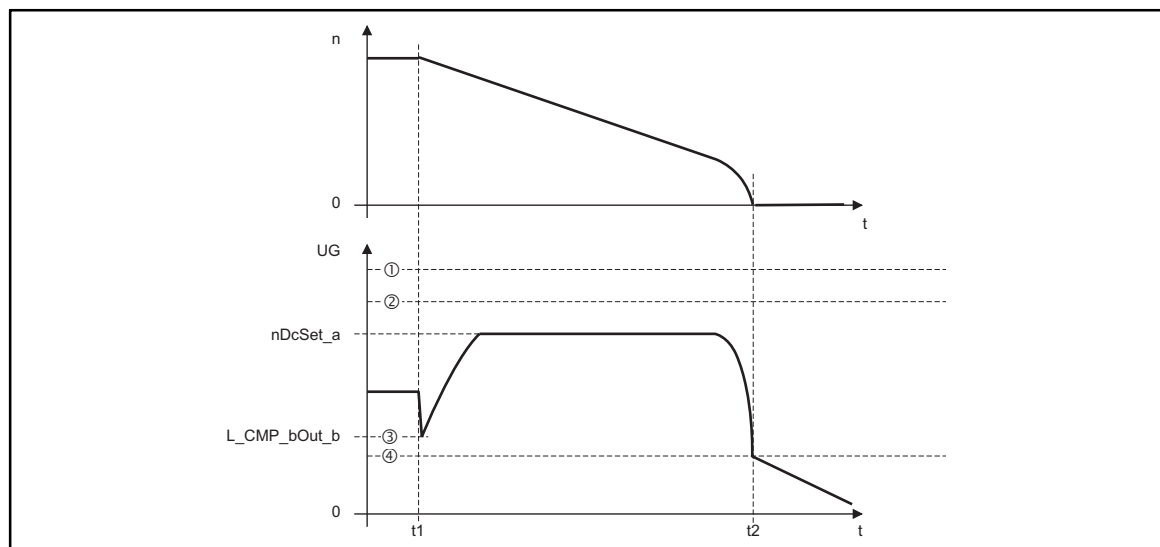
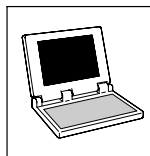


Abb. 2-11

Schematic representation with activated supply-failure control (ideal characteristic)

- ① Switch-off threshold OU
- ② Switch-on threshold for brake unit
- ③ Threshold
- ④ Threshold LU
- n* Speed of the drive
- t1* Supply-failure
- t2* Zero speed reached



Fine setting of the supply-failure control

For the fine setting, you will have to repeat the following points several times.

1. Obtain a very low final speed without the controller reaching the undervoltage threshold LU:
 - Increase the proportional gain nVp .
 - Reduce the integral-action time nTn .
2. Avoid activation of the brake unit or the overvoltage threshold:
 - Increase the integral-action time nTn until the characteristic in Abb. 2-11 is almost reached.
 - If necessary, also reduce the setpoint of the DC-bus voltage at $nDCSet_a$ (in the example C0472/19 (FCODE_nC472_19_a)).
3. An increase of the run-down/deceleration time or reduction of the brake torque (see Abb. 2-12) is only possible with restrictions:
 - Increasing the acceleration time nTi reduces the initial brake torque and simultaneously increases the deceleration time.
 - Increasing the integral-action time nTn reduces the initial brake torque and simultaneously increases the deceleration time. If the integral-action times are too long, the controller reaches the LU threshold before zero speed is reached. The drive is thus no longer under control.
4. Re-establish any signal connections which may be required to the outputs of the drive controller (terminals X6).



Note!

Use C0003 to save all settings in a parameter set, if they are to be retained on power-off.

