

Lenze

Manual



Global Drive

Software package - Cam

Function library

LenzeCamControl.lib

This documentation is valid for the **LenzeCamControl1Vxxx.lib** and **LenzeCamControl2Vxxx.lib** function libraries as of version 2.00.

The function libraries can be used for the following Lenze PLC devices:

	Type designation	As of hardware version	As of software version
9300 Servo PLC	EVS93XX-xT	6A	6.0
ECSxA	ECSxAxxx	1C	7.0

System requirements **Drive PLC Developer Studio:**

DDS-P (ESP-DDS2-P) or DDS-PD (ESP-DDS2-PD) as of version 2.2

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.

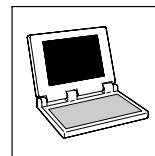
We do not take any liability for direct or indirect damage, e.g. profit loss, order loss or any loss regarding business.

© 2007 Lenze Drive Systems GmbH

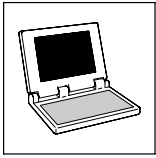
No part of this documentation may be copied or made available to third parties without the explicit written approval of Lenze Drive Systems GmbH.

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

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

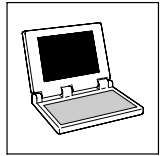


1	Preface and general information	1-1
1.1	About this Manual	1-1
1.1.1	Conventions used in this Manual	1-1
1.1.2	Pictographs used in this Manual	1-1
1.1.3	Structure of the descriptions	1-2
1.1.4	Terminology used	1-2
1.2	Version identifiers of the function libraries	1-3
1.3	Technology function "Cam profiler"	1-4
1.3.1	Concept	1-4
1.3.2	General information about the function blocks	1-4
2	Function blocks LenzeCamControl1V02xx.lib	2-1
2.1	L_CamClutchXAxis - Virtual clutch	2-2
2.2	L_CamContactor - Cam group	2-7
2.3	L_CamContactorVPos - Dynamic cam/hysteresis adjustment	2-16
2.4	L_CamControl - Setpoint conditioning	2-20
2.5	L_CamData - Profile management	2-28
2.6	L_CamExtrapolate - Extrapolation	2-37
2.7	L_CamGetPosition - Unit conversion	2-39
2.8	L_CamLimitVelocity - limiting element	2-40
2.9	L_CamMachineData - Machine parameters	2-41
2.10	L_CamMonitFollowError - Following error monitoring	2-43
2.11	L_CamPosCtrlLin - Linear profile generator	2-45
2.12	L_CamProfileData - Loading of important profile points	2-49
2.13	L_CamSequence - Selection of a profile sequence	2-51
2.14	L_CamSetPosition - Unit conversion	2-57
2.15	L_CamSetXAxisVelocity - Stretching and compression of the master value	2-58
2.16	L_CamSetYAxisVelocity - Stretching and compressing the setpoint	2-63
2.17	L_CamVMasterPosition - Virtual master with positioning function	2-67
2.18	L_CamVMasterVelocity - Virtual master	2-73
3	Function blocks LenzeCamControl2V02xx.lib	3-1
3.1	L_CamAddVelocity - speed addition	3-2
3.2	L_CamAdjustDfIn - master frequency adaptation	3-3
3.3	L_CamDifferentiate - Converter/differentiator	3-4
3.4	L_CamEditProfileData - Online change of profile points	3-5
3.5	L_CamIntegrateVarLim - Phase integrator	3-11
3.6	L_CamSectionShift - Section shift	3-14
3.7	L_CamSelPosition - Master value selection	3-17
3.8	L_CamSelVelocity - Master frequency selection	3-18
3.9	L_CamVelocityFilter - Filter (PT1) for speed signals	3-19
4	Index	4-1



Function library LenzeCamControl.lib

Contents



1 Preface and general information

1.1 About this Manual

This Manual contains information on the function libraries **LenzeCamControl1Vxxx.lib** and **LenzeCamControl2Vxxx.lib** for the **Drive PLC Developer Studio**.

- The function libraries **LenzeCamControl1Vxxx.lib** and **LenzeCamControl2Vxxx.lib** contain functions for implementing cam tasks.
- The function libraries are an integral part of the **Cam software package**.

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 on the conventions used for the variables of Lenze system blocks, function blocks and functions can be obtained from the appendix of the DDS online documentation "Introduction into IEC 61131-3 programming". The conventions ensure universal and uniform labelling and make reading the PLC program easier.

Lenze functions/function blocks

can be recognised by their names. They always begin with an "L_":

- "By means of the **L_CamControlIFB**..."

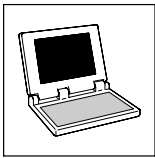
Program listings

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

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

1.1.2 Pictographs used in this Manual

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



Function library LenzeCamControl.lib

Preface and general information

About this Manual

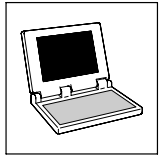
1.1.3 Structure of the descriptions

All function/function block descriptions as well as system blocks contained in this Manual have the same structure:

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

1.1.4 Terminology used

Term	Used below for
Cxxx/y	Subcode y of code Cxxxx (e.g. C0470/3 = subcode 3 of code C0470)
DDS	Drive PLC Developer Studio
FB	Function block
GDC	Global Drive Control (parameter setting program from Lenze)
Parameter codes	Codes for setting the function of a function block
PLC	9300 Servo PLC - ECSxA "Application" axis module
SB	System block
[unit]	Wildcards for the physical unit of the external measuring system used (e. g. "µm", "mm", "cm")



1.2 Version identifiers of the function libraries

Function library version identifiers as of version 1.05

- LenzeCamControl1Vxxxx.lib
- LenzeCamControl2Vxxxx.lib

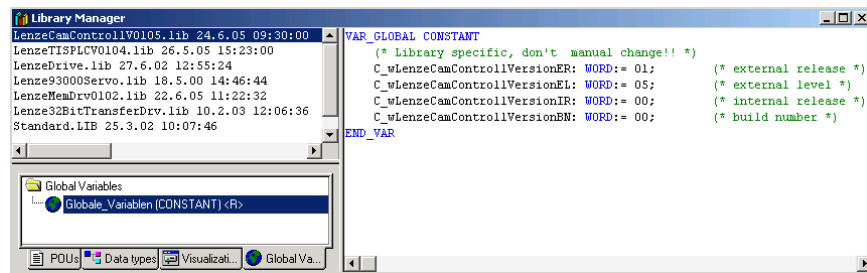
Constant	Meaning	Example value
C_w[function library name]VersionER	External release	01
C_w[function library name]VersionEL	External level	05
C_w[function library name]VersionIR	Internal release	00
C_w[function library name]VersionBN	Build number	00

Version: 01 05 00 00

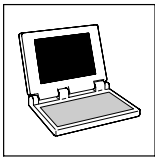
The value of this constant is hex-coded.

- In the example, the value "01050000" stands for version "1.05.00.00".

In the offline mode you can have the value of these constants displayed via **Library management/register Global variables**:



- Select the command **Window→Library manager** to open the Library manager in DDS.
- Select the **Global variables** tab and open the **Global variables** folder.
- Select *Global_variables...* to indicate the version in the Library manager.



Function library *LenzeCamControl.lib*

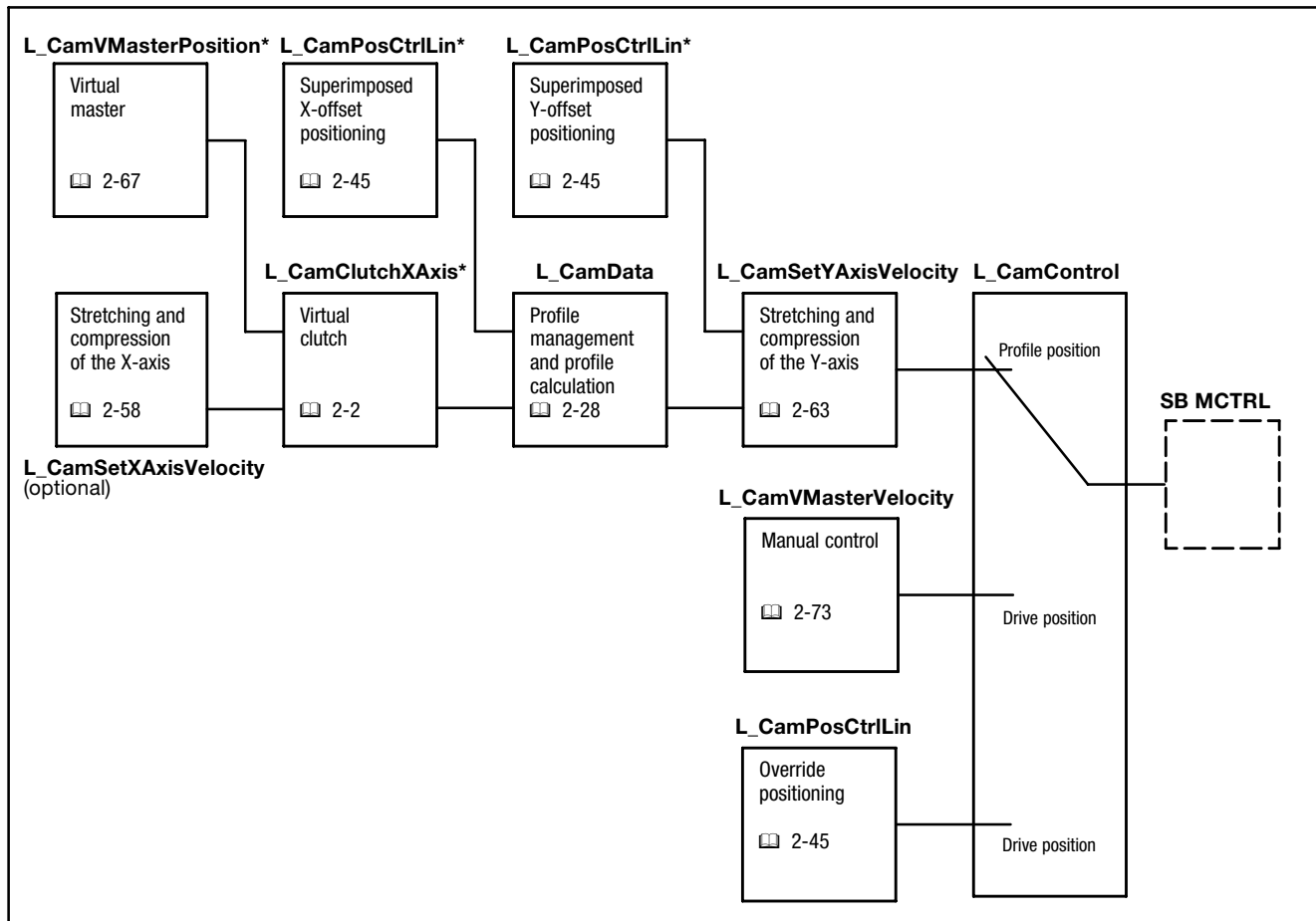
Preface and general information

Technology function "Cam profiler"

1.3 Technology function "Cam profiler"

1.3.1 Concept

The function blocks for the technology function "Cam profiler" are based on the following concept:



1.3.2 General information about the function blocks

Time-equidistant call of function blocks

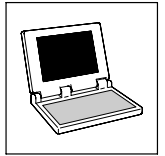
Due to their internal structure, the following function blocks have to be activated in a time-equidistant task (e. g. 2-ms task):

L_CamClutchXAxis	L_CamExtrapolate	L_CamSetXAxisVelocity
L_CamContactor	L_CamIntegrateVarLim	L_CamSetYAxisVelocity
L_CamControl	L_CamPosCtrlLin	L_CamVMasterPosition
L_CamData	L_CamSectionShift	L_CamVMasterVelocity
L_CamDifferentiate		



Note!

The cyclic task **PLC_PRG** is not time-equidistant!



Enabling input "bEnable"

Function blocks with this input can be activated (TRUE) or deactivated to save computing time if not used (FALSE).



Stop!

Please note that state changes at the enabling input *bEnable* may set the drive in motion (compensating movement).

Status output "nState"

Use this output to query the current status of some function blocks.

- Values > "0" indicate warnings, however, the internal function is not interrupted.
- Value "0" indicates "Ok".
- Values < "0" indicate errors and lead to a function stop.



Tip!

The indicated error number is not saved internally and is only available as long as the error exists.



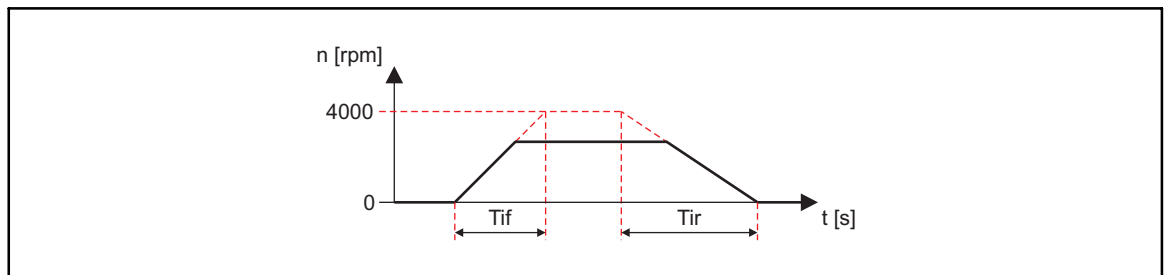
Stop!

Please note that warnings can lead to mistakes!

Acceleration/deceleration ramps

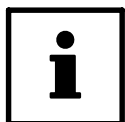
With function blocks providing positioning features, *Ti* times define the falling and rising edge rates.

- *Ti* times can be changed in the corresponding codes.
- The *Ti* times always refer to a speed change of 4000 rpm:



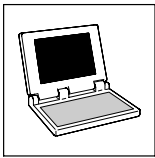
Function blocks with *.lc9 access

The function blocks **L_CamContactor**, **L_CamData**, **L_CamMachineData** and **L_CamProfileData** access the *.lc9-file which contains the data of the motion profiles and is stored in the RAM memory block 1 of the PLC. The *.lc9-file is generated via the **CamDesigner** (see also the corresponding documentation of the CamDesigner/CamLoader).



Note!

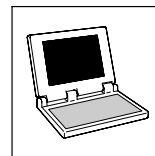
When using the **RAM memory block 1** to store the *.lc9.file, this memory area is not available anymore. Please bear this in mind when using the memory functions of the function library **LenzeMemDrv.lib** (see also the Manual for the function library **LenzeMemDrv.lib**)!



Function library LenzeCamControl.lib

Preface and general information

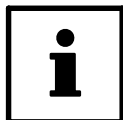
Technology function "Cam profiler"



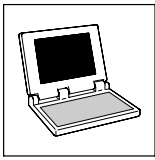
2

Function blocks LenzeCamControl1V02xx.lib

The function blocks for the "Cam profiler" technology function are divided into the two function libraries **LenzeCamControl1V02xx.lib** and **LenzeCamControl2V02xx.lib**.



In this chapter you'll be provided with information on the function blocks which are contained in the **LenzeCamControl1V02xx.lib** function library.



Function library LenzeCamControl.lib

Function blocks

L_CamClutchXAxis - Virtual clutch

2.1 L_CamClutchXAxis - Virtual clutch

Use this FB to engage and disengage the virtual line shaft (x-axis) via adjustable acceleration and deceleration ramps.

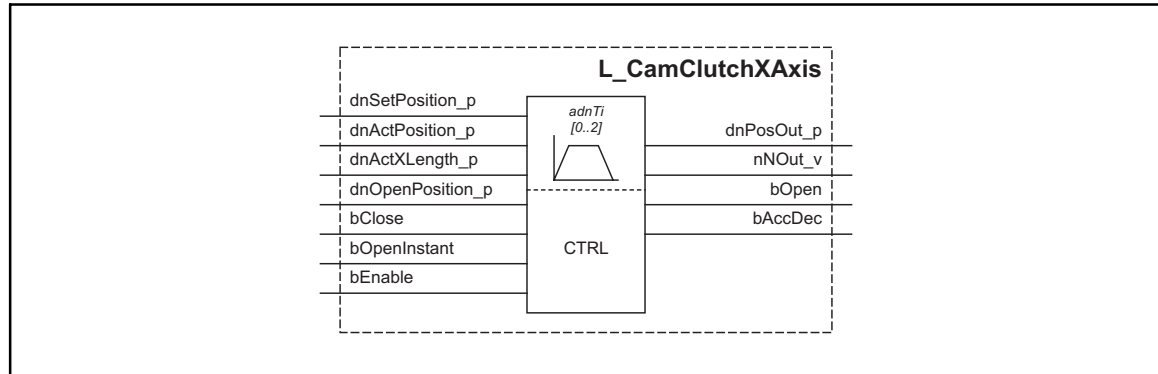
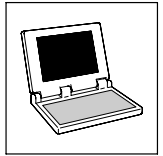


Fig. 2-1 "L_CamClutchXAxis" function block

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
dnSetPosition_p	Double integer	Position	Input of the virtual line shaft (external)	
dnActPosition_p	Double integer	Position	Actual position of the virtual line shaft (internal) e. g. feedback of <i>dnPosOut_p</i> (see connection example)	
dnActXLength_p	Double integer	Position	Cycle of x axis Connect input to <i>dnActXLength_p</i> of L_CamData FB	
dnOpenPosition_p	Double integer	Position	Position setpoint of virtual line shaft after clutch disengagement at standstill	
bClose	Bool	Binary	Clutch	
			FALSE	Disengage clutch, brake along <i>adnTi[2]</i> and stop at <i>dnOpenPosition_p</i> .
			TRUE	Synchronise along <i>adnTi[1]</i> to <i>dnSetPosition_p</i> , engage clutch.
bOpenInstant	Bool	Binary	Forced disengagement - This input has priority over <i>bClose</i>. - The standstill position of the x axis results from the current speed of the x axis and the set deceleration ramp <i>adnTi[0]</i>.	
			FALSE	Clutch function active
			TRUE	The line shaft (internal) is decelerated to standstill along <i>adnTi[0]</i> .
bEnable	Bool	Binary	Activation of the clutch function This input has highest priority.	
			FALSE	<i>dnSetPosition_p</i> is directly output on output <i>dnPosOut_p</i> . The ramp function is inactive. The clutch is closed (engaged).
			TRUE	Clutch function active

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Value/meaning
dnPosOut_p	Double integer	Position	Actual position of the virtual line shaft (internal)
nNOut_v	Integer	Velocity	Speed of the virtual line shaft (internal)
			Note: The phase difference signal considers the incremental cycle skip from $x_{\max}$ to 0 at the end/start of a cycle, but not a possible incremental decimal fraction in the cycle. Therefore do not use this output signal if the cycle cannot be represented incrementally as an integer!
bOpen	Bool	Binary	Disengaging clutch
			<table><tr><td>TRUE</td><td>Drive no longer follows master angle.</td></tr></table>
TRUE	Drive no longer follows master angle.		



Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Value/meaning
bAccDec	Bool	Binary	Acceleration/deceleration (internal line shaft)
			FALSE No acceleration/deceleration.
			TRUE Line shaft is synchronised for master angle (i. e. accelerated or decelerated).

Parameters (Variable type: VAR_CONSTANT_RETAIN)				
Identifiers Code*	Data type	Signal type	Information	
			Possible settings	Presetting
adnTi[0..2]	Array of double integer	-	Ramps for clutch engagement/disengagement (relating to 4000 rpm)	
C1412/1			adnTi[0] = deceleration ramp for forced disengagement (bOpenInstant = TRUE)	
			0.010	{0.001 unit} 130.000 1.000 s
C1412/2			adnTi[1] = acceleration ramp for clutch engagement (bClose = TRUE)	
			0.010	{0.001 unit} 130.000 1.000 s
C1412/3			adnTi[2] = deceleration ramp for clutch disengagement (bClose = FALSE)	
			0.010	{0.001 unit} 130.000 1.000 s
* Code of first instance				

* Code of first instance

Internal connection

Determination of the position of the x axis (master drive) is possible by connecting the input *dnActXLength_p* to the corresponding output *dnActXLength_p* of the **L_CamData** FB:

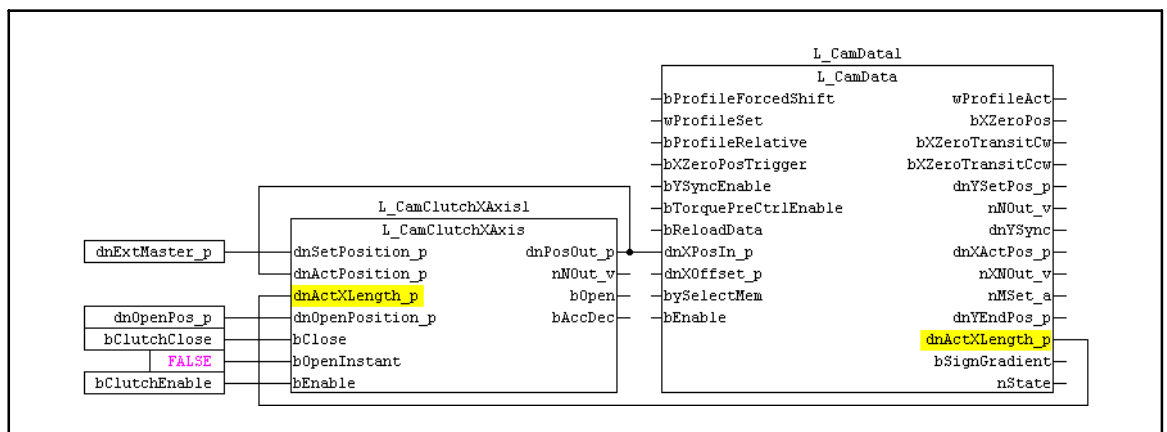


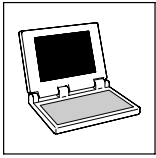
Fig. 2-2

Internal connection for determination of positions



Note!

When changing between profiles with different x axis lengths, the master angle must only rotate in positive direction.



Function library *LenzeCamControl.lib*

Function blocks

L_CamClutchXAxis - Virtual clutch

Activation of the clutch function

The clutch function is activated with *bEnable* = TRUE, the response of the clutch function depends on the input *bClose*:

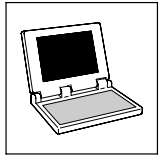
- *bClose* = TRUE: request for engaging the clutch.
- *bClose* = FALSE: request for disengaging the clutch.



Tip!

With *bEnable* = FALSE, the clutch is considered to be engaged.

- The master angle of the line shaft is directly connected, it is not led along ramps (the outputs follow the input *dnSetPosition_p*).
- The function "Direct engagement" can be implemented.



Clutch engagement via ramp function with subsequent synchronous running

After a - mostly process-related - pause, the cam drive can be reconnected to the process by means of this FB if the line shaft is passing through (master drive).

At the time of clutch engagement (bopen = FALSE), internal and external line shafts run at synchronous speed and are in the same position.

The engagement process is started with bClose = TRUE. The synchronisation process is carried out automatically as shown in the following diagram.

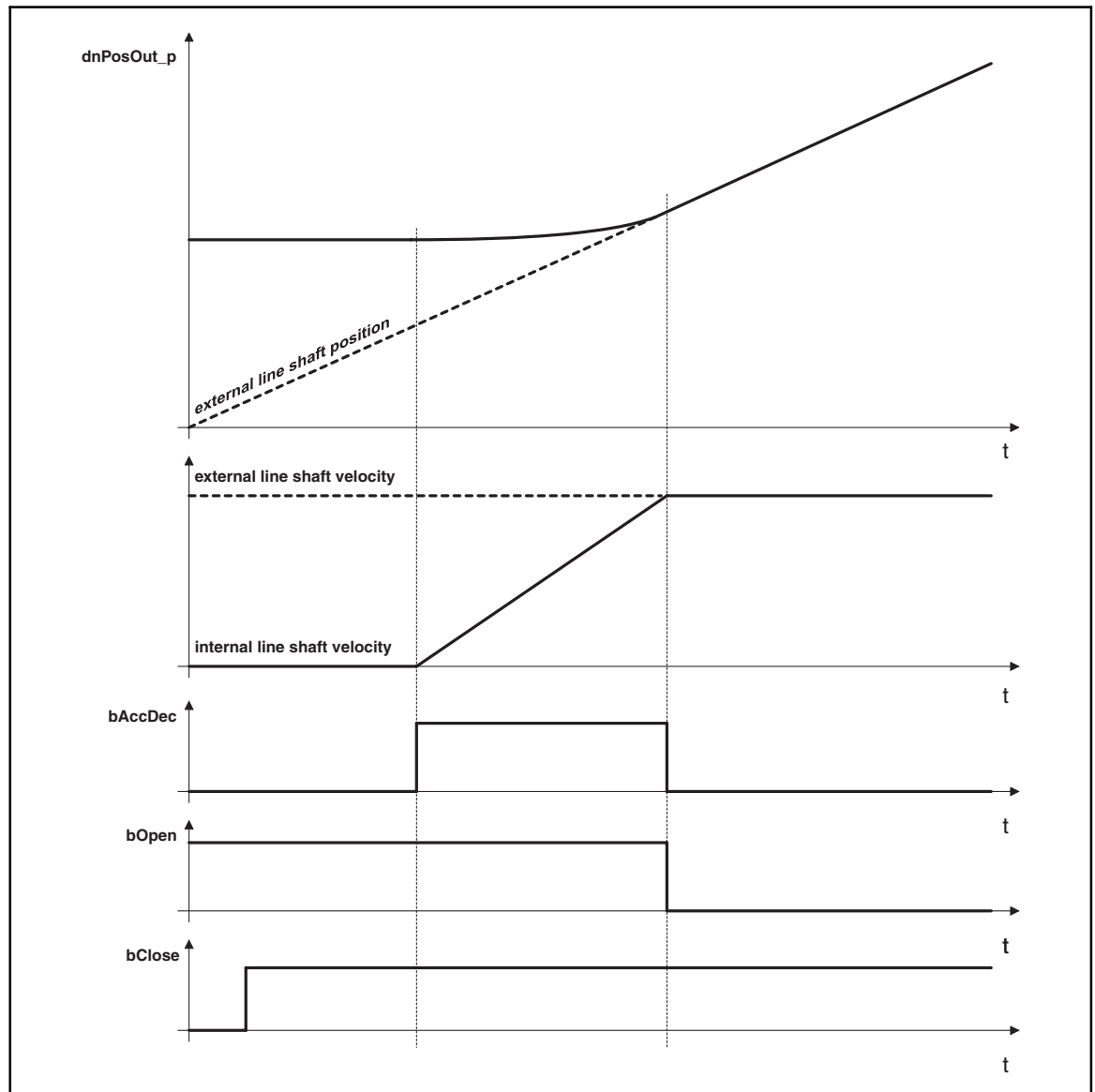
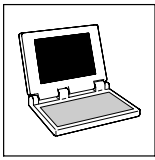


Fig. 2-3

Clutch engagement via ramp function



Function library *LenzeCamControl.lib*

Function blocks

L_CamClutchXAxis - Virtual clutch

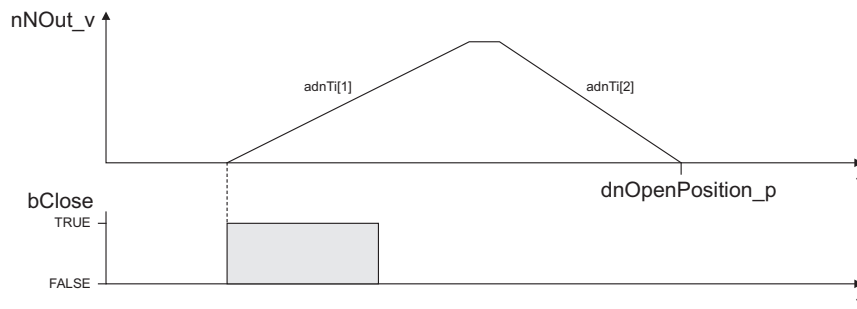
1. The FB calculates the starting time, so that the set ramp *adnTi[1]* meets the synchronous point.
 - Depending on the ramp, the engagement process can continue over several cycles.
 - If the position at *dnSetPosition_p* is only changed slightly per cycle and it is not possible to calculate an acceleration ramp, the clutch is engaged at the synchronous point.
2. The internal line shaft is accelerated as soon as the starting position is reached. During acceleration, *bAccDec* is set to TRUE.
3. As soon as the synchronous point is reached, *bAccDec* and *bOpen* are set FALSE.



Note!

The input *bClose* must remain TRUE as long as the internal line shaft is to run synchronously to the external line shaft.

If *bClose* is reset to FALSE before the engagement process has been completed, the engagement process is completed before disengagement via the ramp function to position *dnOpenPosition_p* is started:



Disengagement via ramp function

Disengagement means that the internal line shaft is disconnected from the external line shaft.

The disengagement process is started by resetting *bClose* to FALSE:

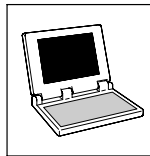
1. The clutch is disengaged.
2. Braking is effected via the *adnTi[2]* deceleration ramp.
 - Depending on the ramp, the disengagement process can continue over several cycles.
3. Stopping is effected on *dnOpenPosition_p*.

Immediate disengagement (forced disengagement)

This function corresponds to a mechanical clutch, internal and external line shafts are disconnected immediately.

The disengagement process is started by setting *bOpenInstant* to TRUE:

1. The output *bOpen* is set to TRUE.
2. The internal line shaft is decelerated to standstill along the ramp *adnTi[0]*. It is not positioned on *dnOpenPosition_p*.
3. As long as the internal line shaft is not at standstill, *bAccDec* is set to TRUE.



2.2 L_CamContactor - Cam group

Use this FB to build up a cam controller providing the following options:

- Travel cams in positive effective direction
- Travel cams in negative effective direction
- Travel cams in both effective directions
- Travel-time cams in positive effective direction
- Travel-time cams in negative effective direction

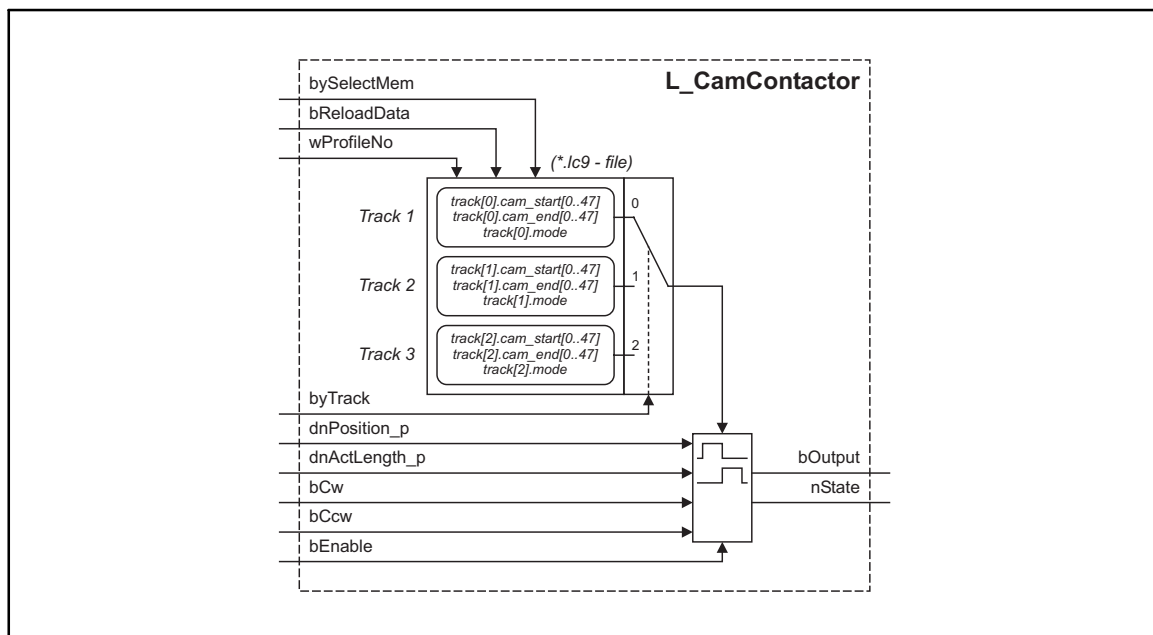
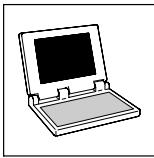


Fig. 2-4 Function block "L_CamContactor"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
bySelectMem	Byte	Number	Selection of data structure (*.lc9 file) in the memory. The selection can only be accepted with a FALSE-TRUE transition at <i>bEnable</i> or <i>bReloadData</i>.
			1RAM block 1 (foreground data field, standard)
			2RAM block 2 (background data field)
bReloadData	BOOL	Binary	Accept selected data structure
			FALSE → TRUEThe data structure selected under <i>bySelectMem</i> is accepted and the cams are initialised.
wProfileNo	Word	-	Profile selection
			0 ... 47Use of data of profile no. 0 ... 47
byTrack	Byte	-	Track selection
			0 ... 2Track 0 ... 2
			Different valueTrack 0; <i>nFailNumber</i> is set to "2".
dnPosition_p	Double Integer	Position	Axis position
dnActLength_p	Double Integer	Position	Length of the X and Y axis
			0linear axes
			Clock pulse lengthcyclic axes



Function library *LenzeCamControl.lib*

Function blocks

L_CamContactor - Cam group

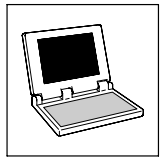
⇒ Inputs (Variable type: VAR_INPUT)				
Identifier	Data type	Signal type	Info/possible settings	
bCw	BOOL	Binary	Selection of direction of rotation	
			Connect this input with the output of the same name <i>bCw</i> of the FB L_CamContactorVPos.	
bCcw	BOOL	Binary	Selection of direction of rotation	
			Connect this input with the output of the same name <i>bCcw</i> of the FB L_CamContactorVPos.	
bEnable	BOOL	Binary	Function block enable	
			TRUE	positive (in CW direction)
			TRUE	negative (in CCW direction)
			Function block enable	
			FALSE	Function block inhibited
			TRUE	Function block enabled (cam controller active)

Outputs ⇒ (Variable type: VAR_OUTPUT)																	
Identifier	Data type	Signal type	Value/meaning														
bOutput	BOOL	Binary	Cam status, depending on the selected data set.														
nState	Integer	-	<table><tr><td colspan="2">Status</td></tr><tr><td>10</td><td>Initialisation active over two clock-pulse cycles.</td></tr><tr><td>2</td><td>Warning: <i>byTrack</i> exceeds the valid range 0 ... 2. • Data of the first track [0] will be used. </td></tr><tr><td>1</td><td>Warning: <i>wProfileNo</i> exceeds the valid range 0 ... 47. • Data of the first profile [0] will be used. </td></tr><tr><td>0</td><td>Ok, no error</td></tr><tr><td>-1</td><td>Error: No data or invalid mode entered in the data structure. • All outputs and started times will be reset. </td></tr><tr><td>-2</td><td>Error: No internal data available. • This FB can only be used together with the cam profiler technology. </td></tr></table>	Status		10	Initialisation active over two clock-pulse cycles.	2	Warning: <i>byTrack</i> exceeds the valid range 0 ... 2. • Data of the first track [0] will be used.	1	Warning: <i>wProfileNo</i> exceeds the valid range 0 ... 47. • Data of the first profile [0] will be used.	0	Ok, no error	-1	Error: No data or invalid mode entered in the data structure. • All outputs and started times will be reset.	-2	Error: No internal data available. • This FB can only be used together with the cam profiler technology.
Status																	
10	Initialisation active over two clock-pulse cycles.																
2	Warning: <i>byTrack</i> exceeds the valid range 0 ... 2. • Data of the first track [0] will be used.																
1	Warning: <i>wProfileNo</i> exceeds the valid range 0 ... 47. • Data of the first profile [0] will be used.																
0	Ok, no error																
-1	Error: No data or invalid mode entered in the data structure. • All outputs and started times will be reset.																
-2	Error: No internal data available. • This FB can only be used together with the cam profiler technology.																



Note!

- The FB must be activated during a cyclic task. It is not possible to activate it during an event-controlled task or **PLC_PRG**.
- If the direction of rotation is not selected with the FB *L_CamContactorVPos*, ensure that the input signals at *bCw* and *bCcw* correspond to the actual direction of rotation and are not both set to TRUE.



Selection of direction of rotation

Connect the inputs *bCw* and *bCcw* with the corresponding outputs of the FB *L_CamContactorVPos* to select the direction of rotation.

- Please observe that the FB **L_CamContactorVPos** needs two cycles to identify the direction of rotation to ensure that the cams do not toggle.
The output *bCw* of the FB **L_CamContactorVPos** will only be set to TRUE after CW rotation has been identified in at least two consecutive cycles.



Tip!

If you have ensured that the axis can only rotate in one direction and the cam data have been generated with the **CamDesigner**, the inputs for the direction of rotation *bCw* and *bCcw* can also be directly assigned.

Input in CamDesigner

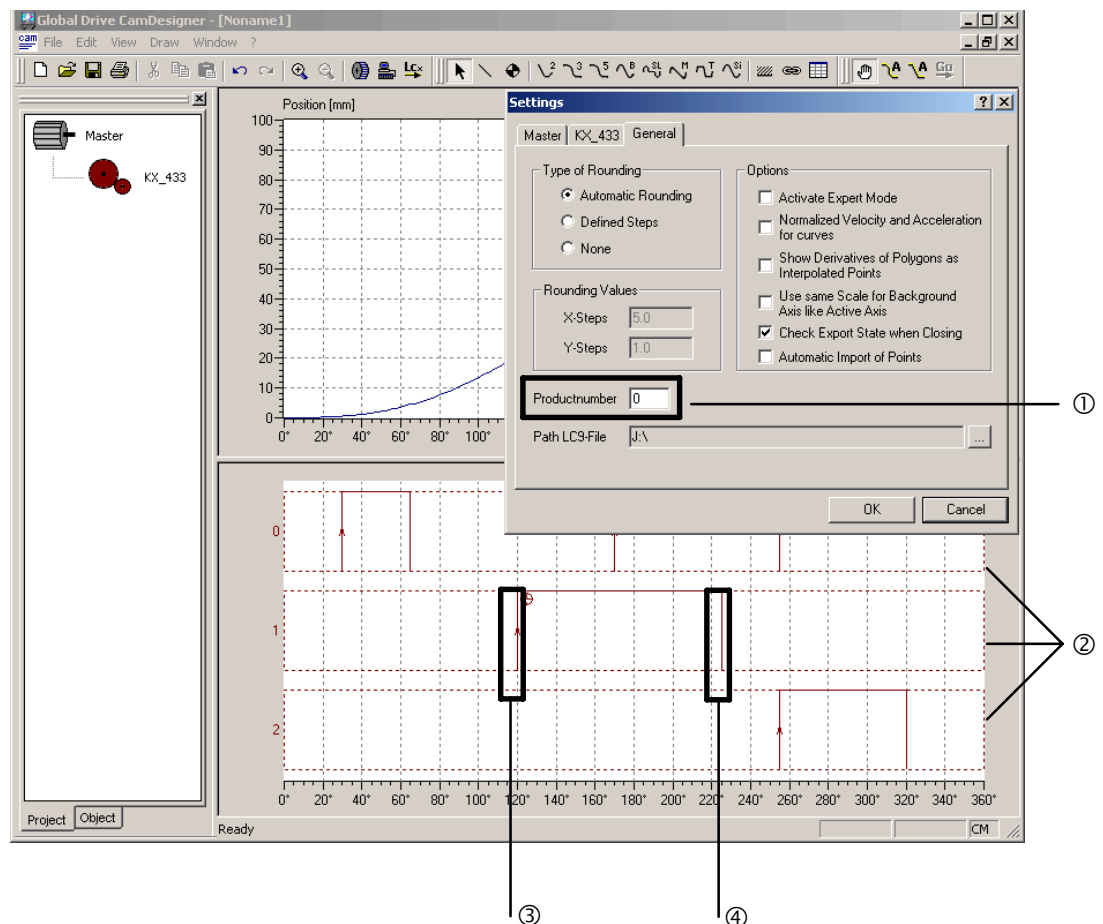
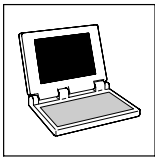


Fig. 2-5

Example: Display in CamDesigner

- ① Indication of product number or profile number *wProductNo* (0 ... 47)
- ② Indication of the three cam tracks (*byTrack*)
- ③ Beginning of a cam
- ④ End of a cam



Function library LenzeCamControl.lib

Function blocks

L_CamContactor - Cam group

Modes for the cam controller

A cam can be determined through five different modes:



Note!

