

Lenze

Manual

IEC 61131-3

inside

***Global Drive
PLC Developer Studio***



Global Drive

Function library

LenzeMotionControl1.lib

This documentation applies to the function library **LenzeMotionControl1.lib** as of version 5.00

The function library **LenzeMotionControl1.lib** can be used for the following Lenze PLCs:

	Type	as from hardware version	as from software version
9300 Servo PLC	EVS93XX-xT	6A	6.0
ECSxA	ECSxAxxx	1C	7.0

Software system requirements:

Drive PLC Developer Studio Professional 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.

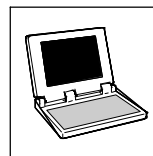
© 2008 Lenze Drive Systems GmbH

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

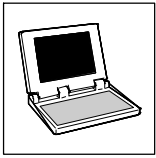
All information given in this documentation has been carefully selected and tested for compliance with the hardware and software described. Nevertheless, discrepancies cannot be ruled out. We do not accept any responsibility or liability for any damage that may occur. Required corrections will be included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

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

Version 4.0 04/2008 - TD29

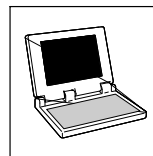


1 Preface and general information	1-1
1.1 About this Manual	1-1
1.1.1 Conventions used in this Manual	1-1
1.1.2 Pictographs used in this Manual	1-1
1.1.3 Layout of the descriptions	1-2
1.1.4 Terminology used	1-2
1.2 Version identifiers of the function libraries	1-3
2 Function blocks for positioning tasks	2-1
2.1 L_MCAutoResetInterface - process control	2-2
2.2 L_MCMachineData - machine parameters	2-4
2.3 L_MCPositionCorrection - feed correction	2-10
2.4 L_MCProfileGenerator - profile generator	2-15
2.5 L_MCResidualDistance - remaining path calculation	2-28
2.6 L_MCSwitchCam - switching points (rotary axis)	2-31
2.7 L_MCSwitchPoint - switching points	2-35
2.8 L_MCTouchProbeCtrl - touch probe preprocessing	2-38
3 Function blocks for converting units	3-1
3.1 L_MCGetPosition	3-2
3.2 L_MCGetVelocity	3-3
3.3 L_MCSetAcceleration	3-4
3.4 L_MCSetDeceleration	3-5
3.5 L_MCSetProfileConverter	3-6
3.6 L_MCSetProfileValues	3-7
3.7 L_MCSetPosition	3-9
3.8 L_MCSetVelocity	3-10
4 Structured data types	4-1
4.1 MC_AXIS_REF	4-1
4.2 MC_PROFILE_REAL_MEASURING	4-3
4.3 MC_PROFILE_INTERNAL_MEASURING	4-4
5 Index	5-1



Function library LenzeMotionControl1.lib

Contents



1 Preface and general information

1.1 About this Manual

This Manual contains information about the function library **LenzeMotionControl1.lib** for the **Drive PLC Developer Studio**.

- The function library **LenzeMotionControl1.lib** comprises functions necessary for positioning tasks.
- The function library **LenzeMotionControl2.lib** contains functions for implementing axis synchronisation tasks.
- The function libraries are part of the **Software Package - Positioner**.

1.1.1 Conventions used in this Manual

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

Variable identifiers

are written in italics in the explanation:

- "Use *bReset*..."



Tip!

Information about the conventions used for the variables of 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_":

- "Use the FB **L_MCProfileGenerator**..."

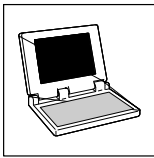
Program listings

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

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

1.1.2 Pictographs used in this Manual

	Use of pictographs	Signal words	
Warning of material damage		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 LenzeMotionControl1.lib

Preface and general information

About this Manual

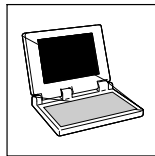
1.1.3 Layout of the descriptions

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

	Function	Function block (FB)/ system block (SB)
①	Headline stating the function and the function identifier	
②	Declaration of the function: - Data type of the return value - Function identifier - List of transfer parameters	-
③	Short description of the most important properties	
④	Function chart including all associated variables - Transfer parameters - Return value	FB/SB chart including all associated variables - Input variables - Output variables
⑤	Table giving information about the transfer parameters: - Identifiers - Data type - Possible settings - Info	Table giving information about input and output variables: - Identifiers - Data type - Type of variable - Possible settings - Info
⑥	Table giving information about the return value: - Data type of the return value - Possible return values and their meaning	-
⑦	Additional information (notes, tips, application examples, etc.)	

1.1.4 Terminology used

Term	In the following text used for
Cxxx/y	Subcode y of code Cxxx (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 functionality of a function block
PLC	9300 Servo PLC - Axis module ECSxA "Application"
SB	System block
[unit]	Wildcard for the physical unit of the external measuring system (e.g. "µm", "mm", "cm")



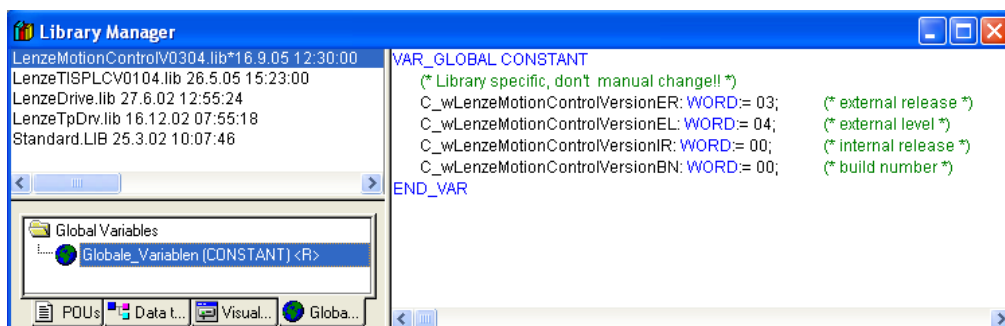
1.2 Version identifiers of the function libraries

Version identifiers as of version 3.04:

Constant	Meaning	Example value
C_w[FunctionLibraryName]VersionER	External release	03
C_w[FunctionLibraryName]VersionEL	External level	04
C_w[FunctionLibraryName]VersionIR	Internal release	00
C_w[FunctionLibraryName]VersionBN	Build number	00

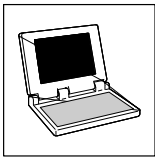
Version: 03 04 00 00

- In the example, "03040000" stands for version "3.04.00.00".



In offline mode, the value of this constant can be displayed via the **Library manager/register Global variables**:

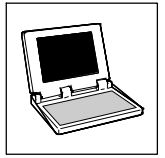
- Use the menu command **Window→Library manager** to open the Library manager in DDS.
- Select the register **Global variables** and open the folder **Global variables**.
- If you highlight the *Global_Variablen...*, the version will be indicated in the Library manager.



Function library LenzeMotionControl1.lib

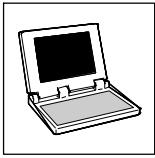
Preface and general information

Version identifiers of the function libraries



2 Function blocks for positioning tasks

This chapter informs about function blocks which can be used for positioning tasks.



Function library LenzeMotionControl1.lib

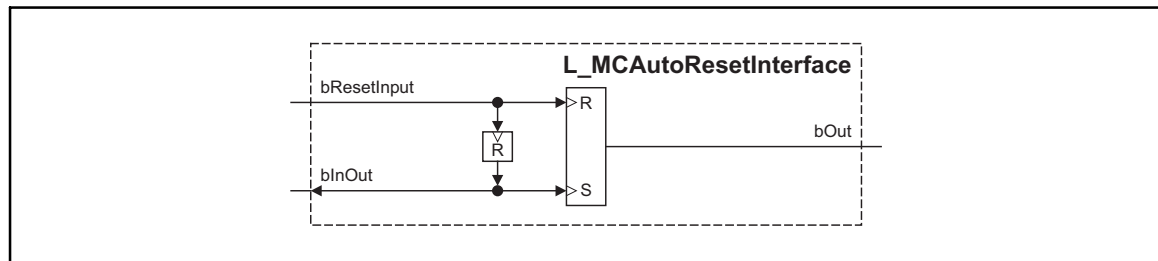
Function blocks for positioning tasks

L_MCAutoResetInterface - process control

2.1 L_MCAutoResetInterface - process control

It is often necessary to start a process (A) from another process (B) and to wait until this process has been processed. This FB makes process control easier.

- The control output *bOut* for starting process (B) is automatically reset when the FB is called again.



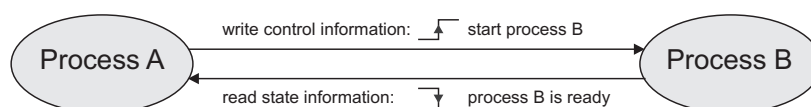
⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
bResetInput	Bool	binary	Control input for resetting <i>bOut</i> and <i>bInOut</i> . FALSE ↗ TRUE <i>bOut</i> and <i>bInOut</i> are reset to FALSE.

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info/possible settings
bInOut	Bool	-	Input/output variable which can be used both as a control input and as a status output. Input information: FALSE ↗ TRUE <i>bOut</i> is set to TRUE until the FB is called again. Output information: TRUE ↘ FALSE <i>bOut</i> was reset to FALSE through <i>bResetInput</i> .

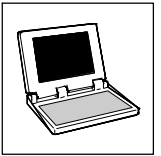
Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
bOut	Bool	-	Control output • A FALSE/TRUE transition at <i>bInOut</i> is used to set this output to TRUE. When the FB is called again, the output is automatically reset to FALSE.

Interface variable

A variable of type VAR_IN_OUT is used as an interface between the two processes. It contains control information (start process B) and status information (process B is ready):



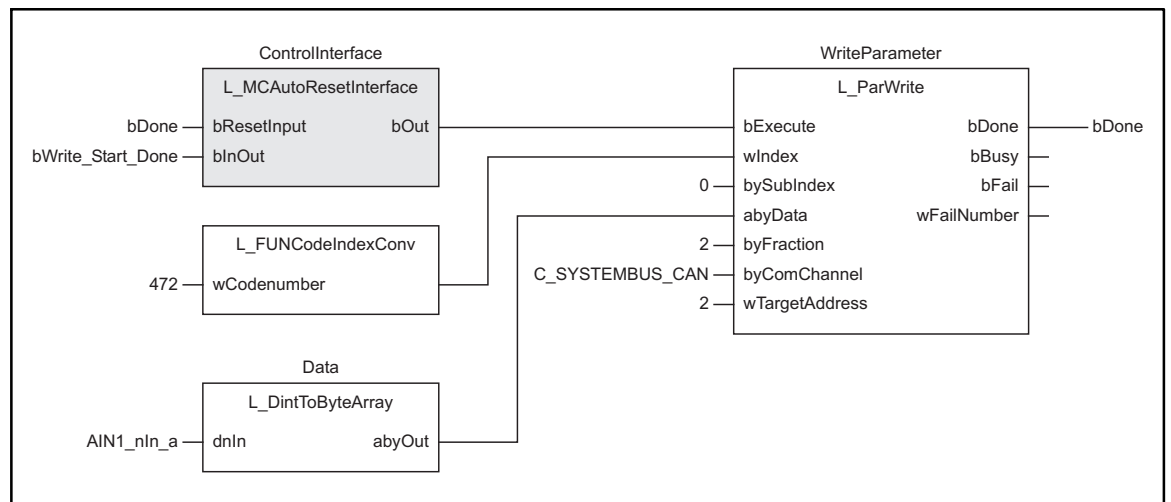
- Connect the interface variable with *bInOut*.

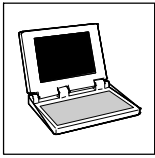


Typical process

1. Process A sets the interface variable connected with *bInOut* from FALSE to TRUE. In doing so, the output *bOut* which is used to start process B is set from FALSE to TRUE.
2. When the FB is called again, *bOut* is automatically reset to FALSE.
3. When process B is ready, a FALSE/TRUE is sent to input *bResetInput*. In this way, the interface variable is reset from TRUE to FALSE.
4. Process A evaluates the interface variable at state FALSE and recognises that process B is ready.

Example connection





Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

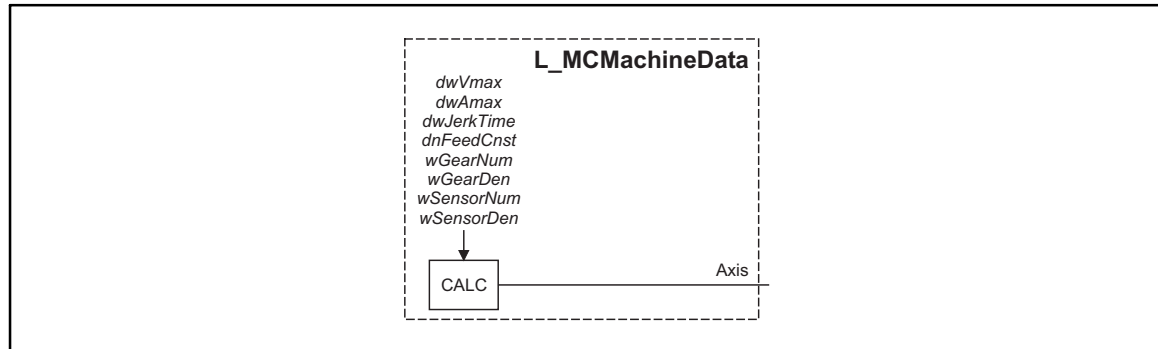
L_MCMachineData - machine parameters

2.2

L_MCMachineData - machine parameters

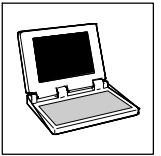
This FB conditions the machine parameters selected as physical units for the internal measuring system of the profile generator.

- The FB stores the conditioned machine parameters in a data structure (MC_AXIS_REF (4-1)).
- Use the output *Axis* to assign this data structure to a global variable to transfer machine parameters to other MotionControl FBs via this "interface".