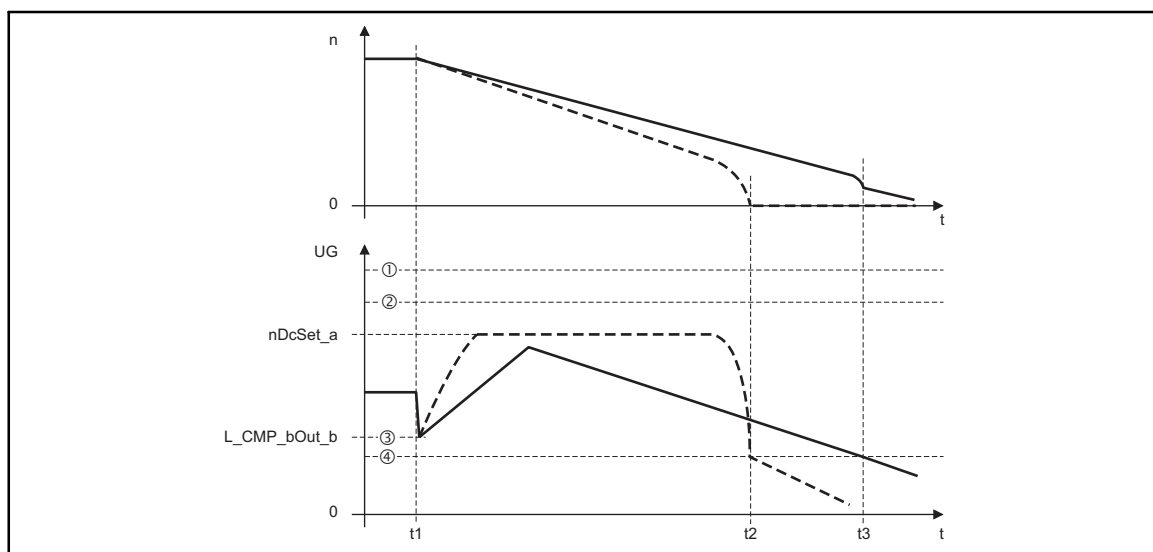
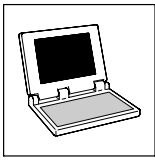


Abb. 2-12

Schematic representation with different brake torques

- | | |
|-----------|---|
| ① | Switch-off threshold OU |
| ② | Switch-on threshold for brake unit |
| ③ | Threshold |
| ④ | Threshold LU |
| n | Speed of the drive |
| $t = t_1$ | Supply-failure |
| $t = t_2$ | Zero speed with higher brake torque (short adjustment time) |
| $t = t_3$ | Drive reaches the LU switch-off threshold with lower brake torque (high adjustment time), without reaching zero speed |
| $t > t_3$ | Drive is no longer under control (is braked by friction) |



Function block library *Lenze9300Servo.lib*

Special functions

Supply-failure control (L_MFAIL)

Reset of the supply failure control

- The supply-failure control is reset with *bReset_b* = TRUE (in the example, through X5/E5 (when *DIGIN_bIn5_b* is connected to *bReset_b*).
- The reset pulse is always required if:
 - The restart protection is active.
 - The restart protection is used and the supply (supply or DC supply) was switched on.

2.1.2.3 Restart protection

The integrated restart protection is to avoid a restart in the lower speed range, after the supply voltage was interrupted for a short time only (supply recovery before the drive has come to standstill).

- Establish the restart protection (□ 2-9: Parameterization of the example in Abb. 2-10)
- In C0472/18 (*FCODE_nC472_18_a*), enter the threshold in [%] of $n_{\max}$ (C0011) below which no automatic start is wanted after supply recovery.
 - If the speed at supply recovery < threshold in C0472/18 (*FCODE_nC472_18_a*): the drive will still be braked under control. This function can only be ended by *bReset_b* = TRUE.
 - If the speed at supply recovery > threshold in C0472/18 (*FCODE_nC472_18_a*): the drive accelerates to its setpoint along the set ramps.
 - The function is deactivated by: *nThreshold_a* = 0%.
- A reset is made by *bReset_b* = TRUE
 - This is required after every supply (re)connection, and is shown by *bStatus_b* = HIGH, when *bFault_b* = FALSE.



Note!

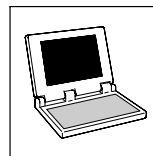
To be able to make the settings through the system block (SB) FCODE, you must also have used the free codes of the SB FCODE.

2.1.2.4 Dynamic adaptation of the control parameters

In special cases, a dynamic modification of the proportional gain may be useful. Two inputs are available for this purpose at FB L_MFAIL (*nConst_a* and *nAdapt_a*). The resulting proportional gain results from:

$$V_p = nV_p \cdot \frac{nConst_a - |nAdapt_a|}{100 \%}$$

(100 % = 16384)



2.1.2.5 Fast supply power recovery (KU)

The fast supply recovery causes a restart of the controller, unless the restart protection is active. The drive accelerates to its setpoint. If this is not wanted, you can delay the restart by *wRetriggerTime* or prevent it in combination with the restart protection.

A fast supply recovery occurs:

- Due to the system, the supply recovery is indicated by the supply-failure detection via the level of the DC-bus voltage. (□ 2-7)
- because of a "short interruption" (KU) of the utility company (e.g. in a thunderstorm).
- Because of faulty components in the supply cables (e.g. slip-rings)

So set *wRetriggerTime* > the measured deceleration time that can be achieved in braking operation.