Two successive cams are identified as single cams if the distance between them is greater than the change of position per clock cycle. This serves to prevent that the "OFF" state between the cams will be skipped.

For cams with very short operating time, the switch-on time must be longer than the change of position per clock cycle to ensure that the cam will be switched for at least one cycle.

Travel cams in positive effective direction



When reaching the reference value *cam_start* and changing in positive direction of the position input *dnPosition_p* occurs, the output *bOutput* will be set to TRUE and only if the value of *cam_end* is exceeded it will be reset to FALSE.

- If the axis goes back after *cam_start* was reached, the output *bOutput* will be reset to FALSE immediately.

Legend:

cam_start : start of a cam
cam_end : end of a cam
▲ / ▼ : edge-controlled

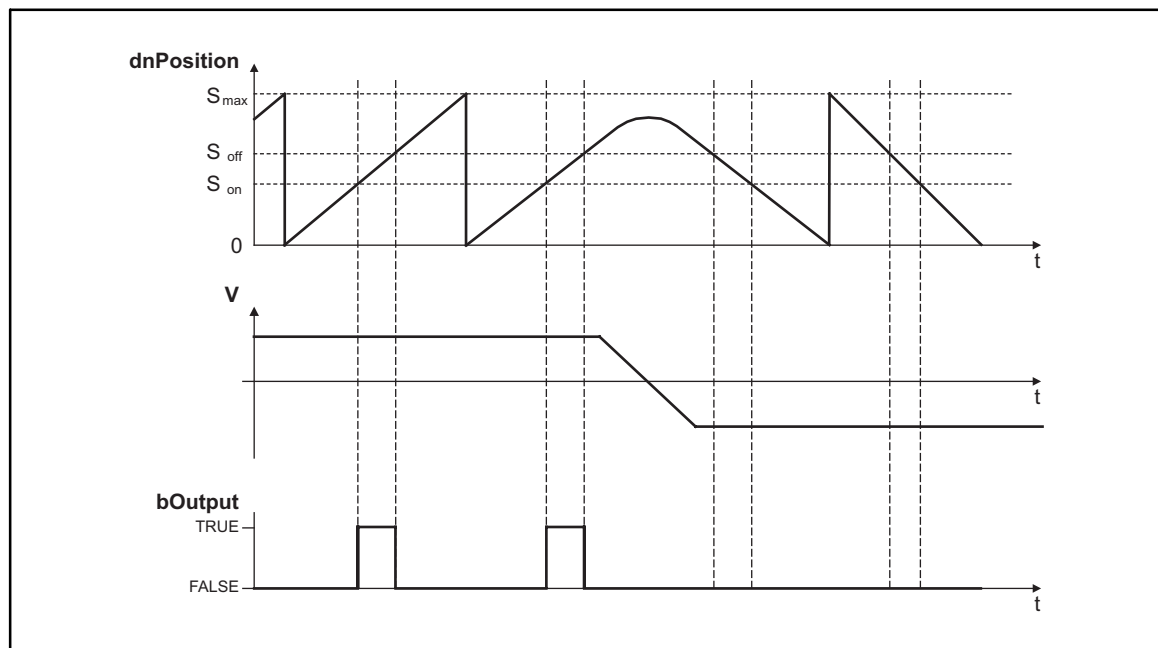
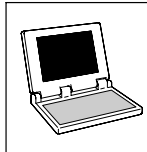
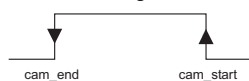


Fig. 2-6

Travel cams in positive effective direction



Travel cams in negative effective direction



When reaching the reference value *cam_end* and changing in negative direction of the position input *dnPosition_p* of the signal, the output *bOutput* is set to TRUE and reset to FALSE if the value of *cam_start* is exceeded.

- If the axis goes back after *cam_end* was reached, the output *bOutput* will be reset to FALSE immediately.

Legend:

cam_start : start of a cam

cam_end : end of a cam

▲ / ▼ : edge-controlled

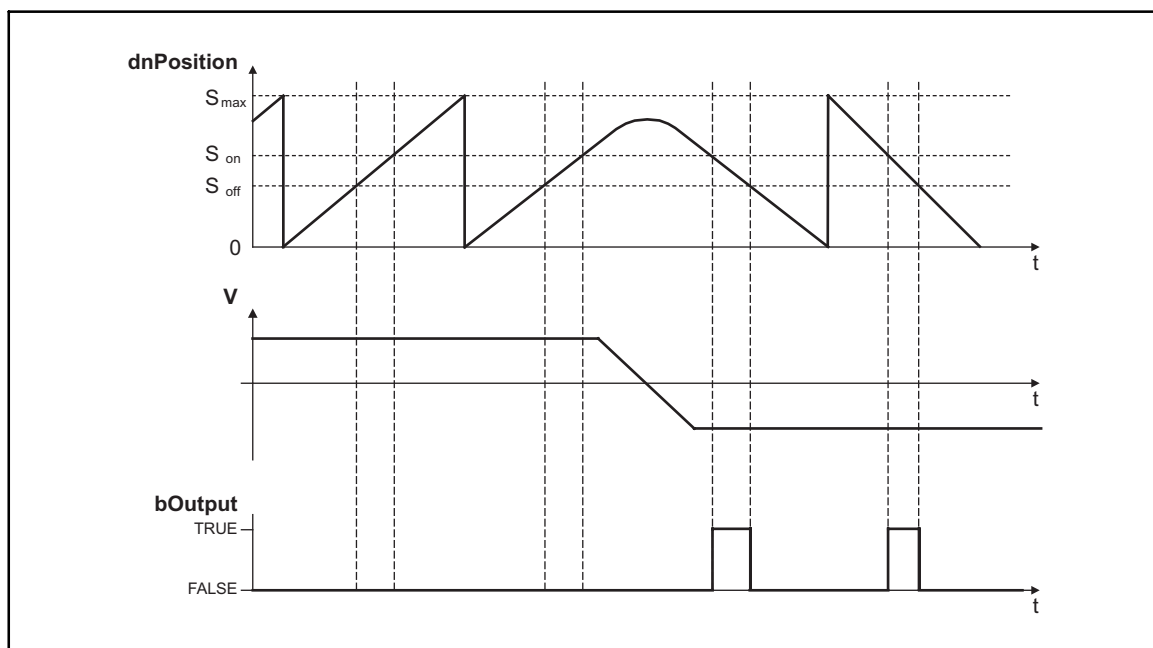
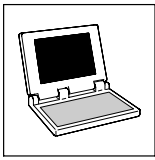


Fig. 2-7

Travel cams in negative effective direction



Function library LenzeCamControl.lib

Function blocks

L_CamContactor - Cam group

Travel cams with bipolar effective direction



Within the home values *cam_start* and *cam_end* the output *bOutput* is set to TRUE.

- The direction of the signal is not important for this mode.

Legend:

cam_start : start of a cam

cam_end : end of a cam

▲ / ▼ : edge-controlled

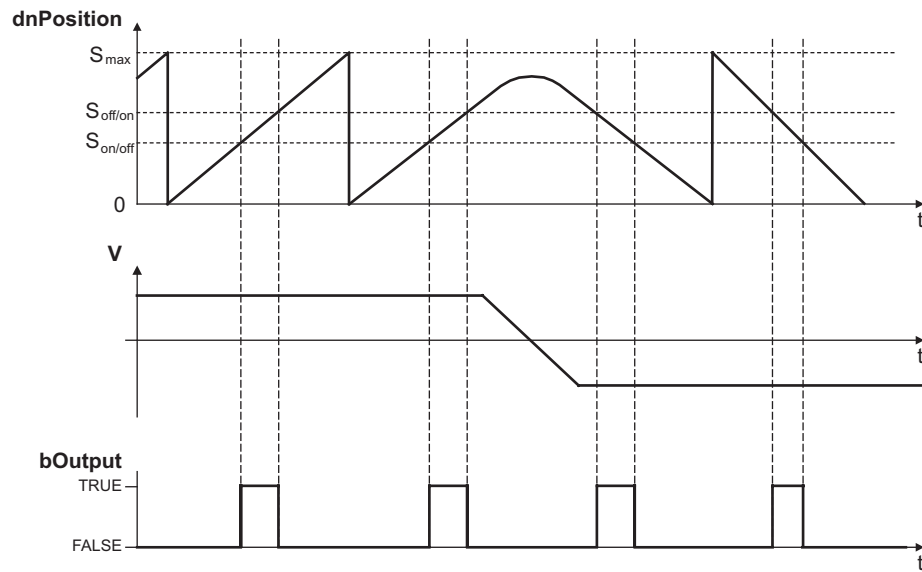
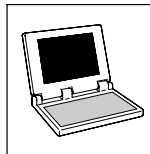


Fig. 2-8

Cam in bipolar effective direction

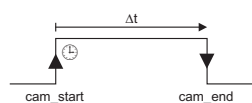


Note!

When using time cams the on-positions of successive cams may overlap if the operating time of the time cam is longer than the interval to the next on-position to the next time cam. (retrigger function)

If the operating time of a time cam is longer than the total clock-pulse time, the signal of the time cam will remain permanently on.

Travel-time cams in positive effective direction



Legend:

When reaching the reference value *cam_start* and changing in positive direction of the position input *dnPosition_p*, the output *bOutput* is set to TRUE and reset to FALSE after the time (in milliseconds) set in *cam_end* has elapsed.

- Once the cam has been set, a change of direction of rotation will not influence the cam signal.
- The set time cam will only be reset after the set time.

cam_start : start of a cam

cam_end : end of a cam

▲ / ▼ : edge-controlled

⌚ : time-controlled (Δt = running time in [ms])

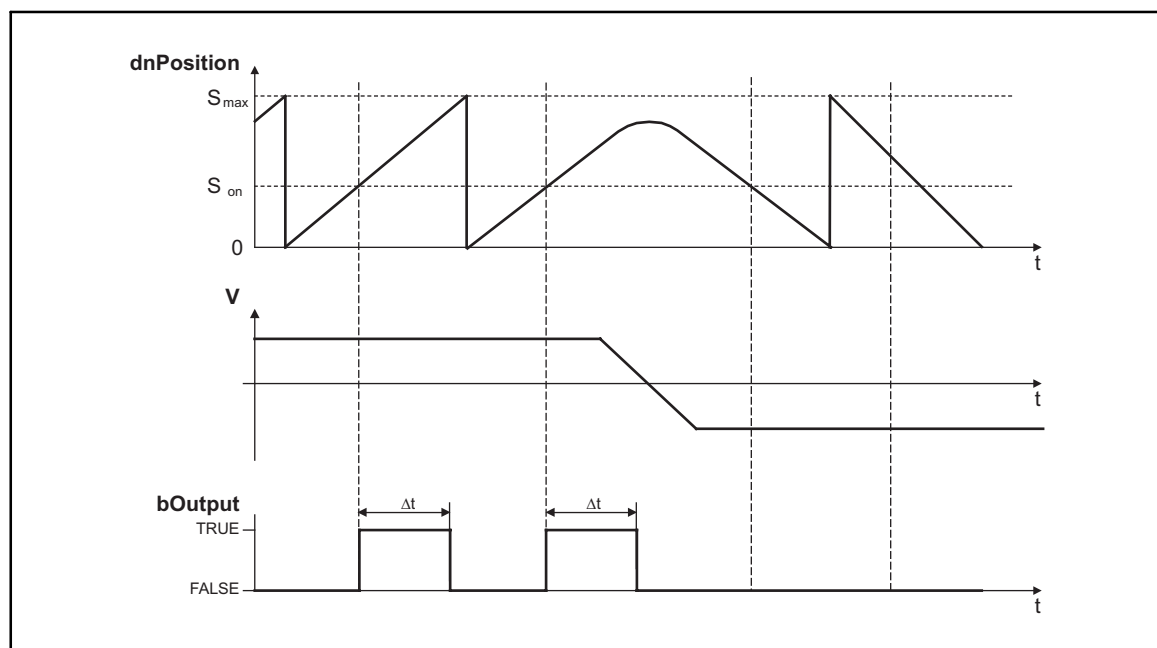
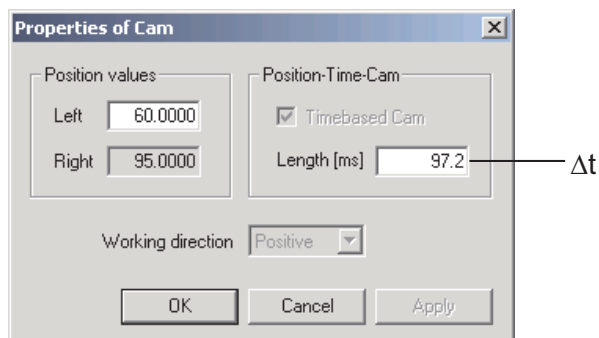
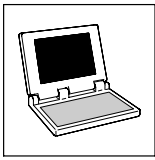


Fig. 2-9

Travel-time cams in positive effective direction



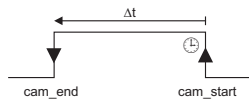


Function library LenzeCamControl.lib

Function blocks

L_CamContactor - Cam group

Travel-time cams in negative effective direction



When reaching the reference value *cam_start* and changing in positive direction of the position input *dnPosition_p* of the signal, the output *bOutput* is set to TRUE and reset to FALSE after the time (in milliseconds) set in *cam_end* has elapsed.

- Once the cam has been set, a change of direction of rotation will not influence the output signal.
- The set time cam will only be reset after the set time.

Legend:

cam_start : start of a cam

cam_end : end of a cam

▲ / ▼ : edge-controlled

⌚ : time-controlled (Δt = running time in [ms])

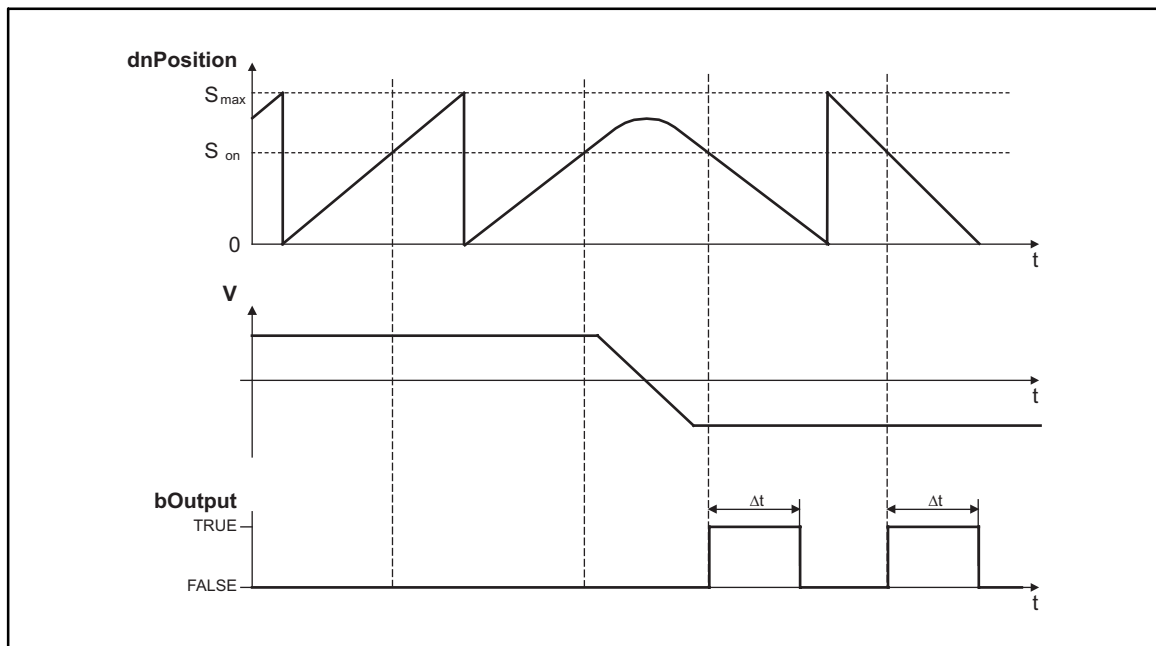
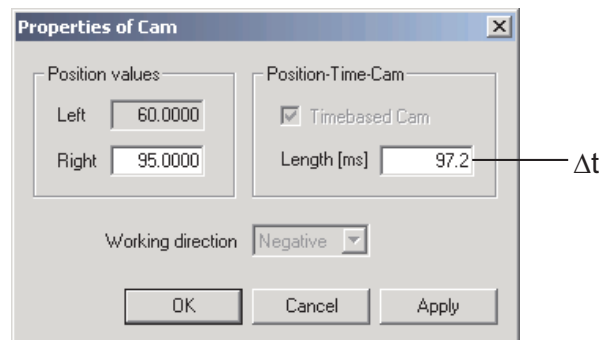
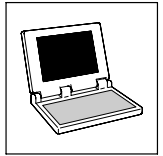


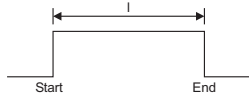
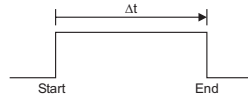
Fig. 2-10

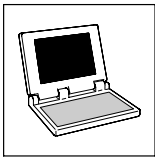
Travel-time cams in negative effective direction





Comparison travel cam/time cam

	Travel cam	Time cam
	 $l = \text{cam length } (\text{cam_end} - \text{cam_start})$	 $\Delta t = \text{running time } (\text{cam_end}) \text{ in [ms]}$
Effective direction	Two effective directions are possible: - Positive: The cam is switched on at the cam start if the axis runs towards the increasing value of position input $dnPosition_p$. - Negative: The cam is switched on at the cam end if the axis runs towards the decreasing value of position input $dnPosition_p$. - It is also possible to use both effective directions.	Two effective directions are possible: - Positive (CW): The cam is switched on at the cam start if the axis runs towards the increasing value of position input $dnPosition_p$. - Negative (CCW): The cam is switched on at the cam start if the axis runs towards the decreasing value of position input $dnPosition_p$.
Switching on	The cam switches on: - At the cam start if the direction of the axis is positive and the effective direction is set to positive. - At the cam end if the direction of the axis is negative and the effective direction is set to negative. - If the value of position input $dnPosition_p$ is within the cam range.	The cam switches on: - At the cam start if the direction of the axis is positive and the effective direction is set to positive. - At the cam start if the direction of the axis is negative and the effective direction is set to negative. After switch on the cam is on for the full time set. - It is also on if the direction of the axis changes. - If a cam start is passed again while the cam is on, the cam switch-on time starts again.
Switching off	The cam switches off if - the value of position input $dnPosition_p$ is not within the cam range. - the effective direction is against the direction of the axis.	The cam switches off after the time set has expired.
Distance	The cam distance is defined by the cam start and the cam end.	The distance of the cam depends on the speed the axis traverses while the cam is on.
Switch-on time	The switch-on time of the cam depends on the speed the axis traverses the cam travel.	The switch-on time of the cam is defined by start and stop.
Notes	The starting value and final value of the cam range are defined by the cam editor of the CamDesigner.	The shortest possible time for the time cam is always longer or the same as the set task time.



Function library LenzeCamControl.lib

Function blocks

L_CamContactorVPos - Dynamic cam/hysteresis adjustment

2.3 L_CamContactorVPos - Dynamic cam/hysteresis adjustment

Use this FB to calculate the current position value by means of a hysteresis and the dead time.

This FB can also be used to compensate for the dead time of connected components (velocity and speed-dependent).

The cam switching points are calculated in the FB **L_CamContactor**.

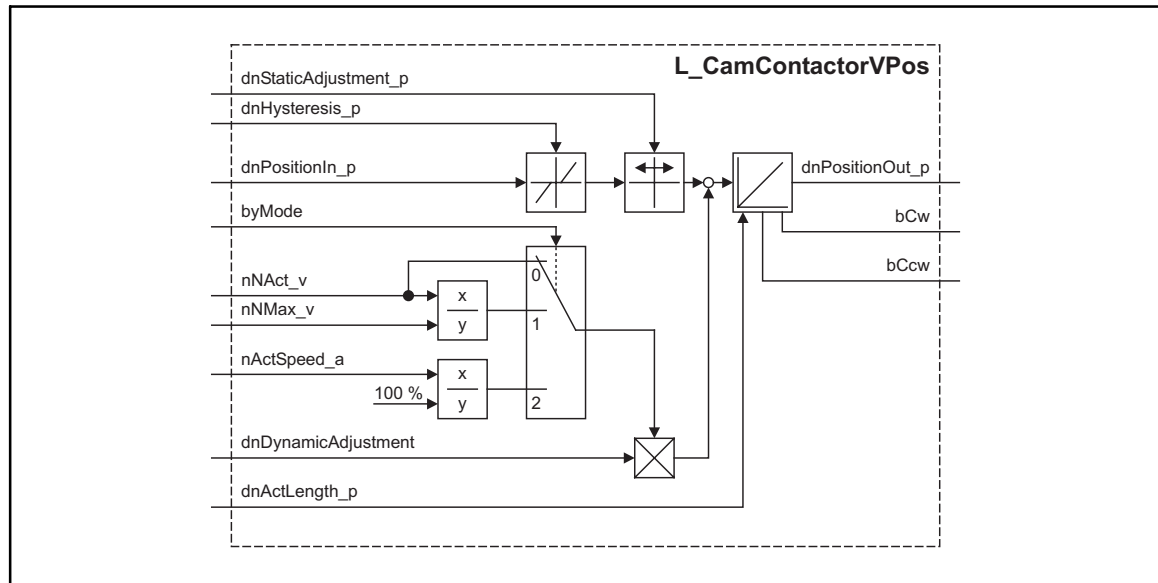
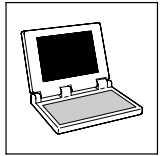


Fig. 2-11 Function block "L_CamContactorVPos"

⇒ Inputs		(Variable type: VAR_INPUT)							
Identifier	Data type	Signal type	Info/possible settings						
dnStaticAdjustment_p	Double Integer	Position	Static adjustment (compensation of paths of deceleration of connected switching elements). This value is added to <i>dnPositionIn_p</i>.						
dnHysteresis_p	Double Integer	Position	Hysteresis The input value <i>dnPositionIn_p</i> must at least change by this value to update the output value <i>dnPositionOut_p</i>.						
dnPositionIn_p	Double Integer	Position	Actual position						
byMode	Byte	-	Selection of the mode to select dynamic adjustment <table><tr><td>0</td><td>Time selected in [ms] at <i>dnDynamicAdjustment</i>. - Speed input <i>nNAct_v</i>. </td></tr><tr><td>1</td><td>Position selected in [inc] at <i>dnDynamicAdjustment</i>. - Only speed inputs <i>nNAct_v</i> and <i>nNMax_v</i>. </td></tr><tr><td>2</td><td>Position selected in [inc] at <i>dnDynamicAdjustment</i>. - Only speed input <i>nActSpeed_a</i>. </td></tr></table>	0	Time selected in [ms] at <i>dnDynamicAdjustment</i> . Speed input <i>nNAct_v</i>.	1	Position selected in [inc] at <i>dnDynamicAdjustment</i> . Only speed inputs <i>nNAct_v</i> and <i>nNMax_v</i>.	2	Position selected in [inc] at <i>dnDynamicAdjustment</i> . Only speed input <i>nActSpeed_a</i>.
0	Time selected in [ms] at <i>dnDynamicAdjustment</i> . Speed input <i>nNAct_v</i>.								
1	Position selected in [inc] at <i>dnDynamicAdjustment</i> . Only speed inputs <i>nNAct_v</i> and <i>nNMax_v</i>.								
2	Position selected in [inc] at <i>dnDynamicAdjustment</i> . Only speed input <i>nActSpeed_a</i>.								
nNAct_v	Integer	Velocity	Actual speed of the X or Y axis in [inc/ms] - Only relevant with <i>byMode</i> = 0 or 1 - This input can be used to apply the speed signal <i>nXNOut_v</i> of the FB L_CamData.						
nNMax_v	Integer	Velocity	Rated clock pulse in [inc/ms] - Only relevant with <i>byMode</i> = 1 - This input can be used to apply the signal <i>g_CamAxis.wXCycles</i>.						
nActSpeed_a	Integer	Analog	Actual speed of the X or Y axis 16384 ≡ 100 % - Only relevant with <i>byMode</i> = 2 - The signal <i>MCTRL_nNAct_a</i> of the SB MCTRL can be assigned to this input.						



⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
dnDynamicAdjustment	Double Integer	Position	Dynamic adjustment of the shift values of the cams connected - If <i>byMode</i>= 0: - Selection of the adjustment in [ms]. - If <i>byMode</i>= 1 or 2: - Selection of the shift in [inc] at maximum speed Negative shift values can also be defined for all modes. In this case, it is shifted in the other direction (lagging adjustment).
dnActLength_p	Double Integer	Position	Number of increments needed for one revolution by a rotary axis. - The output value is always lower than this value. - If you use a direct axis, enter "0" here.

Outputs ⇨ (Variable type: VAR_OUTPUT)					
Identifier	Data type	Signal type	Value/meaning		
dnPositionOut_p	Double Integer	Position	Evaluated position value (hysteresis and dead time calculation) Connect this output with the input <i>dnPositionIn_p</i> of the FB L_CamContactor.		
bCw	BOOL	-	Selection of direction of rotation Connect this output with the input of the same name <i>bCw</i> of the FB L_CamContactor. <table><tr><td>TRUE</td><td>positive (in CW direction)</td></tr></table>	TRUE	positive (in CW direction)
TRUE	positive (in CW direction)				
bCcw	BOOL	-	Selection of direction of rotation Connect this output with the input of the same name <i>bCcw</i> of the FB L_CamContactor. <table><tr><td>TRUE</td><td>negative (in CCW direction)</td></tr></table>	TRUE	negative (in CCW direction)
TRUE	negative (in CCW direction)				

Detection of direction of rotation

The cam switching points are calculated in the FB **L_CamContactor**. Since this FB does not detect the direction of rotation, the FB **L_CamContactorVPos** must be connected to the FB **L_CamContactor**.

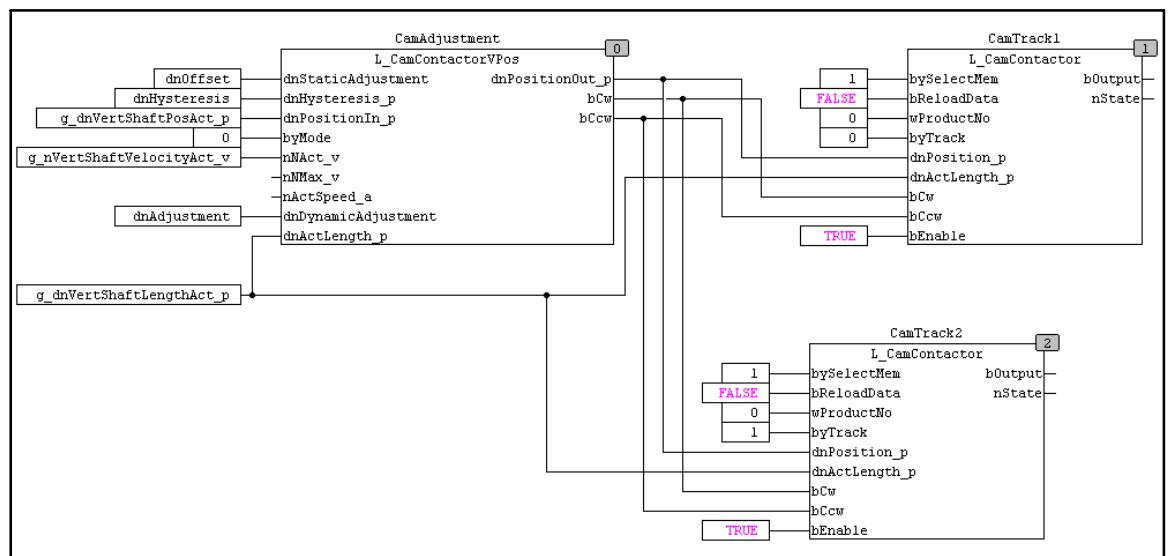
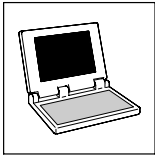


Fig. 2-12

Connection example



Function library LenzeCamControl.lib

Function blocks

L_CamContactorVPos - Dynamic cam/hysteresis adjustment



Note!

If you use several cam groups, the outputs *bCw* and *bCcw* must be connected with the corresponding inputs of all related FBs *L_CamContactor* irrespective whether the cam groups are dynamic or not.

Limitation of the actual position value for a rotary axis

Enter the number of increments per revolution as limit value at the input *dnActLength_p* to ensure that the actual position values do not exceed the limits when you use a rotary axis.

- The length of the X axis can, for instance, be linked with the output *dnActXLength_p* of the FB *L_CamData*.
- The length of the Y axis can, for instance, be linked with the output *dnYEndPos_p* of the FB *L_CamData*.

With a direct axis (e.g. Y axis with limited traversing range), a "0" must be applied to the input *dnActLength_p*.

Dynamic adjustment

The dynamic adjustment (*dnDynamicAdjustment*) compensates for delay times of connected switching elements.

Three modes are available for selecting the adjustment:

byMode	Info	
0	Time selection • Adjustment value in [ms] via <i>dnDynamicAdjustment</i>. • Speed via <i>nNAct_v</i>.	Simple adjustment
1	Position selection 1 • Adjustment value at maximum speed (<i>nNMax_v</i>) via <i>dnDynamicAdjustment</i>. • Speed via <i>nNAct_v</i>. • Maximum speed via <i>nNMax_v</i>.	Selection of maximum speed
2	Position selection 2 • Adjustment value at 100 % via <i>dnDynamicAdjustment</i>. • Speed via <i>nNAct_v</i> (16384 $\equiv$ 100 %).	Adjustment with higher resolution (e.g. suitable for connection to <i>MCTRL_nNAct_a</i>)

- If a positive value is applied to *dnDynamicAdjustment* the position at the output *dnPositionOut_p* leads the position at the input *dnPositionIn_p* and the cam switches earlier.
- If you select a negative value for *dnDynamicAdjustment* the cam can be switched later.

The adjustment affects both edges of a cam:

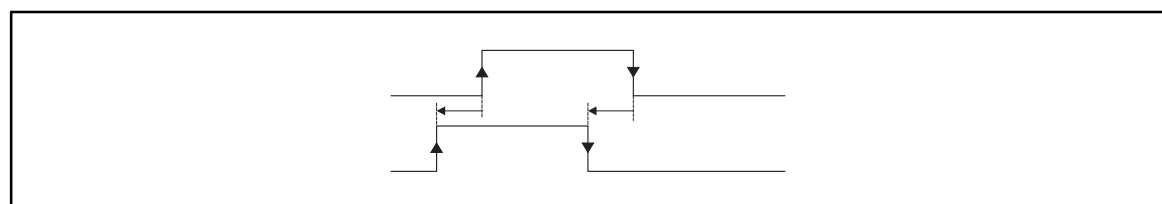
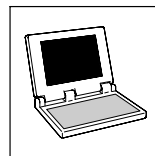


Fig. 2-13

Adjustment of both cam edges



Example for bMode = 2:

A connected valve with a fixed delay value must be switched 2000 increments before the switching point at maximum speed.

- The input *nActSpeed_a* is connected to the output *MCTRL_nNAct_a* of the SB **MCTRL**.
- The setting "2000" in *dnDynamicAdjustment* results in the fact that *dnPositionOut_p* leads at *nActSpeed_a* = 16384 (≡ 100 %) *dnPositionIn_p* by 2000 increments and thus the output of the connected FB **L_CamContactor** is switched earlier.
- The dead time compensation depends on the speed (linear profile), i.e. half speed means switching 1000 increments earlier, standstill means no adjustment at all.

Static adjustment

Additional paths of deceleration can be compensated by an offset at *dnStaticAdjustment*.

Example:

A machine requires a fine adjustment of 50 increments while running.

- By setting "50" in *dnStaticAdjustment*, *dnPositionOut_p* is shifted by exactly this offset in positive direction.

Hysteresis

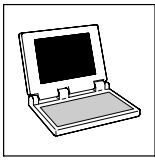
By setting a hysteresis a permanent switching, — also called "toggling" — of cams can be prevented which may result from changes in actual positions due to a mechanical disturbance of the axis.

- Cams switch on and off if the mechanical disturbance is so high that the actual position value fluctuates around a cam transition or within an active cam with just one effective direction.
- The hysteresis is set under *dnHysteresis_p*. It depends on the actual value and applies to all cams. It is also effective if the current axis position does not have a cam.

Rules for the hysteresis:

The following rules apply to a hysteresis when function blocks are interconnected.

- The display of the actual value remains constant within the hysteresis.
- The direction is not changed within the hysteresis.
- Cams are not changed within the hysteresis.
- Time cams are not switched on within a hysteresis, if a time cam is already on, it will be switched off after the time set (not after the hysteresis).
- After the hysteresis, the FB determines the current position, direction of axis movement and states of all cams.
- The hysteresis applies to all cams.



Function library *LenzeCamControl.lib*

Function blocks

L_CamControl - Setpoint conditioning

2.4 *L_CamControl* - Setpoint conditioning

This FB is used to implement the following functions:

- Integrating digital frequency signals to one position
- Loading the auxiliary integrator while maintaining the following error
- Providing an auxiliary position setpoint for separated positioning (e. g. manual mode/referencing)
- Synchronising the Y axis via touch probe/encoder zero pulse
- Speed and torque precontrol

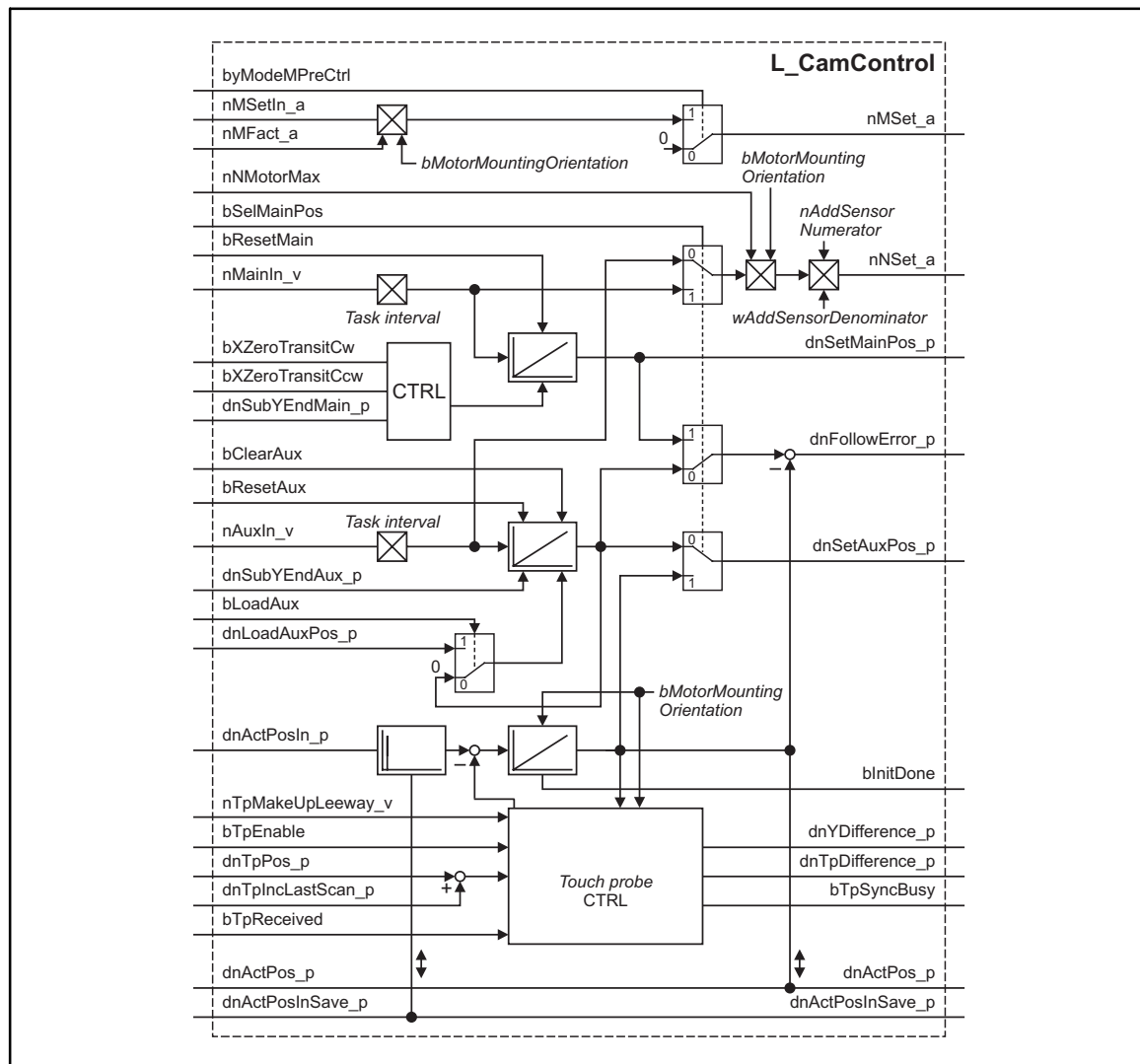
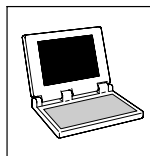
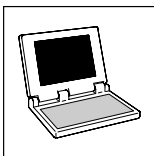


Fig. 2-14 "L_CamControl" function block

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
byModeMPreCtrl	Byte	Analog	Mode of the torque precontrol
			0Off
			1Via a respective acceleration profile on <i>nMSetIn_a</i>
nMSetIn_a	Integer	Analog	Torque precontrol via cam (L_CamData FB, standardised) • Only relevant for <i>byModeMPreCtrl</i> = 1



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
nMFact_a	Integer	Analog	Adjustment of the torque precontrol to the moment of inertia • 16384 $\equiv$ 100 %
nNMotorMax	Integer	-	Maximum speed in [rpm] • Reference value for speed precontrol. • Connect this input to the output <i>MCTRL_nNMaxC11</i> of the MCTRL SB.
bSelMainPos	BOOL	Binary	Selection of the position setpoint
			FALSE Auxiliary position setpoint
bResetMain	BOOL	Binary	Setting the integrator for the main position setpoint (main) to actual value. (see also <i>bResetMainMode</i>)
			TRUE Set to actual value.
nMainIn_v	Integer	Velocity	Speed input for the main position setpoint
bXZeroTransitCw	BOOL	Binary	Input for status signal "Zero crossing of the x axis in positive direction (Cw)" • Connect this input to the corresponding output of the L_CamData FB.
			TRUE Zero crossing of the x axis in positive direction (Cw), main integrator is limited to value on <i>dnSubYEndMain_p</i> .
bXZeroTransitCcw	BOOL	Binary	Input for status signal "Zero crossing of the x axis in negative direction (Ccw)" • Connect this input to the corresponding output of the L_CamData FB.
			TRUE Zero crossing of the x axis in negative direction (Ccw), main integrator is limited to value on <i>dnSubYEndMain_p</i> .
dnSubYEndMain_p	Double integer	Position	Limitation of the main position setpoint to $Y_{\max}$ • Connect this input to the output <i>dnYEndPos_p</i> of the L_CamData FB. • <i>dnSubYEndMain_p</i> = "0" $\equiv$ limitation inactive.
bClearAux	BOOL	Binary	Reset auxiliary integrator • Output <i>dnSetAuxPos_p</i> = 0
			TRUE Reset
bResetAux	BOOL	Binary	Setting auxiliary integrator to actual value. • Output <i>dnSetAuxPos_p</i> = <i>dnActPos_p</i> • Following error is reset. • This input has a higher priority than <i>bClearAux</i> .
			TRUE Set to actual value
nAuxIn_v	Integer	Velocity	Speed input for auxiliary setpoint position
dnSubYEndAux_p	Double integer	Position	Limitation of the auxiliary position setpoint to $Y_{\max}$ • Connect this input to the output <i>dnYEndPos_p</i> of the L_CamData FB if the tool only is positioned within the cycle. • <i>dnSubYEndAux_p</i> = "0" $\equiv$ limitation inactive.
bLoadAux	BOOL	Binary	Loading auxiliary integrator with value on <i>dnLoadAuxPos_p</i> • The actual value integrator is also set while the following error is maintained.
			FALSE $\nrightarrow$ TRUE Load auxiliary integrator
dnLoadAuxPos_p	Double integer	Position	Value by means of which the auxiliary integrator is loaded by setting <i>bLoadAux</i> to TRUE.
dnActPosIn_p	Double integer	Position	Actual position of the motor control • The value is differentiated internally and is given to the actual value integrator as a position change per cycle. • Connect this input to the output <i>dnPos_p</i> of the MCTRL SB.
nTpMakeUp Leeway_v	Integer	Velocity	Speed by means of which the touch probe correction is carried out. • In case of a selection of "0", the Y difference is maintained. • For a smooth adjustment, the L_CamPosCtrlLin FB is to be used (see connection example).