Retain variables (Variable type: VAR CONSTANT RETAIN)			
Identifiers	Data type	Signal type	Info/possible settings
dwVmax	Double word	-	Maximum traversing speed in [unit/s] • 10000 = 1.0000 unit/s
dwAmax	Double word	-	Maximum acceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²
dwJerkTime	Double word	-	Jerk time • 10000 = 1.0000 s
dnFeedCnst	Double integer	-	Feed constant [unit/rev] • Machine feed in units per revolution of the gearbox load side. • 10000 = 1.0000 [unit/rev]
wGearNum	Word	-	Gear factor numerator
wGearDen	Word	-	Gear factor denominator
wSensorNum	Word	-	Position encoder gear factor numerator
wSensorDen	Word	-	Position encoder gear factor denominator
			• Gear factor between motor and position encoder. • Numerator/denominator equals motor speed/encoder speed. • Encoder on motor shaft: 1/1

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	With this output machine parameters can be made available to other MotionControl function blocks.



Parameter codes of instances

Identifiers	L_MCMachineData1	Setting range			Lenze
dwVmax	C1240	0.0001	{0.0001 unit/s}	214000.0000	50.0000 unit/s
dwAmax	C1250	0.0001	{0.0001 unit/s ² }	214000.0000	100.0000 unit/s ²
dwJerkTime	C1256	0	{0.001 s}	10.000	0.000 s
dnFeedCnst	C1204	0.0001	{0.0001 unit/rev}	214000.0000	1.0000 unit/rev
wGearNum	C1202	1	{1}	65535	1
wGearDen	C1203	1	{1}	65535	1
wSensorNum	C1200	1	{1}	65535	1
wSensorDen	C1201	1	{1}	65535	1



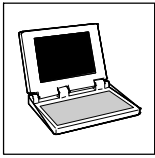
Caution!

- Declare the instance of the FB *L_MCMachineData* locally in PLC_PRG.
- If you do not use the initial instance *L_MCMachineData1* ensure that the parameters of the instance used are > 0 to avoid that the value is divided by zero in following calculations!



Tip!

Connect the output *Axis* with a global variable (e.g. *g_Axis*) to make the machine parameters available to other FBs for positioning via this "interface".



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MCMachineData - machine parameters

Selection and input of machine parameters

Machine parameters are entered via parameter codes which must be created for the FB instance in the **Instance parameter manager**.



Tip!

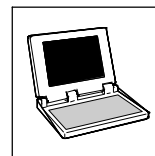
Select the instance name *L_MCMachineData1* for this FB to be able to use Lenze codes (first instance codes).

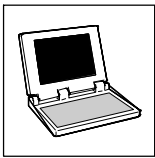
- Even if you use the first instance, corresponding codes must be created in the **Instance parameter manager**.

The following table lists absolutely necessary machine parameters:

Machine parameters	Unit	Code*	Info
Gear factor (numerator)	-	C1202	Input according to the gearbox nameplate.
Gear factor (denominator)	-	C1203	
Position encoder gear factor (numerator)	-	C1200	Input of gear factor between motor and position encoder.
Gear factor position encoder (denominator)	-	C1201	
Feed constant (V_k)	[unit/rev]	C1204	Enter the number in [unit] (e.g. [mm]), which is to be fed when the gearbox output revolves once.
Maximum speed (v_{max})	[unit/s]	C1240	This limit applies to the entire machine. Caution: The limit must be set in a way that it can be reached with max. motor speed: $v_{max} \leq n_{max} \cdot \frac{\text{Feed constant}}{60} \cdot \frac{\text{Gear (denominator)}}{\text{Gear (numerator)}}$ $C1240 \leq C0011 \cdot \frac{C1204}{60} \cdot \frac{C1203}{C1202}$
Maximum acceleration (a_{max})	[unit/s ²]	C1250	
Jerk time	[s]	C1256	
Maximum motor speed (n_{max})	[rpm]	C0011	This limit mainly applies to the motor.
* Parameter code of the first instance L_MCMachineData1			

- The physical unit is defined via the input for [unit], e.g. [mm], [m] or [degree].





Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCMachineData - machine parameters

Use of S-shaped ramps (S-profiles)

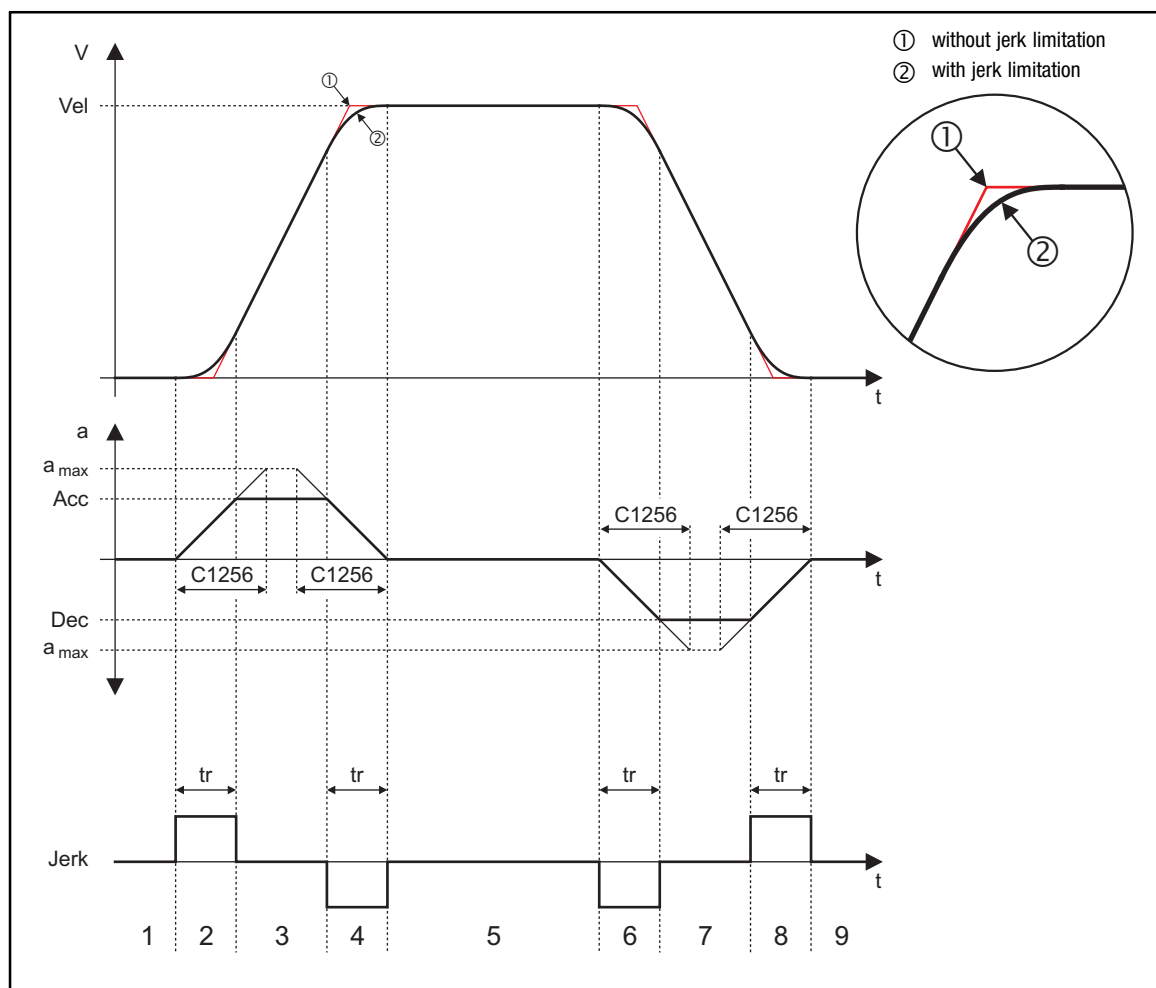
S-shaped ramps (S-profiles) are used for soft acceleration and deceleration. In this way, the jerk and thus the load on the drive components are reduced.

- The selected acceleration/deceleration (Acc/Dec) is only reached after expiry of the jerk time (t_r).

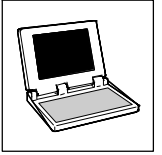


Tip!

Since acceleration rises slower with S-profiles, time required for positioning is longer than with L-profiles.



- | | |
|---|---|
| 1. Standstill | 6. Deceleration with jerk limitation |
| 2. Acceleration with jerk limitation | 7. Deceleration according to deceleration profile (Dec) |
| 3. Acceleration according to acceleration profile (Acc) | 8. Reducing deceleration (jerk limitation) |
| 4. Reducing acceleration (jerk limitation) | 9. Standstill (target position reached) |
| 5. Traversing with Vel according to velocity profile | |



Note!

The selection of disproportionately high S-ramp times (jerk times) with low acceleration times can lead to a faulty profile generation.

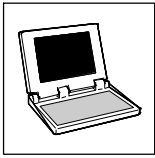
Example: $v = 100 \text{ mm/s}$, $a = 1000 \text{ mm/s}^2$

$$\Rightarrow t_{\text{acceleration}} = 0.1 \text{ s}$$

$$\Rightarrow T_{\text{jerk}} = 1 \text{ s}$$

Therefore only select plausible S-ramp times not higher than half the value of $t_{\text{acceleration}}$.

In the **LenzeMotionControl1.lib**, the function block *L_MCProfileGenerator* is used for profile generation. Using the parameter *wSProfileMode* of the function block *L_MCProfileGenerator* you can select a profile-optimised S-profile generation. If *wSProfileMode* = 1, there are no restrictions for the profile parameters. Profile-optimised S-profile generation requires a much longer processing time. (2-15).



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MCPositionCorrection - feed correction

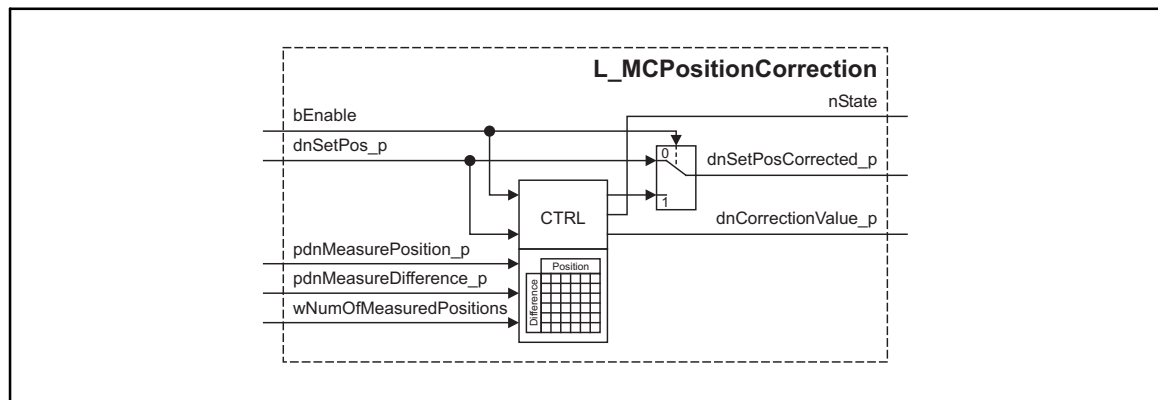
2.3

L_MCPositionCorrection - feed correction

To avoid linearity errors at feed mechanisms (e.g. leadscrews) or measuring systems, in machine tool applications these errors are usually compensated through corresponding correction values for the position setpoints.

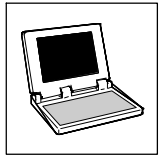
This FB provides corrected position setpoints from a measured-value table which lists the position errors for different measuring positions.

- The FB receives this table in two data fields (arrays).
- The corrected position setpoints can, for instance, be transferred to a profile generator for further processing.



⇒ Inputs (Variable type: VAR_INPUT)							
Identifiers	Data type	Signal type	Info/possible settings				
bEnable	Bool	binary	Activation of position setpoint correction <table><tr><td>FALSE</td><td>Position setpoint correction deactivated, <i>dnSetPos_p</i> is output 1:1 at the output <i>dnSetPosCorrected_p</i>.</td></tr><tr><td>TRUE</td><td>Position setpoint correction activated.</td></tr></table>	FALSE	Position setpoint correction deactivated, <i>dnSetPos_p</i> is output 1:1 at the output <i>dnSetPosCorrected_p</i> .	TRUE	Position setpoint correction activated.
FALSE	Position setpoint correction deactivated, <i>dnSetPos_p</i> is output 1:1 at the output <i>dnSetPosCorrected_p</i> .						
TRUE	Position setpoint correction activated.						
dnSetPos_p	Double integer	position	Setpoint position in [inc]				
pdnMeasurePosition_p	Pointer to double integer	pointer	Pointer to the first value of the data field (array) including the measuring positions				
pdnMeasure-Difference_p	Pointer to double integer	pointer	Pointer to the first value of the data field (array) including the position errors				
wNumOfMeasured-Positions	Word	-	Number of measuring positions in the data field (array) Only 2^n-values are permissible: 2, 4, 8, 16, 32, 64, 128, etc.				

