Lenze

Manual

Global Drive
Function block library
Lenze9300Servo.lib
The function library **Lenze9300Servo.lib (05.2000)** for the **Drive PLC Developer Studio V01.00** can be used for the following Lenze PLCs:

<table>
<thead>
<tr>
<th>Automation system</th>
<th>Type designation</th>
<th>from hardware version</th>
<th>from software version</th>
</tr>
</thead>
<tbody>
<tr>
<td>9300 Servo PLC</td>
<td>EV932XX-xI</td>
<td>2K</td>
<td>10</td>
</tr>
<tr>
<td>9300 Servo PLC</td>
<td>EV933XX-xT</td>
<td>2K</td>
<td>10</td>
</tr>
</tbody>
</table>

**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.
Function block library Lenze9300Servo.lib

Contents

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 ..................................... 1-2
    1.1.3 Terminology used .................................................. 1-2
    1.1.4 What's new? .......................................................... 1-2
  1.2 Lenze software guidelines for variable names ....................... 1-3
    1.2.1 Based on 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

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
  3.2 Index .................................................................................. 3-2
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:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>L_ARIT1</th>
<th>L_ARIT2</th>
<th>Setting range</th>
<th>Lenze</th>
</tr>
</thead>
<tbody>
<tr>
<td>byFunction</td>
<td>C0338</td>
<td>C0600</td>
<td>0 ... 5</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

1.1.3 Terminology used

<table>
<thead>
<tr>
<th>Term</th>
<th>In the following text used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB</td>
<td>Function block</td>
</tr>
<tr>
<td>SB</td>
<td>System block</td>
</tr>
<tr>
<td>Parameter codes</td>
<td>Codes for setting the functionality of a function block</td>
</tr>
<tr>
<td>GDC</td>
<td>Global Drive Control (parameterization program from Lenze)</td>
</tr>
</tbody>
</table>

1.1.4 What's new?

<table>
<thead>
<tr>
<th>Version</th>
<th>ID-No.</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>07/2000</td>
<td>Revised edition for Drive PLC Developer Studio V01.00</td>
</tr>
</tbody>
</table>
1.2 Lenze software guidelines for variable names

The previous concepts for drive controllers in the 9300 series 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 not use direct codes in the programming. The IEC1131-3 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 Based on 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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>array (combined type), field</td>
</tr>
<tr>
<td>p</td>
<td>pointer</td>
</tr>
</tbody>
</table>
Examples of the data-type entry

<table>
<thead>
<tr>
<th>Examples of a data-type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Bool</td>
</tr>
<tr>
<td>by</td>
<td>Byte</td>
</tr>
<tr>
<td>n</td>
<td>Integer</td>
</tr>
<tr>
<td>w</td>
<td>Word</td>
</tr>
<tr>
<td>dn</td>
<td>Double Integer</td>
</tr>
<tr>
<td>dw</td>
<td>Double Word</td>
</tr>
<tr>
<td>s</td>
<td>String</td>
</tr>
<tr>
<td>f</td>
<td>Real (Float)</td>
</tr>
<tr>
<td>sn</td>
<td>Short Integer</td>
</tr>
<tr>
<td>t</td>
<td>Time</td>
</tr>
<tr>
<td>un</td>
<td>Unsigned Integer</td>
</tr>
<tr>
<td>udn</td>
<td>Unsigned Double Integer</td>
</tr>
<tr>
<td>usn</td>
<td>Unsigned Short Integer</td>
</tr>
</tbody>
</table>

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:  anJogValue[10];
- Bool: bIsEmpty;
- Word: wNumberOfValues;
- Integer: nLoop;
- Byte: byCurrentSelectedJogValue;

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:

<table>
<thead>
<tr>
<th>I_&lt;Variablename&gt;</th>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_&lt;Variablename&gt;</td>
<td>VAR_OUTPUT</td>
</tr>
<tr>
<td>IQ_&lt;Variablename&gt;</td>
<td>VAR_IN_OUT</td>
</tr>
<tr>
<td>R_&lt;Variablename&gt;</td>
<td>VAR_RETAIN</td>
</tr>
<tr>
<td>C_&lt;Variablename&gt;</td>
<td>VAR_CONSTANT</td>
</tr>
<tr>
<td>CR_&lt;Variablename&gt;</td>
<td>VAR_CONSTANT RETAIN</td>
</tr>
<tr>
<td>g_&lt;Variablename&gt;</td>
<td>VAR_GLOBAL</td>
</tr>
<tr>
<td>gR_&lt;Variablename&gt;</td>
<td>VAR_GLOBAL RETAIN</td>
</tr>
<tr>
<td>gC_&lt;Variablename&gt;</td>
<td>VAR_GLOBAL CONSTANT</td>
</tr>
<tr>
<td>gCR_&lt;Variablename&gt;</td>
<td>VAR_GLOBAL CONSTANT RETAIN</td>
</tr>
</tbody>
</table>

Example

For a global array of type integer, that includes fixed setpoints analog for a speed setting:

\[ g\_\text{anFixSetSpeedValue}_a \]
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 (rotational) signals.