Function library LenzeCamControl.lib

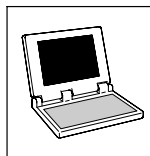
Function blocks

L_CamControl - Setpoint conditioning

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
bTpEnable	BOOL	Binary	Evaluation of the touch probe signals/zero pulse signals relating to the main position setpoint, activating/deactivating. For <i>bSelMainPos</i> = TRUE only.	
			TRUE	The evaluation of the touch probe signals/zero pulse signals is activated.
			FALSE	The evaluation of the touch probe signals/zero pulse signals is deactivated.
dnTpPos_p	Double integer	Position	Position of the touch probe encoder	
dnTpIncLastScan_p	Double integer	Position	Increments since touch probe signal Connect this input to the output <i>MCTRL_dnActIncLastScan</i> of the MCTRL SB.	
bTpReceived	BOOL	Binary	Control signal for touch probe correction, manually or via interruptable input Connect this input to the output <i>MCTRL_bActTpReceived</i> of the MCTRL SB.	
			TRUE	Receive touch probe signal.

Outputs ⇨ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Value/meaning	
nMSet_a	Integer	Analog	Torque precontrol Connect this output to the input <i>MCTRL_nMAdd_a</i> of the MCTRL SB.	
nNSet_a	Integer	Analog	Speed precontrol Connect this output to the input <i>MCTRL_nNSet_a</i> of the MCTRL SB.	
dnSetMainPos_p	Double integer	Position	Setpoint of the motion profile Internal limitation to <i>dnYEndPos_p</i>, position is maintained if auxiliary integrator is used.	
dnFollowError_p	Double integer	Position	Following error Connect this output to the input <i>MCTRL_dnSetPos_p</i> of the MCTRL SB (position controller).	
dnSetAuxPos_p	Double integer	Position	Auxiliary setpoint or tool position If the main integrator is used (<i>bSelMainPos</i> = TRUE), this value corresponds to the actual position.	
blnitDone	BOOL	Binary	Status signal "Initialisation completed"	
			TRUE	Initialisation is completed, FB is ready for operation.
			FALSE	If the blnitDone output remains set to FALSE, this may be due to: - Incomplete/missing profile data in the target system's main memory; repeat the transfer of the LC9 file to the target system. - PLC stop: Check code C2108.
dnYDifference_p	Double integer	Position	Correction value after a touch probe signal	
dnTpDifference_p	Double integer	Position	Difference between two touch probe events	
bTpSyncBusy	BOOL	Binary	Status signal "Synchronisation is active"	
			FALSE	At present setpoints are not being corrected.
			TRUE	Synchronisation is active, Y difference is being compensated.

⇒ Inputs/outputs (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Information/possible settings
dnActPos_p	Double integer	Position	Output of the actual value integrator, limited to Y_{max} . - Relative actual position, relating to the cycle of the motion profile. - Save this value in a manner which is safe against mains failure in a variable of the type PERSISTENT.
dnActPosInSave_p	Double integer	Position	Absolute actual value - Irrespective of the motion profile cycle. - Save this value in a manner which is safe against mains failure in a variable of the type PERSISTENT.



Parameter (Variable type: VAR_CONSTANT_RETAIN)				
Identifiers Code*	Data type	Signal type	Information	
			Possible settings	Presetting
bMotorMounting Orientation C1313	BOOL	-	Motor mounting position	
			• From the view of a machine moving in positive direction	
			FALSE Clockwise (Cw)	FALSE
bResetMainMode C1314	BOOL	-	Mode for the behaviour of the <i>bResetMain</i>	
			FALSE Setting main integrator (main) to actual value.	FALSE
			TRUE Setting main integrator (main) to zero.	
nAddSensor Numerator C1348	Integer	-	Factor for additional position feedback adaptation (numerator)	
			-32767 {1} 32767	0
wAddSensor Denominator C1349	Word	-	Factor for additional position feedback adaptation (denominator)	
			1 {1} 65535	1

* Code of first instance

Integration of digital frequency signals to one position

The main position setpoint (main) as reference value for controlling the cam disc and the auxiliary position setpoint (aux) for separated positioning tasks or the like are provided.



STOP!

A change-over to the main position setpoint by means of *bSelMainPos* = TRUE may only be carried out if the main position setpoint and the auxiliary position setpoint are equal (*dnSetAuxPos_p* = *dnSetMainPos_p*), as otherwise an unintentional compensation at max. torque may be effected.

Main position setpoint

The integrator for the main position setpoint is selected via *bSelMainPos* = TRUE.

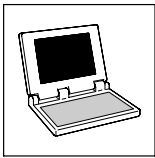
- The main position setpoint is maintained if a change-over via *bSelMainPos* = FALSE to the auxiliary position setpoint is carried out. Like this, the drive can resume its original profile position later on.
- Via *bResetMain* = TRUE, according to the position of the internal variable *bResetMainMode*, the integrator can be set as follows:
 - *bResetMainMode* = FALSE:
The main position setpoint is set to the current actual value (*L_CamControl.dnActPos_p*).
 - *bResetMainMode* = TRUE:
The main position setpoint is set to zero.



STOP!

If the generation of following errors is effected between the main integrator and actual value integrator (*bSelMainPos* = TRUE), this may result in an unintentional compensation at max. torque.

- A limitation of the main position setpoint to the profile pulse is only effected if the inputs *bXZeroTransmitCw*, *bXZeroTransmitCcw* and *dnSubYEndMain_p* are connected to the respective outputs of the **L_CamData** FB; at the time of the zero crossing, $Y_{\max}$ then is subtracted from or added to the main position setpoint once.



Function library *LenzeCamControl.lib*

Function blocks

L_CamControl - Setpoint conditioning

Auxiliary position setpoint for separated positioning

The integrator for the auxiliary position setpoint is selected via *bSelMainPos* = FALSE.

- If the integrator is not selected, the actual value of the drive (*dnActPos_p*) is always applied to its output *dnSetAuxPos_p*.
- The integrator can be set to the actual value via *bResetAux* = TRUE.



Tip!

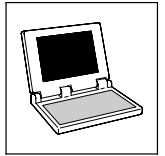
For controller inhibit and quick stop (QSP), always set *bResetAux* to TRUE, in order to rule out rotary motions.

- The integrator can be set to zero via *bClearAux* = TRUE.
- The limitation of the auxiliary position setpoint occurs continuously if a non-zero value is preselected on the input *dnSubYEndAux_p*.

Loading the auxiliary integrator

By setting *bLoadAux* to TRUE, the auxiliary can be loaded with a value which is applied to the *dnLoadAuxPos_p* position input (e. g. for the "Setting dynamic reference" function).

- Loading the auxiliary integrator can only be carried out if the auxiliary integrator is selected via *bSelMainPos* = FALSE.
- The following error here is maintained, the actual value integrator is simultaneously loaded via the following error (*dnActPos_p* = *dnLoadAuxPos_p* - *dnFollowError_p*).



Synchronisation of the Y axis via touch probe (TP)



Note!

In the following text, the synchronisation by means of touch probe is described exemplarily. Detailed information on the touch probe configuration (e. g. activation, edge selection and delay time) can be found in the Manual of the respective PLC device!

For the **9300 servo PLC**, a synchronisation can be effected via the zero pulse of a connected encoder or resolver, or via the digital inputs.

Apart from this, the touch probe correction can also be started manually via the input *bTpReceived*.

- Here no interrupt is actuated. Therefore no value of the **MCTRL** SB is applied on the input *dnTpIncLastScan_p*, which would be required for an exact synchronisation.
- Thus, the synchronisation accuracy depends on the interval of the task in which the **L_CamControl** FB is integrated.

If *bTpReceived* is set to TRUE (by the **MCTRL** SB or manually), and if the touch probe synchronisation is activated via *bTpEnable* = TRUE, the difference between the setpoint and actual value is calculated and output on *dnYDifference_p*.

- If incoming touch probe signals are masked out, by setting *bTpEnable* to FALSE, the touch probe synchronisation can be deactivated in the respective position, e. g. by means of the **L_CamContactor** cam group.
- For compensating the difference, the velocity for catching up can be specified via *nTpMakeUpLeeway_v*. If no velocity is specified, the calculated difference is not reduced, i. e. the synchronisation remains inactive.
- For a smooth compensation of the difference, add the **L_CamPosCtrlLin** FB on *nTpMakeUpLeeway_v* as a ramp generator:

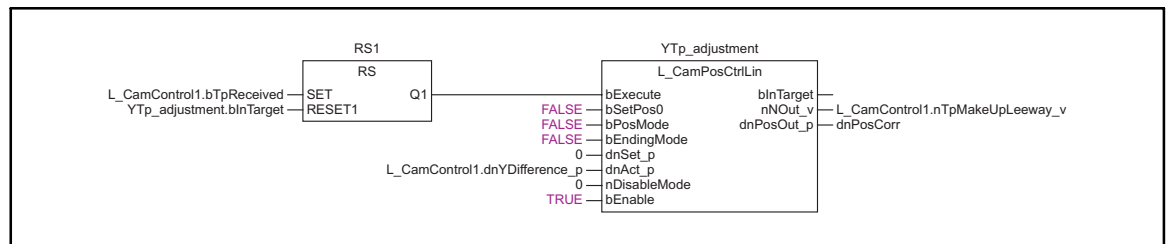
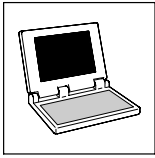


Fig. 2-15

"L_CamPosCtrlLin" function block as ramp generator



Function library LenzeCamControl.lib

Function blocks

L_CamControl - Setpoint conditioning

Separate feedback for speed and position control

For slipping material, an additional sensor can be used to detect the position of the tool or material.

- Select the corresponding feedback system via C0490:

Code	LCD	Possible settings		Information
		Lenze	Selection	
[C0490]	Feedback pos	0		Feedback system for position controller
		0	Resolver on X7	
		1	Encoder TTL on X8	
		2	Sin/Cos encoder on X8	
		3	Absolutevalue encoder ST on X8	
		4	Absolute value encoder MT on X8	

Via the parameters *nAddSensorNumerator* and *wAddSensorDenominator*, you set the factor for the adjustment of the additional position feedback resulting from the given machine constants:

$$\frac{nAddSensorNumerator}{wAddSensorDenominator} = \frac{vk2}{vk1} \cdot \frac{i1}{i2}$$

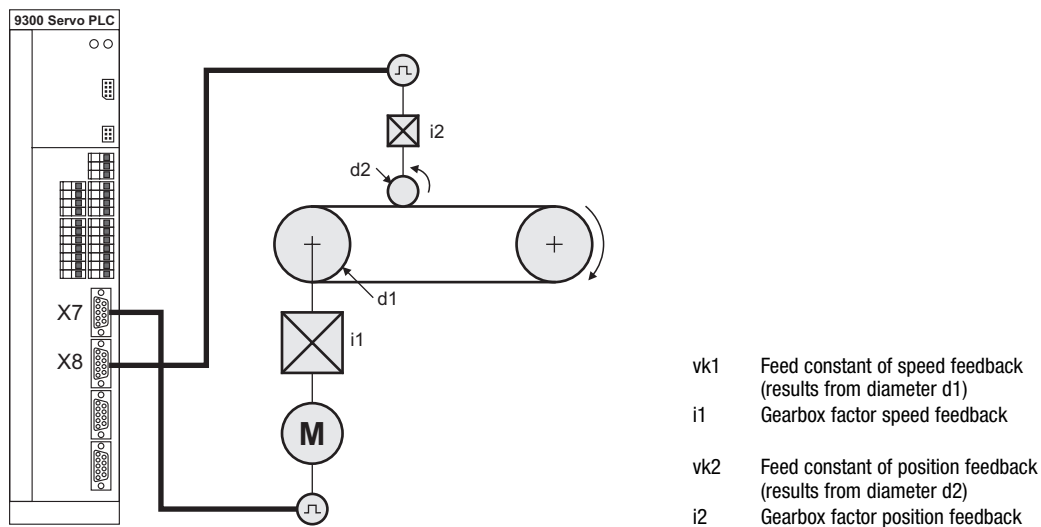


Fig. 2-16

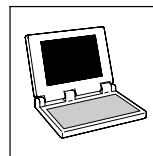
Feedback system for speed and position control

- The set factor only influences the *nNSet_a* speed precontrol value.
- An adjustment only is effected if *nAddSensorNumerator* <> is "0".



Note!

When the **CamDesigner** or **CamLoader** (as import) is used for creating the motion profile, only the machine constants of the position sensor are used for the separate feedback and are entered as machine constants (*vk2* and *i2*) in the *Project settings* dialog box for the slave!



Speed precontrol

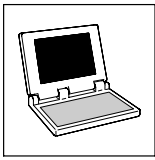
The speed control via the output *nNSet_a* is the velocity of the selected integrator standardised to the *nNMotorMax* reference value.

- Connect the precontrol output *nNSet_a* directly to the input *MCTRL_nNSet_a* of the **MCTRL** SB.

Torque precontrol

The type of the torque precontrol via the output *nMSet_a* is defined via *byModeMPreCtrl*.

- Connect the precontrol output *nMSet_a* directly to the input *MCTRL_nMAdd_a* of the **MCTRL** SB.
- If *byModeMPreCtrl* is set to "1", a precontrol can be effected via an acceleration profile of the **L_CamData** FB.
- Via *nMFact_a*, a stretching/compression or the adjustment to the moment of inertia of the tool to be moved is carried out.



Function library *LenzeCamControl.lib*

Function blocks

L_CamData - Profile management

2.5 L_CamData - Profile management

Use this FB to manage up to 48 different motion profiles (cams) the data of which is stored according to a certain internal structure (*.lc9 file).

- The motion profiles and the resulting *.lc9 file can be easily generated with the Lenze **CamDesigner**.
- In DDS, you can then use the **Cam Support** to link the *.lc9 file to the project and download it together with the PLC program to the PLC memory. The **Cam Support** is an object in the **Resources** register card in the *Object Organizer*.

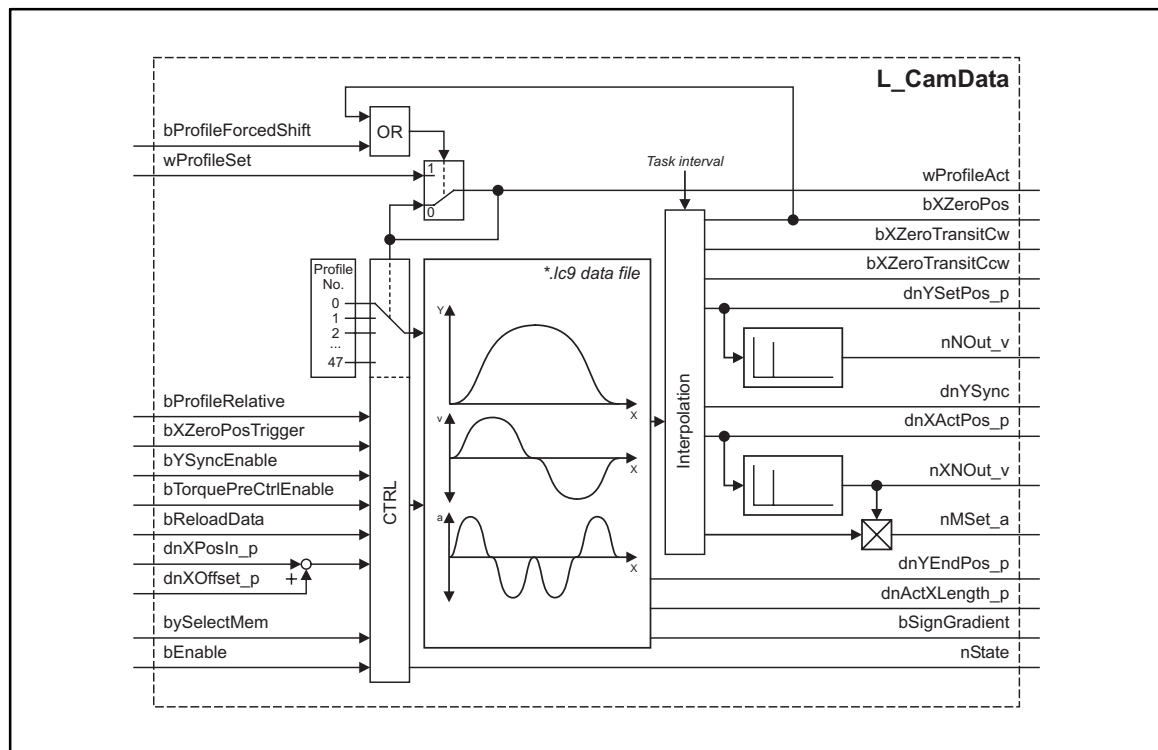
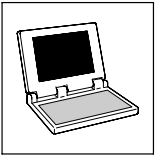
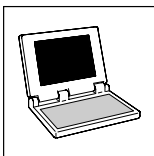


Fig. 2-17

"L_CamData" function block



⇒ Inputs (Variable type: VAR_INPUT)							
Identifiers	Data type	Signal type	Information/possible settings				
bProfileForcedShift	Bool	Binary	Forced changeover of motion profiles within an x clock pulse The Y axis of the new profile is initialised, the x axis remains the same. <table><tr><td>FALSE ↗ TRUE</td><td>Forced changeover</td></tr></table>	FALSE ↗ TRUE	Forced changeover		
FALSE ↗ TRUE	Forced changeover						
wProfileSet	Word	Number	Selection of the motion profile - Selection range 0 ... 47 - The selection range is limited to the number of motion profiles defined in the *.lc9 file. - A change to the new motion profile is only possible during zero crossing of the clock pulse or as a forced changeover.				
bProfileRelative	Bool	Binary	Type or evaluation of the motion profile <table><tr><td>FALSE</td><td>Evaluate motion profile as forward/backward motion (absolute positioning)</td></tr><tr><td>TRUE</td><td>Evaluate motion profile as feed motion (relative positioning)</td></tr></table>	FALSE	Evaluate motion profile as forward/backward motion (absolute positioning)	TRUE	Evaluate motion profile as feed motion (relative positioning)
FALSE	Evaluate motion profile as forward/backward motion (absolute positioning)						
TRUE	Evaluate motion profile as feed motion (relative positioning)						
bXZeroPosTrigger	Bool	Binary	Profile data initialisation through upstream L_CamSectionShift FB If the L_CamSectionShift FB is used connect this input to the corresponding output of the L_CamSectionShift FB. <table><tr><td>TRUE</td><td>Profile data initialisation.</td></tr></table>	TRUE	Profile data initialisation.		
TRUE	Profile data initialisation.						
bYSyncEnable	Bool	Binary	Activation of synchronised stretching/compression via the L_CamSetYAxisVelocity FB <table><tr><td>FALSE</td><td>Synchronised stretching/compression deactivated. (A deactivated Y synchronisation can result in an increment loss.)</td></tr><tr><td>TRUE</td><td>Synchronised stretching/compression activated. (This setting is required for using the L_CamSetYAxisVelocity FB.)</td></tr></table>	FALSE	Synchronised stretching/compression deactivated. (A deactivated Y synchronisation can result in an increment loss.)	TRUE	Synchronised stretching/compression activated. (This setting is required for using the L_CamSetYAxisVelocity FB.)
FALSE	Synchronised stretching/compression deactivated. (A deactivated Y synchronisation can result in an increment loss.)						
TRUE	Synchronised stretching/compression activated. (This setting is required for using the L_CamSetYAxisVelocity FB.)						
bTorquePreCtrl Enable	Bool	Binary	Activation of the torque precontrol calculation Do not activate the calculation if you do not need the torque profile, in order to save computing time. <table><tr><td>FALSE</td><td>Calculation deactivated</td></tr><tr><td>TRUE</td><td>Calculation activated</td></tr></table>	FALSE	Calculation deactivated	TRUE	Calculation activated
FALSE	Calculation deactivated						
TRUE	Calculation activated						
bReloadData	Bool	Binary	Re-initialisation of the profile data - By this, the master value is also reset to X0. - A new profile number assigned to <i>wProfileSet</i> is accepted. - A new data structure selection via <i>bySelectMem</i> is accepted. - When the relative data model is used, re-initialisation should only be carried out when the controller is inhibited. <table><tr><td>FALSE ↗ TRUE</td><td>Re-initialisation of profile data.</td></tr><tr><td>TRUE</td><td>The internal x axis is set to "0".</td></tr></table>	FALSE ↗ TRUE	Re-initialisation of profile data.	TRUE	The internal x axis is set to "0".
FALSE ↗ TRUE	Re-initialisation of profile data.						
TRUE	The internal x axis is set to "0".						
dnXPosIn_p	Double integer	Position	Master value selection (only as position)				
dnXOffset_p	Double integer	Position	Offset for the master value (e. g. via the L_CamPosCtrlLin FB)				
bySelectMem	Byte	Number	Selection of the data structure (*.lc9 file). The selection can only be accepted with a FALSE-TRUE edge on <i>bReloadData</i>. <table><tr><td>1</td><td>RAM block 1 (foreground data field, standard)</td></tr><tr><td>2</td><td>RAM block 2 (background data field)</td></tr></table>	1	RAM block 1 (foreground data field, standard)	2	RAM block 2 (background data field)
1	RAM block 1 (foreground data field, standard)						
2	RAM block 2 (background data field)						
bEnable	Bool	Binary	Activate/Deactivate function block <table><tr><td>TRUE</td><td>FB is activated; the function block accesses the profile data in the memory immediately, processes it and makes the corresponding position and speed signals available at the outputs.</td></tr><tr><td>FALSE</td><td>FB is deactivated; all position outputs remain in their most recent position and all speed outputs are set to zero.</td></tr></table>	TRUE	FB is activated; the function block accesses the profile data in the memory immediately, processes it and makes the corresponding position and speed signals available at the outputs.	FALSE	FB is deactivated; all position outputs remain in their most recent position and all speed outputs are set to zero.
TRUE	FB is activated; the function block accesses the profile data in the memory immediately, processes it and makes the corresponding position and speed signals available at the outputs.						
FALSE	FB is deactivated; all position outputs remain in their most recent position and all speed outputs are set to zero.						

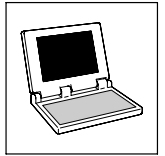


Function library *LenzeCamControl.lib*

Function blocks

L_CamData - Profile management

Outputs ⇨ (Variable type: VAR_OUTPUT)													
Identifiers	Data type	Signal type	Value/meaning										
wProfileAct	Word	Number	Number of the motion profile in process										
bXZeroPos	Bool	Binary	Status signal "Zero crossing of the master value" The signal is only available for a cycle if the x axis speed is > 0. <table><tr><td>TRUE</td><td>Zero crossing (X0 position)</td></tr></table>	TRUE	Zero crossing (X0 position)								
TRUE	Zero crossing (X0 position)												
bXZeroTransitCw	Bool	Binary	Status signal "Zero crossing of x axis in positive direction (CW)" - Connect this output to the corresponding input of the L_CamControl FB. <table><tr><td>TRUE</td><td>Zero crossing of x axis in positive direction (CW). - Not for standstill in X0. </td></tr></table>	TRUE	Zero crossing of x axis in positive direction (CW). Not for standstill in X0.								
TRUE	Zero crossing of x axis in positive direction (CW). Not for standstill in X0.												
bXZeroTransitCcw	Bool	Binary	Status signal "Zero crossing of x axis in negative direction (CCW)" Connect this output to the corresponding input of the L_CamControl FB. <table><tr><td>TRUE</td><td>Zero crossing of x axis in negative direction (CCW).</td></tr></table>	TRUE	Zero crossing of x axis in negative direction (CCW).								
TRUE	Zero crossing of x axis in negative direction (CCW).												
dnYSetPos_p	Double integer	Position	Y setpoint - Absolute profile position, not limited to <i>dnYEndPos_p</i>. - Y setpoint and speed setpoint <i>nNOut_v</i> diverge when the position changes by more than ±16384 [inc/ms]. This may for instance happen during re-initialisation via <i>bReloadData</i>.										
nNOut_v	Integer	Velocity	Speed setpoint 16384 ≙ 15000 rpm - Value is internally limited to ±16384 [inc/ms] ≙ ±15000 [rpm]										
dnYSync	Double integer	-	Synchronisation of the Y axis (via the L_CamSetYAxisVelocity FB) - With <i>bYSyncEnable</i> = TRUE, this output outputs correction increments. - For synchronisation, connect this output with the input <i>dnSyncln</i> of the L_CamSetYAxisVelocity FB.										
dnXActPos_p	Double integer	Position	Current x position										
nXNOut_v	Integer	Velocity	Digital frequency output x axis 16384 ≙ 15000 rpm - If this output is used to detect the clock frequency for the cam controller, the cams must be deactivated if the x axis is reset by setting <i>bReloadData</i> to TRUE. Note: The phase difference signal considers the incremental cycle skip from x_{max} to 0 at the end/start of a cycle, but not a possible incremental decimal fraction in the cycle. Therefore do not use this output signal if the cycle cannot be represented incrementally as an integer!										
nMSet_a	Integer	Analog	Torque precontrol - Output of the acceleration profile as function of the x axis speed - If the L_CamSetYAxisVelocity FB is used, connect this output with its input <i>nMln_a</i>, or the input <i>nMSet_a</i> of the L_CamControl FB.										
dnYEndPos_p	Double integer	Position	Clock pulse of the motion profile 65535 ≙ 1 motor revolution - Calculation according to profile: $dnYEndPos_p = Y_{end}(X_{max}) - Y_{end}(X_0)$										
dnActXLength_p	Double integer	Position	Length of the x axis (clock pulse length in increments)										
bSignGradient	Bool	Binary	Current pitch <table><tr><td>FALSE</td><td>Positive pitch</td></tr><tr><td>TRUE</td><td>Negative pitch</td></tr></table>	FALSE	Positive pitch	TRUE	Negative pitch						
FALSE	Positive pitch												
TRUE	Negative pitch												
nState	Integer	Number	Status <table><tr><td>30</td><td>Initialisation of profile data active</td></tr><tr><td>20</td><td>Warning: master value jump at <i>dnXPosln_p</i> too big. Points per cycle could not be processed. The speed output <i>nNOut_v</i> and position output <i>dnYSetPos_p</i> are asynchronous.</td></tr><tr><td>10</td><td>Warning: profile not available (<i>wProfileSet</i> > max. profile number)</td></tr><tr><td>0</td><td>OK, no error.</td></tr><tr><td>-2</td><td>Error: It was not possible to read profile data (*.lc9 file)</td></tr></table>	30	Initialisation of profile data active	20	Warning: master value jump at <i>dnXPosln_p</i> too big. Points per cycle could not be processed. The speed output <i>nNOut_v</i> and position output <i>dnYSetPos_p</i> are asynchronous.	10	Warning: profile not available (<i>wProfileSet</i> > max. profile number)	0	OK, no error.	-2	Error: It was not possible to read profile data (*.lc9 file)
30	Initialisation of profile data active												
20	Warning: master value jump at <i>dnXPosln_p</i> too big. Points per cycle could not be processed. The speed output <i>nNOut_v</i> and position output <i>dnYSetPos_p</i> are asynchronous.												
10	Warning: profile not available (<i>wProfileSet</i> > max. profile number)												
0	OK, no error.												
-2	Error: It was not possible to read profile data (*.lc9 file)												



Defining the master value (x position) for cam profiler

The master value is selected as an absolute and positive value on $dnXPosIn_p$. It can be generated by a virtual master (**L_CamVMasterPosition**) FB, or it can be directly selected by a position encoder or a bus system.

- If you want to use digital frequencies, they must be integrated into master angles using the **L_CamSetXAxisVelocity** FB.
- If you use an external master angle encoder (programmable multi turn/single turn absolute value encoder), it possibly has to be adjusted via an arithmetic block.
- Stretching and compression of the x axis is only possible via the **L_CamSetXAxisVelocity** FB. This FB also contains the mark synchronisation of the x axis (touch probe) for slipping material and the possibility to select a starting value.



Note!

If the selections at $dnXPosIn_p$ or $dnXOffset_p$ lead to a position jump bigger than half the clock pulse length ($dnActXLength_p$), the zero crossing detection is activated in the FB!

Position jumps on the x axis should therefore be avoided!

Offset for the master angle

Use the input $dnXOffset_p$ to select an offset which is directly added to the master angle (x position):

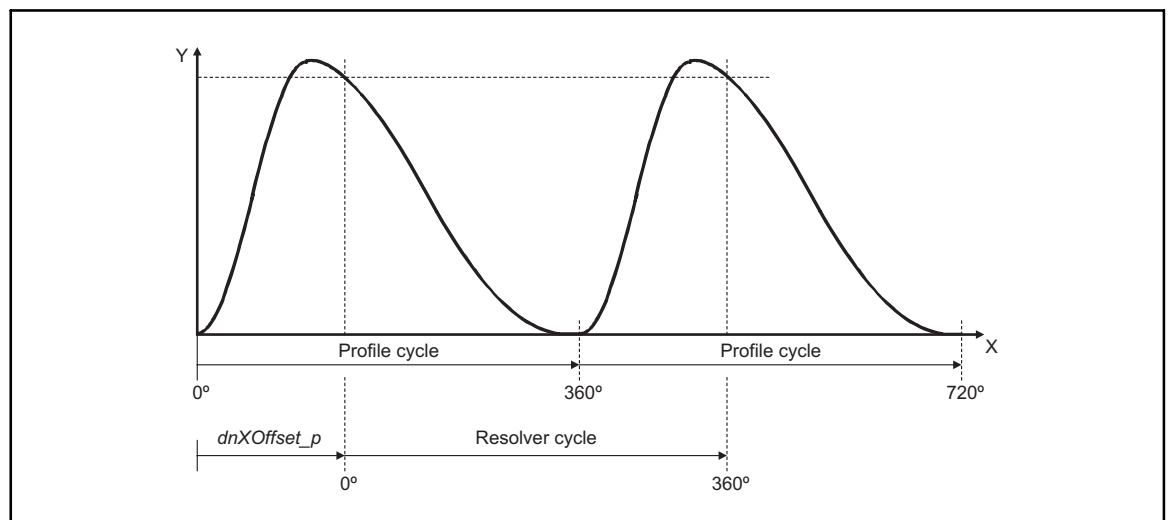


Fig. 2-18

Offset selection for the master value

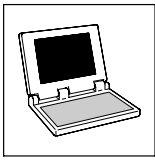
The offset is not limited to a clock pulse length and can therefore also be negative.



STOP!

The drive follows the offset $dnXOffset_p$ immediately, i.e. extreme changes result in sudden torque impulses.

- It is therefore recommended not to activate the input directly, but via a ramp generator, or to change the offset only extremely when the controller is inhibited.
- You can use the **L_CamPosCtrlLin** FB as ramp generator.



Function library *LenzeCamControl.lib*

Function blocks

L_CamData - Profile management

Selection of the motion profile

The motion profile is selected by sending the corresponding number to *wProfileSet*. A maximum of 48 different motion profiles (0 ... 47) can be addressed via this input.

- If the selected number is invalid, *nState* will be displayed.
- The number of the currently processed profile will be displayed via the output *wProfileAct*.

Changeover of the motion profile

A change to a new motion profile is only possible if one of the following conditions is met:

- The motion profile is crossing zero (output *bXZeroPos* = TRUE).
 - Undesirable torque impulses will thus be avoided.
- A FALSE/TRUE edge on *bProfileForcedShift* activates a forced changeover within the profile.
- A FALSE/TRUE edge on *bReloadData* re-initialises the profile data.



Note!

- Where cams with different X cycles are concerned, motion profile switching is only permitted with an X offset of "0".
- A forced changeover via *bProfileForcedShift* should only be carried out if the current and the new profile have the same Y travel and the same pitch at the point of changeover.
 - Forced changeovers from feed to forward/backward profiles are not possible.
 - Please note that the setting of *bProfileForcedShift* to TRUE does not deactivate the changeover of the profile in zero crossing.
 - The profiles must only be created with the absolute data model without optimisation (equidistant distribution of points).

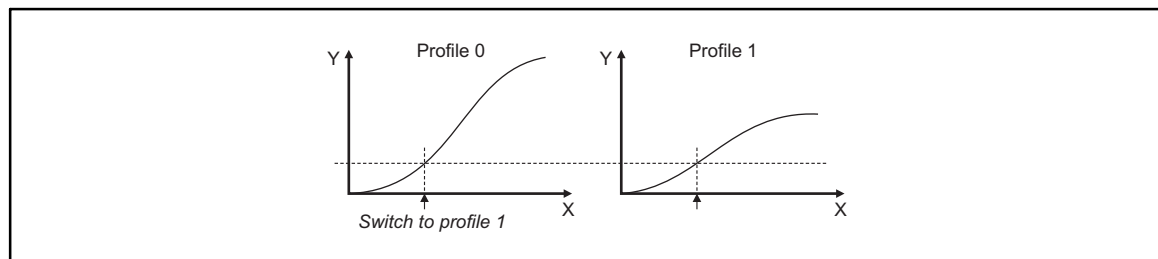


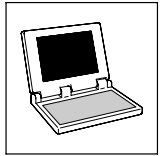
Fig. 2-19

Forced changeover from profile 0 to profile 1 with identical Y travel



Tip!

Use the **L_CamSequence** FB to select a sequence of different motion profiles!



Motion profile type

Use the input *bProfileRelative* to define how the FB is to evaluate the current motion profile:

bProfileRelative = FALSE $\Rightarrow$ **Absolute positioning**

- Use absolute positioning for forward and backward profiles with an identical starting and stop value, and if motion profiles are to be cascaded absolutely.
- Identification: limited travel path (e. g. leadscrew tools)

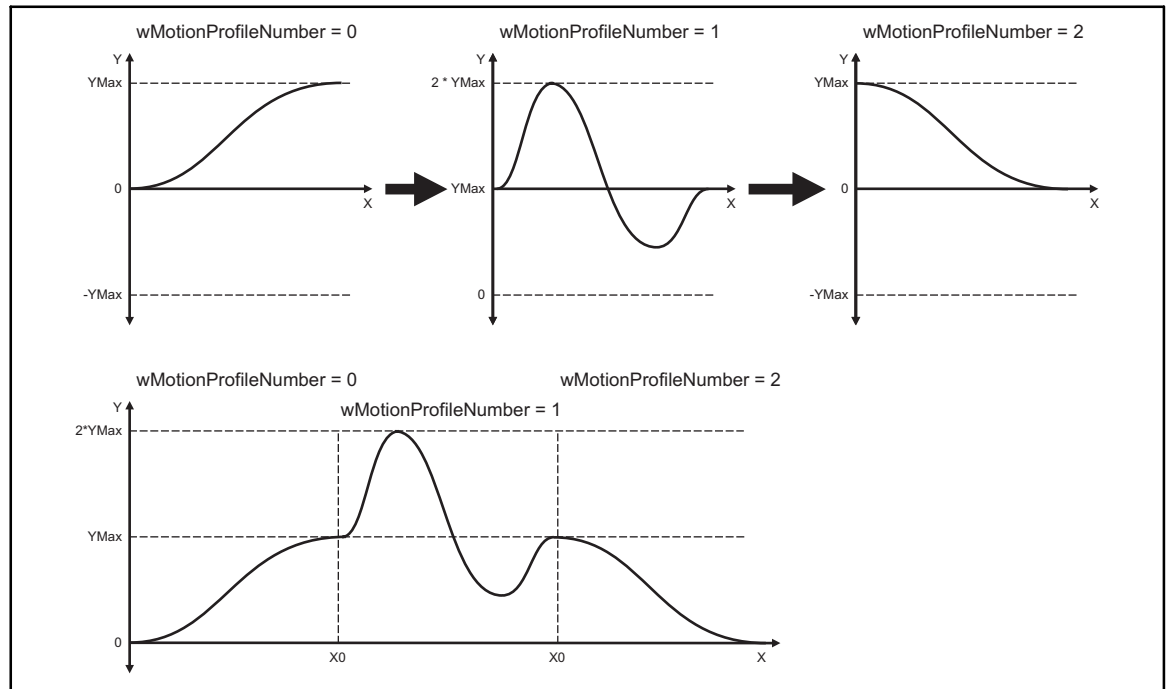


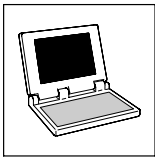
Fig. 2-20

Example: Absolute positioning with forward/backward profiles



Note!