Outputs ⇨ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Info	
nState	Integer	-	Status	
			0	Ok
			10	Number <i>wNumOfMeasurePositions</i> is no 2^n-value, the corrective calculation is not carried out.
			20	Correction is not active.
dnSetPosCorrected_p	Double integer	position	Corrected setpoint position in [inc] ● If <i>bEnable</i> = FALSE, <i>dnSetPosCorrected_p</i> = <i>dnSetPos_p</i> .	
dnCorrectionValue_p	Double integer	position	Correction value (interpolated position correction value)	



Creation of the measured-value table

For this, different measuring points (φ_{act}) are approached along the axis and the deviations ($\Delta\varphi$) from the selected setpoint positions (φ_{set}) are measured by means of a reference measuring system (e.g. laser measuring system):

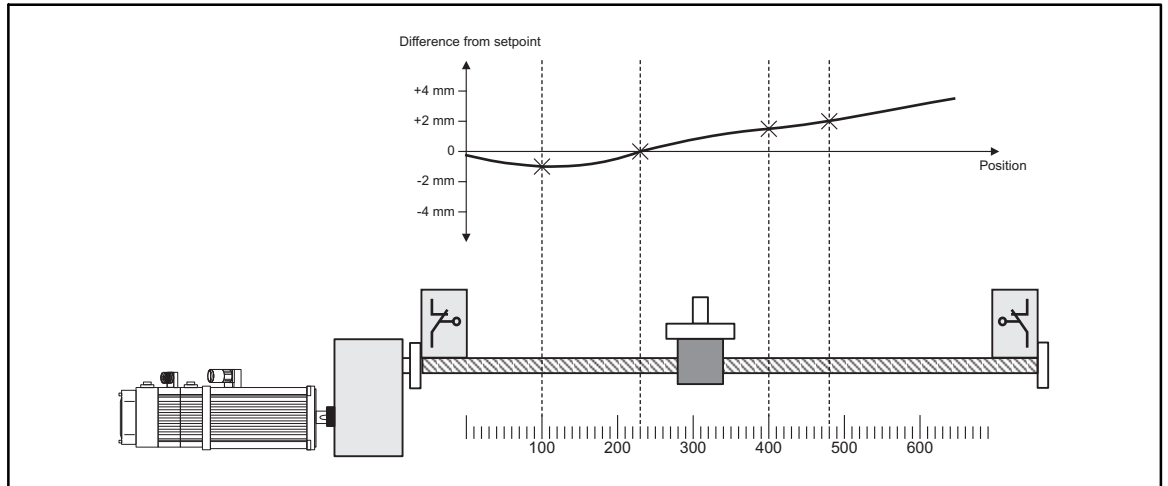


Fig. 2-1

Measurement of position errors at different measuring points

The measured values can be entered into a table:

Measurement No.	Approached setpoint position φ_{set}	Deviation from the setpoint position $\Delta\varphi = \varphi_{act} - \varphi_{set}$
1	100 mm	-1.00 mm
2	230 mm	0.00 mm
3	400 mm	1.80 mm
4	480 mm	2.00 mm
...	...	...
n	___ mm	___ mm

Transferring the measured-value table to the FB

Create two data fields (arrays) in the program. One for the measuring positions and one for the position errors:

VAR

adnMeasurePos: **ARRAY**[0..15] **OF** DINT;

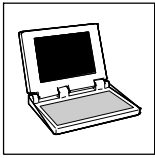
adnMeasureDif: **ARRAY**[0..15] **OF** DINT;

END_VAR

(* 1 rev = 65535 inc = 1 mm *)

```

adnMeasurePos[0] := 6553500;    (* 100 mm *)
adnMeasureDif[0] := -65535;     (* -1 mm *)
adnMeasurePos[1] := 15073050;   (* 230 mm *)
adnMeasureDif[1] := 0;          (* 0 mm *)
adnMeasurePos[2] := 26214000;   (* 400 mm *)
adnMeasureDif[2] := 117963;     (* 1.8 mm *)
adnMeasurePos[3] := 31456800;   (* 480 mm *)
adnMeasureDif[3] := 1310702;    (* 2 mm *)
...
    
```



Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCPositionCorrection - feed correction

Transfer one pointer to the first value of the array including the measured positions at the input *pdnMeasurePosition_p* and one pointer to the first value of the array including the position errors at the input *pdnMeasureDifference_p* to the FB:

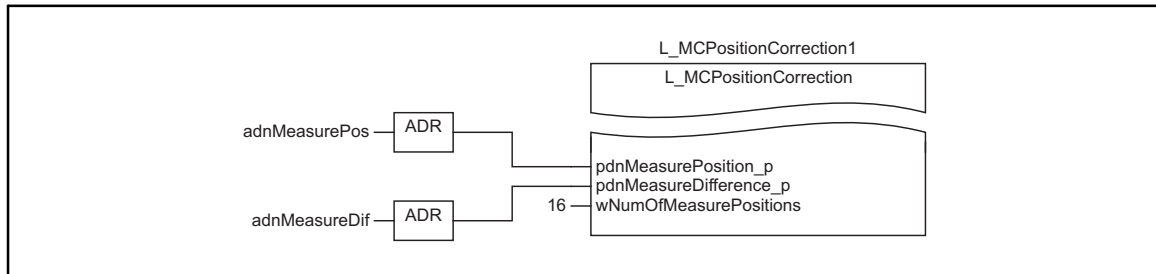


Fig. 2-2

Transferring the measured-value table to the FB



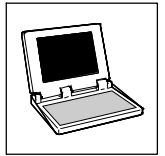
Notes

- The measuring positions need not be equidistant. You need not start at measuring position "0".
- The measuring positions have to be entered in ascending order. Identical measuring positions are not permitted.
- Only 2ⁿ-values are permissible as data field size, i.e. 16, 32, 64, 128, 256, etc.



Tip!

- If you assign user codes to the data fields, you can easily select the measuring points and position errors via codes/sub-codes and adapt them online during operation to the optimum results.
- The computing time of the FB increases linearly with the number of measuring points. To save computing time, an adequate data field size should be selected under *wNumOfmeasurePositions*.



Calculation of correction values

The FB calculates a correction value which is subtracted from the setpoint position. If the setpoint position is between two measured values, the deviation and the corresponding correction value will be determined through interpolation.

The below figure describes the interpolation method for the calculation of the corrected position value between two interpolation points $P_{(n-1)}$ and $P_{(n)}$, with: $P_{(n)} > P_{(n-1)}$

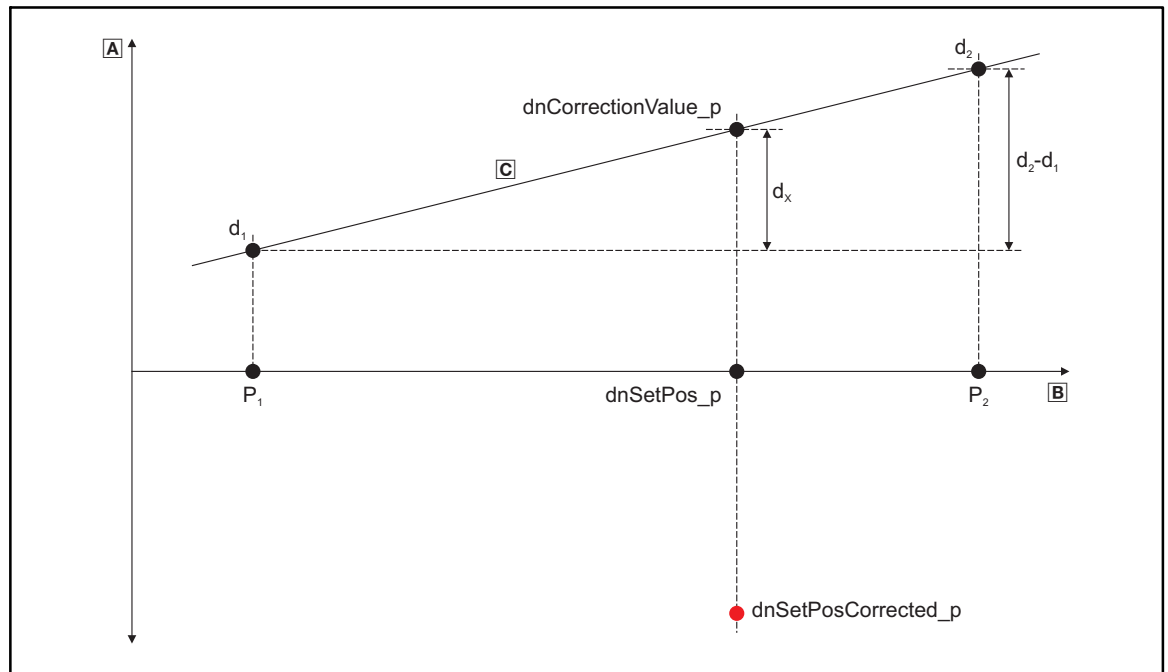


Fig. 2-3

Interpolation method

A Position error

B Measuring position

C Error interpolation line

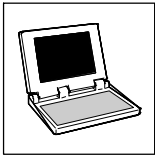
Line gradient (error interpolation line)	$= \frac{d_2 - d_1}{P_2 - P_1}$
d_x	$= \text{Line gradient} \cdot (\text{dnSetPos}_p - P_1)$ $= \frac{d_2 - d_1}{P_2 - P_1} \cdot (\text{dnSetPos}_p - P_1)$
Correction value $\text{dnCorrectionValue}_p$	$= d_x + d_1$ $= \frac{d_2 - d_1}{P_2 - P_1} \cdot (\text{dnSetPos}_p - P_1) + d_1$
Corrected position value $\text{dnSetPosCorrected}_p$	$= \text{dnSetPos}_p - \text{dnCorrectionValue}_p$ $= \text{dnSetPos}_p - \left(\frac{d_2 - d_1}{P_2 - P_1} \cdot (\text{dnSetPos}_p - P_1) + d_1 \right)$



Note!

The correction value is calculated by multiplying $(d_2 - d_1)$ and $(P_2 - P_1)$.

The result of the multiplication $(d_2 - d_1) \times (P_2 - P_1)$ must not be higher than $2^{31}-1$ or lower than -2^{31} . Otherwise, an internal overflow will occur and the interpolation result will not be correct!



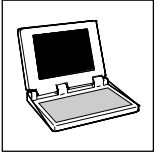
Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MCPositionCorrection - feed correction

Display of setpoint positions on keypads

Use the value of the input *dnSetPos_p* to display the setpoint position on a keypad (e.g. Lenze H410). The output *dnSetPosCorrected_p* sends the corrected value (depending on the status of the FB). The corrected value may differ from the real setpoint position.



2.4 L_MCPProfileGenerator - profile generator

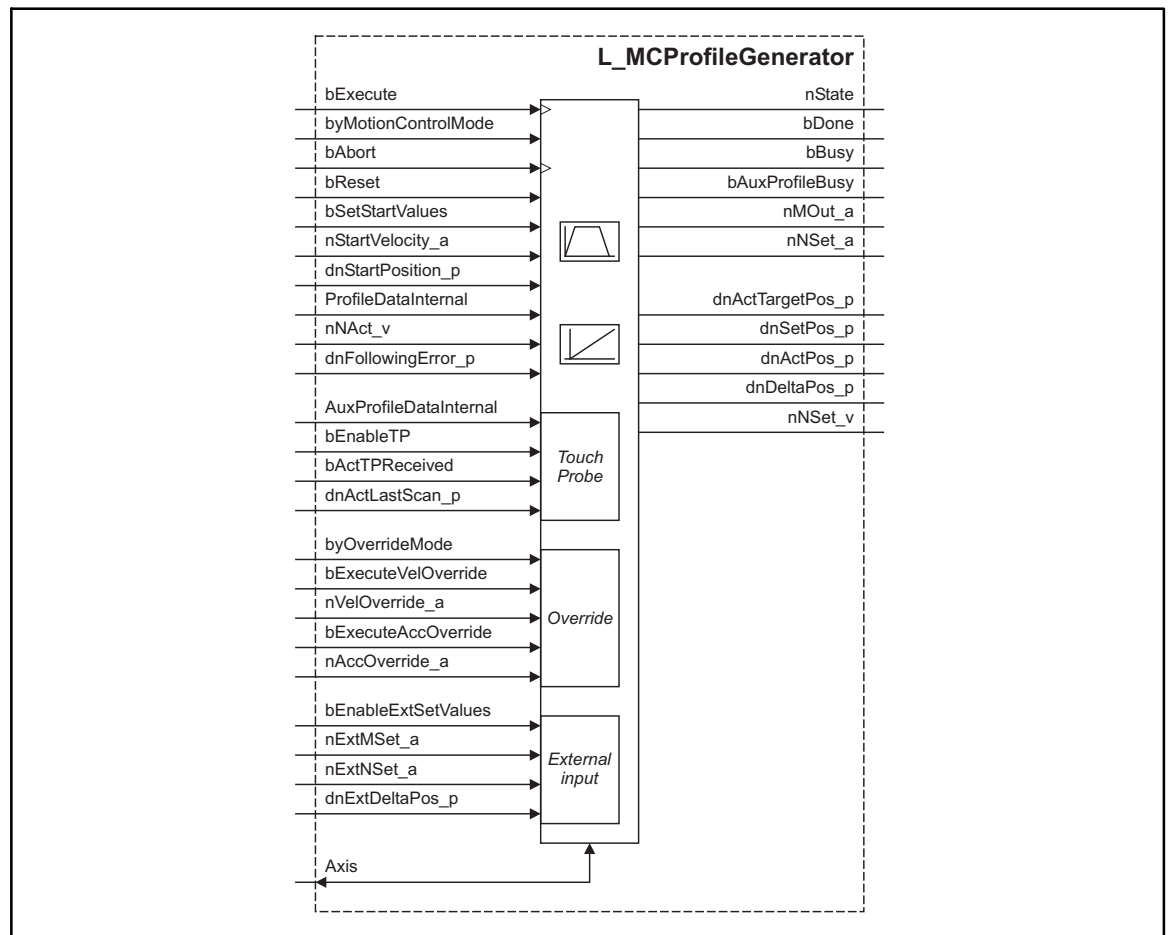
This FB can be used to approach a certain point via a linear speed profile.

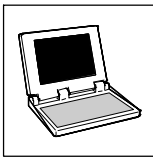


Note!

Due to its internal structure, the FB **L_MCPProfileGenerator** has to be called in a time-equidistant task (e.g. in a 5-ms task).

Caution: The cyclic task PLC_PRG is not time-equidistant!





Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

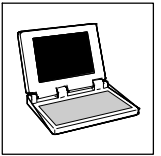
L_MCPProfileGenerator - profile generator

⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
bExecute	Bool	binary	Positioning If you want to start a new positioning process although the new position and the start position are identical, positioning will not be executed. <i>bDone</i> and <i>bBusy</i> are set to TRUE for a short time.
			FALSE → TRUE Activates positioning or constant traversing.
			FALSE → TRUE (again) The current positioning process is stopped and the new positioning process is started with the new position data.
byMotionControlMode	Byte	-	Positioning mode
			0 Absolute positioning
			1 Relative positioning - Internal path integrators are reset before the start, external path integrators are not reset. - Application: Rotary tables
			2 Constant traversing
			3 Relative positioning - max. 2^{30} - Path integrators are not reset before the start.
			Positioning modes with touch probe (remaining path positioning):
			10 Absolute positioning with relative remaining path positioning
			11 Relative positioning with relative remaining path positioning - Internal path integrators are reset before the start, external path integrators are not reset. - Application: Rotary tables
			12 Constant traversing with relative remaining path positioning
			13 Relative positioning with relative remaining path positioning - max. 2^{30} - Path integrators are not reset before the start.
			14 Absolute positioning with absolute remaining path positioning
			15 Relative positioning with absolute remaining path positioning Path integrators are not reset before the start.
bAbort	Bool	binary	Cancellation of positioning This input has a higher priority than <i>bExecute</i>.
			FALSE → TRUE Activates the cancellation of a positioning process or constant traversing, i.e. the system is decelerated to 0 along the deceleration ramp of the profile.
bReset	Bool	binary	Reset of positioning This input has a higher priority than <i>bExecute</i>, <i>bAbort</i> and <i>bSetStartValues</i>.
			TRUE - The setpoint angle integrator has the same value as the actual angle integrator. - Positioning is stopped. - The output speed <i>nNset_a</i> is immediately set to "0". Caution: This causes a sudden speed change. When the motor control is active, this can lead to heavy torque impulses! Recommendation: Set quick stop (QSP) before positioning is reset.
bSetStartValues	Bool	binary	Loading of the start values for positioning This input has a higher priority than <i>bExecute</i> and <i>bAbort</i>.
			FALSE/TRUE - The profile generator is uploaded with the start speed <i>nStartVelocity_a</i>. Caution: This causes a sudden speed change. When the motor control is active, this can lead to heavy torque impulses! Recommendation: Set quick stop (QSP) before loading the start values. - The current positioning process is stopped immediately. - Setpoint angle integrator and actual angle integrator are loaded with the value of <i>dnStartPosition_p</i>.

Function library *LenzeMotionControl1.lib*

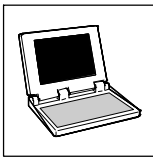
Function blocks for positioning tasks

L_MCPProfileGenerator - profile generator



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
			TRUE/FALSE The following error at the input <i>dnFollowingError</i> is deducted from the actual value integrator
nStartVelocity_a	Integer	analog	Start speed - Value will be accepted if <i>bSetStartValues</i> = TRUE. 16384 equals the maximum speed under C0011.
dnStartPosition_p	Double integer	position	Start position Value will be accepted if <i>bSetStartValues</i> = TRUE.
ProfileDataInternal	MC_PROFILE_INTERNAL_MEASURING	-	Positioning profile Data structure with elements which describe the entire profile.
nNact_v	Integer	velocity	Actual speed for the actual angle integrator In general, this input is to be connected with the system variable <i>MCTRL_nNact_v</i> (speed of the feedback system) of the SB MCTRL.
dnFollowingError_p	Double integer	position	Following error After a TRUE/FALSE transition at the input <i>bSetStartValues</i>, the value will be deducted from the actual value integrator.
AuxProfileDataInternal	MC_PROFILE_INTERNAL_MEASURING	-	Additional positioning profile for remaining path positioning Data structure with elements which describe the entire profile.
bEnableTP	Bool	binary	Activation of touch probe signal processing TRUE Touch probe signal processing is active.
bActTPReceived	Bool	binary	Input for touch probe TRUE Touch probe signal has been detected.
dnActLastScan_p	Double integer	position	Phase difference in [inc] between latch time and start time of the task
byOverrideMode	Byte	-	Definition of the 100 %-reference for override parameters 0 16384 = 100 % = $V_{\max}/Acc_{\max}$ of the axis parameters 1 16384 = 100 % = Vel/Acc of the profile parameters
bExecuteVelOverride	Bool	binary	Activation of speed override TRUE Speed override is active.
nVelOverride_a	Integer	velocity	Override speed 16384 = $v_{\max}$
bExecuteAccOverride	Bool	binary	Activation of acceleration override TRUE Acceleration override is active.
nAccOverride_a	Integer	velocity	Override acceleration 16384 = $a_{\max}$
bEnableExtSetValues	Bool	binary	Changeover to external setpoints TRUE The external setpoints are directly transferred to the corresponding outputs: <i>nExtMSet_a</i> ⇒ <i>nMSet_a</i> <i>nExtNSet_a</i> ⇒ <i>nNSet_a</i> <i>dn_ExtDeltaPos_p</i> ⇒ <i>dnDeltaPos_p</i>
nExtMSet_a	Integer	analog	External setpoint for torque pre-control
nExtNSet_a	Integer	analog	External setpoint for setpoint speed
dnExtDeltaPos_p	Integer	position	External setpoint for position difference

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info/possible settings
Axis	AXIS_REF		Machine parameters - Data structure including the machine parameters in the internal measuring system. - Connect this input with a global variable which is written by FB L_MCMachineData.



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

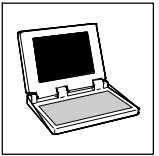
L_MCProfileGenerator - profile generator

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nState	Integer		Indicates the status of positioning.
			Messages:
			0 No action
			1 Positioning completed
			10 Acceleration phase is active
			11 Constant speed phase is active
			12 Deceleration phase is active
			13 Reversing phase is active
			100 Reset is active (<i>bReset</i>)
			101 Start values are loaded
			102 Positioning has been stopped (<i>bAbort</i>)
			103 External setpoint interface activated
			200 Positioning with touch probe (remaining path positioning) has been completed, but no touch probe signal has been detected.
			300 Overflow of internal path integrator
			Errors:
			-200 Profile <i>ProfileDataInternal</i> is invalid: Parameter acceleration = 0
			-201 Profile <i>ProfileDataInternal</i> is invalid: Parameter deceleration = 0
			-202 Profile <i>ProfileDataInternal</i> is invalid: Parameter speed = 0
			-210 Profile <i>AuxProfileDataInternal</i> is invalid: Parameter acceleration = 0
			-211 Profile <i>AuxProfileDataInternal</i> is invalid: Parameter deceleration = 0
			-212 Profile <i>AuxProfileDataInternal</i> is invalid: Parameter speed = 0
			-300 Positioning with the selected profile parameters is not plausible.
bDone	Bool	binary	Message "Positioning hat been executed"
			TRUE Positioning has been executed.
			TRUE (pulse) <i>bDone</i> is set to TRUE for one call-up cycle if the profile is changed in mode Overchange.
bBusy	Bool	binary	Message "Positioning is active"
			TRUE Positioning is active.
			TRUE (pulse) <i>bBusy</i> is set to TRUE for one call-up cycle if target and start position are identical at the start of absolute positioning and the positioning process has therefore not been started.
bAuxProfileBusy	Bool	binary	Message "Remaining path positioning is active"
			TRUE Remaining path positioning is active.

Function library *LenzeMotionControl1.lib*

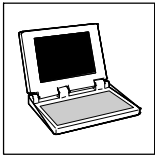
Function blocks for positioning tasks

L_MCPProfileGenerator - profile generator



Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nMOut_a	Integer	analog	Output signal for torque pre-control
nSet_a	Integer	analog	Speed setpoint for speed controller 16384 equals the maximum speed under C0011. - In general, this output is to be connected with the system variable <i>MCTRL_nSet_a</i> of the SB MCTRL.
dnActTargetPos_p	Double integer	position	Target angle position of the current profile 65536 equals one revolution on the motor side 0 with external setpoint selection (<i>bEnableExtSetValues</i> = TRUE) - Indicates either the relative traverse path or the absolute target position (depending on the positioning mode).
dnSetPos_p	Double integer	position	Setpoint angle position 65536 equals one revolution on the motor side - The setpoint integrator can overflow.
dnActPos_p	Double integer	position	Actual angle position 65536 equals one revolution on the motor side - The actual integrator can overflow.
dnDeltaPos_p	Double integer	position	Position difference output for position controller. In general, this output is to be connected with the system variable <i>MCTRL_dnPosSet_p</i> of the SB MCTRL. The phase controller must be activated for this (<i>MCTRL_bPosOn_b</i> = TRUE).
nSet_v	Integer	velocity	Speed setpoint in increments per millisecond 16384 equals 15000 rpm - Only to be used for encoder systems on the motor side! - If a load-side encoder system is used (gearbox factor unequal 1), speed setpoint 0 will be output

Parameters				
Identifiers	Data type	Signal type	Info/possible settings	
wSProfileMode	Word		Mode S-profile generation	
			0	Run-time-optimised (default setting)
			1	S-profile-optimised Requires a much longer processing time
wDecOverride	Word		Activation of deceleration override	
			0	Deceleration override deactivated
			1	The override deceleration (value at the input <i>nAccoverride_a</i> referred to the maximum deceleration) is accepted once when the profile is started.

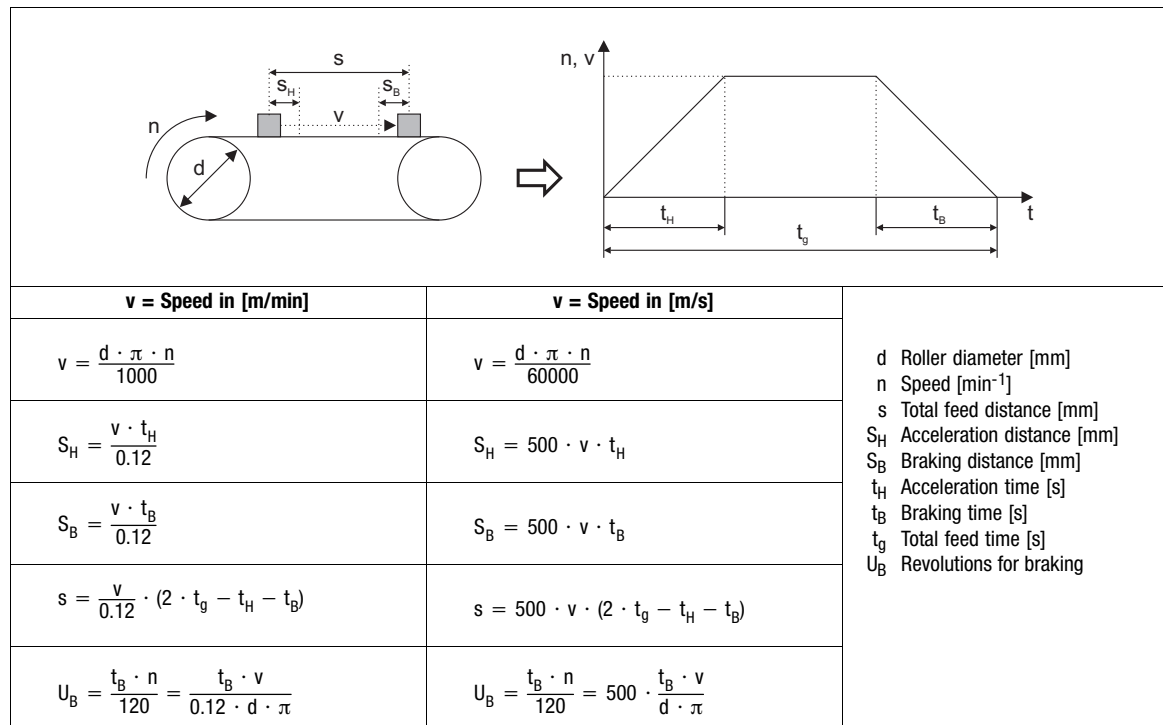


Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCPProfileGenerator - profile generator

Formulas for positioning



Transfer of machine parameters

The input/output *axis* is used to transfer the machine data to the profile generator.

- Connect the input *axis* with a global variable which is written by the FB **L_MCMachineData**. (2-4)

Selection of the positioning profile

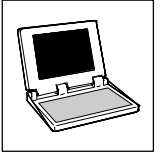
The input *ProfileDataInternal* is used to transfer the positioning profile to the profile generator.

- The FB expects a data structure (*MC_PROFILE_INTERNAL_MEASURING*) with elements which include the positioning data according to the internal measuring system of the profile generator.



Tip!

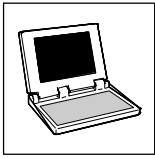
- Information about the function blocks which convert physical units into information that can be processed by the internal measuring system of the profile generator can be found in chapter 3, "Function blocks for unit conversion". (3-1)
- Information about the data structures for the machine parameters and positioning profiles can be found in chapter 4, "Structured data types". (4-1)



Standard profile generator modes

The profile generator can use four different positioning modes:

Mode		Info	Notes
0	Absolute positioning	This mode is based on absolute zero. In general, zero is found by homing and transferred to the actual position integrator of the profile generator. All profiles refer to this zero.	
1	Relative positioning	This mode does not select an absolute position, but uses distances.	Before every positioning process, the internal path integrators are set to 0. It is not possible to run an absolute positioning process after a relative positioning process. Example: If a relative positioning process in mode 1 is activated after an absolute positioning process in mode 0, the 0 position could not be found after the system had switched back to mode 0. The axis then uses the last start position of mode 1 as a reference.
2	Constant traversing	This mode does not approach a defined position, but follows the profile.	- Acceleration and deceleration are based on profile values. - The traversing direction is defined by the sign of the traversing speed. - Traversing is started by a FALSE-TRUE transition at <i>bExecute</i>. - Traversing can be stopped with a FALSE-TRUE transition at <i>bAbort</i>.
3	Relative positioning (max. 2 ³⁰)	Like mode 1, "Relative positioning"; only the internal path integrators are not reset, but continue to be incremented.	The path integrators are limited to 2 ³⁰ .



Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCProfileGenerator - profile generator

Remaining path positioning

Just as with a standard positioning process, with remaining path positioning the drive first processes the standard profile in the corresponding mode.

If touch probe is identified during the traversing process, the drive automatically changes to the profile at *AuxProfileDataInternal*. The profile generator then continues the positioning process with the new parameters in the corresponding positioning mode.