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

<table>
<thead>
<tr>
<th>Signal type</th>
<th>Ending</th>
<th>Previous designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>analog</td>
<td>_a (analog)</td>
<td></td>
</tr>
<tr>
<td>digital</td>
<td>_b (binary)</td>
<td>□</td>
</tr>
<tr>
<td>phase-angle difference or speed (rot.)</td>
<td>_v (velocity)</td>
<td>△</td>
</tr>
<tr>
<td>phase-angle or position</td>
<td>_p (position)</td>
<td>▲</td>
</tr>
</tbody>
</table>

Note!

Normalizing to signal type phase-angle difference/speed: 16384 (INT) △ 15000 rpm
Normalizing top signal type analog: 16384 △ 100 % △ value under [C0011] = N_max
Normalizing to signal type angle or position: 65536 △ 1 motor turn

Examples:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Signal type</th>
<th>Variable type</th>
</tr>
</thead>
<tbody>
<tr>
<td>nIn_a</td>
<td>Analog input value</td>
<td>Integer</td>
</tr>
<tr>
<td>dPhiSet_p</td>
<td>Angle signal</td>
<td>Double Integer</td>
</tr>
<tr>
<td>bLoad_b</td>
<td>Binary value (TRUE/FALSE)</td>
<td>Bool</td>
</tr>
<tr>
<td>nDigitalFrequency_v</td>
<td>Speed input value</td>
<td>Integer</td>
</tr>
</tbody>
</table>

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:

AIL1_nln_a
CAN1_bCtrlTripSet_b
DIGIN_bln3_b
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)

<table>
<thead>
<tr>
<th>VariableName</th>
<th>DataType</th>
<th>SignalType</th>
<th>VariableType</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>bSet_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_INPUT</td>
<td>Speed threshold from which the drive can output the signal &quot;close brake&quot;. The signal source for this input can be a control code, a fixed value or any other analog output of a FB.</td>
</tr>
<tr>
<td>nN_x_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>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.</td>
</tr>
<tr>
<td>nSign_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>QSP can be triggered in conjunction with MCTRL.</td>
</tr>
<tr>
<td>bQSP_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_OUTPUT</td>
<td>Set the brake</td>
</tr>
<tr>
<td>bOut_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_OUTPUT</td>
<td>A controller inhibit can be set in conjunction with DCTRL.</td>
</tr>
<tr>
<td>bMStore_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_OUTPUT</td>
<td>Holding torque of the DC injection brake 16384 = value of C0057 (max. possible torque for the drive configuration)</td>
</tr>
<tr>
<td>nMSetOut_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_OUTPUT</td>
<td>Brake disengaging time</td>
</tr>
<tr>
<td>wReleaseTime</td>
<td>Word</td>
<td>-</td>
<td>VAR CONSTANT</td>
<td>Brake engaging time</td>
</tr>
<tr>
<td>wActivationTime</td>
<td>Word</td>
<td>-</td>
<td>VAR CONSTANT</td>
<td>Holding torque</td>
</tr>
<tr>
<td>nMSet</td>
<td>Integer</td>
<td>-</td>
<td>VAR CONSTANT</td>
<td>These signals are processed as quantities within the FB.</td>
</tr>
</tbody>
</table>

Parameter codes of the instances

<table>
<thead>
<tr>
<th>VariableName</th>
<th>SettingRange</th>
<th>Lenze</th>
</tr>
</thead>
<tbody>
<tr>
<td>wReleaseTime</td>
<td>C0196</td>
<td>0.0</td>
</tr>
<tr>
<td>wActivationTime</td>
<td>C0195</td>
<td>0.0</td>
</tr>
<tr>
<td>nMSet</td>
<td>C0244</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Range of functions
- Close brake
- Open the brake (release)
- Set controller inhibit

2.1.1 Close brake

Functional sequence
1. Select the constant FIXED0% (selection number 1000) using \( b\text{Set}_b = \text{TRUE} \), the function “close brake” is activated.
   - At the same time, \( b\text{QSP}_b \) 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 \( n\text{Nx}_a \), then \( b\text{Out}_b \) 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 \( b\text{Out}_b = \text{TRUE} \), a timer is started. After the time defined by \( w\text{ActivationTime} \) has elapsed, then \( b\text{CInh}_b \) 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 \( b\text{Out}_b = \text{TRUE} \), 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 \( bSet_b = \text{FALSE} \), then \( bCInh_b \) switches = \text{FALSE} immediately. At the same time, \( bMStore_b \) switches immediately = \text{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 \( bOut_b = \text{FALSE} \), immediately.