Ensure that the starting and stop values of the motion profiles are identical when using absolute positioning and profile changeover, otherwise compensation will occur!



Function library *LenzeCamControl.lib*

Function blocks

L_CamData - Profile management

$bProfileRelative = TRUE \Rightarrow$ **Relative positioning**

- Use relative positioning for feed profiles (rotary axes) with different starting and stop value, and if motion profiles are to be cascaded relatively.
- Identification: unlimited travel path (e. g. coupler mechanisms, conveyor belts, rotary tables)

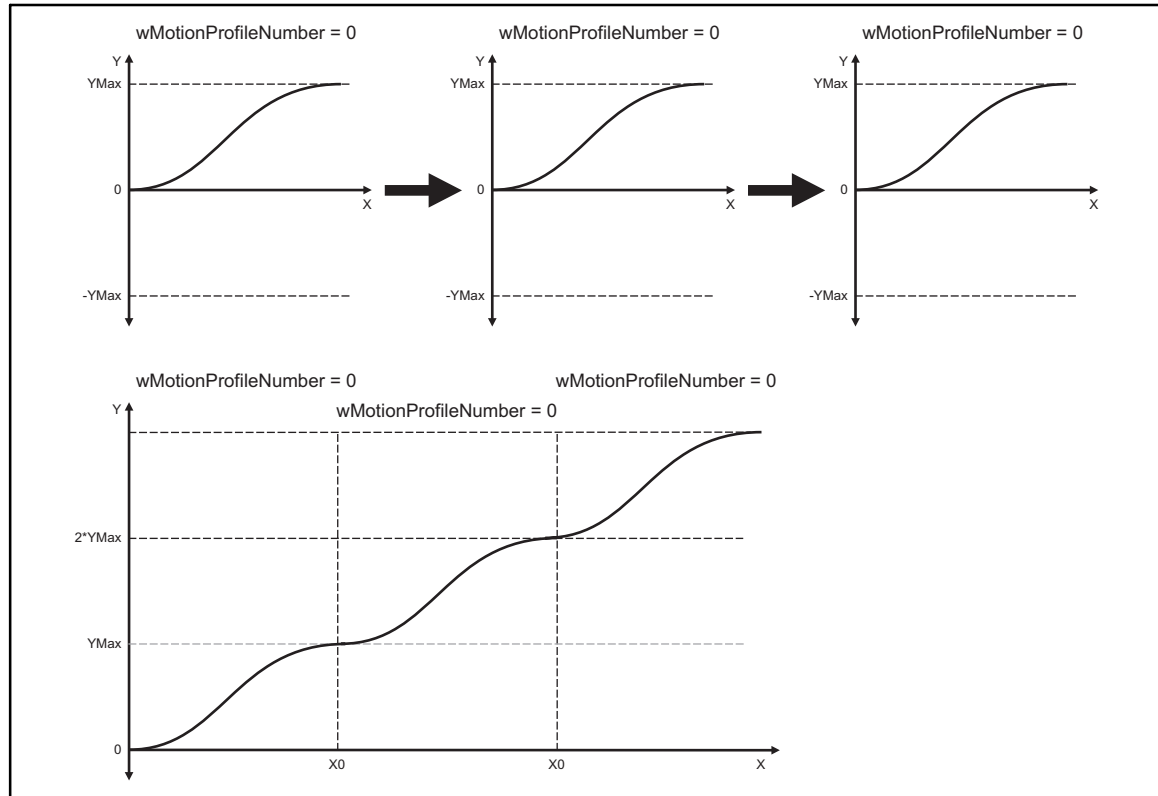


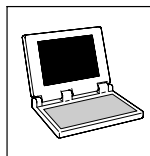
Fig. 2-21

Example: Relative positioning with a feed profile



Note!

- Ensure that the starting values of the motion profiles are identical when using relative positioning and profile changeover!
- When changing between profiles with different x axis lengths, the master angle must only rotate in positive direction.
- If you change the stretching/compression factor during operation, synchronised stretching/compression must be activated for this type of profile.
- For feed profiles, 0° and 360° are not the same position for the Y axis from the calculation point of view. It is therefore not possible to change with $bProfileForcedShift = TRUE$ to a forward/backward profile within a feed profile.



Synchronised stretching and compression of the Y axis

Synchronised stretching and compression is required for absolutely synchronous running of master value and drive and a changeover of the stretching/compression factor during operation.

The stretching/compression itself is carried out by the **L_CamSetYAxisVelocity** FB, which is between the **L_CamData** FB and the **L_CamControl** FB:

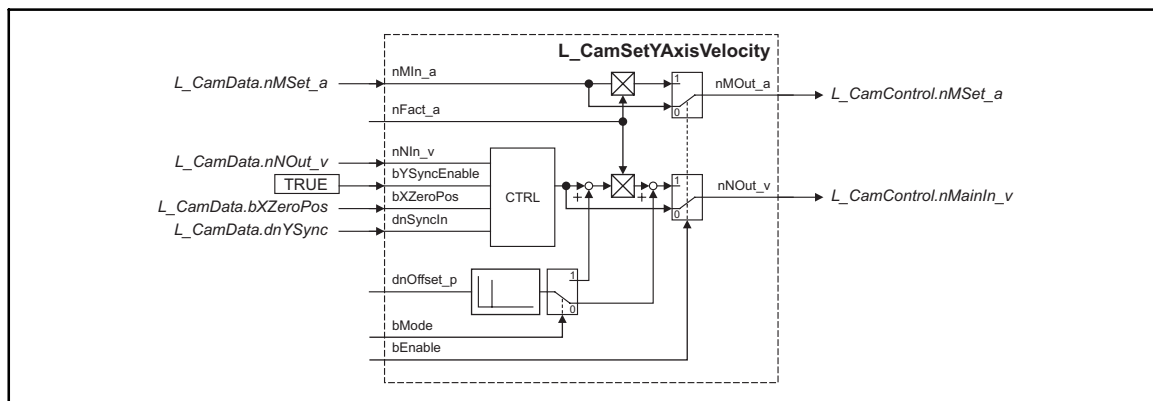


Fig. 2-22

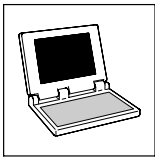
Synchronised stretching/compression with the **L_CamSetYAxisVelocity** FB

The mark synchronisation (touch probe) for the Y axis is carried out via the **L_CamControl** FB.



Note!

If the **L_CamSetYAxisVelocity** FB is not to be used or a synchronised stretching/compression via the **L_CamSetYAxisVelocity** FB is to be effected, set *bYSyncEnable* to FALSE.



Function library *LenzeCamControl.lib*

Function blocks

L_CamData - Profile management

Torque precontrol

For highly dynamic applications, the **L_CamData** FB at the same time can extract a torque profile according to the motion profile from the *.lc9 file and output it at *nMSet_a* for torque precontrol.

- Use the **L_CamSetYAxisVelocity** FB and the **L_CamControl** FB to send the torque profile to the motor control (**MCTRL**) SB of the **9300 servo PLC**.
- If you do not need the torque profile, *bTorquePreCtrlEnable* should be set to FALSE to deactivate this function and save computing time.
- If *bTorquePreCtrlEnable* is set to TRUE, the torque profile from the *.lc9 file will be used, whereby the amplitude or degree of precontrol depends on the speed of the line shaft (x axis).



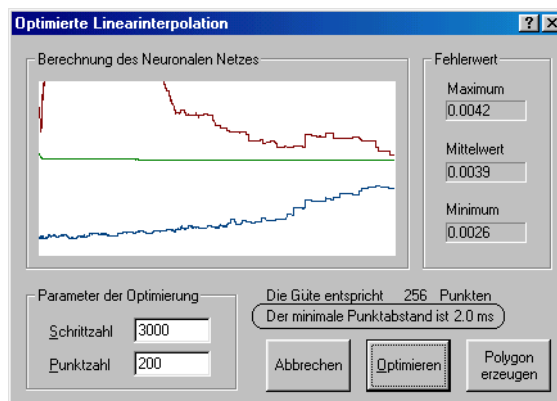
Tip!

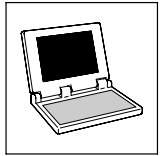
- If you create a motion profile with **CamDesigner**, enter the rated clock pulse in the *Features of the master* dialog box to ensure that the signal is normalised for this rate.
- The **L_CamControl** FB is used for adjustment to the moment of inertia.

Maximum number of points

The number of points of the motion profile should not be higher than one point per processing cycle of the **L_CamData** FB, otherwise points will be skipped.

- If the FB skips more than 20 points, the output *nState* will send the warning "20". An appropriate troubleshooting should be provided (e.g. reduction of the x axis speed) to ensure that the drive does not react in an uncontrolled way.
- In the **CamDesigner** the distance between the points for a selected clock pulse number can be checked via the *Interpolation* dialog box (only for the absolute data model):





2.6 L_CamExtrapolate - Extrapolation

This FB can be used to carry out an extrapolation of position information, for instance to compensate bus transmission cycles greater than 1 ms or to smooth absolute value encoders with a low resolution. A forced extrapolation can also be effected.

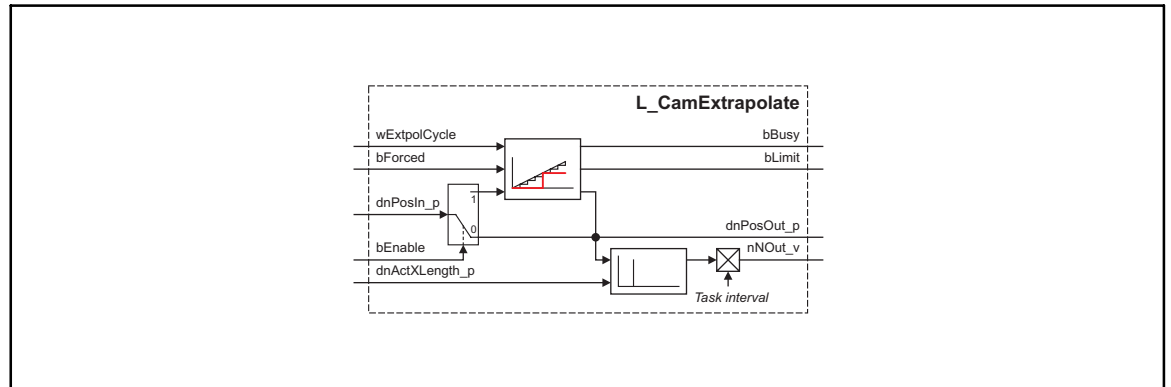
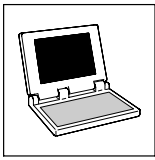


Fig. 2-23 "L_CamExtrapolate" function block

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
wExtpolCycle	Word	-	Number of desired extrapolation cycles - Value internally limited to 1...127. - Depending on the task interval and bus cycle.	
bForced	Bool	Binary	Forced extrapolation	
			FALSE	Extrapolation is only carried out if the input value has not changed.
			TRUE	Extrapolation is carried out in accordance with the number in <i>nExtpolCycle</i> , even if the input value has changed.
dnPosIn_p	Double Integer	Position	Position information	
bEnable	Bool	Binary	Activating extrapolation	
			FALSE	The <i>dnPosIn_p</i> master value is output 1:1 on the <i>dnPosOut_p</i> position output without extrapolation. The <i>nNOut_v</i> speed output is calculated (differentiation of the x axis).
			TRUE	The <i>dnPosIn_p</i> master value is output extrapolated on the <i>dnPosOut_p</i> position output. The duration of the extrapolation depends on the settings <i>bForced</i> and <i>wExtpolCycle</i>.
dnActXLength_p	Double integer	Position	Length of x axis	

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Value/meaning
bBusy	Bool	Binary	Status signal "Extrapolation is carried out" TRUE Extrapolation is carried out.
bLimit	Bool	Binary	Status signal "Extrapolation limit reached" TRUE Extrapolation limit <i>wExtpolCycle</i> reached.
dnPosOut_p	Double integer	Phase	Output of the position information
nNOut_v	Integer	Velocity	Digital frequency output 16384 ≙ 15000 rpm - Limitation + overflow memory Note: The phase difference signal considers the incremental cycle skip from $x_{\max}$ to 0 at the end/start of a cycle, but not a possible incremental decimal fraction in the cycle. Therefore do not use this output signal if the cycle cannot be represented incrementally as an integer!



Function library *LenzeCamControl.lib*

Function blocks

L_CamExtrapolate - Extrapolation

Compensation of bus transmission cycles > 1ms

If position information (e. g. the position of the vertical shaft) are exchanged between master and slave drives via a bus system with a transmission cycle > 1 ms, or between tasks with different cycle times, the setpoints cannot be processed in the control cycle of the drives. The consequence of this is that the position at a constant speed does not change linearly, but in a staircase-shaped manner, which in turn brings about torque impulses.

By means of this FB, the staircase-shaped gradation can be smoothed mathematically by extrapolation:

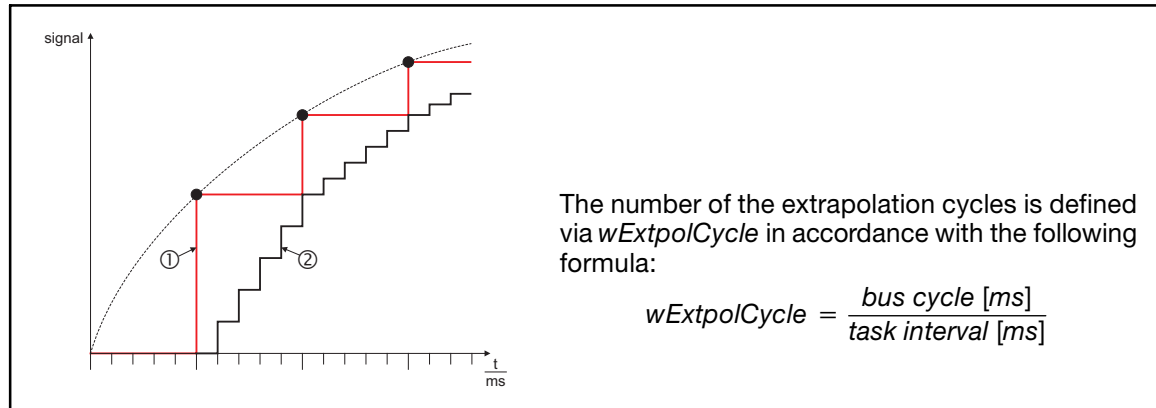


Fig. 2-24

Signal characteristic
 ① Setpoint signal via bus system
 ② Interpolated setpoint signal



Tip!

Ensure that an integer value results for *wExtrapolCycle*, so that the position information increases linearly and a speed calculation can be carried out in a correct manner.

- If a value smaller than the one calculated in accordance with the above formula is set, the extrapolation limit is reached and this is displayed by the output *bLimit*. In this case, *dnPosIn_p* is directly interconnected to *dnPosOut_p*.
- If a value greater than the one calculated in accordance with the above formula is set, this has no consequences.

- The extrapolation is activated by setting *bEnable* to TRUE.
- If *bEnable* is set to FALSE, the input signal only is looped through, or the formation of the difference for the speed output is carried out.

Smoothing a low-resolution absolute value encoder

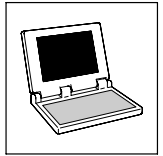
If the resolution of the absolute value encoder is so low that not a new setpoint is available for each task call, this FB is used to effect a "fine interpolation" between the setpoints, providing for a smoother running.

For this purpose, instead of the bus cycle time, enter the resolution converted at minimum speed in the above-mentioned formula for calculating *wExtrapolCycle*.

Forced extrapolation

Forced extrapolation means that irrespective of whether the input signal has changed or not, extrapolation to the maximum value is effected under *wExtrapolCycle*.

- The forced extrapolation is activated by setting *bForced* to TRUE.



2.7 L_CamGetPosition - Unit conversion

This FB converts the incremental value of a position into a physical unit.

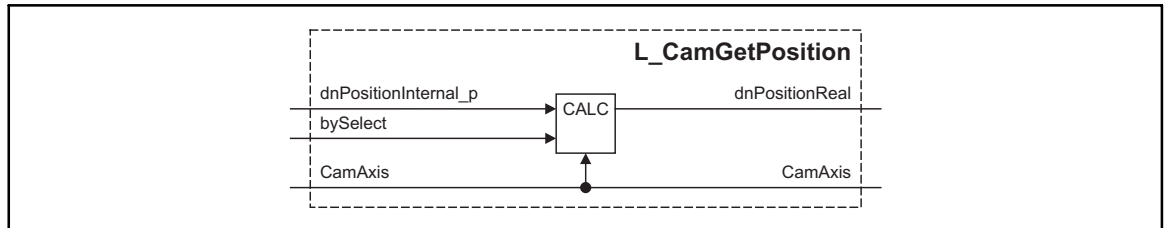


Fig. 2-25

Function block "L_CamGetPosition"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
dnPositionInternal_p	Double Integer	Position	Position in [inc] • 65535 ≡ 1 Motor revolution or encoder revolution
bySelect	Byte	Number	Selection of either master or slave positions for conversion. 0 Master positions 1 Slave positions

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
dnPositionReal	Double Integer	Unit	Position in [unit] • 10000 = 1.0000 [unit]

Inputs/outputs (Variable type: VAR_IN_OUT)			
Identifier	Data type	Signal type	Info
CamAxis	CAM_AXIS_REF	-	Profile parameters • Data structure with elements which contain machine parameters in the internal measuring system. (For description see FB L_CamMachineData. 2-41) • Connect this input with a global variable which is written by the FB L_CamMachineData.

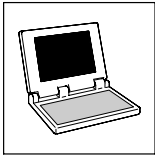
Formula

$$dnPositionReal \text{ [unit]} = dnPositionInternal_p \text{ [inc]} \cdot \frac{10000 \cdot \text{Feed constant [unit/rev.]}}{65536 \text{ [inc/rev.]}} \cdot \frac{\text{Gearbox_factor_denominator}}{\text{Gearbox_factor_numerator}}$$



Tip!

- Due to computing time reasons this FB should only be activated when an input signal is changed.
- The output value *dnPositionReal* is output in fixed comma format and has four decimal positions.
– Example: the value "1205900" means 120.5900 [units].



Function library LenzeCamControl.lib

Function blocks

L_CamLimitVelocity - limiting element

2.8 L_CamLimitVelocity - limiting element

This FB provides a speed limiter without position loss.

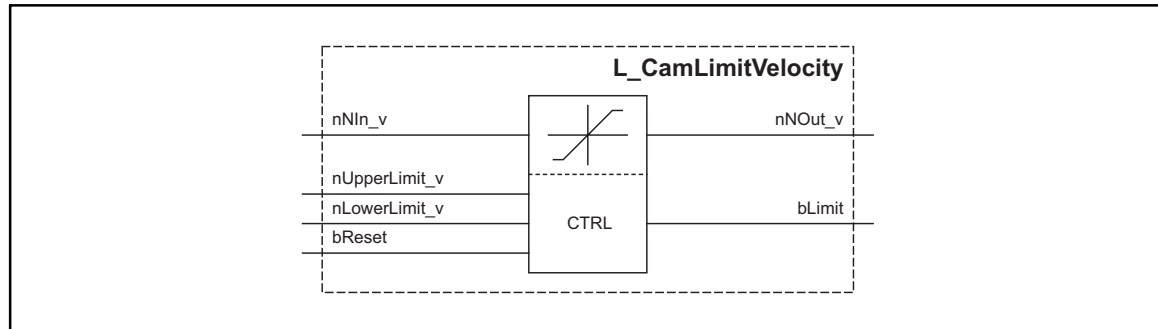


Fig. 2-26

Function block "L_CamLimitVelocity"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
nNIn_v	Integer	Velocity	Speed in [inc/ms] • 16384 [inc/ms] ≈ 15000 rpm
nUpperLimit_v	Integer	Velocity	Upper speed limitation
nLowerLimit_v	Integer	Velocity	Lower speed limitation
bReset	BOOL	Binary	Delete overflow buffer TRUE Overflow buffer deleted.

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
nNOut_v	Integer	Velocity	Speed in [inc/ms] • 16384 [inc/ms] ≈ 15000 rpm
bLimit	BOOL	Binary	Message signal "Limitation active" TRUE Upper or lower limitation is active.

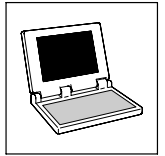
Speed limitation without position loss

While the speed limitation is active, an overflow buffer is filled. It makes sure that the position does not get lost. This is an asynchronous state, which means that the cam profiler does not run in synchronism with the master drive or line shaft if the input signal *nNIn_v* is outside the limits.

- The overflow buffer will be deleted when the input signal is lower than the upper and higher than the lower limit value.
- It is possible to delete the overflow buffer for set-up operation without influencing other functions via the boolean input *bReset*.

Limitation inputs

- Make sure that the value at *nUpperLimit_v* is always higher than the one at *nLowerLimit_v*.
- If *nUpperLimit_v* = 0 AND *nLowerLimit_v* = 0, limiting is not active and the input signal is switched directly to the output. The overflow buffer is deactivated and deleted.
- Similarly, limiting is not active if *nUpperLimit_v* = +16384 AND *nLowerLimit_v* = -16384.



2.9 L_CamMachineData - Machine parameters

This FB conditions the machine parameters for the internal measuring system.

- The machine parameters with the physical units are entered in the Lenze tool **CamDesigner**; they are then stored in an *.lc9 file together with the motion profile data.
- In DDS, you can then use the **Cam Support** to link the *.lc9 file to the project and download it together with the PLC program to the PLC memory. The **Cam Support** is an object in the **Resources** register card in the *Object Organizer*.
- The FB stores the conditioned machine parameters in a data structure (CAM_AXIS_REF).
- Via the output *CamAxis* you can assign this data structure to a global variable to transmit the machine parameters to other LenzeCamControl FBs via this "interface".

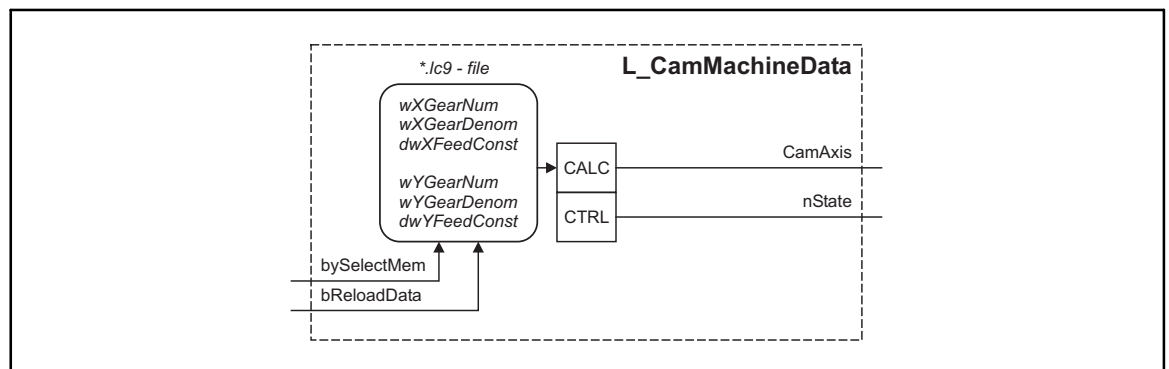
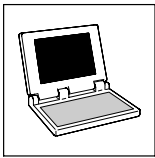


Fig. 2-27

"L_CamMachineData" function block

Internal parameters (Machine constants)			
Identifiers	Data type	Signal type	Information/possible settings
wXGearNum	Word	-	Master (X) - gearbox factor numerator
wXGearDenom	Word	-	Master (X) - gearbox factor denominator
dwXFeedConst	Double word	-	Master (X) - feed constant
wYGearNum	Word	-	Slave (Y) - gearbox factor numerator
wYGearDenom	Word	-	Slave (Y) - gearbox factor denominator
dwYFeedConst	Double word	-	Slave (Y) - feed constant

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
bySelectMem	Byte	Number	Selection of data structure (*.lc9 file). The selection can only be accepted with a FALSE-TRUE edge on <i>bReloadData</i>.
			1 RAM block 1 (foreground data field, standard)
			2 RAM block 2 (background data field)
bReloadData	Bool	Binary	Updating of outputs FALSE ↗ TRUE Re-initialisation of profile data and updating of the outputs.



Function library *LenzeCamControl.lib*

Function blocks

L_CamMachineData - Machine parameters

Outputs ⇒ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Value/meaning	
CamAxis	CAM_AXIS_REF	-	Data structure with machine parameters in the internal measuring system By means of a global (temporary) variable you can provide these machine parameters to other LenzeCamControl FBs.	
nState	Integer	Number	Status	
			10	Initialisation: reading the profile data
			0	Ok, no error
			-2	Error: It was not possible to read profile data (*.lc9 file)

Transferring the machine parameters

Define a global variable of the *CAM_AXIS_REF* data type for the purpose of transferring the machine parameters to other FBs and connect this variable to

- the output *CamAxis* of the **L_CamMachineData**
- FB
- the corresponding inputs of the FBs requiring the machine parameters in the internal measuring system.

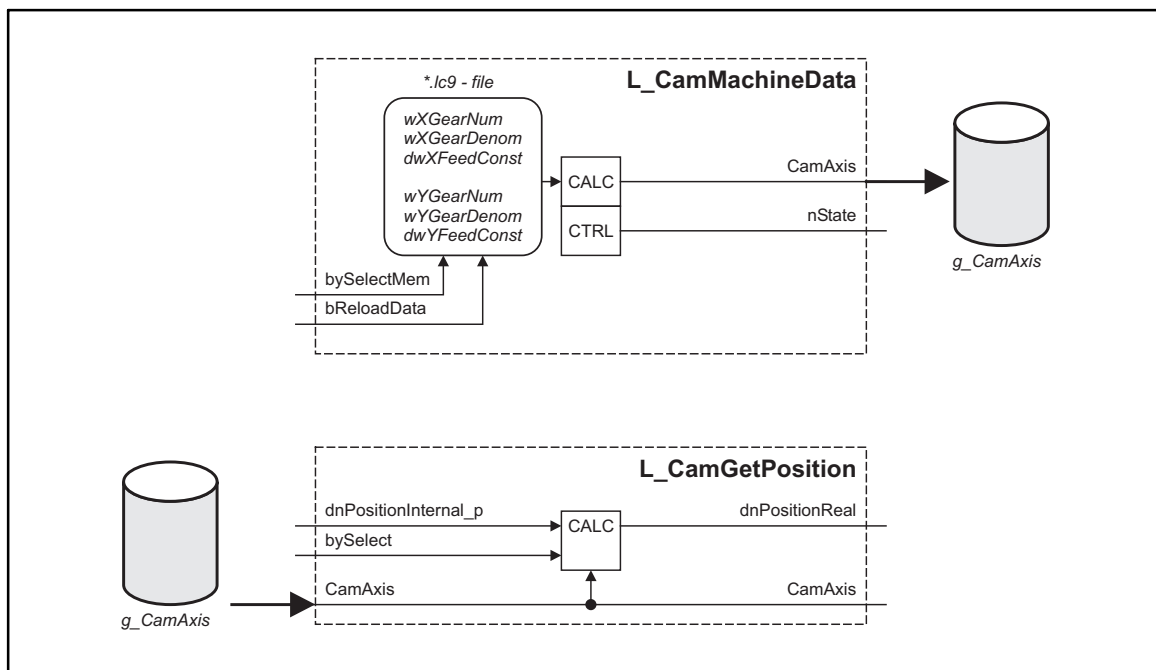


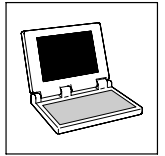
Fig. 2-28

Transferring the machine parameters by means of a global variable (here: *g_CamAxis*)



Tip!

The call of the **L_CamMachineData** FB for instance can be effected in the main program **PLC_PRG**. The internal machine parameters are updated by a FALSE-TRUE edge on *bReloadData*.



2.10 L_CamMonitFollowError - Following error monitoring

This FB serves to monitor following errors. The following error signal of the FB **L_CamControl** is evaluated here.

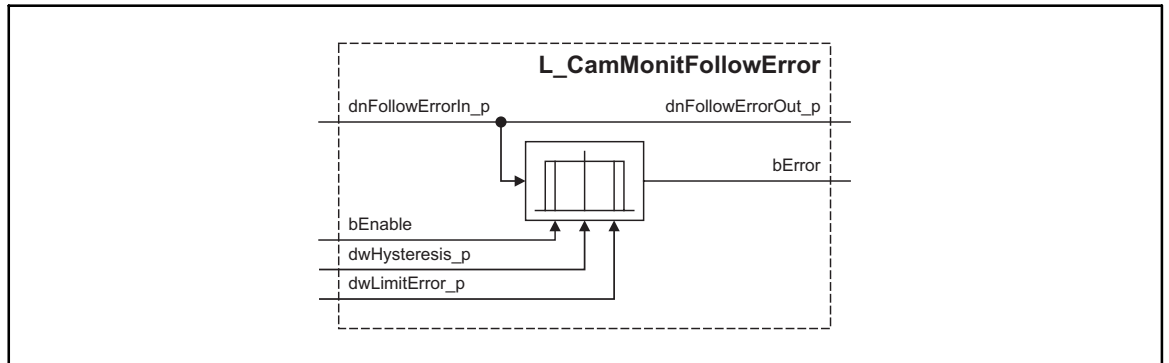
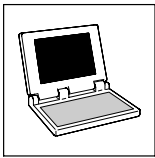


Fig. 2-29

Function block "L_CamMonitFollowError"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
bEnable	BOOL	Binary	Activate monitoring TRUE Monitoring activated.
dnFollowErrorIn_p	Double integer	Position	Following error input • Connect this input with the output <i>dnFollowError_p</i> of the FB L_CamControl .
dwHysteresis_p	Double Word	Position	Hysteresis for the error output
dwLimitError_p	Double Word	Position	Threshold for positive and negative following errors

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
dnFollowErrorOut_p	Double integer	Position	Following error output • Connect this output e.g. with the input <i>dnPosSet_p</i> of the SB MCTRL (deviation of actual phase from the set phase for phase controllers).
bError	BOOL	Binary	Status signal "Following error" TRUE Following error via threshold



Function library *LenzeCamControl.lib*

Function blocks

L_CamMonitFollowError - Following error monitoring

Evaluation

For evaluating the following error signal *dnFollowError_p* of the FB **L_CamControl** it must be connected with the input *dnFollowErrorIn_p*.

- The FB compares it with the configurable following error limit *dwLimitError_p*. If this limit is exceeded in positive or negative direction, it is indicated by the TRUE state at the output *bError*.
- For resetting the output *bError* a hysteresis can be defined at *dwHysteresis_p* to avoid a permanent change in the switching point.
- Moreover, the following error input signal is output 1:1 to *dnFollowErrorOut_p*.

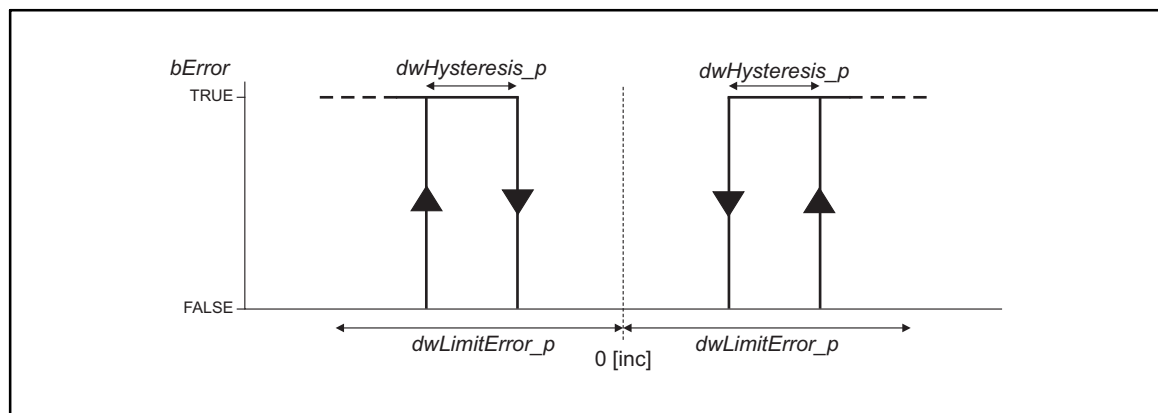


Fig. 2-30

Switching threshold with hysteresis for (early warning stage) output *bError*



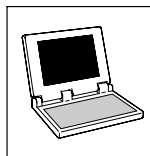
Tip!

In case of the **9300 Servo PLC** you can connect e.g. the output *dnFollowErrorOut_p* with the input *dnPosSet_p* of the SB **MCTRL**, to report to the motor control the deviation of the actual phase from the set phase for the phase controller.

Deactivation of the following error monitoring

By setting *bEnable* to FALSE the following error monitoring can be deactivated.

Even if the following error monitoring is deactivated, the following error input signal is output 1:1 to the output *dnFollowErrorOut_p* so that the FB does not need to be dismantled for this purpose.



2.11 L_CamPosCtrlLin - Linear profile generator

Use this FB for the following positioning functions:

- Bringing the drive into cam position (e. g. after mains connection, manual control, homing)
- Detaching the drive from the profile and bringing it into a safe position (position override function)
- Positioning the x axis via x offset (superimposed positioning)
- Positioning the Y axis via Y offset (superimposed positioning)

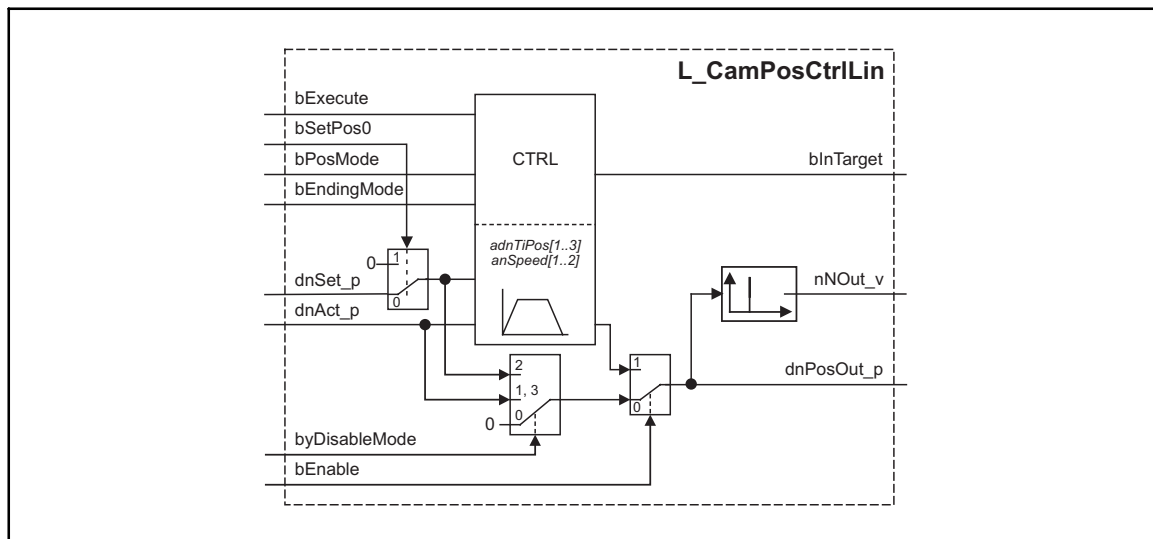
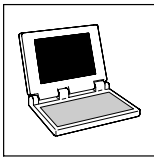


Fig. 2-31

"L_CamPosCtrlLin" function block

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
bExecute,	Bool	Binary	Carrying out the positioning profile Only possible if input <i>bEnable</i> is set = TRUE.	
			FALSE	No positioning / positioning is cancelled. The speed is directed to 0 rpm (<i>nOut_v</i> = 0).
			TRUE	Positioning is carried out. For using the <i>bEndingMode</i> = TRUE switch-off mode, a new positive edge for a subsequent positioning process is required.
bSetPos0	Bool	Binary	The set position internally is set to "0". Only if <i>bEnable</i> = TRUE.	
			FALSE	As set position, the value on the input <i>dnSet_p</i> is used.
			TRUE	The value on the input <i>dnSet_p</i> is ignored. The set position internally is set to "0".
bPosMode	Bool	Binary	Positioning mode	
			FALSE	Absolute positioning
			TRUE	Relative positioning - Use for <i>bEndingMode</i> = TRUE (with limit stop) only! - With a FALSE-TRUE edge on the input <i>bExecute</i>, the outputs further traverse by the value <i>dnSet_p</i>.
bEndingMode	Bool	Binary	Limit stop	
			FALSE	Without limit stop The outputs continually follow the setpoint on <i>dnSet_p</i>.
			TRUE	With limit stop - The outputs follow the setpoint on <i>dnSet_p</i> once. - If the set position changes, a new FALSE-TRUE edge on the input <i>bExecute</i> is required for a further positioning process.