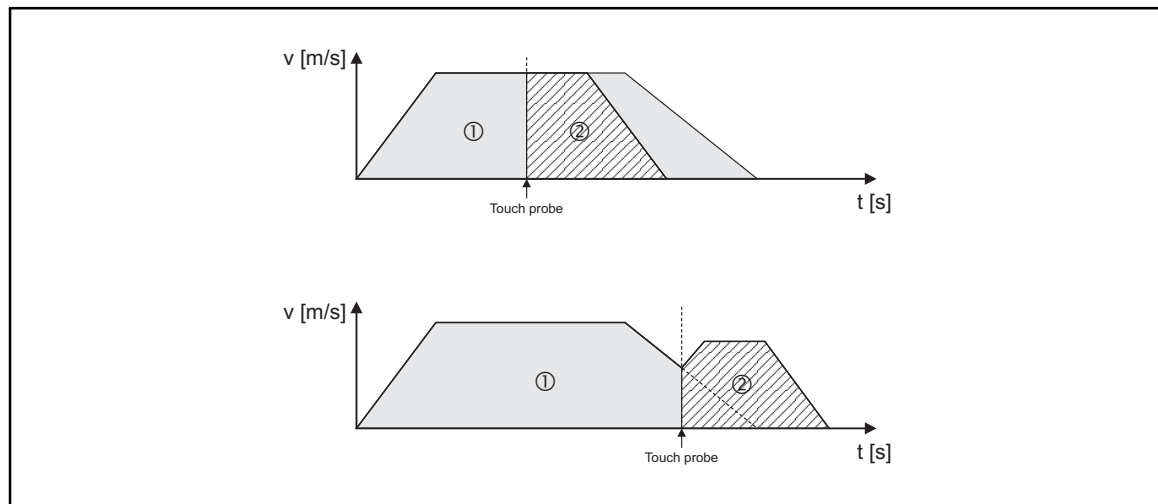
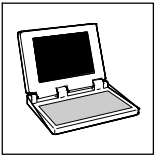


Fig. 2-4

Example: Remaining path positioning after touch probe identification

- ① Standard profile (*ProfileDataInternal*)
- ② Additional profile for remaining path positioning (*AuxProfileDataInternal*)

- If remaining path positioning is active TRUE will be displayed at the output *bAuxProfileBusy*.
- If touch probe is activated once again at input *bActTpReceived* the positioning process starts again, i.e. the second remaining path positioning profile is processed once again from the current position. This is only possible if the corresponding positioning mode and the remaining path positioning profile are still selected.
- Via a FALSE-TRUE transition at input *bExecute* you can cancel an active remaining path positioning process and change into another positioning mode.



Profile generator modes for remaining path positioning

The profile generator uses the following modes for remaining path positioning with touch probe:

Mode		Info
10	Absolute positioning with relative remaining path positioning	See mode 0, "Absolute positioning". • If touch probe is detected the profile generator automatically changes to the additional profile <i>AuxProfileDataInternal</i> and executes a relative positioning process with the corresponding profile parameters.
11	Relative positioning with relative remaining path positioning	See mode 1, "Relative positioning". • If touch probe is detected the profile generator automatically changes to the additional profile <i>AuxProfileDataInternal</i> and executes a relative positioning process with the corresponding profile parameters.
12	Constant traversing with relative remaining path positioning	See mode 2, "Constant traversing". • If touch probe is detected the profile generator automatically changes to the additional profile <i>AuxProfileDataInternal</i> and executes a relative positioning process with the corresponding profile parameters.
13	Relative positioning with relative remaining path positioning	See mode 3, "Relative positioning". • If touch probe is detected the profile generator automatically changes to the additional profile <i>AuxProfileDataInternal</i> and executes a relative positioning process with the corresponding profile parameters. • In this mode, the internal path integrators are not reset, but continue to be incremented (internally limited to 2^{30}).
14	Absolute positioning with absolute remaining path positioning	See mode 0, "Absolute positioning". • If touch probe is detected the profile generator automatically changes to the additional profile <i>AuxProfileDataInternal</i> and executes an absolute positioning process with the corresponding profile parameters.
15	Relative positioning with absolute remaining path positioning	See mode 3, "Relative positioning". • If touch probe is detected the profile generator automatically changes to the additional profile <i>AuxProfileDataInternal</i> and executes an absolute positioning process with the corresponding profile parameters. • In this mode, the internal path integrators are not reset, but continue to be incremented (internally limited to 2^{30}).



Tip!

The FB **L_MCTouchProbeCtrl** can be used for touch probe signal pre-processing. The FB can organise up to five touch probe signals and allows you to define position windows to hide touch probe signals depending on the current position.

Profile change during positioning (override)

"Override" means a change of profile parameters and their acceptance during positioning.

The geometry of the traversing profile changes through an override to ensure exact positioning even if the speed is changed during positioning (speed override):

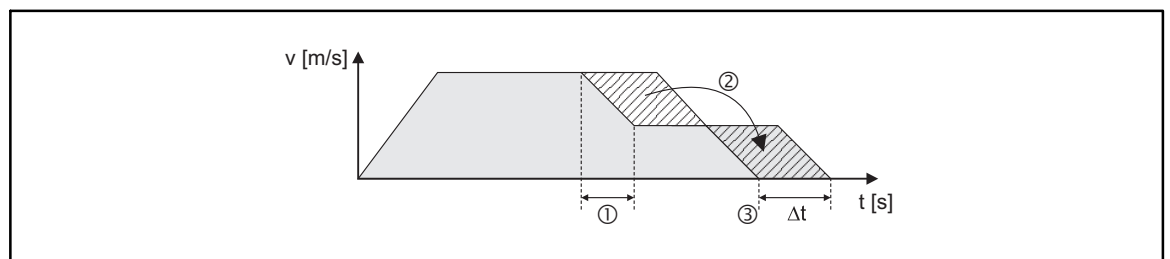
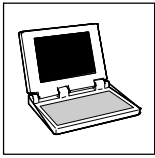


Fig. 2-5

Override (here: speed override)

- ① Speed is reduced during positioning.
- ② The defined position can only be reached if the missing area is "added" to the profile.
- ③ The reduced speed results in a longer positioning process (Δt).



Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCProfileGenerator - profile generator

Definition of the reference value for override

The reference for the override values is defined under *g_byOverrideMode*:

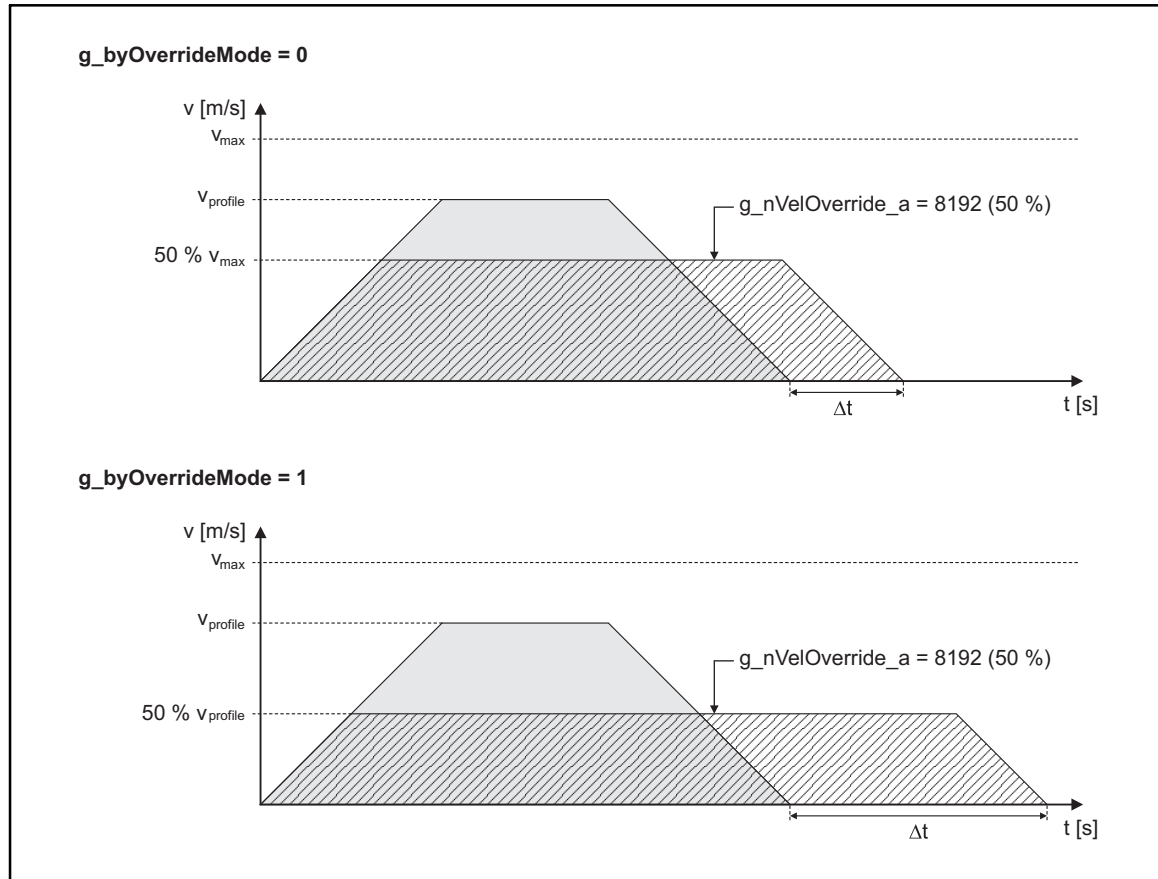


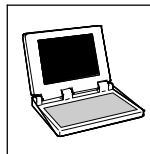
Fig. 2-6

Definition of the reference value for override (here: speed override)



Tip!

Override mode "1" can be used to reduce the clock pulse rate of the tool, e.g. for machine set-up. In this mode, the profile remains unchanged, only profile processing takes longer.



Execution of an override

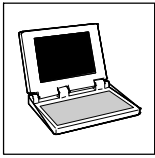
The FB **L_MCProfileGenerator** can be used to change the speed ("speed override") and acceleration ("acceleration override").

- The speed override is activated by setting *bExecuteVelOverride* to TRUE.
- The acceleration override is activated by setting *bExecuteAccOverride* to TRUE.
- The speed override is selected via the input *nVelOverride_a*, the acceleration override via the input *nAccOverride_a*:

Identifiers	Data type	Info/possible settings
nVelOverride_a	Integer	1 ... 16384 Speed (referred to $v_{\max}$) • $16384 \equiv v_{\max}$ • The speed can only be changed in the acceleration phase and during constant traversing (trapezium). • If a speed override is executed during constant traversing the drive decelerates or accelerates until the new speed is reached. The current profile parameters are used for acceleration/deceleration. If an acceleration override is also activated the drive accelerates with the new acceleration value.
nAccOverride_a	Integer	1 ... 16384 Acceleration (referred to $a_{\max}$) • $16384 \equiv a_{\max}$ • The acceleration can only be changed in the acceleration phase.

The parameter *wDecoverride* can be used to activate the deceleration override. If the deceleration override is activated (*wDecoverride* = 1), the override deceleration (value at the input *nAccoverride_a* referred to the maximum deceleration) is accepted once when the profile is started. During the subsequent course of the positioning process, the profile deceleration will not be changed again.

Parameters							
Identifiers	Data type	Signal type	Info/possible settings				
wDecOverride	Word		Activation of deceleration override <table><tr><td>0</td><td>Deceleration override deactivated</td></tr><tr><td>1</td><td>When the profile is started, the profile deceleration is set once to the deceleration override value (value at the input <i>nAccoverride_a</i> referred to the maximum deceleration).</td></tr></table> Definition of the 100% reference for deceleration override parameters: byOverrideMode = 0: 16384 ≡ 100 % ≡ Accmax from the axis parameters byOverrideMode = 1: 16384 ≡ 100 % ≡ Dec from the profile parameters	0	Deceleration override deactivated	1	When the profile is started, the profile deceleration is set once to the deceleration override value (value at the input <i>nAccoverride_a</i> referred to the maximum deceleration).
0	Deceleration override deactivated						
1	When the profile is started, the profile deceleration is set once to the deceleration override value (value at the input <i>nAccoverride_a</i> referred to the maximum deceleration).						



Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCProfileGenerator - profile generator



Tip!

If an override is activated changes at the inputs *nVelOverride_a* or *nAccOverride_a* **will have a direct effect** on the profile.

If the override is deactivated by resetting *bExecuteVelOverride* to FALSE or *bExecuteAccOverride* to FALSE profile processing will be completed with the current speed value.