3. When the input is reset, a time element is triggered. After the time defined by \( wReleaseTime \) has elapsed, then \( bQSP_b = \text{FALSE} \), 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 = \text{FALSE} and \( bMStore_b \) = \text{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.
2.1.1.3 Set controller inhibit

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

**Functional sequence**

1. By setting the controller inhibit \((\text{DCTRL}\_\text{bCInh}\_\text{b} = \text{TRUE})\) \(\text{bOut}\_\text{b}\) switches immediately = TRUE.
   
   – The drive is then braked by its mechanical brake.

2. If the controller inhibit \((\text{DCTRL}\_\text{bCInh}\_\text{b} = \text{FALSE})\) before the actual speed falls below the threshold at \(\text{nNx}\_\text{a}\), then \(\text{bOut}\_\text{b}\) = 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)
Special functions

Holding brake (L_BRK)

Abb. 2-5 Switching cycle when braking
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.

### Variable Descriptions

<table>
<thead>
<tr>
<th>VariableName</th>
<th>DataType</th>
<th>SignalType</th>
<th>VariableType</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAdapt_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Dynamic adjustment of the proportional gain of the UGsetp -control in [%] of nVp. (16384 = 100 %)</td>
</tr>
<tr>
<td>nConst_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Proportional gain of the UGsetp controller in [%] from nVp. (16384 = 100 %)</td>
</tr>
<tr>
<td>nDcSet_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Voltage setpoint at which the DC bus voltage is to be maintained. (1000 V = 100 %)</td>
</tr>
<tr>
<td>nNSet_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Speed setpoint in [%] of C0011 (C0011 = 100 % = 16384)</td>
</tr>
<tr>
<td>tFault_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_INPUT</td>
<td>TRUE = activates the supply-failure control</td>
</tr>
<tr>
<td>bReset_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_INPUT</td>
<td>TRUE = reset</td>
</tr>
<tr>
<td>tThreshold_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Restart threshold in [%] of C0011 (C0011 = 100 % = 16384)</td>
</tr>
<tr>
<td>nAct_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Comparison value for the restart threshold in [%] of C0011</td>
</tr>
<tr>
<td>nSet_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_INPUT</td>
<td>Speed starting point for run-down/deceleration in [%] of C0011 (C0011 = 100 % = 16384)</td>
</tr>
<tr>
<td>nOut_a</td>
<td>Integer</td>
<td>analog</td>
<td>VAR_OUTPUT</td>
<td>Speed setpoint in [%] of C0011 (C0011 = 100 % = 16384)</td>
</tr>
<tr>
<td>bStatus_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_OUTPUT</td>
<td>TRUE = supply-failure control is active</td>
</tr>
<tr>
<td>bIReset_b</td>
<td>Bool</td>
<td>binary</td>
<td>VAR_OUTPUT</td>
<td>TRUE = supply-failure control active, the drive is braking</td>
</tr>
<tr>
<td>nIp</td>
<td>Integer</td>
<td>-</td>
<td>VAR_CONSTANT</td>
<td>Gain</td>
</tr>
<tr>
<td>nTn</td>
<td>Integer</td>
<td>-</td>
<td>VAR_CONSTANT</td>
<td>Integral-action time T</td>
</tr>
<tr>
<td>nTi</td>
<td>Integer</td>
<td>-</td>
<td>VAR_CONSTANT</td>
<td>Acceleration time</td>
</tr>
<tr>
<td>wRetriggerTime</td>
<td>Unsigned Integer</td>
<td>-</td>
<td>VAR_CONSTANT</td>
<td>Retrigger time</td>
</tr>
</tbody>
</table>

### Parameter Codes

<table>
<thead>
<tr>
<th>VariableName</th>
<th>L_MFAIL1</th>
<th>SettingRange</th>
<th>Lenze</th>
</tr>
</thead>
<tbody>
<tr>
<td>nIp</td>
<td>C0980</td>
<td>0.001 ... 31.000</td>
<td>0.500</td>
</tr>
<tr>
<td>nTn</td>
<td>C0981</td>
<td>20 ... 2000 ms</td>
<td>100</td>
</tr>
<tr>
<td>nTi</td>
<td>C0982</td>
<td>0.001 ... 16.000 s</td>
<td>2.000</td>
</tr>
<tr>
<td>wRetriggerTime</td>
<td>C0983</td>
<td>0.001 ... 60.000 s</td>
<td>1.000</td>
</tr>
</tbody>
</table>
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).