Function library LenzeCamControl.lib

Function blocks

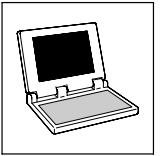
L_CamPosCtrlLin - Linear profile generator

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
dnSet_p	Double integer	Position	Positioning setpoint
dnAct_p	Double integer	Position	Actual position Connect this input for instance to the output <i>dnSetAuxPos_p</i> of the L_CamControl FB.
byDisableMode	Byte	-	Behaviour for deactivated positioning function (<i>bEnable</i> = FALSE)
			0The output <i>dnPosOut_p</i> skips to "0". The speed signal on the output <i>nOut_v</i> represents this position skip as a phase signal. Note: If the output <i>nOut_v</i> takes effect on the setpoint generation for the drive, undefined motions (impacts, following errors) may be the consequence.
			1The outputs <i>dnPosOut_p</i> and <i>nOut_v</i> follow the value on the input <i>dnAct_p</i> (actual position).
			2The outputs <i>dnPosOut_p</i> and <i>nOut_v</i> follow the value on the input <i>dnSet_p</i> (setpoint).
			3The outputs <i>dnPosOut_p</i> and <i>nOut_v</i> follow the value on the input <i>dnAct_p</i> (actual position), though without limitation.
bEnable	Bool	Binary	Activating/deactivating positioning function This input has highest priority
			FALSEDeactivate positioning function For behaviour, see <i>nDisableMode</i>
			TRUEActivate positioning function

Outputs ⇨ (Variable type: VAR_OUTPUT)					
Identifiers	Data type	Signal type	Value/meaning		
dnPosOut_p	Double integer	Position	Position setpoint		
nNOut_v	Integer	Velocity	Speed output of the profile generator 16384 ≡ 15000 rpm		
bInTarget	Bool	Binary	Status signal "Position target reached" <table><tr><td>TRUE</td><td>Position target reached</td></tr></table>	TRUE	Position target reached
TRUE	Position target reached				

Parameters (Variable type: VAR_CONSTANT_RETAIN)				
Identifiers	Data type	Signal type	Information	
Code*			Possible settings	Presetting
adnTiPos[1..3]	Double integer array	-	Ramps for acceleration/deceleration (ref. to 4000 rpm)	
			adnTiPos[1] = deceleration ramp for stop before the set position is reached (bExecute = FALSE)	
C1079/1			0.010 {0.001 unit} 130.000	1.000 s
			adnTiPos[2] = acceleration ramp for the positioning process	
C1079/2			0.010 {0.001 unit} 130.000	1.000 s
			adnTiPos[3] = deceleration ramp for the positioning process	
C1079/3			0.010 {0.001 unit} 130.000	1.000 s
anSpeed[1..2]	Array of integers	-	Traversing speed	
			anSpeed[1] = feed (positive speed)	
C1078/1			-15000 {1 rpm} 15000	200 rpm
			anSpeed[2] = return motion (negative speed)	
C1078/2			-15000 {1 rpm} 15000	200 rpm
* Code of first instance				

* Code of first instance



Absolute/relative positioning

Absolute positioning	Relative positioning
- Selection: <i>bPosMode</i>= FALSE - The setpoint on <i>dnSet_p</i> is considered as an absolute set position (relating to the zero position).	- Selection: <i>bPosMode</i>= TRUE - The outputs from the current position are further traversed by the value <i>dnSet_p</i>. - Condition: <i>bEndingMode</i> = TRUE (with limit stop)

Without/with limit stop

Without limit stop	With limit stop
- Selection: <i>bEndingMode</i>= FALSE - The outputs continually follow the setpoint on <i>dnSet_p</i>.	- Selection: <i>bEndingMode</i>= TRUE - The outputs follow the setpoint on <i>dnSet_p</i> once. - If the set position changes, a new FALSE-TRUE edge on the input <i>bExecute</i> is required for a further positioning process.

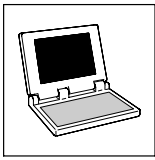
Stopping the positioning

By means of *bExecute* = FALSE, an active positioning procedure can be stopped.

- Via the *adnTiPos[1]* deceleration ramp, the speed on output *nNOut_v* is brought into standstill.
- At the same time, the output *dnPosOut_p* is stopped.
- By means of *bExecute* = TRUE, positioning can be continued or a new positioning process can be started. In the operating mode "with limit stop" (*bEndingMode* = TRUE), a FALSE-TRUE edge on *bExecute* is required for this purpose.

Approaching zero position

By means of *bSetPos0* = TRUE, *dnSet_p* internally is set to zero and the output *dnPosOut_p* can be traversed to zero position.



Function library *LenzeCamControl.lib*

Function blocks

L_CamPosCtrlLin - Linear profile generator

Example: Selecting an offset for the Y axis

The following travel profile is to be realised by means of the **L_CamPosCtrlLin** FB:

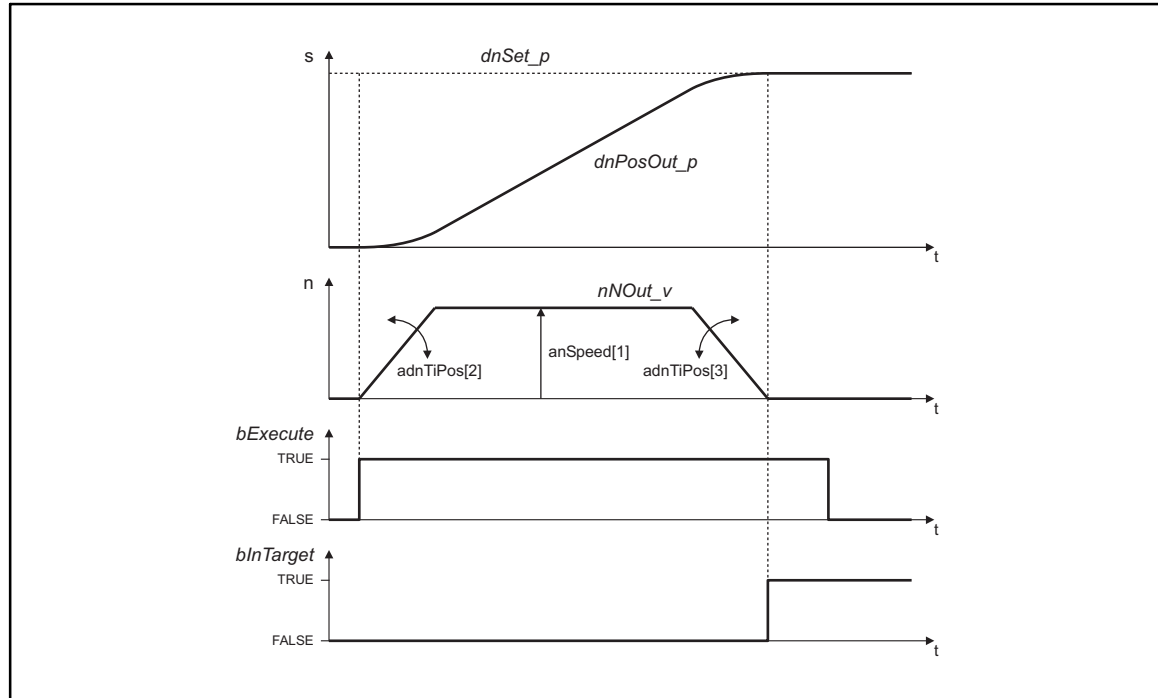


Fig. 2-32

Travel profile

Circuit proposal:

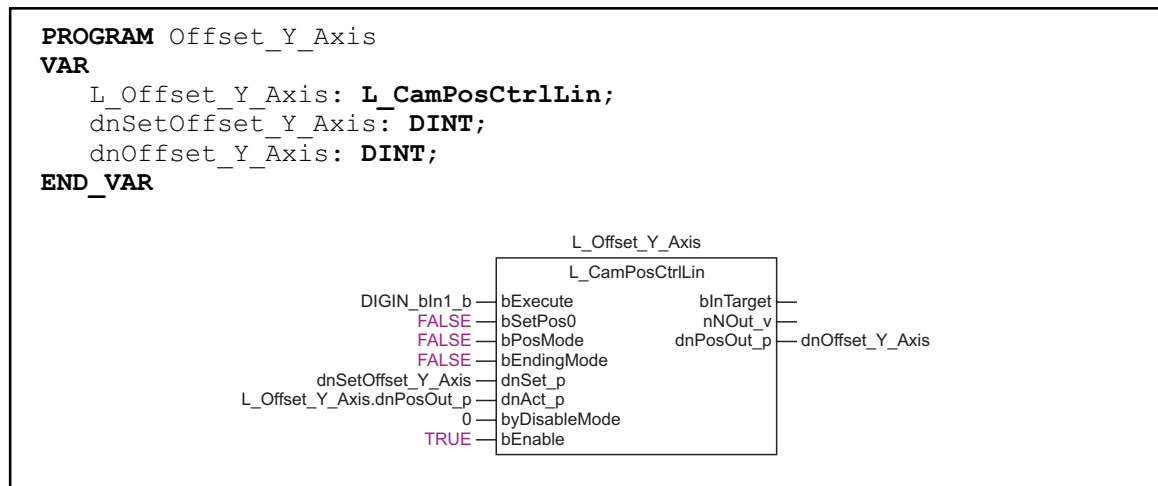
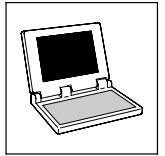


Fig. 2-33

Circuit proposal for the selection of an offset for the Y axis

- The set position here is defined via the *dnSetOffset_Y_Axis* variable.
- The actual position can be directly read from the output *dnPosOut_p*.
- Via the digital input *DIGIN_bln1_b*, positioning is enabled.
- The input *bEnable* has to be set to TRUE during positioning.



2.12 L_CamProfileData - Loading of important profile points

This FB provides important profile points as internal increments.

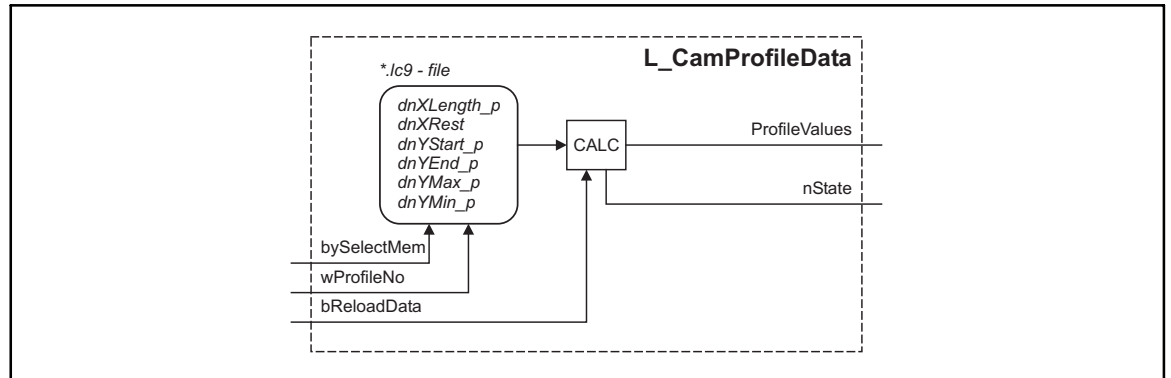
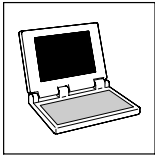


Fig. 2-34 Function block "L_CamProfileData"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
bySelectMem	Byte	Number	Selection of data structure (*.lc9 file) in the memory. The selection can only be accepted with a FALSE-TRUE transition at <i>bReloadData</i>.
			1RAM memory block 1 (foreground data field, standard)
			2RAM memory block 2 (background data field)
wProfileNo	Word	Number	Selection of the motion profile - Selection range 0 ... 47 (The selection range is limited to the number of motion profiles defined in the *.lc9 file.)
bReloadData	BOOL	Binary	Updating of outputs A new data structure selection through <i>bySelectMem</i> is accepted.
			FALSE ↗ TRUEThe outputs are updated.

Outputs ⇨ (Variable type: VAR_OUTPUT)				
Identifier	Data type	Signal type	Value/meaning	
ProfileValues	CAM_PROFILE_REF	Position	Data structure with the most important points of the selected profile Positions in [inc]	
nState	Integer	Number	Status	
			15	Initialisation or reading of data from the *.lc9 file
			10	Warning: Profile not available (wProfileNo > max. profile number)
			0	FB in "StandBy"
			-2	Error: It was not possible to read profile data (*.lc9 file)



Function library *LenzeCamControl.lib*

Function blocks

L_CamProfileData - Loading of important profile points

Data structure CAM_PROFILE_REF

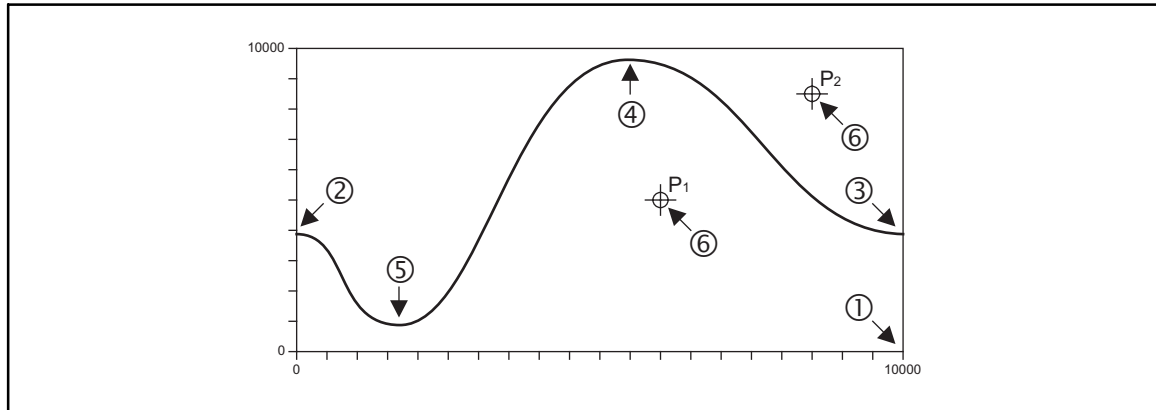


Fig. 2-35

Example profile

Elements of data structure CAM_PROFILE_REF		
Identifier	Data type	Meaning
① dnXLength_p	Double Integer	Clock pulse length (length of X axis) in [inc]
dnXRest	Double Integer	Decimal positions of the clock pulse length
② dnYStart_p	Double Integer	Start position of Y axis in [inc] at X position = 0
③ dnYEnd_p	Double Integer	End position of Y axis in [inc] at X position = <i>dnXLength_p</i>
④ dnYMax_p	Double Integer	Maximum Y position/maximum Y travel in [inc]
⑤ dnYMin_p	Double Integer	Minimum Y position/minimum Y travel in [inc]
⑥	Via CamDesigner you can enter up to four additional base points. The Y position of the base points do not need to be on the motion profile. - These base points are entered via the command Draft→Position flag. - During compilation, the data of the points, the motion profile and the cams are written to the *.IC9 file.	
dnXAddPos_p[0..3]	Double-integer array	X positions of the additional points (P1...P4) in [inc]
dnYAddPos_p[0..3]	Double-integer array	Y positions of the additional points (P1...P4) in [inc]

Output/update of data

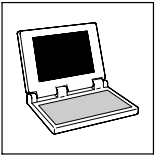
The positions at the output *ProfileValues* are updated in the following cases:

- after a profile change at *wProfileNo*.
- by a FALSE/TRUE edge at *bReloadData*.
 - This is necessary e.g. after the profile data have been changed by the FB **L_CamEditProfileData**.



Tip!

The FB must only be activated during a cyclic task, i.e. it must not be addressed during a task with a fixed cycle time (e.g. T#10ms).



2.13 L_CamSequence - Selection of a profile sequence

This FB is used to define a sequence of profiles for the **L_CamData** FB; here among other things, the following functions are supported:

- Selection of any sequence
- Cyclic and continuous operation
- Hold and reset function
- Selection of a first step

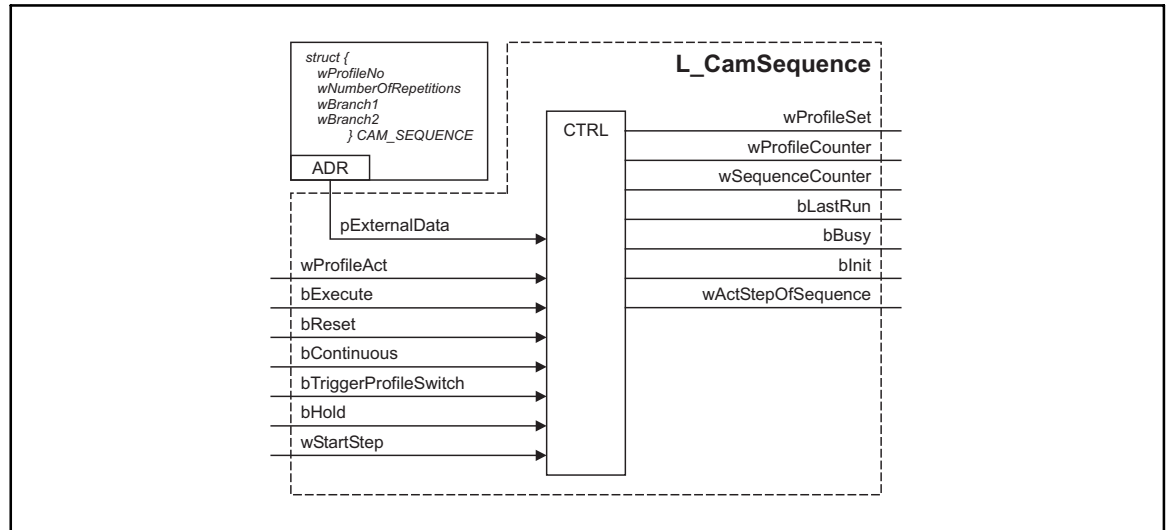
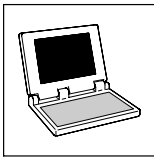


Fig. 2-36

"L_CamSequence" function block

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
pExternalData	Pointer to CAM_SEQUENCE	-	Pointer to the starting address of the sequence (array from <i>CAM_SEQUENCE</i> data type in the user program).
wProfileAct	Word	Number	Number of the profile currently executed (actual profile). • Connect this input to the corresponding output of the L_CamData FB. 0 ... 47 Profile No. 0 ... 47
bExecute	BOOL	Binary	Starting/aborting the sequence FALSE Abort execution of the sequence • The current profile is still executed. • By reset to TRUE, the execution of the sequence is continued. TRUE Start/continue the execution of the sequence
bReset	BOOL	Binary	Carrying out the reset • 4 call cycles of the FB are required to carry out the reset. • During the reset is carried out, output <i>bInit</i> is set to TRUE. TRUE The data is reinitialised with the starting values, and the counter outputs are reset.
bContinuous	BOOL	Binary	Selection of continuous/cyclic operation FALSE Cyclic operation • The sequence is executed exactly once. • If the sequence has been executed, <i>bBusy</i> is set to FALSE. TRUE Continuous operation • The sequence is continuously executed again; when the end of the sequence is reached, execution starts with <i>wStartStep</i> again. • Via <i>wSequenceCounter</i> it is output how often the sequence has already been executed.



Function library *LenzeCamControl.lib*

Function blocks

L_CamSequence - Selection of a profile sequence

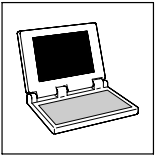
⇒ Inputs (Variable type: VAR_INPUT)					
Identifiers	Data type	Signal type	Information/possible settings		
bTriggerProfile Switch	BOOL	Binary	Trigger input for profile or step change Connect this input with the output <i>bXZeroPos</i> of the L_CamData FB.		
			<table><tr><td>TRUE ⇒ FALSE</td><td>A changeover to the next profile or step is effected.</td></tr></table>	TRUE ⇒ FALSE	A changeover to the next profile or step is effected.
TRUE ⇒ FALSE	A changeover to the next profile or step is effected.				
bHold	BOOL	Binary	Holding function The output <i>bBusy</i> remains set to TRUE during the holding function.		
			<table><tr><td>FALSE</td><td>Continuing sequence.</td></tr></table>	FALSE	Continuing sequence.
			FALSE	Continuing sequence.	
<table><tr><td>TRUE</td><td>The sequence is stopped and the same step is executed continually. The counter output <i>wProfileCounter</i> runs downwards to zero.</td></tr></table>	TRUE	The sequence is stopped and the same step is executed continually. The counter output <i>wProfileCounter</i> runs downwards to zero.			
TRUE	The sequence is stopped and the same step is executed continually. The counter output <i>wProfileCounter</i> runs downwards to zero.				
wStartStep	Word	Number	Number of the first sequence step. A new value only is accepted for <i>bExecute</i> = FALSE and if the sequence is completed (<i>bBusy</i> = FALSE).		
			<table><tr><td>0 ... 47</td><td>Step No. 0 ... 47</td></tr></table>	0 ... 47	Step No. 0 ... 47
0 ... 47	Step No. 0 ... 47				

Outputs ⇨ (Variable type: VAR_OUTPUT)					
Identifiers	Data type	Signal type	Value/meaning		
wProfileSet	Word	Number	Number of the profile of the current step (set profile). Connect this output to the corresponding input of the L_CamData FB.		
wProfileCounter	Word	Number	Number of times the profile of the current step will still be executed. The value "0" refers to the "last cycle".		
wSequenceCounter	Word	Number	Number of times the sequence has already been executed in continuous operation. Maximum count value = 65535		
bLastRun	BOOL	Binary	Status signal "Executing the last profile" Can be used to stop the virtual master. <table><tr><td>TRUE</td><td>Last profile within the sequence is executed.</td></tr></table>	TRUE	Last profile within the sequence is executed.
TRUE	Last profile within the sequence is executed.				
bBusy	BOOL	Binary	Status signal "Execution active" <table><tr><td>TRUE</td><td>Sequence is still being executed.</td></tr></table>	TRUE	Sequence is still being executed.
TRUE	Sequence is still being executed.				
blnit	BOOL	Binary	Status signal "initialisation active" <table><tr><td>TRUE</td><td>Initialisation via <i>bReset</i> active (max. 4 call cycles of the FB).</td></tr></table>	TRUE	Initialisation via <i>bReset</i> active (max. 4 call cycles of the FB).
TRUE	Initialisation via <i>bReset</i> active (max. 4 call cycles of the FB).				
wActStepOf Sequence	Word	Number	Display of the currently edited step of a sequence		

Selection of the sequence

The sequence is defined in the user program by means of an array of the *CAM_SEQUENCE* data structure:

CAM_SEQUENCE (structure)		
Identifiers	Data type	Information
wProfileNo	Word	Set profile for this step.
wNumberOfRepetitions	Word	Number of times the step is to be run through in succession.
wBranch1	Word	Sequence step if number of defined repetitions has not yet been achieved.
wBranch2	Word	Sequence step if number of defined repetitions has been achieved.



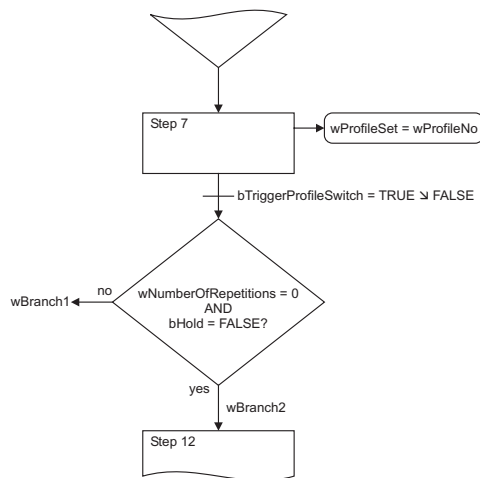
The starting address of the arrays has to be communicated to the FB on the input *pExternalData* via an address operator (**ADR**).

Max. 48 different steps (array index 0 ... 47) for a sequence can be defined in accordance with the following pattern:

```

PROGRAM SequenceInit
VAR
    MySequenceStep: ARRAY[0..47] OF CAM_SEQUENCE;
END VAR

MySequenceStep[7]
{
    wProfileNo := 17;
    wNumberOfRepetitions := 3000;
    wBranch1 := 7;
    wBranch2 := 12
}
    
```



The above example shows step No. 7 of a sequence.

- The set profile to be executed for this step is profile No. 17 (*wProfileNo*).
- The profile is to be executed at a total of 3000 times (*wNumberOfRepetitions*).
- The repetitions are effected directly in succession, as branch 1 contains the same step number (*wBranch1*).
- After the repetitions have been executed, step No. 12 (*wBranch2*) is carried out.

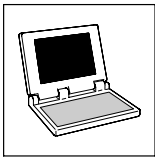


Tip!

Changes with regard to the parameters for the branches and repetitions are only accepted in the initial state (*bExecute* = FALSE).

Sequence end

The end of the sequence is defined by branch 2 (*wBranch2*). If the first step (*wStartStep*) is indicated within branch 2 of a step, the sequence is completed in cyclic operation (*bContinuous* = FALSE).



Function library *LenzeCamControl.lib*

Function blocks

L_CamSequence - Selection of a profile sequence

Cyclic operation

In cyclic operation the sequence is executed exactly once.

- By setting *bContinuous* to FALSE, cyclic operation is activated.
- By setting *bExecute* to TRUE, execution of the sequence is started.
- By resetting *bExecute* to FALSE, the running execution can be interrupted, by reset to TRUE it can be continued (breakpoint).
- During execution, *bBusy* is = TRUE. The FALSE status signals that the execution of the sequence has been completed or interrupted via *bExecute*.
- When starting the last profile of the sequence, *bLastRun* is set to TRUE. By means of this status signal, other FBs (e. g. the **L_CamVMasterPosition**) FB can be controlled.
- After completion of a cycle, the execution of the sequence can be restarted via *bExecute* (cyclic operation).

Continuous operation

In contrast to cyclic operation, the sequence is continually executed in continuous operation.

- By setting *bContinuous* to TRUE, continuous operation is activated.
- By setting *bExecute* to TRUE, execution of the sequence is started.
- By resetting *bExecute* to FALSE, the running execution can be interrupted, by reset to TRUE it can be continued (breakpoint).
- During execution, *bBusy* is = TRUE. The FALSE status signals that the execution of the sequence has been completed or interrupted via *bExecute*.
- After completion of a sequence cycle, execution is automatically started again with the step *wStartStep*.
- Via *wSequenceCounter* the number of times the sequence has already been executed is output.
- By resetting *bContinuous* to FALSE, continuous operation is stopped after the sequence has been passed through completely.

Hold function

By setting *bHold* to TRUE, the sequence cycle can be stopped.

- Only the set profile of the current step is executed.
- During execution, *bBusy* is = TRUE.

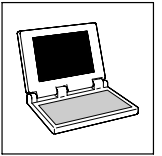
Reset function

By setting *bReset* to TRUE, the initial state (starting values) can be loaded.

- Sequence and profile counters are reset.
- The reset function proceeds via 4 cycles, the output *blnit* meanwhile is set to TRUE.

Selecting the first step

In the initial state (*bExecute* = FALSE) each defined step within the sequence can be selected as a first step via *wStartStep*.



Exemplary sequence

Initialisation:

```
IF SYSTEM_b1LoopTask2 THEN
  MySequence[0].wProfileNo := 5;
  MySequence[0].wNumberOfRepetitions := 3;
  MySequence[0].wBranch1 := 1;
  MySequence[0].wBranch2 := 1;

  MySequence[1].wProfileNo := 4;
  MySequence[1].wNumberOfRepetitions := 3;
  MySequence[1].wBranch1 := 2;
  MySequence[1].wBranch2 := 2;

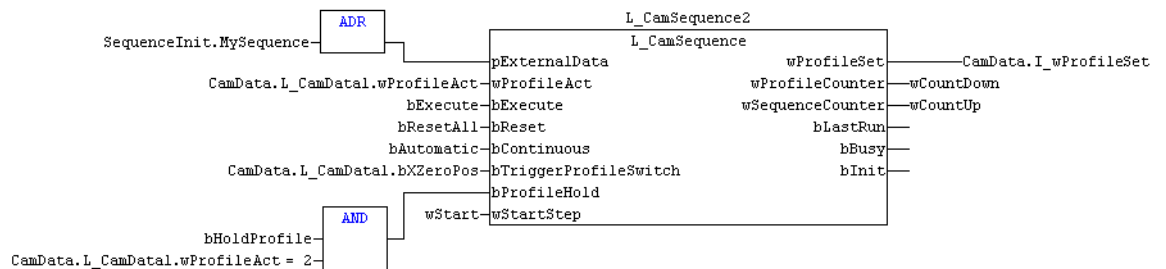
  MySequence[2].wProfileNo := 3;
  MySequence[2].wNumberOfRepetitions := 3;
  MySequence[2].wBranch1 := 0;
  MySequence[2].wBranch2 := 3;

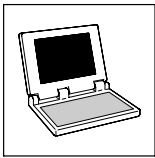
  MySequence[3].wProfileNo := 2;
  MySequence[3].wNumberOfRepetitions := 1;
  MySequence[3].wBranch1 := 3;
  MySequence[3].wBranch2 := 4;

  MySequence[4].wProfileNo := 1;
  MySequence[4].wNumberOfRepetitions := 1;
  MySequence[4].wBranch1 := 4;
  MySequence[4].wBranch2 := 5;

  MySequence[5].wProfileNo := 0;
  MySequence[5].wNumberOfRepetitions := 1;
  MySequence[5].wBranch1 := 5;
  MySequence[5].wBranch2 := 0;
END IF
```

FB wiring:



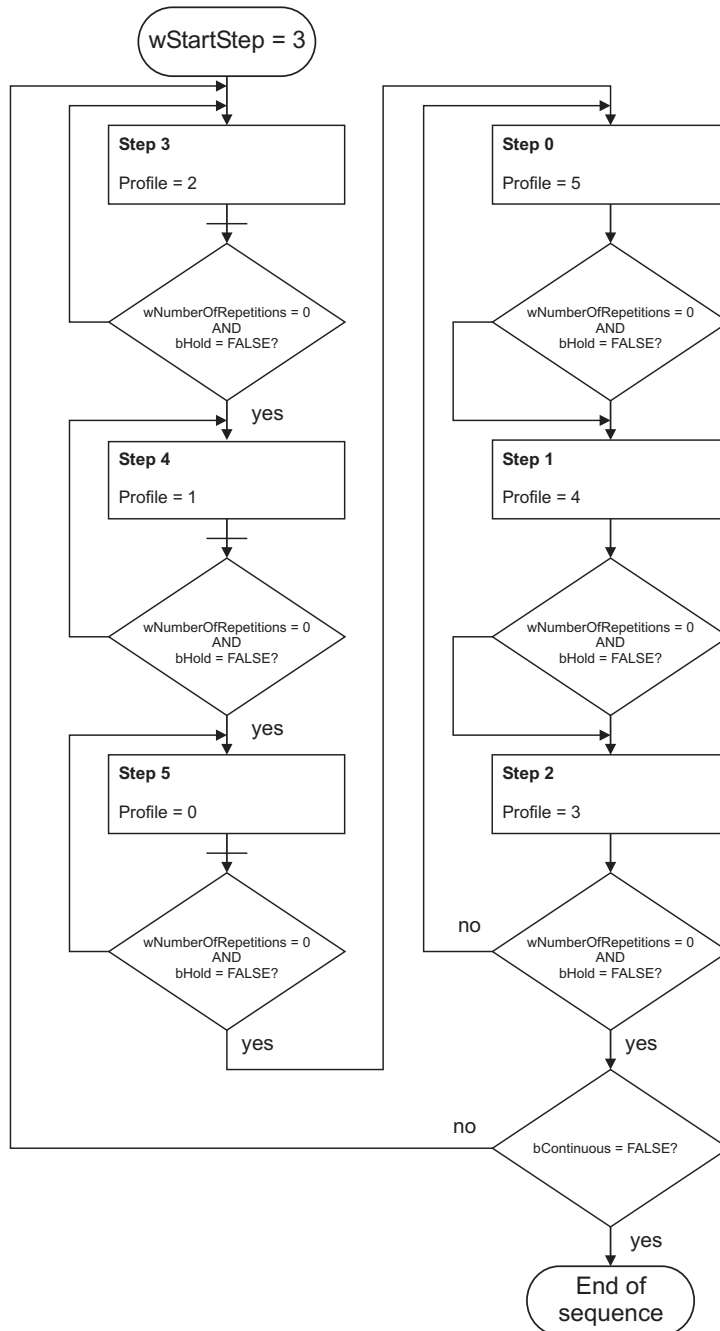


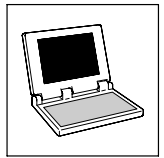
Function library *LenzeCamControl.lib*

Function blocks

L_CamSequence - Selection of a profile sequence

Operating sequence for *wStartStep*= 3:





2.14 L_CamSetPosition - Unit conversion

This FB converts the physical unit of a position into an internal unit.

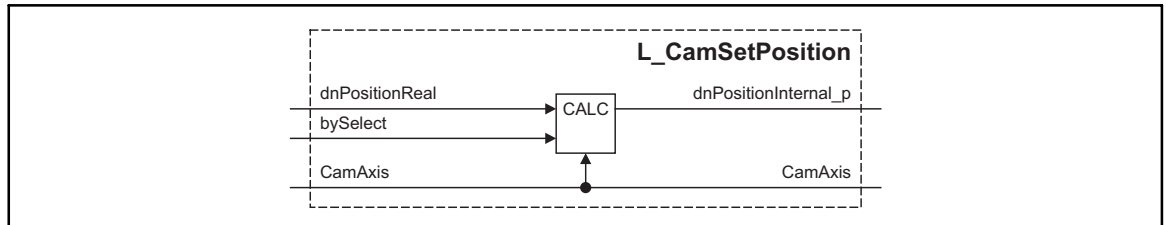


Fig. 2-37

Function block "L_CamSetPosition"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
dnPositionReal	Double Integer	Unit	Position in [unit] • 10000 ≙ 1.0000 [unit]
bySelect	Byte	Number	Selection of either master or slave positions for conversion
			0 Master positions
			1 Slave positions

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
dnPositionInternal_p	Double Integer	Position	Position in [inc] • 65536 ≙ 1 motor revolution

Inputs/outputs (Variable type: VAR_IN_OUT)			
Identifier	Data type	Signal type	Info
CamAxis	CAM_AXIS_REF	-	Machine parameters • Data structure with elements which contain machine parameters in the internal measuring system. (For description see FB L_CamMachineData. 2-41) • Connect this input with a global variable which is written by the FB L_CamMachineData.

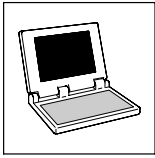
Formula

$$dnPositionInternal_p \text{ [inc]} = \frac{dnPositionReal \text{ [unit]}}{10000} \cdot \frac{65536 \text{ [inc/rev.]}}{\text{Feed constant [unit/rev.]}} \cdot \frac{\text{Gearbox_factor_numerator}}{\text{Gearbox_factor_denominator}}$$



Tip!

- For computing time reasons, only activate this FB if the input value *dnPositionReal* has changed.
- The input value must be in fixed comma format and have 4 decimal positions.
– Example: 120.59 [unit] must be selected as "1205900".



Function library LenzeCamControl.lib

Function blocks

L_CamSetXAxisVelocity - Stretching and compression of the master value

2.15

L_CamSetXAxisVelocity - Stretching and compression of the master value

This FB is used to stretch/compress the x axis and carry out a synchronisation via touch probe. The digital frequency is integrated to the position within the cycle.

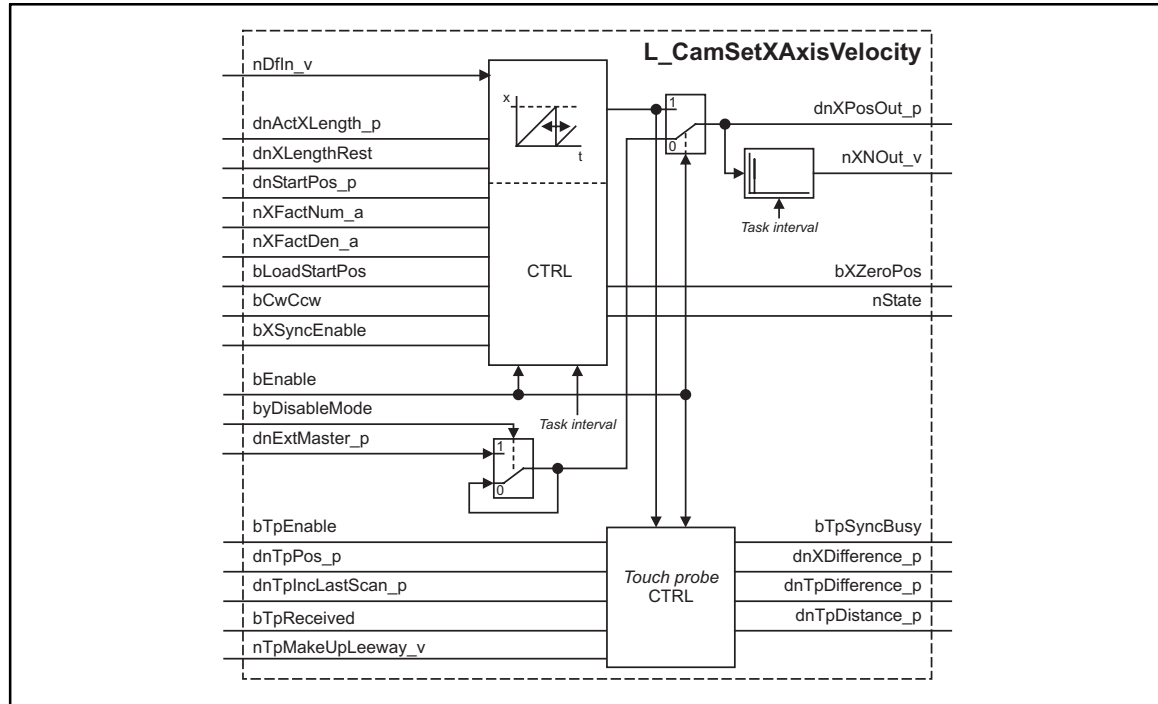
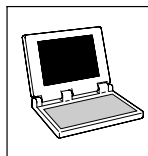


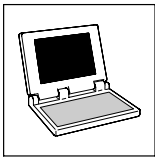
Fig. 2-38 "L_CamSetXAxisVelocity" function block

⇒ Inputs (Variable type: VAR_INPUT)					
Identifiers	Data type	Signal type	Information/possible settings		
nDfln_v	Integer	Velocity	Digital frequency input		
nTpMakeUpLeeway_v	Integer	Velocity	Velocity for touch probe correction • If "0" is selected, the x difference is maintained. • For a smooth compensation use the L_CamPosCtrlLin FB.		
dnActXLength_p	Double integer	Position	Length of x axis • Connect this input to the output <i>dnActXLength_p</i> of the L_CamData FB.		
dnXLengthRest	Double integer	-	Decimal fraction of the x axis (<i>dnActXLength_p</i>) • In case of a gearbox ratio for the x axis which is not integer • This value is provided by the L_CamProfileData FB.		
dnStartPos_p	Double integer	Position	Starting position of the master value integrator • Only positive values between 0 and <i>dnActXLength_p</i> possible. • Acceptance is effected by the FALSE-TRUE edge on <i>bLoadStartPos</i> (if the FB is enabled).		
nXFactNum_a	Integer	Analog	Factor for stretching/compression: numerator • The product of the resulting factor, digital frequency and the task interval (maximum position change per cycle) may not leave the integer value range. • Changes are carried out "online".		
nXFactDen_a	Integer	Analog	Factor for stretching/compression: denominator • Only select values <> "0". • Changes are carried out "online".		
bLoadStartPos	Bool	Binary	Loading starting position in the master value integrator <table><tr><td>TRUE</td><td>Load the <i>dnStartPos_p</i> starting position in the master value integrator.</td></tr></table>	TRUE	Load the <i>dnStartPos_p</i> starting position in the master value integrator.
TRUE	Load the <i>dnStartPos_p</i> starting position in the master value integrator.				



⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
bCwCcw	Bool	Binary	Inversion of the vertical shaft (x axis)	
			FALSE	CW rotation
			TRUE	CCW rotation
bXSyncEnable	Bool	Binary	Activating synchronised stretching/compression	
			TRUE	A new stretching/compression factor only is accepted in the zero crossing.
			FALSE	A new stretching/compression factor is accepted immediately.
bEnable	Bool	Binary	Function block enable	
			FALSE	Function block deactivated, for behaviour see <i>byDisableMode</i>
			TRUE	Function block enabled
byDisableMode	Byte	Number	Behaviour for deactivated function block (<i>bEnable</i> = FALSE)	
			0	Speed output inactive, position is maintained.
			1	Position on <i>dnExtMaster_p</i> is accepted, speed output active.
dnExtMaster_p	Double integer	Position	External master angle - This input is directly interconnected to output <i>dnXPosOut_p</i> if <i>bEnable</i> = FALSE and <i>byDisableMode</i> = "1". - The value is not limited to the <i>dnActXLength_p</i> cycle!	
bTpEnable	Bool	Binary	Activating the touch probe evaluation	
			TRUE	Touch probe evaluation activated
dnTpPos_p	Double integer	Position	Position of the touch probe signal encoder (distance to profile zero point) - Value is internally limited to 0 ... 2 x <i>dnActXLength_p</i>	
dnTpIncLastScan_p.	Double Integer	Position	Number of increments counted since the touch probe. Connect this input for instance to the output <i>DFIN_dnIncLastScan_p</i> of the DFIN_IO_DigitalFrequency SB.	
bTpReceived	Bool	Binary	Control signal for touch probe correction, manually or via interruptible input In case of a manual start, no interrupt is actuated. Therefore no value of the DFIN_IO_DigitalFrequency SB will be applied on the input <i>dnTpIncLastScan_p</i>, which would be required for an exact synchronisation.	
			TRUE	Receive the touch probe signal.

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Value/meaning
dnXPosOut_p	Double integer	Position	Position output Connect this output for instance to the input <i>dnXPosIn_p</i> of the L_CamData FB if both operate in the same task.
nXNOut_v	Integer	Velocity	Vertical shaft speed output (x axis)
			Note: The phase difference signal considers the incremental cycle skip from $x_{\max}$ to 0 at the end/start of a cycle, but not a possible incremental decimal fraction in the cycle. Therefore do not use this output signal if the cycle cannot be represented incrementally as an integer!
bXZeroPos	Bool	Binary	Message signal "Profile zero crossing"
			<table><tr><td>TRUE</td><td>Profile zero crossing</td></tr></table>
TRUE	Profile zero crossing		



Function library *LenzeCamControl.lib*

Function blocks

L_CamSetXAxisVelocity - Stretching and compression of the master value

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Value/meaning
nState	Integer	Number	Status
			13 Warning: resulting position change per cycle is too great for normal operation.
			12 Warning: stretching/compression factor denominator = 0
			11 Warning: starting value inappropriate ($dnXStartPos_p > dnActXLength_p$)
			10 Warning: starting value inappropriate ($dnXStartPos_p < 0$)
			0 Ok, no error
			-10 Error: cycle = "0"; no operation possible.
			-11 Error: resulting position change per cycle is too great for synchronised stretching/compression in the zero crossing.
bTpSyncBusy	Bool	Binary	Status signal "Touch probe correction"
			TRUE Touch probe correction is carried out.
dnXDifference_p	Double integer	Position	Correction value after a touch probe signal Deviation to $dnTpPos_p$ during touch probe.
dnTpDifference_p	Double integer	Position	Difference between two touch probe events
dnTpDistance_p	Double integer	Position	Distance between touch probe signal and profile zero crossing

Enabling the function block and integrating the digital frequency

If *bEnable* is set to TRUE, the function block is enabled and the *nDfln_v* master frequency signal is integrated to one position.

- This position is purely positive and is shown on the output *dnXPosOut_p* within the limits of the cycle.
- The cycle is defined via *dnActXLength_p*.
 - Connect this input to the output *dnActXLength_p* of the **L_CamData** FB.
- For the integration, a starting value can be defined on *dnStartPos_p*. The acceptance of this value is effected by setting *bLoadStartPos* to TRUE.



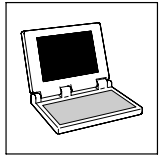
Note!

When changing between profiles with different x axis lengths, the master angle must only rotate in positive direction.

Changing the direction of rotation

Via *bCwCcw* you define the direction of rotation of the master value:

- FALSE: direction of rotation is positive (Cw).
- TRUE: direction of rotation is negative (Ccw).



Looping through the external master signal

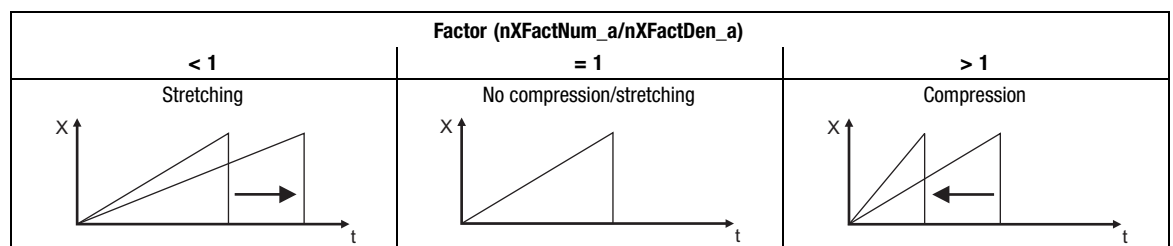
If the functions of the FB are not required, the FB can be switched inactive by setting *bEnable* to FALSE.

If the FB is switched inactive and *byDisableMode* is set to "1", an external master angle applied to *dnExtMaster_p* (e. g. the position signal of a phase encoder) can be directly switched to the output *dnXPosOut_p*.

- In this connection, the output signal is not limited to the cycle.
- The formation of a difference is carried out, so that the respective speed can also be output.
- By means of this function, you can easily change to an alternative master value without having to remove the FB for reasons of calculating time from the project.
- If you change back to the digital frequency by setting *bEnable* to TRUE, you can load a new starting value (*dnXStartPos_p* by setting *bLoadStartPos*) to TRUE.

Stretching/compression of the conductivity signal

Via *nXFactNum_a* and *nXFactDen_a* the factor by which the master value axis is to be stretched or compressed is defined:



Tip!

In this case stretching/compression does not mean an incremental stretching or compression of the x axis, but a change in speed for following the master value axis.

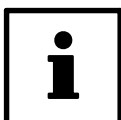
Stretching/compression therefore always has an impact on the speed output *nXNOut_v*!

Synchronised stretching and compression

Stretching/compression usually is effected "online", i. e. a factor change has an immediate impact on the x speed output, by which, however, increments are lost.

In order to ensure that no increments are lost (e. g. for feed applications), a synchronised stretching/compression has to be effected.

- Synchronised stretching/compression means that a change in the stretching/compression factor only is accepted in the zero crossing of the x axis.
- By setting *bXSync* to TRUE, you activate the synchronised stretching/compression.

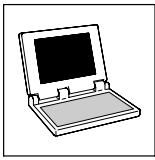


Note!

Observe the following for the synchronised stretching/compression (*bXSync* = TRUE):

- The maximum position change is ± 32767 increments. The product of *nDfln_v*, task interval and the stretching/compression factor may not exceed this value!

Check the result with your maximum values.



Function library *LenzeCamControl.lib*

Function blocks

L_CamSetXAxisVelocity - Stretching and compression of the master value

Synchronising the x axis via touch probe



Note!

The following text describes the synchronisation via touch probe in an exemplified manner.

For the **9300 servo PLC** a synchronisation can be effected via the zero pulse of a connected encoder or via the digital inputs.

Apart from this, the touch probe correction can also be started manually via the input *bTpReceived*.

- Here, no interrupt is actuated. Therefore no value of the **DFIN_IO_DigitalFrequency** SB will be applied on the input *dnTpIncLastScan_p*, which would be required for an exact synchronisation.
- The synchronisation accuracy therefore depends on the interval of the task, in which the **L_CamSetXAxisVelocity** FB is integrated.

If *bTpReceived* is set to TRUE (by the **DFIN_IO_DigitalFrequency** SB or manually), and if *bTpEnable* is set to TRUE, the difference between the setpoint and actual value is calculated and is output on *dnXDifference_p*.

- The position of the touch probe encoder is defined via *dnTpPos_p* and has to be settled in the range 0 ... 2 x *dnActXLength_p*. The comparison value refers to the current value on the position output *dnXPosOut_p*.
- If incoming touch probe signals are to be masked out, you can deactivate the touch probe synchronisation in the respective position by setting *bTpEnable* to FALSE, e. g. by means of the **L_CamContact** cam group. This can be useful if synchronisation is to be carried out on the print marks or zero pulses of an encoder.
- For compensating the difference, the speed for picking up can be specified via *nTpMakeUpLeeway_v*. If no speed is preselected, the calculated difference is not reduced, i. e. the synchronisation remains inactive.
- For a smooth compensation of the difference, switch the **L_CamPosCtrlLin** FB as a ramp generator on *nTpMakeUpLeeway_v*:

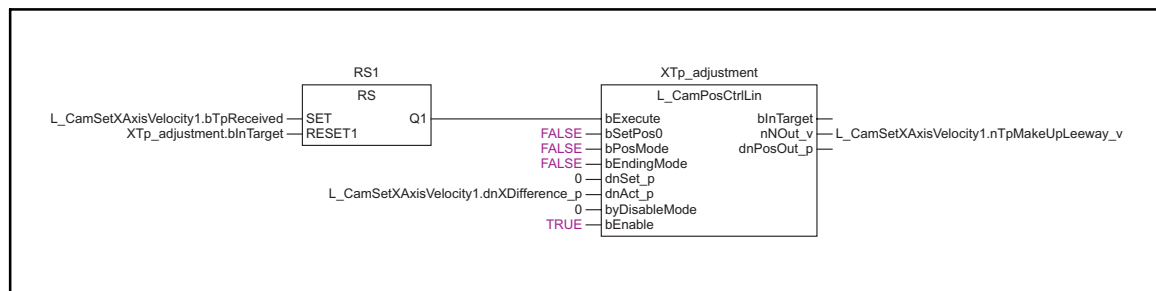


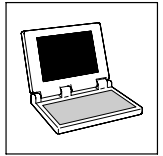
Fig. 2-39

"L_CamPosCtrlLin" function block as ramp generator



Tip!

Detailed information on the configuration of the touch probe (e. g. activation, edge selection and delay time) can be found in the Manual for the **9300 servo PLC**!



2.16 L_CamSetYAxisVelocity - Stretching and compressing the setpoint

This FB is used to change the set position of the drive "online" by stretching/compressing the motion profile and by adding an offset.

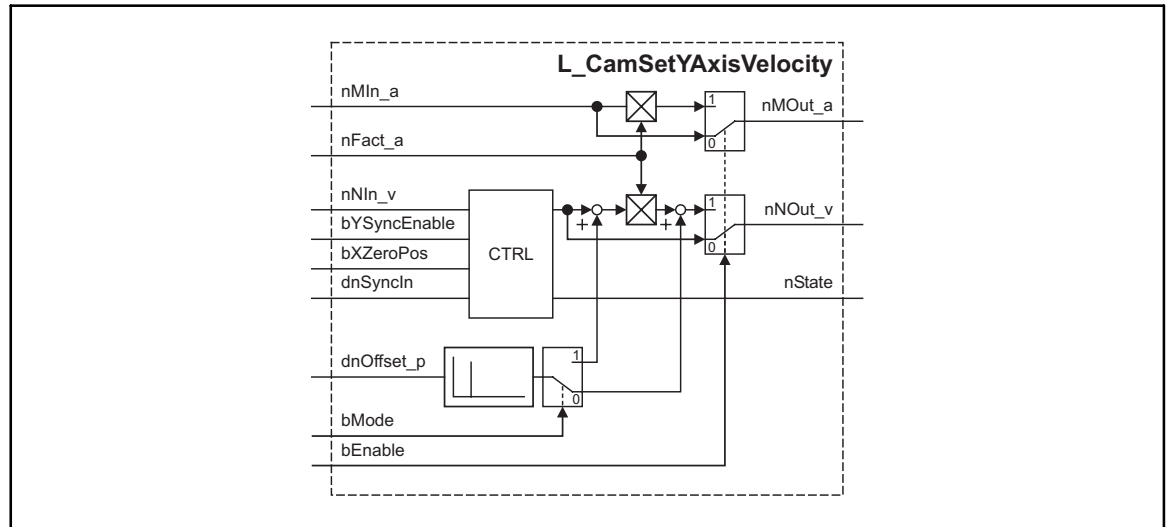
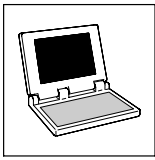


Fig. 2-40

"L_CamSetYAxisVelocity" function block

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
nMin_a	Integer	Analog	Torque setpoint 16384 ≡ 100 % - Connect this input to the output <i>nMSet_a</i> of the L_CamData FB.	
nFact_a	Integer	Analog	Stretching/compression factor for the motion profile. 16384 ≡ 100 % - For selection of negative values, inversion of the direction of rotation. - For <i>bMode</i>= TRUE, also stretching/compression factor for the offset <i>dnOffset_p</i>.	
			< 16384	Compression
			= 16384	No stretching/compression (factor = 100 %)
			> 16384	Stretching
nNIn_v	Integer	Velocity	Velocity - Connect this input to the output <i>nNOut_v</i> of the L_CamData FB. - Maximum selection: ±16384 [inc/ms] ≡ ±15000 [rpm]	
bYSyncEnable	BOOL	Position	Activating synchronised stretching/compression	
			FALSE	A new stretching/compression factor is accepted immediately.
			TRUE	A new stretching/compression factor is only accepted in the zero crossing.
bXZeroPos	BOOL	Position	Control signal for synchronised stretching/compression Connect this input to the output <i>bXZeroPos</i> of the L_CamData FB.	
			FALSE ↗ TRUE	Acceptance of the stretching/compression factor if the drive is not in standstill.
dnSyncln	Double integer	-	Correction value for synchronised stretching/compression Connect this input to the output <i>dnYSync</i> of the L_CamData FB.	
dnOffset_p	Double integer	Position	Superimposed motion/position to the motion profile For <i>bMode</i> = TRUE, this value is output by multiplication by the <i>nFact_a</i> stretching/compression factor at the output.	
bMode	BOOL	Binary	Applying stretching/compression also to the offset	
			FALSE	<i>dnOffset_p</i> is added to the output in an unchanged manner.



Function library *LenzeCamControl.lib*

Function blocks

L_CamSetYAxisVelocity - Stretching and compressing the setpoint

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
			TRUE	<i>dnOffset_p</i> is added to the speed signal and is output by multiplication by the <i>nFact_a</i> stretching/compression factor at the output <i>nNOut_v</i> .
bEnable	BOOL	Binary	Activating the stretching/compression function	
			FALSE	The stretching/compression function is deactivated. <i>nMin_a</i> is directly output on <i>nMOut_a</i>. - The sum of <i>nNIn_v</i> and <i>dnSyncln</i> is directly output on <i>nNOut_v</i>. - The <i>dnOffset_p</i> offset is not considered.
			TRUE	The stretching/compression function is activated.

Outputs ⇒ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Value/meaning	
nMOut_a	Integer	Analog	Torque 16384 ≡ 100 % - Depending on <i>bMode</i> and <i>nFact_a</i>. - Connect this output to the input <i>nMSet_a</i> of the L_CamControl FB.	
nNOut_v	Integer	Velocity	Speed setpoint Connect this output to the input <i>nMainIn_v</i> of the L_CamControl FB.	
nState	Integer	Number	Status	
			20	<i>nNOut_v</i> speed output limitation active.
			0	OK, no error.

Activating the stretching/compression function

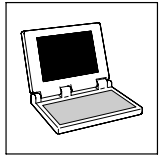
If *bEnable* is set to TRUE, the stretching/compression function as well as the selection of an offset is activated. In case of FALSE, the torque precontrol and the velocity are directly looped through to the respective outputs.

Stretching & compression

The motion profile can be changed online via the stretching/compression factor.

Example for the travel change of a mechanical cam:	Stretching/compression factor		Stretching/compression	Reversal of direction
	Selection on <i>nFact_a</i>	≡ Value in percent		
	+16384	+100 %	No	No
	-16384	-100 %	No	Yes, in Y position
	> 16384	>100 %	Stretching	No
	< 16384	< 100 %	Compression	No

- The stretching/compression factor may only be changed if the motor is in standstill and the motion profile is in zero position.
- If a change-over during the motion is required, use the "Synchronised stretching/compression" function for this purpose.



Examples

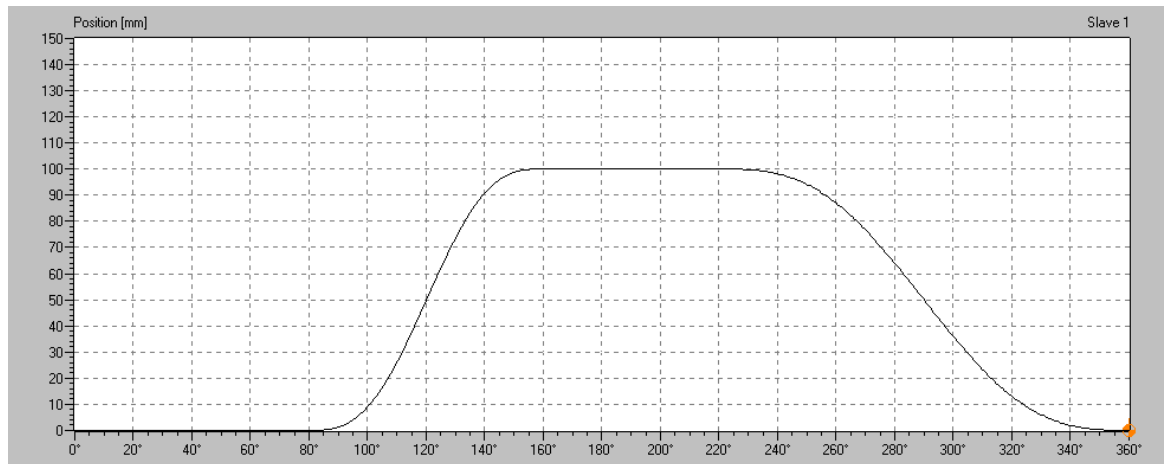


Fig. 2-41

$nFact_a = 16384$ (100 % of 100 mm stroke)

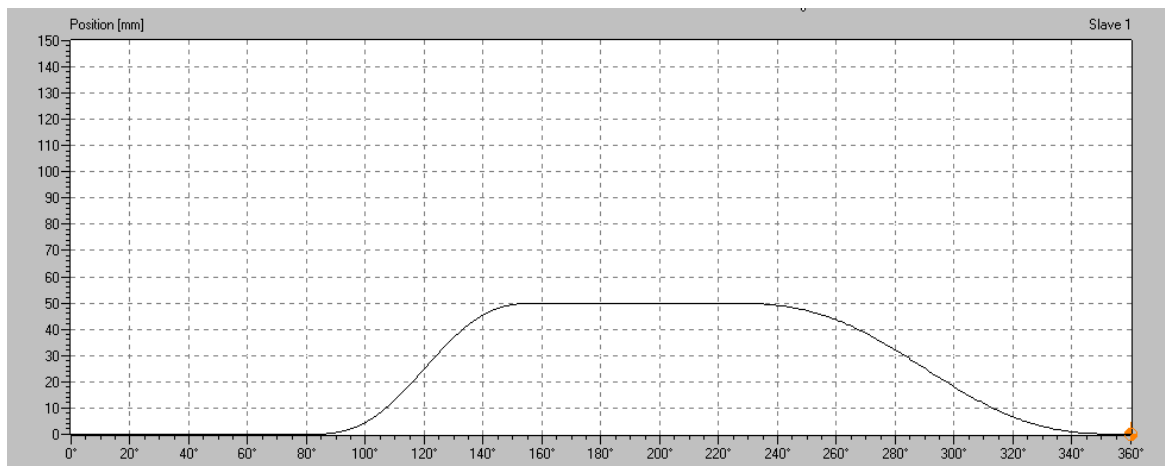


Fig. 2-42

Compression: $nFact_a = 8192$ (50 % of 100 mm stroke)

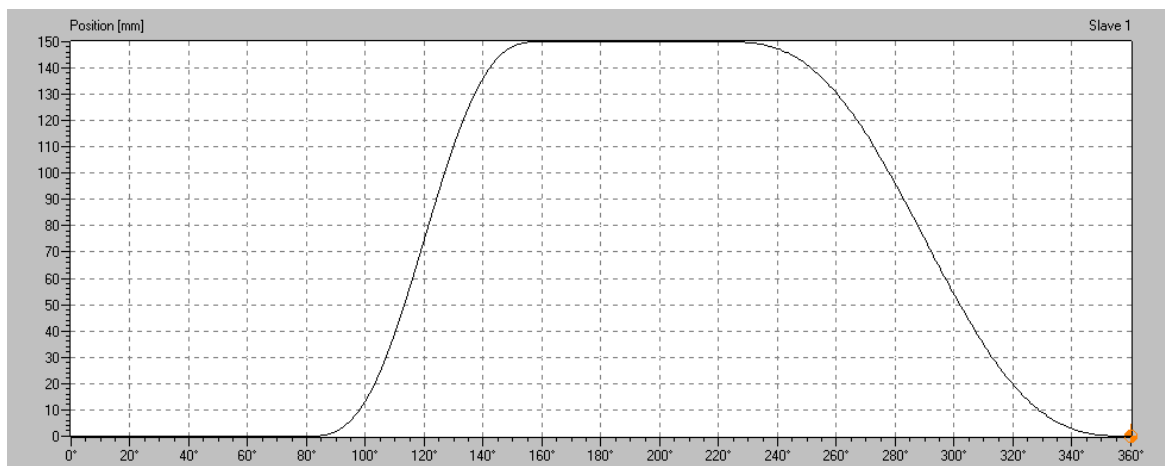
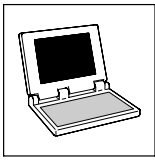


Fig. 2-43

Stretching: $nFact_a = 24576$ (150 % of 100 mm stroke)



Function library *LenzeCamControl.lib*

Function blocks

L_CamSetYAxisVelocity - Stretching and compressing the setpoint

Synchronised stretching/compression

A new stretching/compression factor on the input *nFact_a* is accepted immediately and is implemented. If the motion profile up to this time has no break (gradient ≤ 0), the new factorisation results in the loss of the y zero position or in the increment loss with regard to a feed profile.

Synchronised stretching/compression therefore is only required if the stretching/compression of the profile is to be changed during the processing of products. A new *nFact_a* stretching/compression factor is only accepted in the zero crossing of the cam.

The synchronisation is effected in connection with the **L_CamData** FB in the cam zero crossing of the profile, whereby the following requirements have to be met:

- For the **L_CamData** FB, *bYSyncEnable* is also set to TRUE, and thus the synchronisation is activated.
- The *dnSyncIn* input is connected to the *dnYSync* output of the FB **L_CamData**.
- The *bSyncIn* input is connected to the *bXZeroPos* output of the FB **L_CamData**.



Note!

Only activate the synchronised stretching/compression by setting *bYSyncEnable* to TRUE if the drive is already positioned on the cam.

Offset

Via *dnOffset_p*, you can define an offset to shift the Y position:

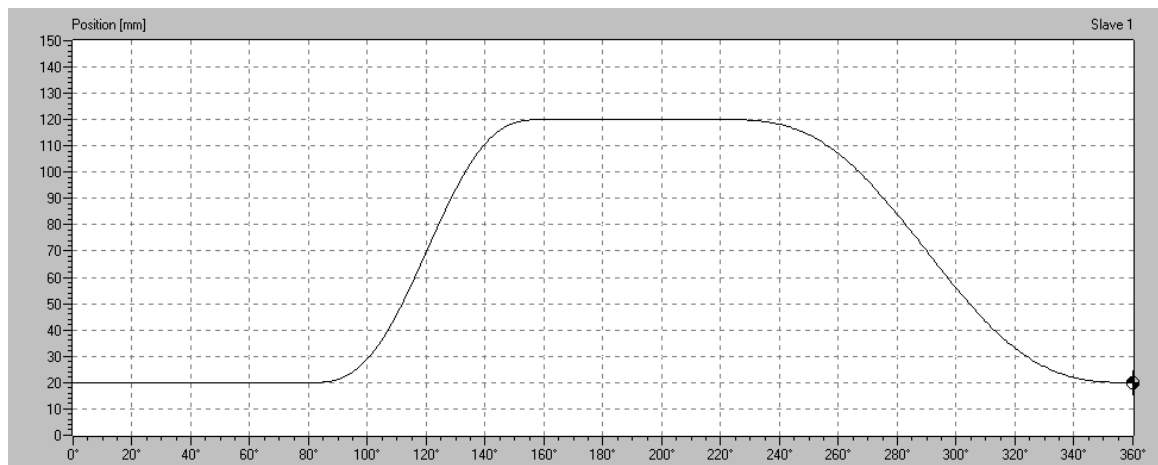
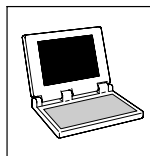


Fig. 2-44

Shifting the profile by 20 mm in Y position by defining an offset.

If *bMode* is set to TRUE, the *dnOffset_p* offset is multiplied by the *nFact_a* stretching/compression factor and is displayed at the output.



2.17 L_CamVMasterPosition - Virtual master with positioning function

Use this FB for the following functions:

- Generating and positioning the vertical shaft (x axis)
- Normal operation or cyclic operation with stop via adjustable ramp in desired position
- Manual homing (inching)
- Handwheel function via incremental encoders
- Emergency stop

Connection between machine clock pulse and virtual master

Due to the gearbox ratio and the difference between feed constant and x profile length, the required number of cycles must be converted to the vertical shaft of the master.

The master speed to be selected in the virtual master is calculated as follows:

$$\text{master speed [rpm]} = \text{number of cycles [1/min]} \cdot \frac{\text{x profile length [mm]}}{\text{feed constant [mm]}} \cdot \frac{\text{gearbox numerator}}{\text{gearbox denominator}}$$

x profile length/feed constant ratio	
$\frac{\text{x profile length [mm]}}{\text{feed constant [mm]}}$	If x profile length and feed constant are not identical, one revolution of the feed roll does not correspond to a complete clock pulse. A corresponding x profile length (ΔX)/feed constant ($\pi \cdot d$) ratio must be considered.

Gearbox ratio	
$\frac{\text{gearbox numerator}}{\text{gearbox denominator}}$	If the motor of the master drive rotates by the gearbox ratio faster than the feed roller, a corresponding ratio must be considered.

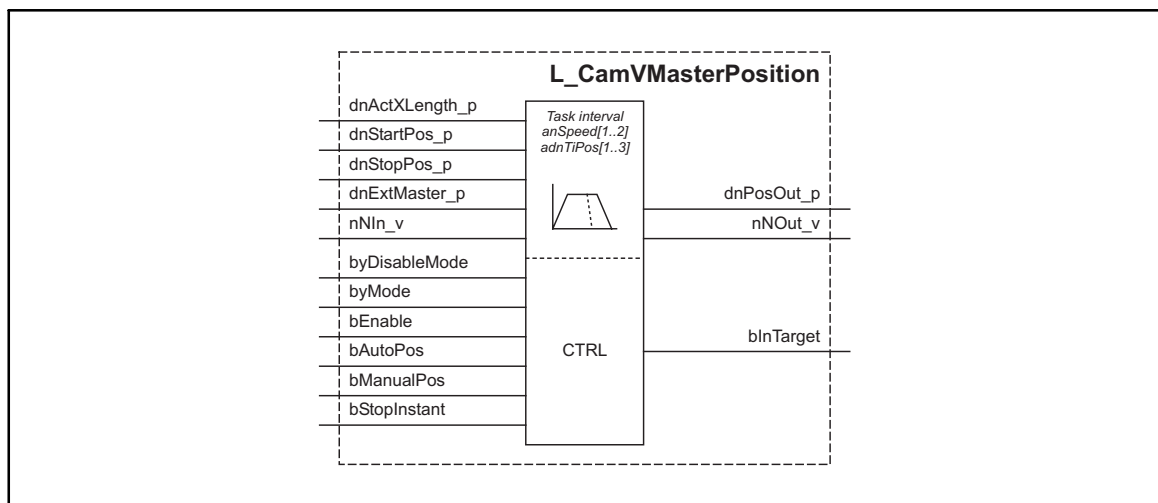
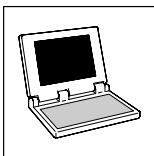


Fig. 2-45

"L_CamVMasterPosition" function block

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
dnActXLength_p	Double integer	Position	Length of the x axis in increments
dnStartPos_p	Double integer	Position	Starting position
dnStopPos_p	Double integer	Position	Stop position
dnExtMaster_p	Double integer	Position	Position of an external master (e. g. absolute value encoder) • Limitation: 0 ... dnActXLength_p

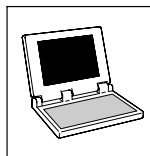


Function library *LenzeCamControl.lib*

Function blocks

L_CamVMasterPosition - Virtual master with positioning function

⇒ Inputs (Variable type: VAR_INPUT)													
Identifiers	Data type	Signal type	Information/possible settings										
nNln_v	Integer	Velocity	Digital frequency setpoint										
byDisableMode	Byte	Number	Behaviour for disenabled function block (<i>bEnable</i> = FALSE) <table><tr><td>0</td><td>Speed output inactive, position is maintained.</td></tr><tr><td>1</td><td>Speed output inactive, position is set to starting position.</td></tr><tr><td>2</td><td>Speed output active, handwheel function (via incremental encoders on <i>nNln_v</i>)</td></tr><tr><td>3</td><td>Speed output active, position is set to zero.</td></tr><tr><td>4</td><td>Speed output active, loop through of the position on <i>dnExtMaster_p</i>.</td></tr></table>	0	Speed output inactive, position is maintained.	1	Speed output inactive, position is set to starting position.	2	Speed output active, handwheel function (via incremental encoders on <i>nNln_v</i>)	3	Speed output active, position is set to zero.	4	Speed output active, loop through of the position on <i>dnExtMaster_p</i> .
0	Speed output inactive, position is maintained.												
1	Speed output inactive, position is set to starting position.												
2	Speed output active, handwheel function (via incremental encoders on <i>nNln_v</i>)												
3	Speed output active, position is set to zero.												
4	Speed output active, loop through of the position on <i>dnExtMaster_p</i> .												
byMode	Byte	Number	Selection of functions <table><tr><td>0</td><td>Normal operation - Activation by means of <i>bAutoPos</i> = TRUE </td></tr><tr><td>1</td><td>Cyclic operation - Activation by means of a FALSE-TRUE edge on <i>bAutoPos</i> </td></tr><tr><td>2</td><td>Manual operation in positive direction (CW) - Activation by means of <i>bManualPos</i> = TRUE </td></tr><tr><td>3</td><td>Manual operation in negative direction (CCW) - Activation by means of <i>bManualPos</i> = TRUE </td></tr><tr><td>4</td><td>Positioning to <i>dnExtMaster_p</i> - Activation by means of a FALSE-TRUE edge on <i>bAutoPos</i> </td></tr></table>	0	Normal operation Activation by means of <i>bAutoPos</i> = TRUE	1	Cyclic operation Activation by means of a FALSE-TRUE edge on <i>bAutoPos</i>	2	Manual operation in positive direction (CW) Activation by means of <i>bManualPos</i> = TRUE	3	Manual operation in negative direction (CCW) Activation by means of <i>bManualPos</i> = TRUE	4	Positioning to <i>dnExtMaster_p</i> Activation by means of a FALSE-TRUE edge on <i>bAutoPos</i>
0	Normal operation Activation by means of <i>bAutoPos</i> = TRUE												
1	Cyclic operation Activation by means of a FALSE-TRUE edge on <i>bAutoPos</i>												
2	Manual operation in positive direction (CW) Activation by means of <i>bManualPos</i> = TRUE												
3	Manual operation in negative direction (CCW) Activation by means of <i>bManualPos</i> = TRUE												
4	Positioning to <i>dnExtMaster_p</i> Activation by means of a FALSE-TRUE edge on <i>bAutoPos</i>												
bEnable	Bool	Binary	Function block enable <table><tr><td>FALSE</td><td>Function block inhibited, for behaviour see <i>byDisableMode</i></td></tr><tr><td>TRUE</td><td>Function block enabled</td></tr></table>	FALSE	Function block inhibited, for behaviour see <i>byDisableMode</i>	TRUE	Function block enabled						
FALSE	Function block inhibited, for behaviour see <i>byDisableMode</i>												
TRUE	Function block enabled												
bAutoPos	Bool	Binary	Activating automatic operation with stop in desired position <table><tr><td>TRUE</td><td>In normal operation (<i>byMode</i>= 0): start normal operation - The vertical shaft accelerates to set speed via the ramp. </td></tr><tr><td>TRUE ↘ FALSE</td><td>In normal operation (<i>byMode</i>= 0): - The vertical shaft decelerates to stop position via the ramp (automatic stop process). </td></tr><tr><td>FALSE ↗ TRUE</td><td>In cyclic operation (<i>byMode</i>= 1): stop normal operation - The vertical shaft accelerates to set speed via the ramp. - The automatic stop process is activated immediately. </td></tr></table>	TRUE	In normal operation (<i>byMode</i> = 0): start normal operation The vertical shaft accelerates to set speed via the ramp.	TRUE ↘ FALSE	In normal operation (<i>byMode</i> = 0): The vertical shaft decelerates to stop position via the ramp (automatic stop process).	FALSE ↗ TRUE	In cyclic operation (<i>byMode</i> = 1): stop normal operation - The vertical shaft accelerates to set speed via the ramp. - The automatic stop process is activated immediately.				
TRUE	In normal operation (<i>byMode</i> = 0): start normal operation The vertical shaft accelerates to set speed via the ramp.												
TRUE ↘ FALSE	In normal operation (<i>byMode</i> = 0): The vertical shaft decelerates to stop position via the ramp (automatic stop process).												
FALSE ↗ TRUE	In cyclic operation (<i>byMode</i> = 1): stop normal operation - The vertical shaft accelerates to set speed via the ramp. - The automatic stop process is activated immediately.												
bManualPos	Bool	Binary	Activate manual homing - Without automatic positioning - This input has a higher priority than <i>bAutoPos</i> (normal operation/cyclic operation) <table><tr><td>FALSE</td><td>No manual control</td></tr><tr><td>TRUE</td><td>Manual homing (inching) - If the direction of travel has not been explicitly defined via <i>byMode</i> = 2/3, traversing is effected in positive direction (Cw). </td></tr></table>	FALSE	No manual control	TRUE	Manual homing (inching) If the direction of travel has not been explicitly defined via <i>byMode</i> = 2/3, traversing is effected in positive direction (Cw).						
FALSE	No manual control												
TRUE	Manual homing (inching) If the direction of travel has not been explicitly defined via <i>byMode</i> = 2/3, traversing is effected in positive direction (Cw).												
bStopInstant	Bool	Binary	Stop function: forced stop via separate deceleration ramp This input has highest priority <table><tr><td>TRUE</td><td>The current traversing function is interrupted and the drive is brought into standstill via the set deceleration ramp.</td></tr></table>	TRUE	The current traversing function is interrupted and the drive is brought into standstill via the set deceleration ramp.								
TRUE	The current traversing function is interrupted and the drive is brought into standstill via the set deceleration ramp.												



Outputs ⇨ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Value/meaning	
dnPosOut_p	Double integer	Position	Position (within the limits of the X axis)	
nNOut_v	Integer	Velocity	Digital frequency	
			Note: The phase difference signal considers the incremental cycle skip from $x_{\max}$ to 0 at the end/start of a cycle, but not a possible incremental decimal fraction in the cycle. Therefore do not use this output signal if the cycle cannot be represented incrementally as an integer!	
blnTarget	Bool	Binary	Status signal: current position <i>dnPosOut_p</i> has reached the stop function <i>dnStopPos_p</i> .	
			• Signal is available until a new traversing action is selected.	
			<table><tr><td>FALSE</td><td>The current position <i>dnPosOut_p</i> has not yet left the stop position once (after initialisation) or is not placed on the <i>dnStopPos_p</i> stop position.</td></tr><tr><td>TRUE</td><td>The current position <i>dnPosOut_p</i> has approached the stop function <i>dnStopPos_p</i>.</td></tr></table>	FALSE
FALSE	The current position <i>dnPosOut_p</i> has not yet left the stop position once (after initialisation) or is not placed on the <i>dnStopPos_p</i> stop position.			
TRUE	The current position <i>dnPosOut_p</i> has approached the stop function <i>dnStopPos_p</i> .			
			Note: The output is not set to TRUE if the virtual master is in the stop position after initialisation of the target system, but has not yet left this position.	