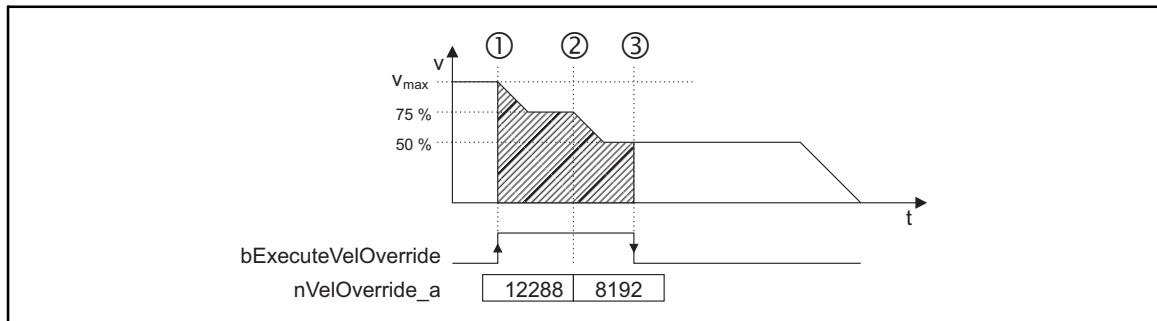
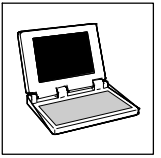


Fig. 2-7

Override (here: speed override)

- ① Speed override is activated (*bExecuteVelOverride* = TRUE);
Speed is reduced to 75 % of $v_{\max}$ (*nVelOverride_a* = 12288)
- ② Speed is once again reduced to 50% of $v_{\max}$ (*nVelOverride_a* = 8192)
- ③ Speed override is deactivated (*bExecuteVelOverride* = FALSE)



Profile linkage with overchange

If profiles are linked to each other by means of an overchange, the drive traverses at a selected final speed (unequal "0") from one traversing profile ① to another ②:

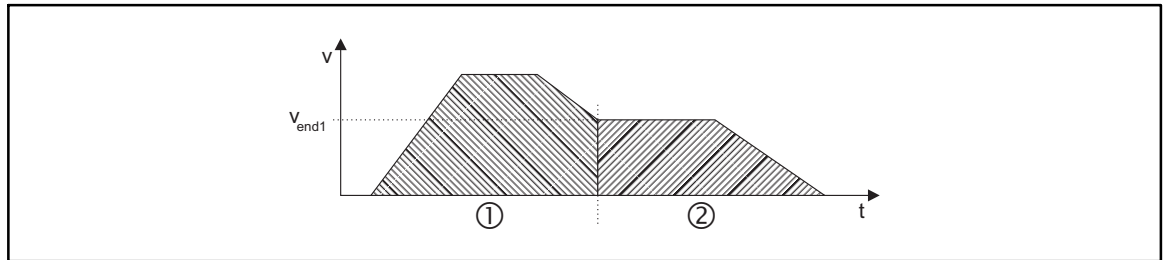
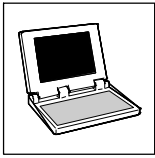


Fig. 2-8

Example: Profile linkage with overchange

- ① First traversing profile with final speed v_{end1} unequal "0".
- ② Second traversing profile with initial speed = final speed of the first traversing profile.

- The final speed v_{end} is selected in the corresponding positioning profile under v_{end} in [units] and will be transferred to the profile generator as soon as the positioning process is started.
- The final speed v_{end} is limited to the value of the positioning speed set via the profile parameters.
- When the positioning process has been completed and the selected final speed v_{end} has been reached the drive automatically starts another positioning process with the profile selected at *ProfileDataInternal*. *bExecute* need not be set again.
- If the final speed of the following profile is also unequal "0", the profile linkage will be continued until a profile with a final speed equal "0" will be processed. After this, *bExecute* must be set to TRUE to start another positioning process.
- If the target position of a profile is within a range which cannot be reached due to the final speed of the previous profile and the deceleration of the current profile, the drive reverses.
- After remaining path positioning (activated through touch probe), the profile will also be linked to the profile selected at *ProfileDataInternal* if the final speed of the remaining path positioning process is unequal "0" (profile data at *AuxProfileDataInternal*).



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

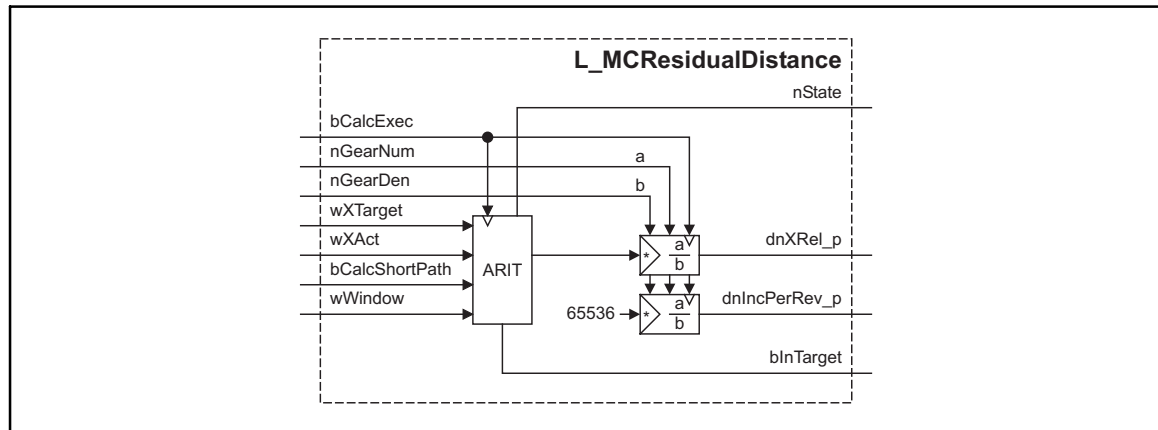
L_MCResidualDistance - remaining path calculation

2.5

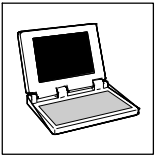
L_MCResidualDistance - remaining path calculation

This FB calculates the distance to be traversed until the target position is reached for rotary-axis applications.

- The remaining path calculation uses absolute values to reach an optimum resolution for the remaining path. The current position and the target position have to be selected in the machine measuring system between 0 ... 65535.
- The remaining path is evaluated with the gear factor and indicated as a distance in motor increments.
- In addition, the number of increments which equal one revolution in the target measuring system is output.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
bCalcExec	Bool	binary	Start of the calculation
			FALSE FB deactivated (nState = 0)
			FALSE → TRUE Start calculation
			TRUE Check, if the target has been reached
nGearNum	Integer	-	Gear factor numerator
nGearDen	Integer	-	Gear factor denominator
wXTarget	Word	-	Target position in the machine measuring system (0 ... 65535 [units])
wXAct	Word	-	Current position in the machine measuring system (0 ... 65535 [units])
bCalcShortPath	Bool	binary	Calculation of the shortest remaining path
			TRUE The FB always calculates the shortest remaining path, i.e. negative results are also possible in this mode.
wWindow	Word	-	Width of the message window "Target position reached" • Due to the symmetrical structure of the window around the target position, odd values will be rounded off to even values. • If the current position is within the window <i>bInTarget</i> will be set to TRUE.



Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nState	Integer	-	Status
			-10 Calculation has not been carried out because gear factor denominator = 0 or gear factor numerator = -32768 ($dnXRel_p = 0$; $dnIncPerRev_p = 0$; $blnTarget = FALSE$).
			0 Input parameters ok, calculation has been carried out.
			10 FB is not active.
			20 Shortest remaining path has been calculated.
dnXRel_p	Double integer	position	Motor increments until target position
dnIncPerRev_p	Double integer	position	Motor increments per 1 rev. of target measuring system
blnTarget	Bool	binary	Message "Target position reached"
			TRUE The current position is indicated in the window selected under wWindow.

Definition of the message window "Target position reached"

Under *wWindow* you can define a window which shows if the target position has been reached.

- If the current position is within the window *blnTarget* will be set to TRUE.
- The input *bCalcExec* must be set to TRUE after the FALSE-TRUE transition to start the calculation.
- Due to the symmetrical structure of the window around the target position, odd values for *wWindow* will be rounded off to even values (see ②).

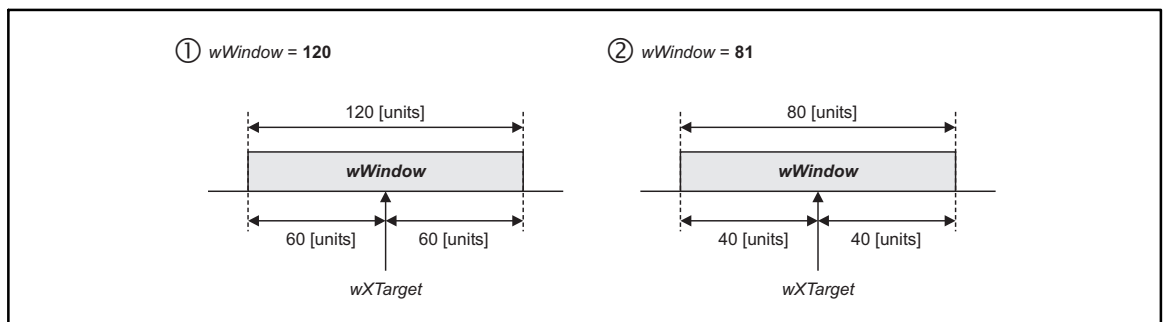


Fig. 2-9

Symmetrical structure of *wWindow* around the target position *wXTarget*



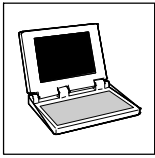
Note!

The output *blnTarget* works without hysteresis.

Calculation of the shortest remaining path

If *bCalcShortPath* is set to TRUE the shortest remaining path will be calculated.

- In this case, the calculated distance can also be negative.
- Examples can be found on the next page under "Remaining path calculation".



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MCResidualDistance - remaining path calculation

Calculation of the remaining path

A FALSE-TRUE transition at *bCalcExec* starts the calculation of the output values as follows:

- If $wXAct > wXTarget$:

<i>bCalcShortPath</i> = FALSE	$dnXRel_p = (65536 - wXAct + wXTarget) \cdot \frac{nGearNum}{nGearDen}$ $dnIncPerRev_p = 65536 \cdot \frac{nGearNum}{nGearDen}$
<i>bCalcShortPath</i> = TRUE	$dnXRel_p = - (wXAct - wXTarget) \cdot \frac{nGearNum}{nGearDen}$ $dnIncPerRev_p = 65536 \cdot \frac{nGearNum}{nGearDen}$

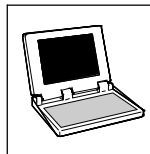
- If $wXAct \leq wXTarget$:

<i>bCalcShortPath</i> = FALSE	$dnXRel_p = (wXTarget - wXAct) \cdot \frac{nGearNum}{nGearDen}$ $dnIncPerRev_p = 65536 \cdot \frac{nGearNum}{nGearDen}$
<i>bCalcShortPath</i> = TRUE	$dnXRel_p = - (65536 - wXTarget + wXAct) \cdot \frac{nGearNum}{nGearDen}$ $dnIncPerRev_p = 65536 \cdot \frac{nGearNum}{nGearDen}$



Note!

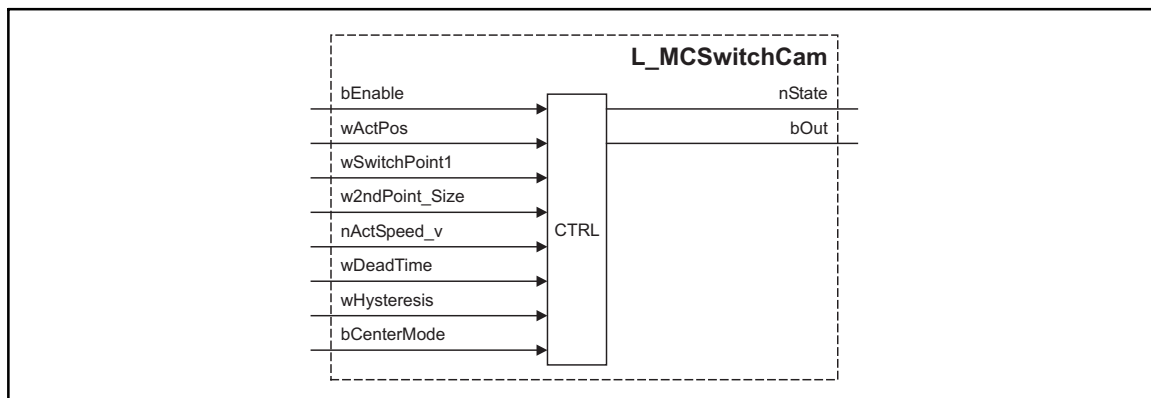
This FB does not process residual values.



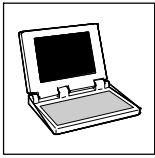
2.6 L_MCSwitchCam - switching points (rotary axis)

This FB is used to implement switching points ("cams") on a rotary axis, i.e. digital switches whose binary state (FALSE/TRUE) depends on the actual position.

- Switching points are used to start, for instance, peripheral devices such as spray jets or cutters depending on the tool position.
- The FB furthermore supports the compensation of delay times of external switching elements (dead time compensation).



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
bEnable	Bool	binary	Activation of the FB TRUE Function block activated.
wActPos	Word	position	Actual position in [unit] • Entry in machine measuring system: 0 ... 65535 inc $\equiv$ 0° ... 360° of the rotary axis
wSwitchPoint1	Word	position	Position of the first switching point in [unit] • Entry in machine measuring system: 0 ... 65535 inc $\equiv$ 0° ... 360° of the rotary axis
w2ndPoint_Size	Word	position	With <i>bCenterMode</i> = FALSE: Position of the second switching point in [unit] With <i>bCenterMode</i> = TRUE: Size of the switching window in [unit] • Due to the symmetrical structure of the window around the first switching point, odd values will be rounded off to even values.
nActSpeed_v	Integer	velocity	Actual speed in [inc/ms] • 16384 $\equiv$ 15000 inc/ms
wDeadTime	Word	-	Dead time in [μ s] for dead time compensation • Selection in 10 μ S-steps • The switching position resulting from this is not checked for plausibility.
wHysteresis	Word	-	Switching hysteresis in [unit] • Entry in machine measuring system: 0 ... 16000 inc (limited to 16000 inc)
bCenterMode	Bool	binary	Definition how the selection <i>w2ndPoint_Size_p</i> is to be interpreted. FALSE <i>w2ndPoint_Size</i> defines the second switching point. TRUE <i>w2ndPoint_Size</i> defines the size of the switching window.



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MGSwitchCam - switching points (rotary axis)

Outputs ⇨ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Info	
nState	Integer	binary	Status	
			10	FB is not active.
			0	Ok
			-10	The values from which the switching points are calculated are not plausible (<i>wSwitchPoint1</i> , <i>w2ndPoint_Size</i> , <i>wHysteresis</i>).
bOut	Bool	binary	Switching output	
			TRUE	The actual position is within the defined switching window.