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 (nIn1 < nIn2)
Supply-failure detection of the supply module

- A digital output of the supply module 934x is switched to the function block \text{L\_MFAIL} via the digital inputs \text{DIGIN} of the 93XX controller. In the example, input X5/E4 is used.

![Diagram](image)

**Abb. 2-8** Example of a supply-failure detection by an external monitoring system

Programming the example in Abb. 2-8:
1. Set the signal links according to Abb. 2-8
2. 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.

![Diagram](image)

**Abb. 2-9** Example of a supply-failure detected by different sources

Programming the example in Abb. 2-9:
1. Set the signal links according to Abb. 2-9
2. In FB \text{L\_CMP}, set \text{byFunction} = 3 (\text{nln1} < \text{nln2})
2.1.2.2 Supply failure control

Integration of the FB into the signal flow of the controller

Programming the example (SpeedModeInternal24VSupply.lpc) in Abb. 2-10:

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Inputs/outputs of L_MFAIL</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed setpoint path</td>
<td>nNSet_a, nNOut_a</td>
<td>Here, the actual speed value</td>
</tr>
<tr>
<td>Start value for deceleration</td>
<td>nSet_a</td>
<td></td>
</tr>
<tr>
<td>Source for the setpoint of the DC bus voltage</td>
<td>nDCSet_a</td>
<td>Here, from the freely linkable code</td>
</tr>
<tr>
<td>Source for the activation of the supply failure control:</td>
<td>bFault_b</td>
<td>□ 2-7: Supply-failure detection</td>
</tr>
<tr>
<td>Proportional gain and adaptation of the DC bus voltage controller:</td>
<td>n_Adapt_a, nConst_a</td>
<td></td>
</tr>
<tr>
<td>Restart protection</td>
<td>nThreshold_a, nAct_a</td>
<td>In FCODE_nC472_18_a first, enter approx. 2% (reference: nmax, C0011)</td>
</tr>
<tr>
<td>Reset input</td>
<td>bReset_b</td>
<td>Here, with terminal DIGIN_bIn5_b</td>
</tr>
</tbody>
</table>

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

- $bFault_b = \text{TRUE}$ activates the supply-failure control
- $bFault_b = \text{FALSE}$ 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 $bReset_b = \text{TRUE}$.

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 (adjsutment 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).

<table>
<thead>
<tr>
<th>Supply voltage range</th>
<th>C0173 =</th>
<th>Switch-off threshold LU</th>
<th>Switch-on threshold LU</th>
<th>Switch-off threshold OU</th>
<th>Switch-on threshold OU</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 400 V</td>
<td>0</td>
<td>285 V</td>
<td>430 V</td>
<td>770 V</td>
<td>755 V</td>
</tr>
<tr>
<td>400 V</td>
<td>1</td>
<td>285 V</td>
<td>430 V</td>
<td>770 V</td>
<td>755 V</td>
</tr>
<tr>
<td>400 V ... 460 V</td>
<td>2</td>
<td>328 V</td>
<td>473 V</td>
<td>770 V</td>
<td>755 V</td>
</tr>
<tr>
<td>480 V without brake chopper</td>
<td>3</td>
<td>342 V</td>
<td>487 V</td>
<td>770 V</td>
<td>755 V</td>
</tr>
<tr>
<td>Operation with brake chopper (up to 480 V)</td>
<td>4</td>
<td>342 V</td>
<td>487 V</td>
<td>800 V</td>
<td>785 V</td>
</tr>
</tbody>
</table>

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.
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

![Diagram](image-url)

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
- \( t_1 \) Supply-failure
- \( t_2 \) Zero speed reached
Function block library Lenze9300Servo.lib
Special functions
Supply-failure control (L_MFAIL)