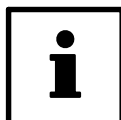
Parameters (Variable type: VAR_CONSTANT_RETAIN)																											
Identifiers Code*	Data type	Signal type	Information																								
			Possible settings Presetting																								
adnTiPos[1..3]	Double-integer array	-	Ramps for acceleration/deceleration (ref. to 4000 rpm) <table border="1"> <tr> <td colspan="4"><i>adnTiPos[1]</i> = deceleration ramp for stop function</td></tr> <tr> <td>C1452/1</td><td>{0.001 unit}</td><td>130.000</td><td>1.000 s</td></tr> <tr> <td colspan="4"><i>adnTiPos[2]</i> = acceleration/deceleration ramp for normal operation</td></tr> <tr> <td>C1452/2</td><td>{0.001 unit}</td><td>130.000</td><td>1.000 s</td></tr> <tr> <td colspan="4"><i>adnTiPos[3]</i> = acceleration/deceleration ramp for manual control</td></tr> <tr> <td>C1452/3</td><td>{0.001 unit}</td><td>130.000</td><td>1.000 s</td></tr> </table>	<i>adnTiPos[1]</i> = deceleration ramp for stop function				C1452/1	{0.001 unit}	130.000	1.000 s	<i>adnTiPos[2]</i> = acceleration/deceleration ramp for normal operation				C1452/2	{0.001 unit}	130.000	1.000 s	<i>adnTiPos[3]</i> = acceleration/deceleration ramp for manual control				C1452/3	{0.001 unit}	130.000	1.000 s
<i>adnTiPos[1]</i> = deceleration ramp for stop function																											
C1452/1	{0.001 unit}	130.000	1.000 s																								
<i>adnTiPos[2]</i> = acceleration/deceleration ramp for normal operation																											
C1452/2	{0.001 unit}	130.000	1.000 s																								
<i>adnTiPos[3]</i> = acceleration/deceleration ramp for manual control																											
C1452/3	{0.001 unit}	130.000	1.000 s																								
anSpeed[1..2]	Array of integers	-	Manual homing speed <table border="1"> <tr> <td colspan="4"><i>anSpeed[1]</i> = speed for manual homing in positive direction (CW)</td></tr> <tr> <td>C1451/1</td><td>{1 rpm}</td><td>15000</td><td>10 rpm</td></tr> <tr> <td colspan="4"><i>anSpeed[2]</i> = speed for manual homing in negative direction (CCW)</td></tr> <tr> <td>C1451/2</td><td>{1 rpm}</td><td>15000</td><td>10 rpm</td></tr> </table>	<i>anSpeed[1]</i> = speed for manual homing in positive direction (CW)				C1451/1	{1 rpm}	15000	10 rpm	<i>anSpeed[2]</i> = speed for manual homing in negative direction (CCW)				C1451/2	{1 rpm}	15000	10 rpm								
<i>anSpeed[1]</i> = speed for manual homing in positive direction (CW)																											
C1451/1	{1 rpm}	15000	10 rpm																								
<i>anSpeed[2]</i> = speed for manual homing in negative direction (CCW)																											
C1451/2	{1 rpm}	15000	10 rpm																								

* Code of first instance

Generating and positioning the vertical shaft (x axis)

The setpoint axis (x axis) is generated by means of the length defined on *dnActXLength_p*.

- The x axis is purely positive, negative values are not permissible (internal limitation).
- By selecting a negative speed on the input *nNIn_v*, the x axis can be traversed backwards.
- A non-zero starting value can be defined via the input *dnStartPos_p*. This starting value has to be smaller than the length of the x axis. A change in value only is effective for *byDisableMode* = 1 and for inhibited FB.

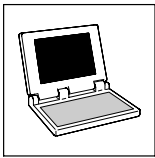


Note!

When changing between profiles with different x axis lengths, the master angle must only rotate in positive direction.

Activating the traversing function

The traversing function of the FB set via *byMode* can only be started if the FB is enabled via *bEnable* = TRUE.



Function library *LenzeCamControl.lib*

Function blocks

L_CamVMasterPosition - Virtual master with positioning function

Normal operation (with automatic stop process)

Normal operation is selected by means of $byMode = 0$.

- If $bAutoPos$ is set to TRUE, the vertical shaft accelerates to the set speed (nN/n_v via the set ramp $adnTiPos[2]$).
- The profile is executed as long as $bAutoPos$ remains set to TRUE.
- If $bAutoPos$ is reset to FALSE, the vertical shaft is braked to the stop position (automatic stop process) via the set ramp $adnTiPos[2]$.
- A normal operation interrupted by manual control has to be restarted by means of a FALSE-TRUE edge on $bAutoPos$.

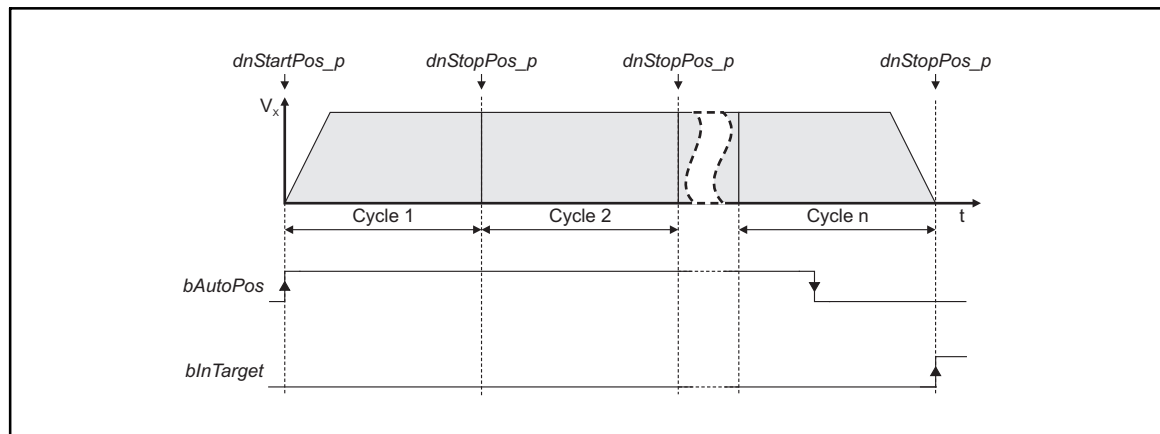
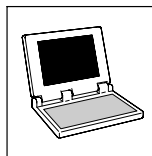


Fig. 2-46

Normal operation (continual execution of the profile)



Cyclic operation (with automatic stop process)

Cyclic operation is selected by means of $byMode = 1$.

- If $bAutoPos$ receives a FALSE-TRUE edge, the vertical shaft accelerates to the set speed ($nNIn_v$ via the set ramp $adnTiPos[2]$) and is automatically braked to the stop position (automatic stop process).
- In cyclic operation, the profile is executed once at a maximum (current position to stop position).
- If the acceleration/deceleration ramps are set too low, the setpoint velocity cannot be reached.
- For a further cycle, a new FALSE-TRUE edge on $bAutoPos$ is required.
- A cyclic operation interrupted by manual control also has to be restarted by means of a new FALSE-TRUE edge on $bAutoPos$.

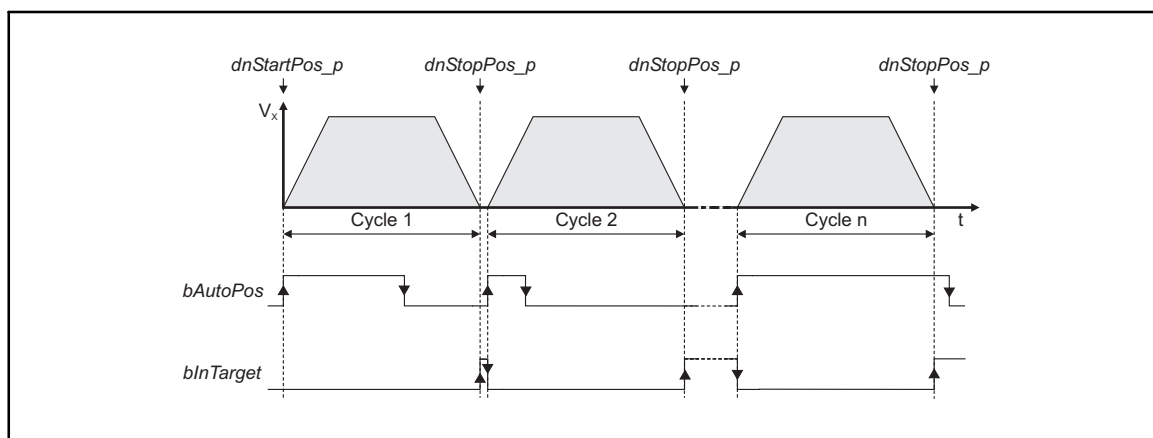


Fig. 2-47

Cyclic operation (one-time execution of the profile)

Manual homing (inching)

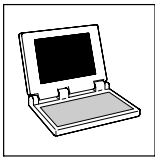
Via $bManualPos = TRUE$, manual homing is activated.

- Manual homing has a higher priority than normal/cyclic operation.
- Use $adnTiPos[1..3]$ to set individual acceleration and deceleration ramps for manual homing.
- If the traversing direction has not been explicitly defined via $byMode = 2/3$, traversing is effected in positive direction (Cw).

Handwheel function via incremental encoders

The handwheel function can only be carried out if the traversing function is deactivated via $bEnable = FALSE$ and $byDisableMode = 2$ is set.

- The encoder signal specified on $nNIn_v$ is integrated and output within the limits of the cycle on the $dnPosOut_p$ position output.



Function library *LenzeCamControl.lib*

Function blocks

L_CamVMasterPosition - Virtual master with positioning function

Forced stop

By setting *bStopInstant*= TRUE, an immediate stop can be activated.

- The drive is brought into standstill immediately via the set deceleration ramp *adnTiPos[1]*, therefore no positioning to the target position is effected.



Note!

The function can only be carried out if the FB is enabled (*bEnable* = TRUE) and has a higher priority than all other traversing functions.

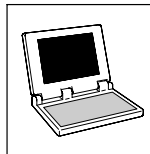
- A traversing function interrupted by a forced stop has to be restarted by the corresponding signal.

External master position

By setting *bEnable* to FALSE and *byDisableMode* = 4, an external master position in the range of 0 to the maximum cycle (*dnActXLength_p* can be defined via *dnExtMaster_p*).

Beforehand, though, the actual position *dnPosOut_p* has to be brought to this external master position.

- For this purpose, use the traversing mode *byMode* = 4, which is started by means of a FALSE-TRUE edge on *bAutoPos*.



2.18 L_CamVMasterVelocity - Virtual master

Use this FB for the following functions:

- Generation of a virtual master frequency
- Manual homing (inching)
- Stop along adjustable ramp

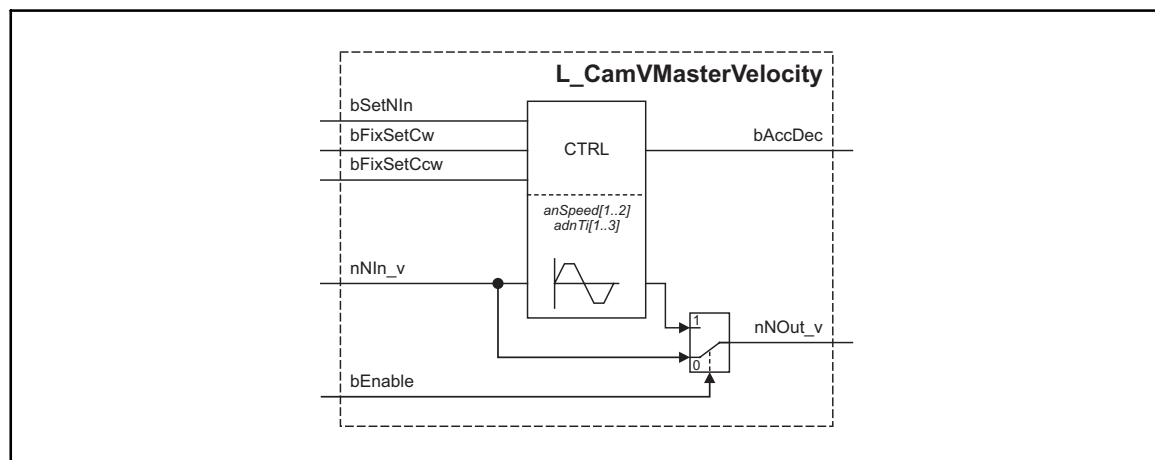
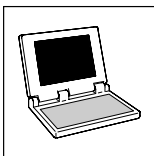


Fig. 2-48

Function block "L_CamVMasterVelocity"

⇒ Inputs (Variable type: VAR_INPUT)				
Identifier	Data type	Signal type	Info/possible settings	
bEnable	BOOL	Binary	Activation/deactivation of ramp function generator	
			FALSE	Ramp function generator is deactivated The master frequency setpoint is directly connected to the master frequency output.
			TRUE	Ramp function generator is activated If the inputs <i>bSetNIn</i>, <i>bFixSetCw</i> and <i>bFixSetCcw</i> are set to FALSE, the drive will be decelerated to standstill along the deceleration ramp <i>adnTi[1]</i> (stop function).
bSetNIn	BOOL	Binary	Ramp function generator target speed	
			FALSE	Target speed = 0 rpm
			TRUE	Target speed = Selection at <i>nNIn_v</i>
bFixSetCw	BOOL	Binary	Manual homing This input has priority over <i>bFixSetCcw</i>.	
			TRUE	Manual homing in positive direction (CW)
bFixSetCcw	BOOL	Binary	Manual homing	
			TRUE	Manual homing in negative direction (CCW)
nNIn_v	Integer	Velocity	Master frequency setpoint	



Function library LenzeCamControl.lib

Function blocks

L_CamVMasterVelocity - Virtual master

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
bAccDec	BOOL	Binary	Ramp function generator status
			TRUE Ramp function generator function is active, i.e. the drive is being accelerated or decelerated.
nNOut_v	Integer	Velocity	Master frequency

Parameter (Variable type: VAR_CONSTANT_RETAIN)			
Identifier	Data type	Signal type	Info
Code*			Possible settings Default setting
adnTi[1..3]	Array of Double Integer	-	Ramps for acceleration/deceleration (ref. to 4000 rpm)
			adnTi[1] = Deceleration ramp for stop function
C1462/1			0.010 {0.001 s} 130.000 1.000 s
			adnTi[2] = Acceleration/deceleration ramp for normal operation
C1462/2			0.010 {0.001 s} 130.000 1.000 s
			adnTi[3] = Acceleration/deceleration ramp for manual homing
C1462/3			0.010 {0.001 s} 130.000 1.000 s
anSpeed[1..2]	Array of integers	-	Manual homing speed
			anSpeed[1] = Speed for manual homing in positive direction (CW)
C1461/1			0 {1 rpm} 15000 200 rpm
			anSpeed[2] = Speed for manual homing in negative direction (CCW)
C1461/2			0 {1 rpm} 15000 200 rpm

* Code of first instance

Generation of a virtual master frequency

The cam profiler requires a master system (speed or position of X axis) for operation. If the cam profiler is not part of a network of drives and if it is not connected to an encoder, this FB will generate a virtual master frequency.

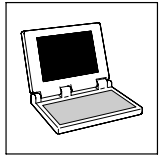
- The speed setpoint is selected through *nNIn_v* in inc/ms.

Ramp function generator enable

If *bEnable* is TRUE, the ramp function generator will be enabled, if it is FALSE, the speed setpoint will be directly passed on to the master frequency output.

Normal operation

If *bSetNIn* is TRUE, the (enabled) ramp function generator follows the ramp *adnTi[1]* until it reaches the speed setpoint selected under *nNIn_v*.



Manual homing (inching)

Via *bFixSetCw* and *bFixSetCcw* manual homing is controlled according to the following table:

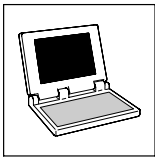
<i>bFixSetCw</i>	<i>bFixSetCcw</i>	
0	0	Manual homing deactivated • <i>bSetNln</i> = FALSE: master frequency <i>nNout_v</i> = 0 rpm • <i>bSetNln</i> = TRUE: master frequency <i>nNout_v</i> = <i>nNln_v</i>
0	1	Manual homing in negative direction (CCW) at speed <i>anSpeed</i> [2].
1	0 or 1	Manual homing in positive direction (CW) at speed <i>anSpeed</i> [1].

- Manual homing has priority over normal operation.
- Use *anSpeed*[1..2] to set individual acceleration and deceleration ramps for manual homing.

Stop function

If the inputs *bSetNln*, *bFixSetCw* and *bFixSetCcw* are all set FALSE and the ramp function generator is enabled (*bEnable* = TRUE), the drive will be decelerated to standstill along the deceleration ramp *adnTi*[1].

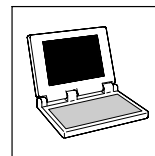
- The stop function has priority over other functions.



Function library LenzeCamControl.lib

Function blocks

L_CamVMasterVelocity - Virtual master

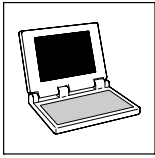


3 Function blocks LenzeCamControl2V02xx.lib

The function blocks for the "Cam profiler" technology function are divided into the two function libraries **LenzeCamControl1V02xx.lib** and **LenzeCamControl2V02xx.lib**.



In this chapter you'll be provided with information on the function blocks which are contained in the **LenzeCamControl2V02xx.lib** function library.



Function library *LenzeCamControl.lib*

Function blocks

L_CamAddVelocity - speed addition

3.1 L_CamAddVelocity - speed addition

Use this FB to add and subtract speed signals.

The output is limited to ± 16383 inc/ms. The internal "overflow buffer" prevents a position loss when the limits are exceeded.

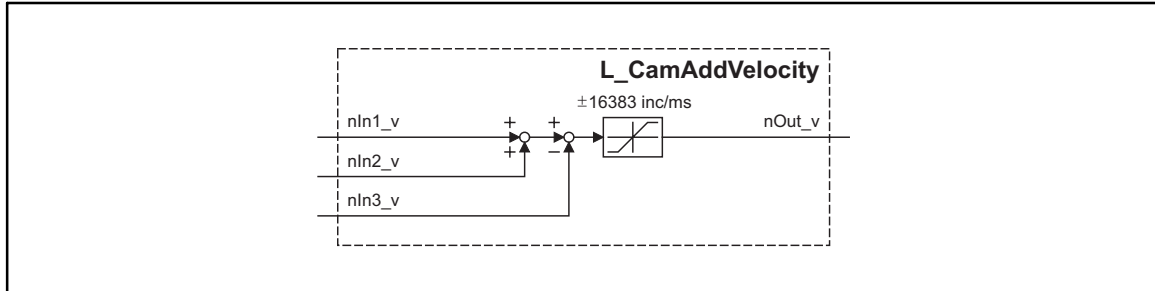
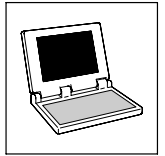


Fig. 3-1 Function block "L_CamAddVelocity"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
nIn1_v	Integer	Velocity	Addition input
nIn2_v	Integer	Velocity	Addition input
nIn3_v	Integer	Velocity	Subtracting input

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
nOut_v	Integer	Velocity	Output - The output is limited to ± 16383 inc/ms (without position loss). ± 16383 inc/ms ≈ 14999 rpm



3.2 L_CamAdjustDfln - master frequency adaptation

Use this FB to adapt the incremental encoder used.

- The FB gets its input signal directly from the system block **DFIN**.
- The adaptation is done via the encoder constant $wEncConst$ and the stretching factor x/y .

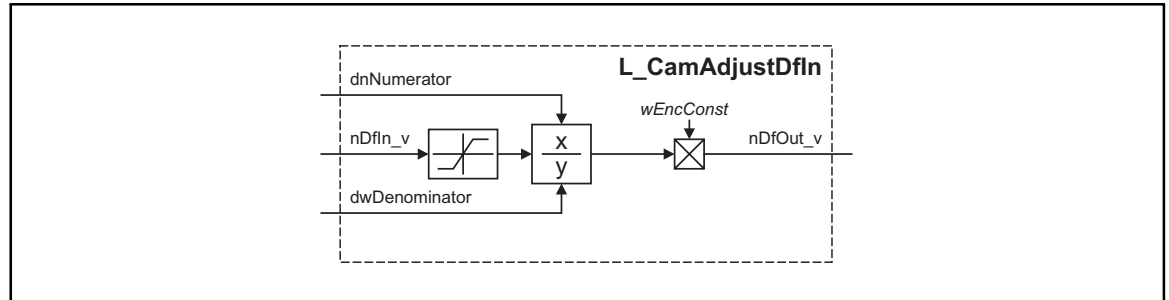


Fig. 3-2

Function block "L_Ca mAd justDfln"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
nDfln_v	Integer	Velocity	Input for the speed signal of the SB DFIN in [inc/ms] • Limitation: 2147 [inc/ms]
dnNumerator	Double Integer	Position	Stretching factor (numerator) • Limitation: -100000 ... +100000
dwDenominator	Double Word	Position	Stretching factor (denominator) • Limitation: 1 ... 2 ³¹

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
nDfOut_v	Integer	Velocity	Adapted velocity in [inc/ms] • 16384 [inc/ms] ≙ 15000 rpm

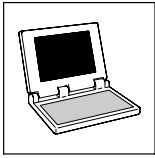
Parameter (Variable type: VAR_CONSTANT_RETAIN)				
Identifier	Data type	Signal type	Info	
			Possible settings	Default setting
wEncConst	Word	Number	Encoder constant at nDfln_v	
C1480			10 {1}	32767 16384

* Code of first instance



Note!

Do not use this FB if a touch probe is carried out for the master value. (See FB **L_CamSetXAxisVelocity** (2-58))



Function library *LenzeCamControl.lib*

Function blocks

L_CamDifferentiate - Converter/differentiator

3.3 L_CamDifferentiate - Converter/differentiator

This FB serves to convert a phase change at the input *dnIn_p* into speed.

The overflow buffer of the FB ensures that positions and increments do not get lost if the speed output is limited by a too big position change at the input.

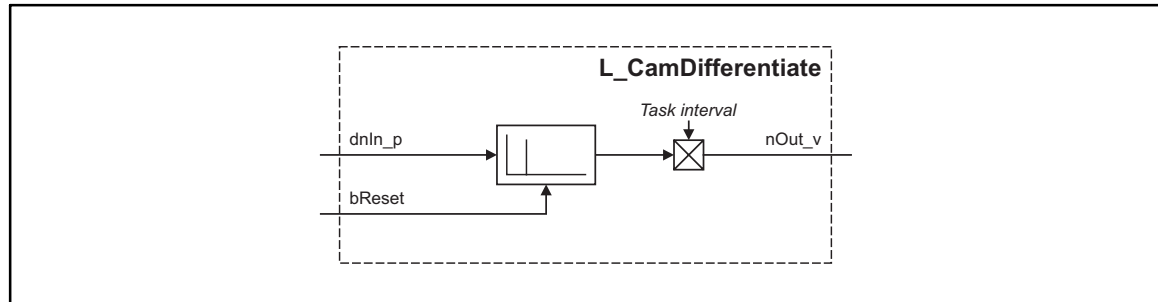


Fig. 3-3 Function block "L_CamDifferentiate"

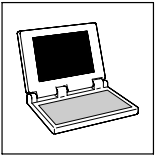
⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
dnIn_p	Double Integer	Position	Phase signal to be converted into speed.
bReset	BOOL	Binary	Reset output
			FALSE The speed converted from a phase signal is assigned to the output.
			TRUE The output is reset, 0 rpm are output as speed.

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
nOut_v	Integer	Velocity	Speed 16384 ≙ 15000 rpm - max. output value $\pm 32767 \equiv \pm 30000$ rpm



Tip!

For differentiating the master value (X axis) in the cam applications use the FB **L_CamExtrapolate** (with the setting *bEnable* = FALSE).



3.4 L_CamEditProfileData - Online change of profile points

Use this FB to change profile or acceleration points of a motion profile and to check the new points in x direction.



Note!

This FB cannot be used to generate motion profiles (*.lc9). You can only change existing motion profiles created with **CamDesigner** in the absolute data model.

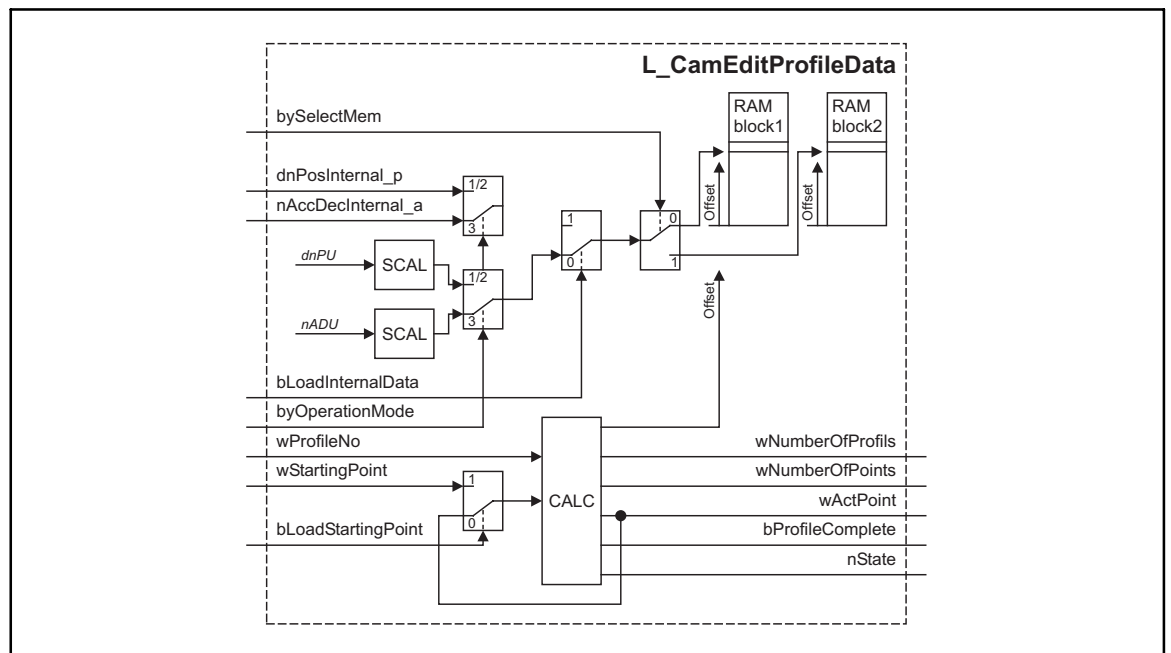
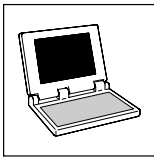


Fig. 3-4 "L_CamEditProfileData" function block

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
bySelectMem	Byte	Number	Selection of the *.lc9 file.	
			1	RAM block 1 (foreground data field, standard)
			2	RAM block 2 (background data field)
dnPosInternal_p	Double integer	Position	Positions in [inc] - Define the new x or y position value for the current point according to the selected operating mode of these input variables. - X positions must be positive. - The position is accepted with a FALSE-TRUE edge on <i>bLoadInternalData</i> for the current point <i>wActPoint</i>.	
nAccDeclInternal_a	Integer	Analog	Selection of the acceleration (analog values) ±16384 ≡ ±100 %	
bLoadInternalData	BOOL	Binary	Acceptance of the position at <i>dnPosInternal_p</i> or the acceleration at <i>nAccDeclInternal_a</i> for the current point.	
			TRUE ⇒ FALSE	Acceptance of the value at <i>dnPosInternal_p</i> or <i>nAccDeclInternal_a</i> and automatic transition to the next point (auto-increment function).



Function library LenzeCamControl.lib

Function blocks

L_CamEditProfileData - Online change of profile points

⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Information/possible settings	
byOperationMode	Byte	Number	Selection of the control mode	
			0	Reset (<i>nState</i> , <i>wActPoint</i>)
			1	Change points of x axis
			2	Change points of Y axis
			3	Change acceleration values
			4	Check whether points of the x axis are in ascending order (saw-tooth).
wProfileNo	Word	Number	Selection of the motion profile whose points are to be changed. ● Profile number 0 ... <i>wNumberOfProfiles</i>	
wStartingPoint	Word	Number	Selection of the number of the point the x/y position values or acceleration values of which (see <i>byOperationMode</i>) are to be changed. ● Enter number of point 0 ... <i>wNumberOfPoints</i>	
bLoadStartingPoint	BOOL	Binary	Skip to the selected point number <i>wStartingPoint</i> , e. g. for changing the values again.	
			TRUE ⇨ FALSE	Current point <i>wActPoint</i> = <i>wStartingPoint</i>

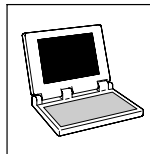
Outputs ⇒ (Variable type: VAR_OUTPUT)					
Identifiers	Data type	Signal type	Value/meaning		
wNumberOfProfiles	Word	Number	Number of profiles in the *.lc9 file. The profile with the highest number is output. (e. g. <i>wNumberOfProfiles</i> = "2": profiles 0.1 and 2 are contained.)		
wNumberOfPoints	Word	Number	Number of points in the currently selected profile. The <i>wNumberOfPoints</i> variable contains the number of profile points - 1. (e. g. <i>wNumberOfPoints</i> = "99": the profile has 100 points (0 ... 99))		
wActpoint	Word	Number	Current point number By a TRUE-FALSE edge on <i>bLoadInternalData</i> or by writing of the parameters <i>dnPU</i> (C1491) or <i>nADU</i> (C1492), the values of this point, changed in the respective mode (<i>byOperationMode</i>) are accepted.		
bProfileComplete	BOOL	Binary	Status signal " Last profile point reached" <table><tr><td>TRUE</td><td>Last point of the profile has been reached and changed.</td></tr></table>	TRUE	Last point of the profile has been reached and changed.
TRUE	Last point of the profile has been reached and changed.				
nState	Integer	Number	Status		
			20	Selection of the first point to be changed (<i>wStartingPoint</i>) is higher than the number of profile points.	
			15	Initialisation active, data are read from *.lc9 file.	
			10	Selection of the motion profile to be changed (<i>wProfileNo</i>) is higher than the number of motion profiles in the *.lc9 file.	
			0	FB is ready for operation.	
			-2	No profile data in the *.lc9 file.	
			-4	Profile data is not available in the absolute data model.	
			-10	Points of the x axis are not in ascending order.	



Stop!

If you use the **L_CamEditProfileData FB** to change cam interpolation points, you must make sure that the motion profile to be processed is not called in an instance of the **L_CamData FB**. Otherwise, the modified motion profile may exhibit jerk. Please also note that the x or y residual values of a profile cannot be changed or recreated in parallel.

If the motion profile remains selected in the **L_CamData FB** for reasons associated with process technology, please make sure that the x and y final values of the motion profile remain identical to the previous values (only intermediate values may be changed).



Parameters (Variable type: VAR_CONSTANT_RETAIN)				
Identifiers	Data type	Signal type	Information	
			Possible settings	Presetting
dnPU	Double integer	Position [unit]	Selected position in [unit] converted by the scaling routine and accepted for the current point <i>wActPoint</i> . • According to the selected control mode, X or Y positions are selected. X coordinates must be positive.	
C1491			-214000.0000 {0.001 unit/s} 214000.0000	0.0000 [unit]
nADU	Integer	Analog [%]	Selected acceleration in [%] converted by the scaling routine and accepted for the current point <i>wActPoint</i> . • $\pm 100\%$ ± 16384	
C1492			-100.00 {0.01 %} 100.00	0.00 %
* Code of first instance				

Selection of the axis, motion profile and point number to change values

By means of the **L_CamEditProfileData** FB, either the x or y coordinates of a motion profile can be changed. If the x and y coordinates are to be altered both at the same time, the FB has to be instanced for this purpose. Furthermore the acceleration values of a motion profile can be altered by means of this FB.

- Use *byOperationMode* to select the type of value to be changed:
 - 1 = points of the x axis
 - 2 = points of the Y axis
 - 3 = acceleration values
- Use *wProfileNo* to select the motion profile to be changed.
 - The output *wNumberOfPoints* indicates the number of points contained in the selected profile.
- Via *wStartingPoint* you select the number of the point the values of which are to be altered.
 - Enter number of point 0 ... *wNumberOfPoints*.
 - The acceptance is carried out via a TRUE-FALSE edge on *bLoadStartingPoint* (*wActPoint* = *wStartingPoint*).

If a point number beyond 0 ... *wNumberOfPoints* is to be accessed, a fault message occurs (*nState* = 20), and the current number of the point is set to "0" (*wActPoint* = 0).

Selection of new positions

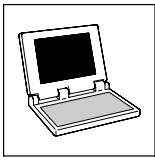
A new position value can be selected as:

- a physical value [unit] using the parameter *dnPU* (C1491)
- increments [inc] at *dnPosInternal_p*

Selection of new acceleration values

A new acceleration value can be selected as:

- a physical value [%] using the parameter *nADU* (C1492)
- increments at *nAccDecInternal_a*



Function library *LenzeCamControl.lib*

Function blocks

L_CamEditProfileData - Online change of profile points

Acceptance of new values for the current point

- If the parameters *dnPU* or *nADU* are used for the selection, the values will be converted by the scaling routine and automatically accepted for the current point *dnActPoint*.
- If the input variables *dnPosInternal_p* or *nAccDecInternal_a* are used for the selection, the acceptance of each point requires a TRUE-FALSE edge on *bLoadInternalData* is required.
- After the value for the current point has been accepted, the next profile point automatically becomes the current point (auto-increment function).

Checking of new points

It is possible to check the data selected for the x axis. This is an automatic process in which the drive checks if the points of the x axis are arranged in ascending order.

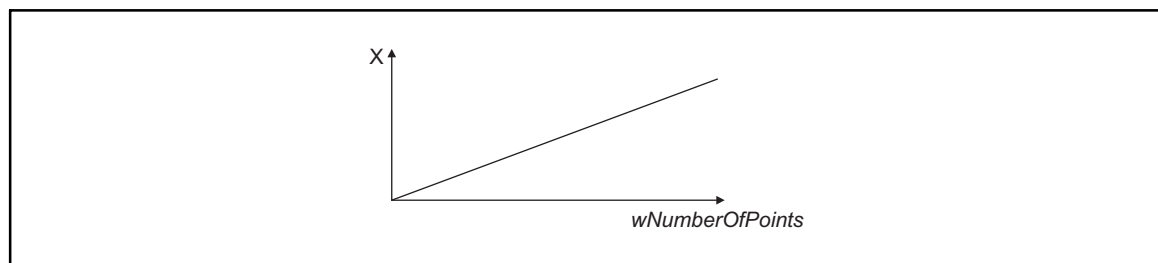
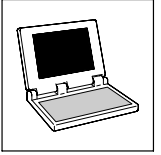


Fig. 3-5

Checking of new points

Set *byOperationMode* to "4" to activate the check.

- If all points of the x axis are in ascending order the output *nState* is set to the value "0" and the output *bProfileComplete* is set to TRUE.
- If an x value lower than the preceding X value is detected the check is stopped and the error message (*nState* = "-10") is indicated.
 - Set *byOperationMode* to "0" to reset the error message.
- For a new check, set *byOperationMode* to "0" to reinitialise the check.
- The time required for the check depends on the number of points and the task interval in which the FB is being processed. 10 points are checked per FB call.



Example: changing the positions y-sided via the variable *dnPosInternal_p*

1. Switch on FB.
2. Select RAM block via the input *bySelectMem* = 1 or 2.
3. Set mode: *byOpMode* = 2
4. Set profile to be changed: e. g. *wProfileNo* = 2
 - If n profiles can be selected, a profile can be selected via numerical values (*wProfileNo* = 0 ... n-1).
5. Select number of the point to be changed (optional):
 - *wStartingPoint* = point number 0 ... *wNumberOfPoints*
 - *bLoadStartPoint* = TRUE => FALSE
 - (skip to the selected point: *wActPoint* = *wStartingPoint*)
6. Alter the value of the point numbers set under 5. by writing the parameter *dnPU* (C1491).
7. The next point is selected (the output *wActPoint* is incremented automatically).

The FB now is ready for writing the next point. If the last point has been reached and written, the output *bProfileComplete* is set to TRUE. This output is reset to FALSE if another point is aimed (see 5.) or if the *byOpMode* mode is set to the value "0" (see 3.).

For changes via the *nADU* parameter, follow the same steps. In this connection, though, set *byOpMode* = 3, as otherwise the values are not accepted.

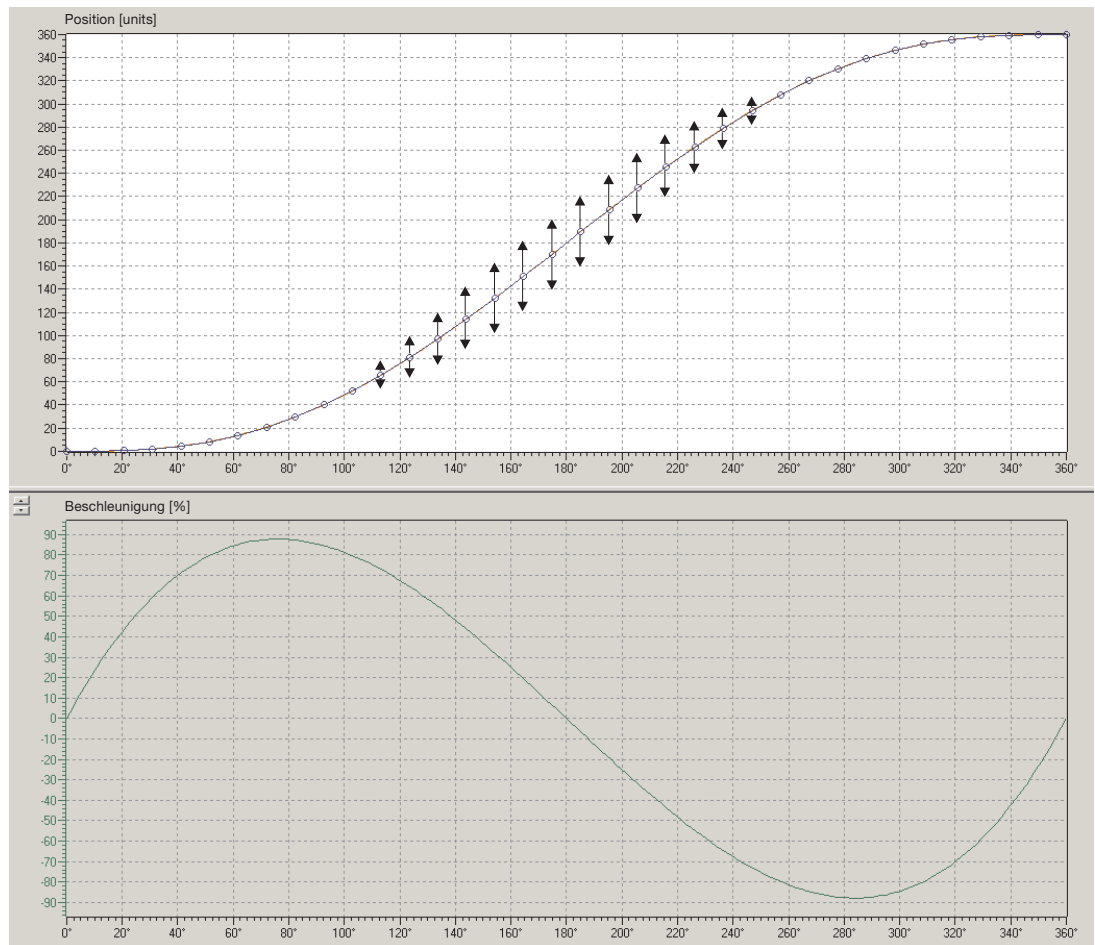
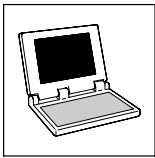


Fig. 3-6

Example: changing the positions y-sided



Function library *LenzeCamControl.lib*

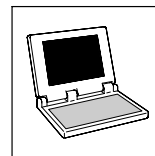
Function blocks

L_CamEditProfileData - Online change of profile points

Example: changing the positions x-sided via the *dnPU* parameter

1. Switch on FB.
2. Select RAM module via the input *bySelectMem* = 1 or 2.
3. Set mode: *byOpMode* = 1
4. Set profile to be changed: e. g. *wProfileNo* = 2
 - If n profiles can be selected, a profile can be selected via numerical values (*wProfileNo* = 0 ... n-1).
5. Select number of the point to be changed (optional):
 - *wStartingPoint* = point number 0 ... *wNumberOfPoints*
 - *bLoadStartPoint* = TRUE => FALSE
(skip to the selected point: *wActPoint* = *wStartingPoint*)
6. Alter the value of the point number set under 5. by writing of the variable *dnPosInternal_p*.
7. *SetbLoadInternalData* = TRUE => FALSE.
 - The output *wActPoint* indicates the next point.