Definition of the switching range

Depending on the setting of *bCenterMode*, the switching range can be defined via the start and end position or by entering the center point/size of the switching range on the rotary axis:

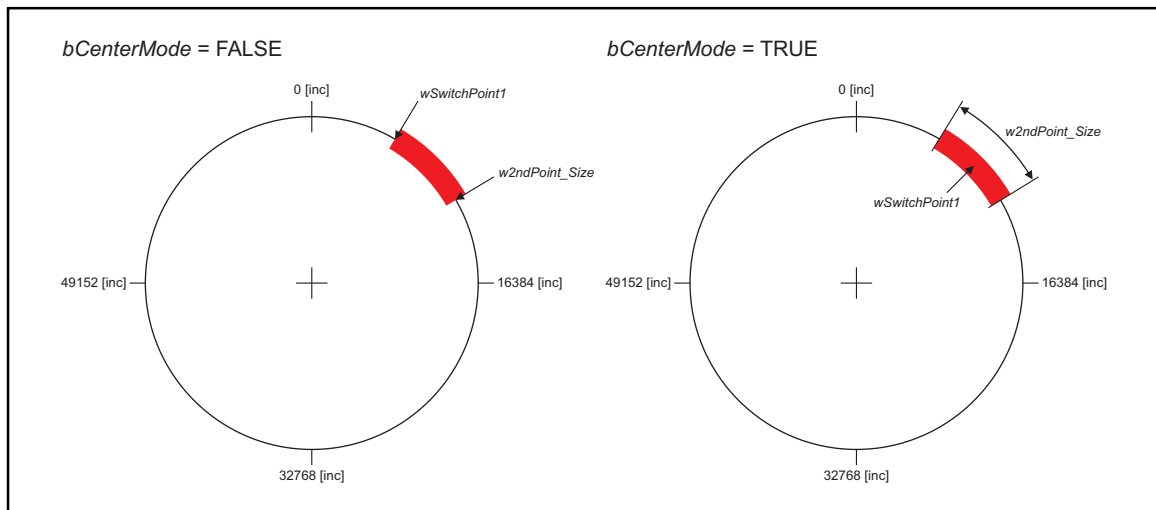
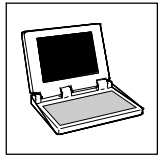


Fig. 2-10

Definition of the switching range



Dead time compensation

The dead time of external switching elements can be compensated by a corresponding dead time selection at $wDeadTime$. The FB calculates a preliminary stop position from the selected dead time value and the current speed, i.e. the switching output $bOut$ will be set earlier.

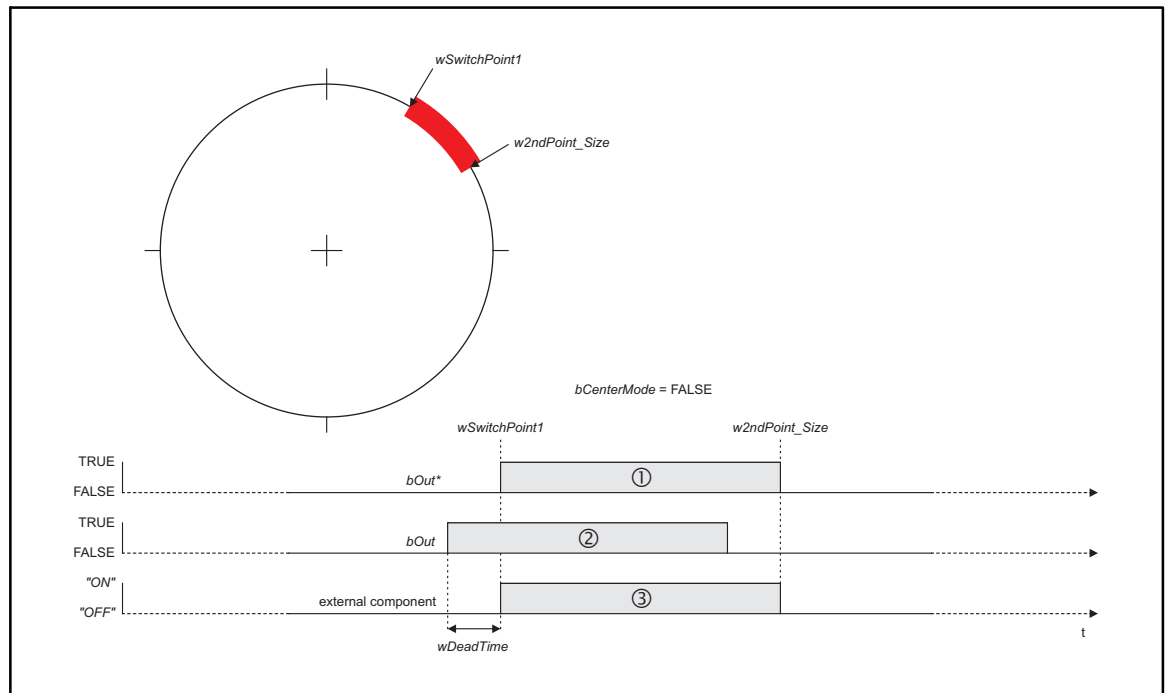


Fig. 2-11

Dead time compensation

- ① Required switching range
- ② Switching with dead time compensation
- ③ Switching of the external switching element (with dead time)

Switching hysteresis

Permanent status changes of $bOut$ resulting from changes of actual position values due to mechanical disturbances at the axis can be avoided by selecting a hysteresis at the input $wHysteresis$.

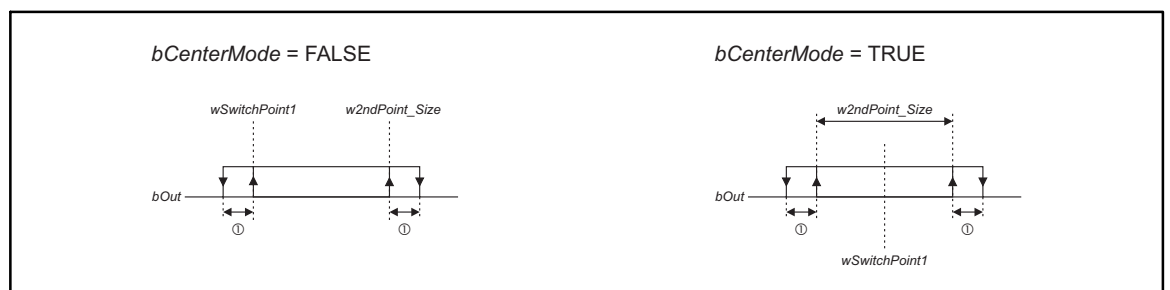
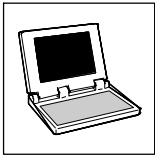


Fig. 2-12

Switching hysteresis

- ① $wHysteresis$



Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

L_MCSwitchCam - switching points (rotary axis)

Implementation of several cams

By calling several instances of the function block you can implement a corresponding number of cams on a rotary axis:

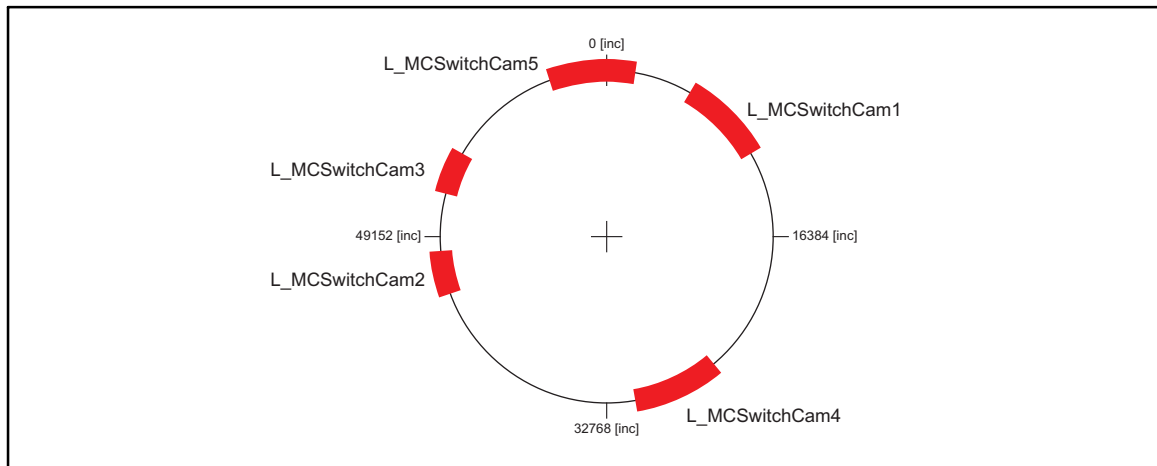
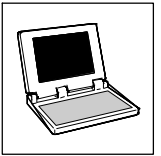


Fig. 2-13

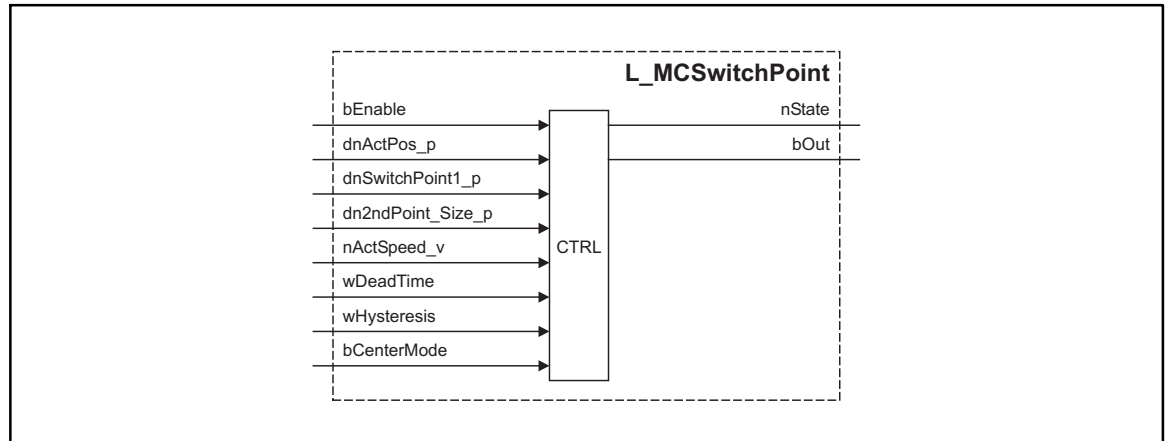
Implementation of several cams by calling several FB instances



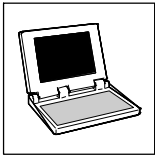
2.7 L_MCSwitchPoint - switching points

This FB is used to implement switching points, i.e. digital switches whose binary state (FALSE/TRUE) depends on the actual position.

- Switching points are used to start, for instance, peripheral devices such as spray jets or cutters depending on the tool position.
- The FB furthermore supports the compensation of delay times of external switching elements (dead time compensation).



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
bEnable	Bool	binary	Activation of the FB TRUE Function block activated.
dnActPos_p	Double integer	position	Actual position in [inc]
dnSwitchPoint1_p	Double integer	position	Position of the first switching point in [inc]
dn2ndPoint_Size_p	Double integer	position	With <i>bCenterMode</i> = FALSE: Position of the second switching point in [inc] • <i>dn2ndPoint_Size_p</i> must be higher than <i>dnSwitchPoint1_p</i> . With <i>bCenterMode</i> = TRUE: Size of the switching window in [inc] • Due to the symmetrical structure of the window around the first switching point, odd values will be rounded off to even values. • Only positive values are permissible.
nActSpeed_v	Integer	velocity	Actual speed in [inc/ms] • 16384 ≙ 15000 rpm
wDeadTime	Word	-	Dead time in [μs] for dead time compensation • Selection in 10 μs-steps • The switching position resulting from this is not checked for plausibility.
wHysteresis	Word	-	Switching hysteresis in [inc]
bCenterMode	Bool	binary	Definition how the selection <i>dn2ndPoint_Size_p</i> is to be interpreted. FALSE <i>dn2ndPoint_Size_p</i> defines the second switching point. TRUE <i>dn2ndPoint_Size_p</i> defines the size of the switching window.



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MGSwitchPoint - switching points

Outputs ⇨ (Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Info	
nState	Integer	binary	Status	
			10	FB is not active.
			0	Ok
			-10	The values from which the switching points are calculated are not plausible (<i>dnSwitchPoint1_p</i> , <i>dn2ndPoint_Size_p</i> , <i>whysteresis</i>).
bOut	Bool	binary	Switching output	
			TRUE	The actual position is within the defined switching window.

Definition of the switching range

Depending on the setting of *bCenterMode*, the switching range can be defined via the start and end position or by entering the center point/size of the switching range:

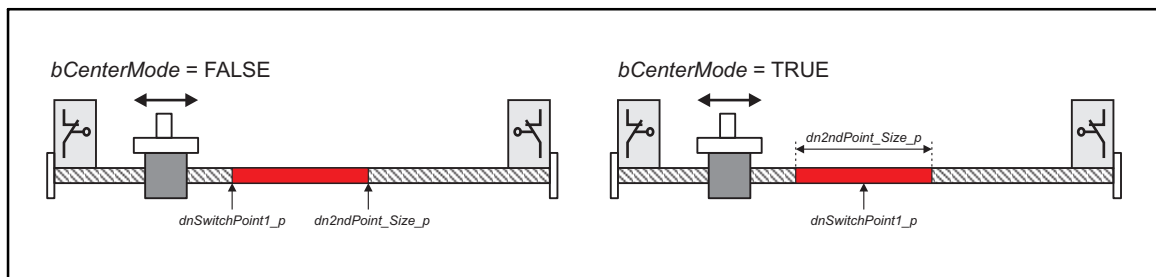


Fig. 2-14

Definition of the switching range

Dead time compensation

The dead time of external switching elements can be compensated by a corresponding dead time selection at *wDeadTime*. The FB calculates a preliminary stop position from the selected dead time value and the current speed, i.e. the switching output *bOut* will be set earlier.