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 = t1 Supply-failure
- t = t2 Zero speed with higher brake torque (short adjustment time)
- t = t3 Drive reaches the LU switch-off threshold with lower brake torque (high adjustment time), without reaching zero speed
- t > t3 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 \( b\text{Reset}_b = \text{TRUE} \) (in the example, through X5/E5 (when DIGIN_bIn5_b is connected to \( b\text{Reset}_b \)).
- 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 (\( c94 \): Parameterization of the example in Abb. 2-10)
- In C0472/18 ( FCOD\_nC472\_18\_a ), enter the threshold in [%] of \( n_{\text{max}} \) (C0011) below which no automatic start is wanted after supply recovery.
  - If the speed at supply recovery < threshold in C0472/18 ( FCOD\_nC472\_18\_a ): the drive will still be braked under control. This function can only be ended by \( b\text{Reset}_b = \text{TRUE} \).
  - If the speed at supply recovery > threshold in C0472/18 ( FCOD\_nC472\_18\_a ): the drive accelerates to its setpoint along the set ramps.
  - The function is deactivated by: \( n\text{Threshold}_a = 0\% \).
- A reset is made by \( b\text{Reset}_b = \text{TRUE} \)
  - This is required after every supply (re)connection, and is shown by \( b\text{Status}_b = \text{HIGH} \), when \( b\text{Fault}_b = \text{FALSE} \).

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 ( \( n\text{Const}_a \) and \( n\text{Adapt}_a \)). The resulting proportional gain results from:

\[
V_p = nV_p \cdot \frac{n\text{Const}_a - |n\text{Adapt}_a|}{100}\% \\
(100\% = 16384)
\]
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 `+UG, -UG` 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:

<table>
<thead>
<tr>
<th>Column</th>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>C0039</td>
<td>Code C0039</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Subcode 1 of code C0039</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Subcode 2 of code C0039</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Subcode 14 of code C0039</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Subcode 15 of code C0039</td>
</tr>
<tr>
<td></td>
<td>[CD156]</td>
<td>Parameter value of the code can only be modified when the controller is inhibited</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LCD</th>
<th>Keypad LCD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• DIS ... display only</td>
</tr>
<tr>
<td></td>
<td>• all others are parameter values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lenze</th>
<th>Factory setting of the code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>The column &quot;important&quot; contains further information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Choice</th>
<th>1 (% )</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum value</td>
<td>(smallest step/unit)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMPORTANT</th>
<th>Additional, important explanation of the code</th>
</tr>
</thead>
</table>

### 3.1.1 L_BRK

FB description: [□ 2-1]

<table>
<thead>
<tr>
<th>Code</th>
<th>LCD</th>
<th>Possible settings</th>
<th>IMPORTANT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wActivationTime</td>
<td>99.9</td>
<td>Brake closing time of L_BRK1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0</td>
<td>Engaging time of the mechanical holding brake (see technical data of the brake).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99.9 s infinite</td>
<td>After the time under C0195 has elapsed, the status &quot;mechanical brake closed&quot; is reached</td>
</tr>
<tr>
<td></td>
<td>wReleaseTime</td>
<td>0.0</td>
<td>Brake opening time of L_BRK1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0</td>
<td>Disengaging time of the mechanical holding brake (see technical data of the brake).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.1 s)</td>
<td>After the time under C0196 has elapsed, the status &quot;mechanical brake opened&quot; is reached</td>
</tr>
<tr>
<td></td>
<td>nMSet</td>
<td>0.00</td>
<td>Holding moment of the DC-brake of L_BRK1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-100.00</td>
<td>100 % = value of C0057</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.01 %)</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### 3.1.2 L_MFAIL

FB description: [□ 2-6]

<table>
<thead>
<tr>
<th>Code</th>
<th>LCD</th>
<th>Possible settings</th>
<th>IMPORTANT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nVp</td>
<td>0.500</td>
<td>Gain Up of L_MFAIL1</td>
</tr>
<tr>
<td></td>
<td>nTn</td>
<td>100</td>
<td>Time constant of L_MFAIL1</td>
</tr>
<tr>
<td></td>
<td>nTir</td>
<td>2.000</td>
<td>Acceleration time Tir of L_MFAIL1</td>
</tr>
<tr>
<td></td>
<td>wRetriggerTime</td>
<td>1.000</td>
<td>Retrigger time of L_MFAIL1</td>
</tr>
</tbody>
</table>
3.2 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, 1-3
Warning of damage to material, 1-3
Signal type, Explanation of, 1-3
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

V
Variable names
Conventions, Hungarian Notation, 1-3
Lenze software guidelines, Explanation of, 1-3
Variable type, Labelling, 1-4