The FB now is ready for writing the next point. If the last point has been reached and written, the output *bProfileComplete* is set to TRUE. This output is reset to FALSE if another point is aimed (see 5.) or if the *byOpMode* mode is set to the value "0" (see 3.).



3.5 L_CamIntegrateVarLim - Phase integrator

This FB provides an angle integrator with limit which can be changed online.

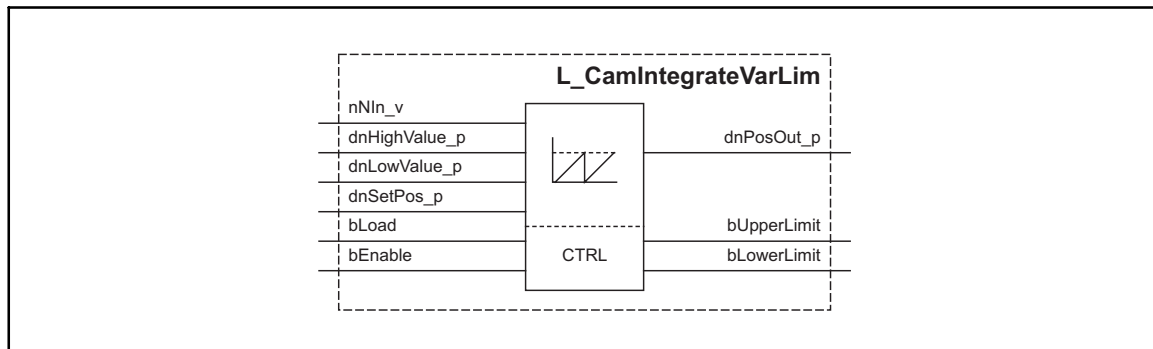
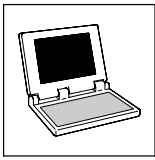


Fig. 3-7

Function block "L_CamIntegrateVarLim"

⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
nNIn_v	Integer	Velocity	Speed/master frequency in [inc/ms]
dnHighValue_p	Double Integer	Position	Upper value for return to <i>dnLowValue_p</i>
dnLowValue_p	Double Integer	Position	Lower value for return to <i>dnHighValue_p</i>
dnSetPos_p	Double Integer	Position	Start value for the integrator • Is loaded into the integrator with a Low/High edge at <i>bLoad</i> .
bLoad	BOOL	Binary	Loading of start value to the integrator FALSE ↗ TRUE Load starting value <i>dnSetPos_p</i> into the integrator.
bEnable	BOOL	Binary	Load start value permanently into integrator. FALSE Starting value <i>dnSetPos_p</i> is permanently loaded into the integrator (reset function). TRUE Integrator active.

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
dnPosOut_p	Double Integer	Position	Output of the position • 65535 = a motor/encoder revolution
bUpperLimit	BOOL	Binary	Message signal "Upper value reached" FALSE ↗ TRUE Integrator value has reached <i>dnHighValue_p</i> positive counting direction, the counting process is continued from <i>dnLowValue_p</i> . TRUE Return is not possible since <i>dnHighValue_p</i> is too low for the speed at <i>nNIn_v</i> .
bLowerLimit	BOOL	Binary	Message signal "Lower value reached" FALSE ↗ TRUE Integrator value has reached <i>dnLowValue_p</i> negative counting direction, the counting process is continued from <i>dnHighValue_p</i> . TRUE Return is not possible since <i>dnLowValue_p</i> is too low for the speed at <i>nNIn_v</i> .



Function library *LenzeCamControl.lib*

Function blocks

L_CamIntegrateVarLim - Phase integrator

Integrating the speed to a phase

- A positive input value at *nNIn_v* increases the output value at *dnPosOut_p*.
- A negative input value at *nNIn_v* reduces the output value at *dnPosOut_p*.

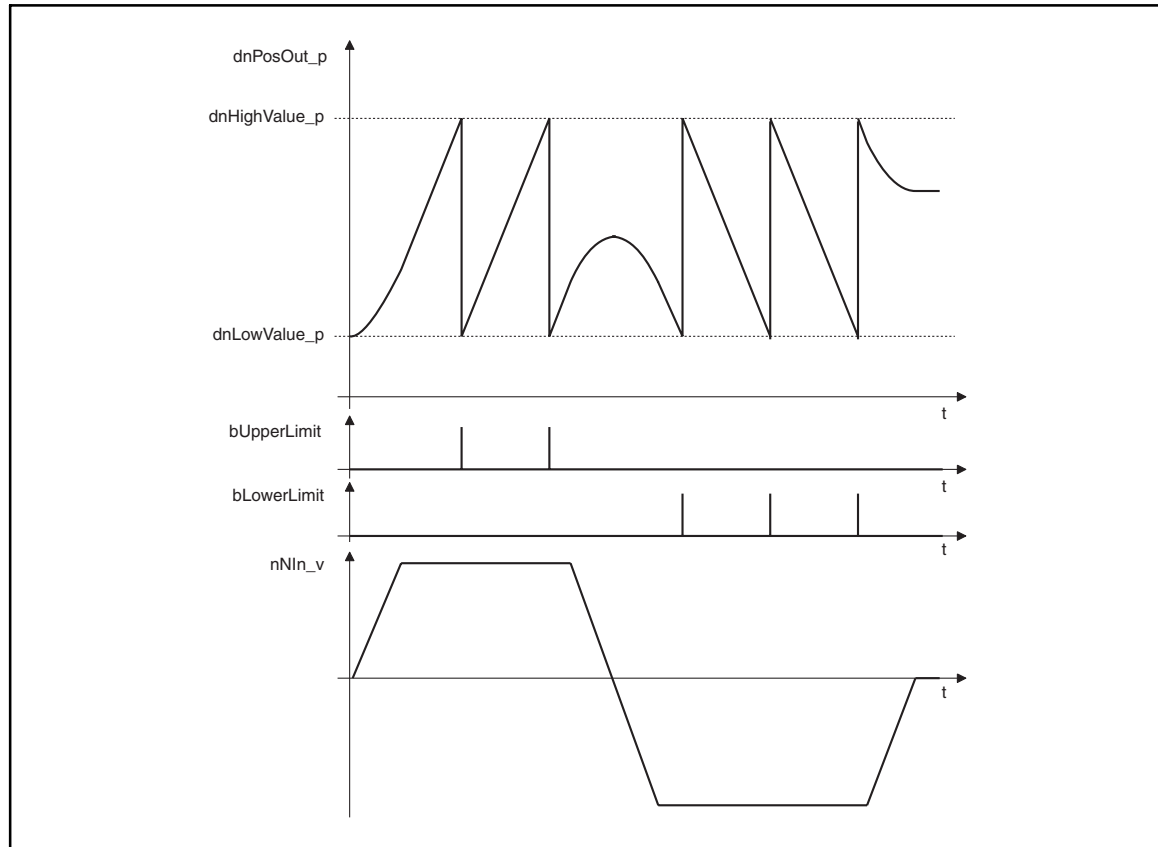


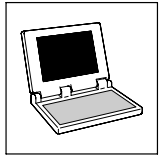
Fig. 3-8

Signal characteristic: Integrating the speed to a phase



Tip!

- The values set for the return (*dnHighValue_p* and *dnLowValue_p*) are no "hard" limits, i.e. the integrator can also assume other values (e.g. via *dnSetPos_p*).
- If *dnHighValue_p* and *dnLowValue_p* are set to the value "0", the integrator runs freely.
- All positions can be loaded. The FB behaves as described in the figures on the following pages.



Positions can be loaded as follows:

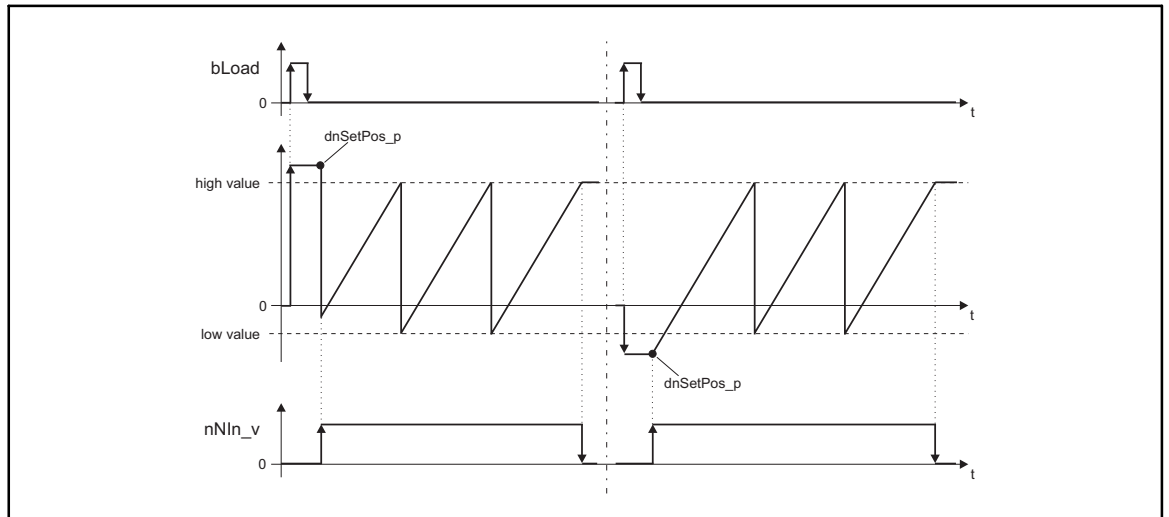


Fig. 3-9

Signal characteristic: loading a position and traversing with positive speed

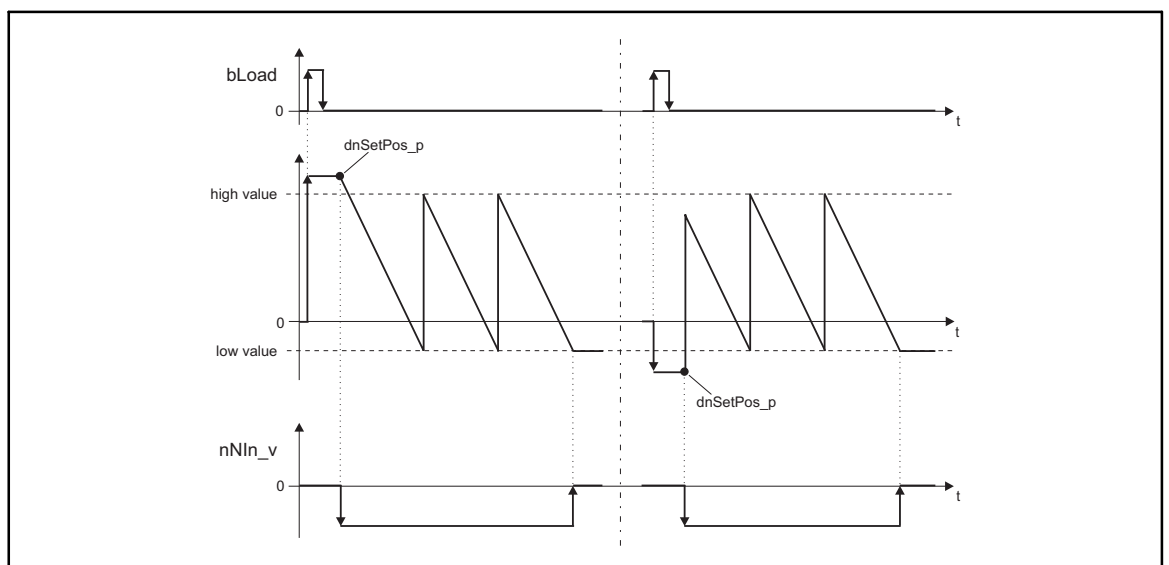
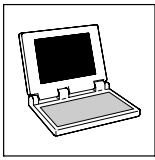


Fig. 3-10

Signal characteristic: loading a position and traversing with negative speed



Function library LenzeCamControl.lib

Function blocks

L_CamSectionShift - Section shift

3.6 L_CamSectionShift - Section shift

This FB is used to shift a profile section by decelerating the master axis.

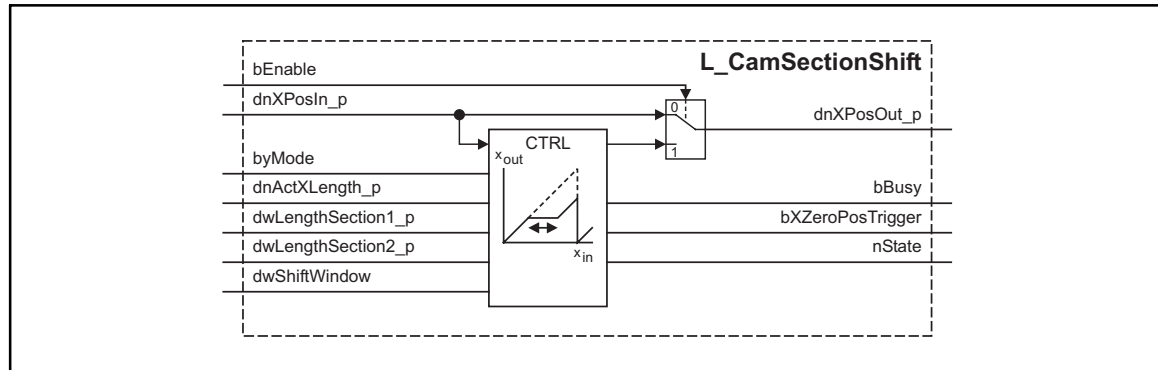
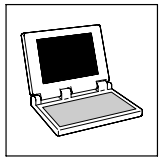


Fig. 3-11

"L_CamSectionShift" function block

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Information/possible settings
bEnable	BOOL	Binary	Function block enable
			FALSE Function block inhibited, x axis is looped through 1:1
			TRUE Function block enabled
dnXPosIn_p	Double integer	Position	Position of the x axis Limitation: 0 ... <i>dnActXLength_p</i>
byMode	Byte	Number	Shift modes A change in mode can only be effected if the x axis is in standstill in the first section.
			0 Time-controlled
			1 Phase-controlled with saved shift setpoint
			2 Phase-controlled with dynamic shift setpoint
dnActXLength_p	Double integer	Position	Length of x axis Connect this input to the output <i>dnActXLength_p</i> of the L_CamData FB.
dwLengthSection1_p	Double word	Position	Length of section 1 in [inc] - From this position on, the shift commences. - Limitation: 0 ... 2³¹ [inc]
dwLengthSection2_p	Double word	Position	Length of section 2 in [inc] - This value is required for checking the shift width <i>dwShiftWindow</i>. - Limitation: 0 ... 2³¹ [inc]
dwShiftWindow	Double word	Position	Adjustment value - depending on the set mode: - For <i>byMode</i>= 0: - selection of the shift in [ms], limited to 0 ... 60000 [ms]. - For <i>byMode</i>= 1 or 2: - selection of the shift in [inc]. - Maximum shift value = length of the 3. profile section.



Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Value/meaning
dnXPosOut_p	Double integer	Position	Position output • For <i>bEnable</i> = FALSE, the input value <i>dnXPosIn_p</i> is output unchanged.
bBusy	BOOL	Binary	Status signal "Section shift active" TRUE Section shift active.
bXZeroPosTrigger	BOOL	Binary	Status signal "zero crossing" • For initialisation of the profile data, connect this output to the corresponding input of the L_CamData FB. FALSE ⇨ TRUE Zero crossing • The TRUE status is only available for one FB call cycle.
nState	Integer	Number	Status 12 <i>dwShiftWindow</i> shift value too great in mode 2 11 <i>dwShiftWindow</i> shift value too great in mode 1 10 <i>dwShiftWindow</i> shift value too great in mode 0 0 OK, no error -1 Error: incorrect direction of rotation of the x axis (input <i>dnXPosIn_p</i>) -2 Error: selection on <i>dnXPosIn_p</i> beyond the limitation

Preconditions for a shift

For carrying out a shift, the following requirements have to be met:

1. The motion profile has to comprise 3 sections.
2. The last section has to be a latch phase.
3. The direction of rotation of the x axis has to be positive (change in position on *dnXPosIn_p*)

Example:

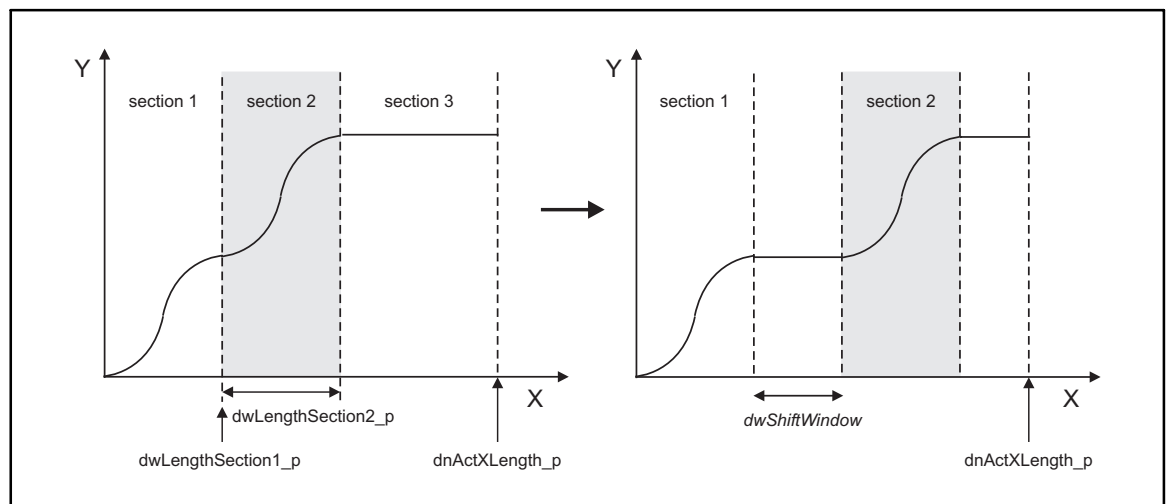
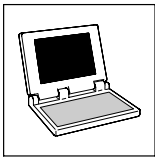


Fig. 3-12

Example: shifting the 2. profile section



Function library LenzeCamControl.lib

Function blocks

L_CamSectionShift - Section shift

Shift modes

The following shift modes are provided:

Mode 0 - time-controlled shift

- On the input *dwShiftWindow*, a time or the ratio of shift time [ms] and cycle time [ms] is defined. The maximum value for *dwShiftWindow* is 60000, which for a cycle time of 1 ms corresponds to a shift of one minute:

$$\text{Vorgabe an } dwShiftWindow = \frac{\text{gewünschte Verschiebungszeit}}{\text{Taskintervall}}$$

- A new value only is accepted if the x axis is located in the first profile section.



Tip!

The welding bar control is a typical application for this mode, requiring a defined welding time or a defined standstill of the master value.

Mode 1 - phase-controlled shift with a saved shift setpoint

- In this mode, the shift is defined as a position value. As for all modes, the maximum value is the remaining length of the third profile section.
- As for mode 0, a new setpoint only is accepted in the first section.

Mode 2 - phase-controlled shift with a dynamic shift setpoint

- In this mode a new shift phase is accepted immediately.



STOP!

The immediate acceptance of a new shift phase in mode 2 may result in torque leaps!

A switch-over between the modes can only be effected if the x axis is in standstill in the first profile section.

Effecting the shift

For effecting and checking the shift, define the lengths of sections 1 and 2 (*dwLengthSection1_p* and *dwLengthSection2_p*) as well as the length of the x axis (*dnActXLength_p*).

The specified shift phase *dwShiftWindow* is monitored internally by means of these parameters and, where required, is limited to the duration of the latch phase (section 3):

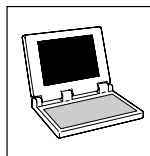
$$dwShiftWindow \leq dnActXLength_p - (dwLengthSection1_p + dwLengthSection2_p)$$

- The shift occurs if the x position defined on *dnXPosIn_p* reaches the *dwLengthSection1_p* starting position.
- As long as the x position defined on *dnXPosIn_p* is smaller than the sum of *dwShiftWindow_p* + *dwLengthSection1_p*, the x position that is output on *dnXPosOut_p* remains constant.
- During this shift phase, the status output *bBusy* is set to TRUE.



Tip!

If a shift temporarily is not desired, the FB can be deactivated by setting *bEnable* to FALSE. The input value *dnXPosIn_p* then is output unchanged on *dnXPosOut_p*.



3.7 L_CamSelPosition - Master value selection

Use this FB to select a master value (master angle) (1 of 4).

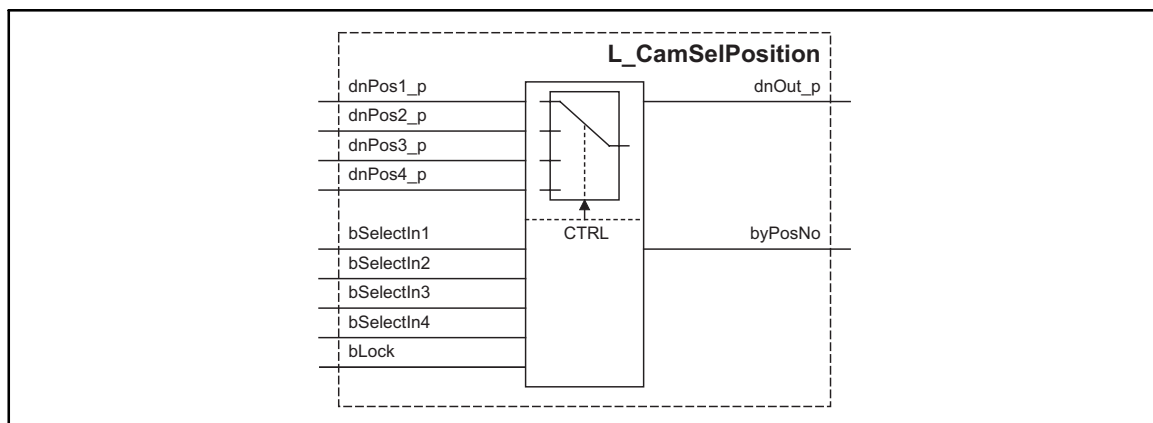


Fig. 3-13 Function block "L_CamSelPosition"

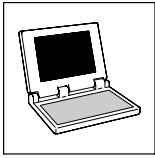
⇒ Inputs (Variable type: VAR_INPUT)			
Identifier	Data type	Signal type	Info/possible settings
dnPosIn1_p ... dnPosIn4_p	Double Integer	Position	Master value 1 ... Master value 4
bSelectIn1 ... bSelectIn4	BOOL	Binary	Master value 1 selection ... Selection of master value 4
bLock	BOOL	Binary	Selection locking
			TRUE It is not possible to select another master value input

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
dnOut_p	Double Integer	Position	Selected master value (<i>dnPosIn1_p</i> ... <i>dnPosIn4_p</i>)
byPosNo	Byte	Number	Number (1 ... 4) of the currently selected master value

"1 of 4" selection

The four inputs *dnPosIn1_p* ... *dnPosIn4_p* can be assigned to master value sources one of which can be connected to the master value output *dnOut_p* according to the following logic:

Cycle	Input				Lock bLock	Output nOut_v	See Remarks
	bSelectIn1	bSelectIn2	bSelectIn3	bSelectIn4			
n	FALSE ↗ TRUE	FALSE	FALSE ↗ TRUE	FALSE	FALSE	<i>dnPosIn1_p</i>	①
n + 1	TRUE	FALSE ↗ TRUE	FALSE	FALSE	FALSE	<i>dnPosIn2_p</i>	②
n + 2	TRUE	TRUE	FALSE	FALSE ↗ TRUE	TRUE	<i>dnPosIn2_p</i>	③
Remarks:							
①	The master value input to be connected is selected by means of a FALSE/TRUE edge at the corresponding selected input <i>bSelectIn1</i> ... <i>bSelectIn4</i> . - The selected input must remain TRUE after the FALSE-TRUE transition. - If FALSE/TRUE edges occur at several selected inputs at the same time in the same calculating cycles, the selected input with the lowest number has priority (here: <i>bSelectIn1</i> has priority over <i>bSelectIn3</i>).						
②	The input last triggered with a FALSE-TRUE transition has priority.						
③	By setting <i>bLock</i> to TRUE a change-over to another master value can be locked.						



Function library LenzeCamControl.lib

Function blocks

L_CamSelVelocity - Master frequency selection

3.8 L_CamSelVelocity - Master frequency selection

Use this FB to select a master frequency "1 of 4".

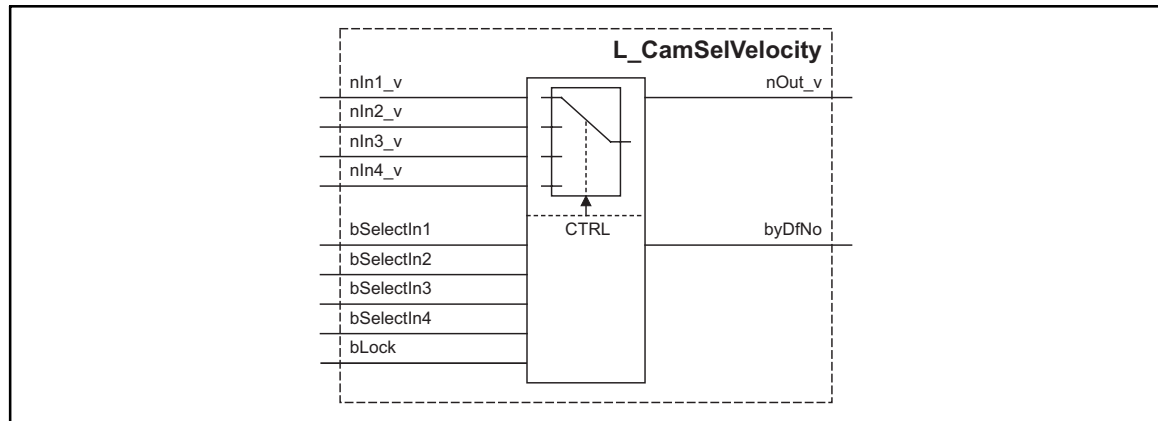


Fig. 3-14 Function block "L_CamSelVelocity"

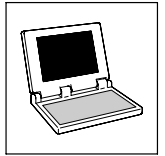
⇒ Inputs (Variable type: VAR_INPUT)				
Identifier	Data type	Signal type	Info/possible settings	
nIn1_v	Integer	Velocity	Master frequency 1	
...			...	
nIn4_v			Master frequency 4	
bSelectIn1	BOOL	Binary	Selection of master frequency 1	
...			...	
bSelectIn4			Selection of master frequency 4	
bLock	BOOL	Binary	Selection locking	
			TRUE	It is not possible to select another master frequency input.

Outputs ⇒ (Variable type: VAR_OUTPUT)				
Identifier	Data type	Signal type	Value/meaning	
nOut_v	Integer	Velocity	Selected master frequency (<i>nIn1_v</i> ... <i>nIn4_v</i>)	
byDfNo	Byte	Number	Number (1 ... 4) of the currently selected master frequency	

"1 of 4" selection

The four inputs *nIn1_v* ... *nIn4_v* can be assigned to master frequency sources one of which can be connected to the master frequency output *nOut_v* according to the following logic:

Cycle	Input				Lock bLock	Output nOut_v	See Remarks
	bSelectIn1	bSelectIn2	bSelectIn3	bSelectIn4			
n	FALSE ↗ TRUE	FALSE	FALSE ↗ TRUE	FALSE	FALSE	<i>nIn1_v</i>	①
n + 1	TRUE	FALSE ↗ TRUE	FALSE	FALSE	FALSE	<i>nIn2_v</i>	②
n + 2	TRUE	TRUE	FALSE	FALSE ↗ TRUE	TRUE	<i>nIn2_v</i>	③
Remarks:							
①	The master frequency input to be connected is selected by means of a FALSE/TRUE edge at the corresponding selected input <i>bSelectIn1</i> ... <i>bSelectIn4</i> . - The selected input must remain TRUE after the FALSE-TRUE transition. - If FALSE/TRUE edges occur at several selected inputs at the same time in the same calculating cycles, the selected input with the lowest number has priority (here: <i>bSelectIn1</i> has priority over <i>bSelectIn3</i>).						
②	The input last triggered with a FALSE-TRUE transition has priority.						
③	By setting <i>bLock</i> to TRUE a change-over to another master frequency can be locked.						



3.9 L_CamVelocityFilter - Filter (PT1) for speed signals

Use this FB to filter or delay speed signals according to the PT1 principle. The resulting position loss will be compensated by an internal compensator.



Note!

Only use this FB in tasks with fixed time interval (e.g. T#2ms).

Deactivate the filter function during acceleration and deceleration to avoid a drifting of positions. For this, set *bEnable* to FALSE.

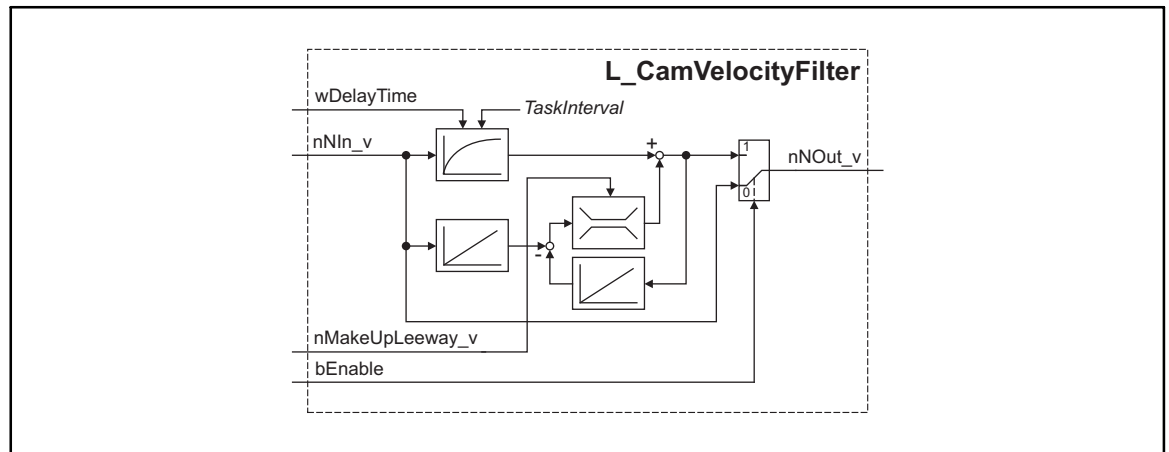
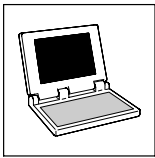


Fig. 3-15 Function block "L_CamVelocityFilter"

⇒ Inputs (Variable type: VAR_INPUT)				
Identifier	Data type	Signal type	Info/possible settings	
wDelayTime	Word	Time	Delay time in [ms] - Setting range: 1 ... 65535 [ms] - When the filter is activated, the delay time must be longer than the task interval.	
nInln_v	Integer	Velocity	Speed input 16384 inc/ms ≙ 15000 rpm	
nMakeUpLeeway_v	Integer	Velocity	Compensation speed for position loss 16384 inc/ms ≙ 15000 rpm	
bEnable	BOOL	Binary	Filter activation	
			FALSE	Filter deactivated
			TRUE	Filter activated

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifier	Data type	Signal type	Value/meaning
nOut_v	Integer	Velocity	Speed output 16384 inc/ms ≙ 15000 rpm



Function library *LenzeCamControl.lib*

Function blocks

L_CamVelocityFilter - Filter (PT1) for speed signals

Settings

- Use *wDelayTime* to select the delay time T in [ms]. For trouble-free operation of the filter, the delay time must be longer than the task interval.
- The proportional component is predefined with $K=1$:

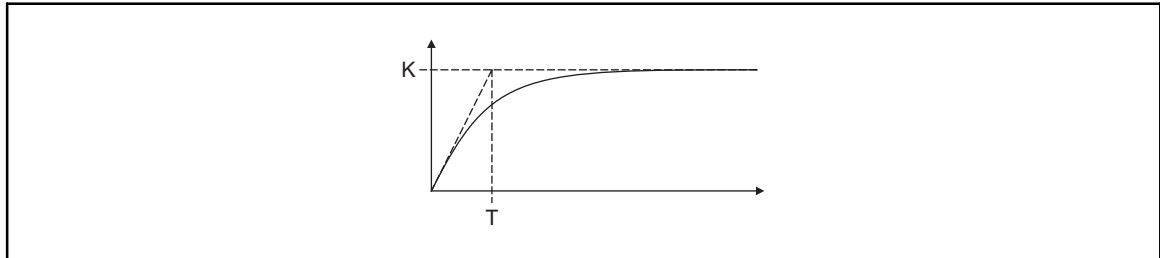
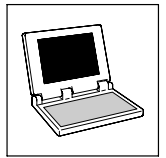


Fig. 3-16

Deceleration time T in [ms]

- Use *nMakeUpLeeway_v* to select a compensation speed > 0 [inc/ms] to avoid position losses.



4 Index

A

Absolute positioning, 2-33 , 2-47
Acceleration/deceleration ramps, 1-5
Activating the traversing function, 2-69
Activation of the clutch function, 2-4
Approaching zero position, 2-47
Auxiliary position setpoint for separated positioning, 2-24

C

Cam group, 2-7
CAM_PROFILE_REF, 2-50
Changing profile points online, 3-5
Changing the direction of rotation, 2-60
Clutch engagement via ramp function , 2-5
Compensation of bus transmission cycles > 1ms, 2-38
Continuous operation, 2-54
Cyclic operation, 2-54
Cyclic operation (with automatic stop process), 2-71

D

Data structure CAM_PROFILE_REF, 2-50
Deactivation of the following error monitoring, 2-44
Defining the master value (x position) for the cam profiler, 2-31
Detection of direction of rotation, 2-17
Differentiator, 3-4
Disengagement via ramp function, 2-6
Dynamic adjustment, 2-18
Dynamic cam/hysteresis adjustment, 2-16

E

Effecting the shift, 3-16
Enabling input "bEnable", 1-5
External master position, 2-72
Extrapolation, 2-37

F

Feed drive via cams, 2-34
Feedback for speed and position control, 2-26
Filter (PT1) for speed signals, 3-19
Filtering speed signals, 3-19
Following error monitoring, 2-43
Forced extrapolation, 2-38
Forced stop, 2-72
Function blocks with *.lc9 access, 1-5
Function libraries, Version info, 1-3

G

Generating and positioning the vertical shaft (x axis), 2-69
Generation of a virtual master frequency, 2-74

H

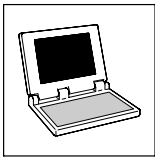
Handwheel function, 2-71
Hysteresis, 2-19

I

Immediate disengagement (forced disengagement), 2-6
Integrating the speed to a phase, 3-12
Integration of digital frequency signals to one position, 2-23

L

L_CamAddVelocity, 3-2
L_CamAdjustDfIn, 3-3
L_CamClutchXAxis, 2-2
L_CamContactor, 2-7
L_CamContactorVPos, 2-16
L_CamControl, 2-20
L_CamData, 2-28
L_CamDifferentiate, 3-4
L_CamEditProfileData, 3-5
L_CamExtrapolate, 2-37
L_CamGetPosition, 2-39
L_CamIntegrateVarLim, 3-11
L_CamLimitVelocity, 2-40
L_CamMachineData, 2-41
L_CamMonitFollowError, 2-43
L_CamPosCtrlLin, 2-45



Function library *LenzeCamControl.lib*

Index

L_CamProfileData, 2-49
L_CamSectionShift, 3-14
L_CamSelPosition, 3-17
L_CamSelVelocity, 3-18
L_CamSequence, 2-51
L_CamSetPosition, 2-57
L_CamSetXAxisVelocity, 2-58
L_CamSetYAxisVelocity, 2-63
L_CamVelocityFilter, 3-19
L_CamVMasterPosition, 2-67
L_CamVMasterVelocity, 2-73
Ic9-file, 1-5
Limit stop, 2-47
Limitation inputs, 2-40
Limitation of the actual position value for a rotary axis, 2-18
limiting element, 2-40
Linear profile generator, 2-45
Loading of important profile points, 2-49
Loading positions, 3-13
Loading the auxiliary integrator, 2-24
Looping through the external master signal, 2-61

M

Machine parameters, 2-41 , 2-42
Main position setpoint, 2-23
Manual homing (inching), 2-71 , 2-75
master frequency adaptation, 3-3
Master frequency selection, 3-18
Master value selection, 3-17
Maximum number of points, 2-36
Motion profile type, 2-33

N

Normal operation, 2-74
Normal operation (with automatic stop process), 2-70

O

Offset, 2-66
Offset for the master angle, 2-31
Online change of profile points, 3-5

P

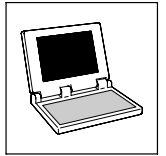
Phase integrator, 3-11
Preconditions for a shift, 3-15
Profile management, 2-28

R

Ramp function generator enable, 2-74
Relative positioning, 2-34 , 2-47

S

Safety information
 Layout, Warning of damage to material, 1-1
 layout, other notes, 1-1
Section shift, 3-14
Selecting an offset for the Y axis, 2-48
Selection of a profile sequence, 2-51
Selection of direction of rotation, 2-9
Selection/changeover of the motion profile, 2-32
Separate feedback for speed and position control, 2-26
Setpoint conditioning, 2-20
Shift modes, 3-16
Smoothing a low-resolution absolute value encoder, 2-38
speed addition, 3-2
Speed control, 2-27
Speed limitation without position loss, 2-40
Static adjustment, 2-19
Status input "nState", 1-5
Stop function, 2-75
Stopping the positioning, 2-47
Stretching & compressing the setpoint, 2-63
Stretching & compression of the master value, 2-58
Synchronisation of the Y axis via touch probe, 2-25
Synchronised stretching and compression of the Y axis, 2-35
Synchronised stretching and compression of the y axis, 2-66
Synchronising the x axis via touch probe, 2-62



T

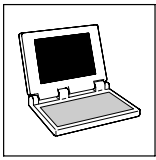
Term definitions, 1-2
time cam, 2-15
Time-equidistant call, 1-4
Torque precontrol, 2-27 , 2-36
Touch probe, 2-25 , 2-62
Transferring the machine parameters, 2-42
travel cam, 2-15

U

unit, 1-2
Unit conversion, 2-39 , 2-57

V

Virtual clutch, 2-2
Virtual master, 2-73
Virtual master with positioning function, 2-67



Function library LenzeCamControl.lib

Index