2.1.2.6 Application example

Drive network with digital frequency coupling

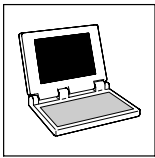


Stop!

Für drive networks which are connected via digital frequency (a master and one or more slaves):

- the supply-failure detection and control must only be activated for the master.
 - You must link the supply-failure control into the signal flow to meet this requirement.
- You must operate all the controllers through the terminals +U_G, -U_G in a DC-bus configuration.

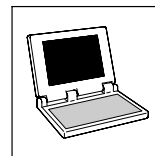
Observe the specifications in the System Manual "Servo controller 9300", Part F.



Function block library Lenze9300Servo.lib

Special functions

Supply-failure control (L_MFAIL)



3 Appendix

3.1 Code table

How to read the code table:

Column	Abbreviation	Meaning
Code	C0039	Code C0039
	1	Subcode 1 of code C0039
	2	Subcode 2 of code C0039
	...	...
	14	Subcode 14 of code C0039
	15	Subcode 15 of code C0039
	[C0156]	Parameter value of the code can only be modified when the controller is inhibited
LCD		Keypad LCD - DIS: ... display only - all others are parameter values
Lenze		Factory setting of the code
	*	The column "Important" contains further information
Choice	1 {1 %} 99	Minimum value {smallest step/unit} maximum value
IMPORTANT	-	Additional, important explanation of the code

3.1.1 L_BRK

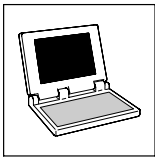
FB description: (2-1)

Code	LCD	Possible settings		IMPORTANT
		Lenze	Choice	
C0195	wActivationTime	99.9	0.0 {0.1 s} 99.9 s infinite	99.9
				Brake closing time of L_BRK1 - Engaging time of the mechanical holding brake (see technical data of the brake). - After the time under C0195 has elapsed, the status "mechanical brake closed" is reached
C0196	wReleaseTime	0.0	0.0 {0.1 s}	60.0
				Brake opening time of L_BRK1 - Disengaging time of the mechanical holding brake (see technical data of the brake). - After the time under C0196 has elapsed, the status "mechanical brake opened" is reached
C0244	nMSet	0.00	-100.00 {0.01 %} 100.00	100.00
			100 % = value of C0057	Holding moment of the DC-brake of L_BRK1

3.1.2 L_MFAIL

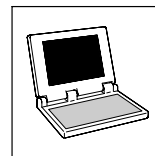
FB description: (2-6)

Code	LCD	Possible settings		IMPORTANT
		Lenze	Choice	
C0980	nVp	0.500	0.001 {0.001}	31.000
				Gain Vp of L_MFAIL1
C0981	nTn	100	20 {1 msec}	2000
				Time constant of L_MFAIL1
C0982	nTir	2.000	0.001 {0.001 sec}	16.000
				Acceleration time Tir of L_MFAIL1
C0983	wRetriggerTime	1.000	0.001 {0.001 sec}	60.000
				Retrigger time of L_MFAIL1



Function block library Lenze9300Servo.lib

Appendix



4 Index

A

Appendix, 3-1

C

Code table, 3-1

Codes

Holding brake (L_BRK), 3-1

Supply-failure control (L_MFAIL), 3-1

D

data-type entry, Explanation of, 1-4

Definitions, 1-2

F

Fast supply recovery (KU), 2-15

Function blocks

Holding brake (L_BRK), 2-1

Close brake, 2-2

Open the brake, 2-3

Setting controller inhibit, 2-4

Supply-failure control (L_MFAIL), 2-6

H

Holding brake (L_BRK), 2-1

I

identifier, Explanation of, 1-4

L

L_BRK, 2-1

L_MFAIL, 2-6

Lenze software guidelines, Hungarian Notation, 1-3

P

prefix, Explanation of, 1-3

R

Restart protection, 2-14

S

Safety information, Layout

Other notes, 1-1

Warning of damage to material, 1-1

Signal type, Explanation of, 1-5

Supply-failure control, 2-9

Supply-failure control (L_MFAIL), 2-6

Fast supply recovery (KU), 2-15

Restart protection, 2-14

Supply-failure control, 2-9

Supply-failure detection, 2-7

Supply-failure detection, 2-7

System variables, Explanation of, 1-5

T

Type of variable, Identification, 1-4

V

Variable names

Conventions, Hungarian Notation, 1-3

Lenze software guidelines, Explanation of, 1-3

Version identifiers of the function library, 1-6