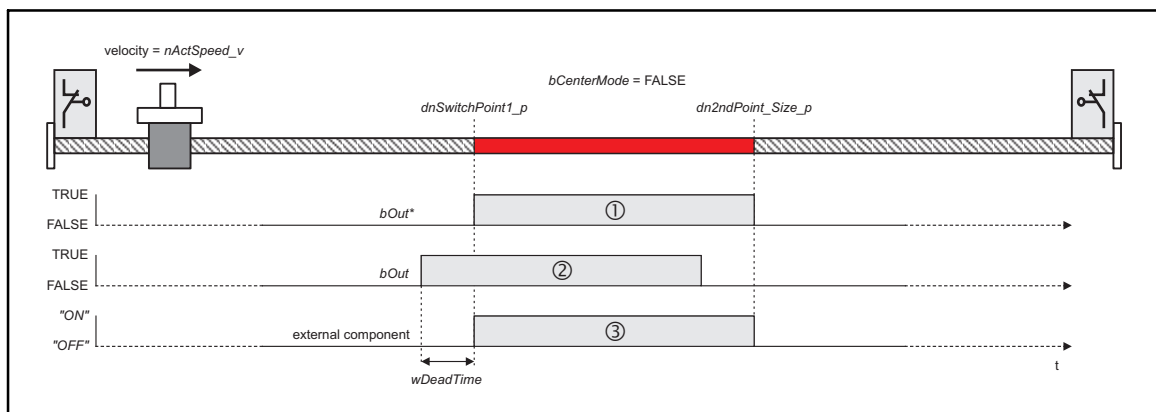
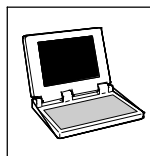


Fig. 2-15

Dead time compensation

- ① Switching without dead time compensation
- ② Switching with dead time compensation
- ③ Switching of the external switching element (with dead time)



Switching hysteresis

Permanent status changes of *bOut* resulting from changes of actual position values due to mechanical disturbances at the axis can be avoided by selecting a hysteresis at the input *wHysteresis*.

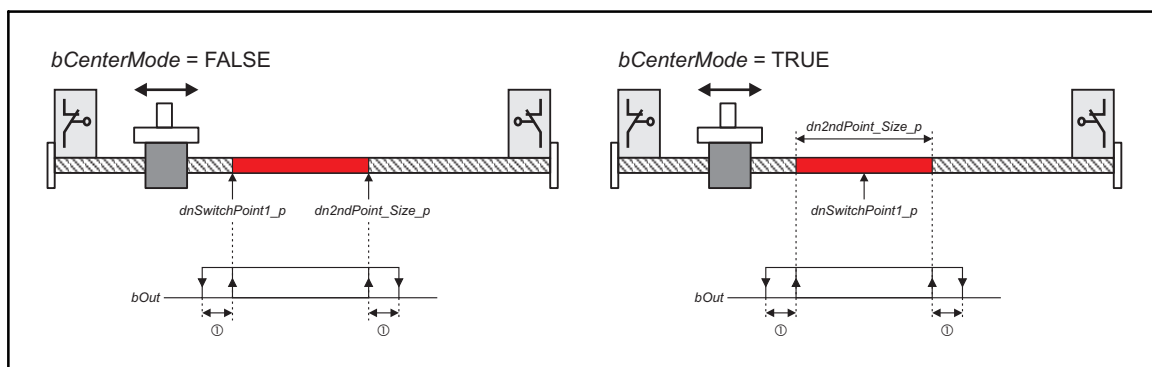


Fig. 2-16

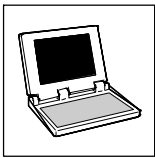
Switching hysteresis

① *wHysteresis*



Tip!

Please note that the switching points and the hysteresis will be shifted when setting a dead time under *wDeadTime*.



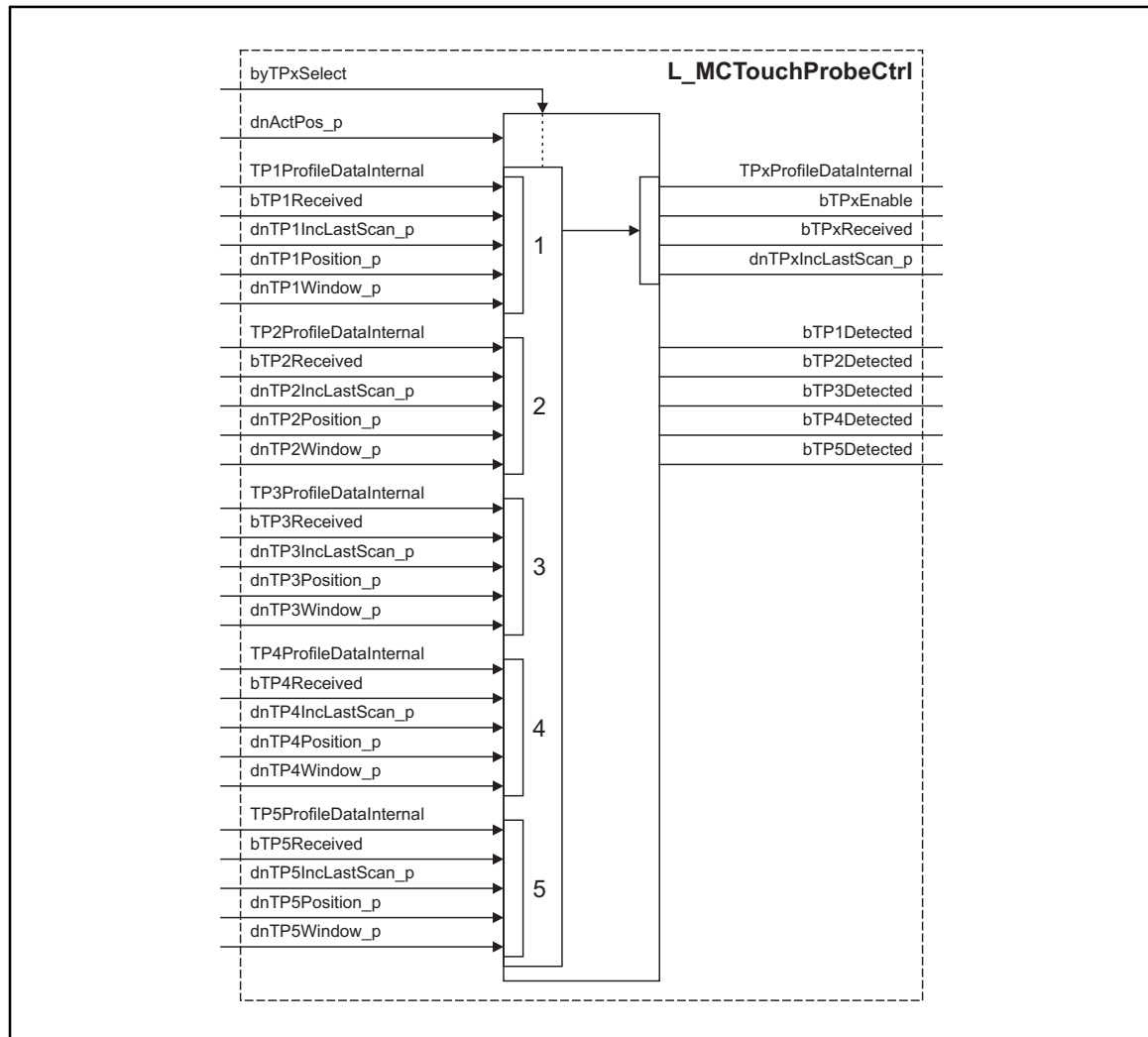
Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MCTouchProbeCtrl - touch probe preprocessing

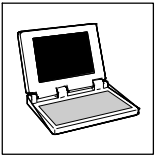
2.8 L_MCTouchProbeCtrl - touch probe preprocessing

This FB is used to change between five different touch probe signals. It is possible to define a window for a position-dependent release for every touch probe.

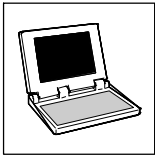


Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks *L_MCTouchProbeCtrl - touch probe preprocessing*



(Variable type: VAR_INPUT)					
Identifiers	Data type	Signal type	Info/possible settings		
byTPxSelect	Byte	binary	Bit-coded activation of the touch probe channels		
			0 All outputs are set to “0” and all touch probe channels are deactivated.		
			Bit code:		
			Bit 0 0: Touch probe channel 1 deactivated 1: Touch probe channel 1 activated		
			Bit 1 0: Touch probe channel 2 deactivated 1: Touch probe channel 2 activated		
			Bit 2 0: Touch probe channel 3 deactivated 1: Touch probe channel 3 activated		
			Bit 3 0: Touch probe channel 4 deactivated 1: Touch probe channel 4 activated		
			Bit 4 0: Touch probe channel 5 deactivated 1: Touch probe channel 5 activated		
			Bit 5 Reserved		
			Bit 6 Reserved		
Bit 7 Reserved					
dnActPos_p	Double integer	position	Actual position value in [inc]		
With the following identifiers the x stands as a wildcard for touch probe number 1 ... 5 of the touch probe.					
TPxProfileDataInternal	MC_PROFILE_INTERNAL_MEASURING	-	Positioning profile for remaining path positioning - Data structure with elements which describe the entire profile. - The data structure x will be switched through to the output <i>TPxProfileDataInternal</i> if the actual position value is within the touch probe enable window x and the corresponding touch probe signal has been detected.		
bTPxReceived	Bool	binary	Input for touch probe - In general, a touch probe signal of the operating system is connected here. - The input x will be switched through to the output <i>bTPxReceived</i> if the actual position value is in the touch probe enable window x. <table><tr><td>TRUE</td><td>Touch probe signal has been detected.</td></tr></table>	TRUE	Touch probe signal has been detected.
TRUE	Touch probe signal has been detected.				
dnTPxIncLastScan_p	Double integer	position	Phase difference in [inc] between latch time and start time of the task The input x will be switched through to the output <i>dnTPxIncLastScan_p</i> if the actual position value is in the touch probe enable window x and the corresponding touch probe signal has been detected.		
dnTPxPosition_p	Double integer	position	Position of the touch probe sensor in [inc] referred to the reference position. The position value and the window size must not exceed the range double integer.		
dnTPxWindow_p	Double integer	position	Size of the touch probe enable window in [inc] before and after the touch probe sensor position. If the window size is set to 0 [inc] the touch probe channel will be processed independently of the actual position. Important: The individual windows must not overlap each other. Otherwise the touch probe channels cannot be correctly assigned.		



Function library LenzeMotionControl1.lib

Function blocks for positioning tasks

L_MCTouchProbeCtrl - touch probe preprocessing

Outputs ⇨		(Variable type: VAR_OUTPUT)				
Identifiers	Data type	Signal type	Info			
TPxProfileDataInternal	MC_PROFILE_INTERNAL_MEASURING	-	Positioning profiles for remaining path positioning - Data structure with elements which describe the entire profile. - Output to which the profile structure of the touch probe channel is switched through which is – activated via <i>byTPxSelect</i> and – whose enable window includes the actual value.			
bTPxEnable	Bool	binary	Status signal "Activation of touch probe signal processing" <table><tr><td>TRUE</td><td>The actual position is in the enable window of a touch probe channel enabled via <i>byTPxSelect</i>.</td></tr></table>		TRUE	The actual position is in the enable window of a touch probe channel enabled via <i>byTPxSelect</i> .
TRUE	The actual position is in the enable window of a touch probe channel enabled via <i>byTPxSelect</i> .					
bTPxReceived	Bool	binary	Status signal "Touch probe detected" - Output to which the input signal of the touch probe channel is switched through which is – activated via <i>byTPxSelect</i> and – whose enable window includes the actual value. <table><tr><td>TRUE</td><td>Touch probe signal has been identified. - The status signal will be reset to FALSE after every task cycle. </td></tr></table>		TRUE	Touch probe signal has been identified. The status signal will be reset to FALSE after every task cycle.
TRUE	Touch probe signal has been identified. The status signal will be reset to FALSE after every task cycle.					
dnTPxIncLastScan_p	Double integer	position	Phase difference in [inc] between latch time and start time of the task - Output to which the input signal of the touch probe channel is switched through which is – activated via <i>byTPxSelect</i> and – whose enable window includes the actual value.			
Status signals "Touch probe detected"						
bTP1Detected	Bool	binary	TRUE	Touch probe 1 has been detected.		
bTP2Detected				Touch probe 2 has been detected.		
bTP3Detected				Touch probe 3 has been detected.		
bTP4Detected				Touch probe 4 has been detected.		
bTP5Detected				Touch probe 5 has been detected.		

Activation/deactivation of the touch probe channels

The touch probe channels are activated and deactivated through bit codes via the input *byTPxSelect*:

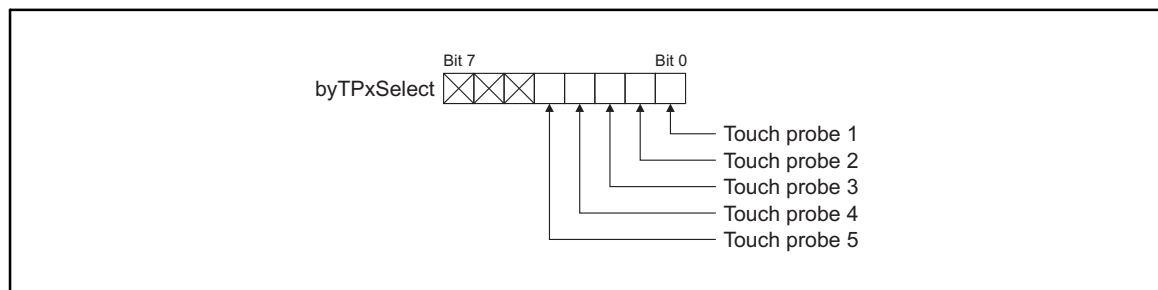
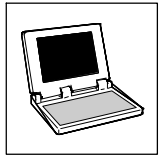


Fig. 2-17

Bit-coded activation/deactivation

- If *byTPxSelect* = "0" all outputs are set to ZERO and all touch probe channels are deactivated.



Position-dependent touch probe enable

This FB supports the position-dependent enabling of touch probe through the selection of an enable window. This prevents an accidental touch probe activation if the tool is at a position outside the window:

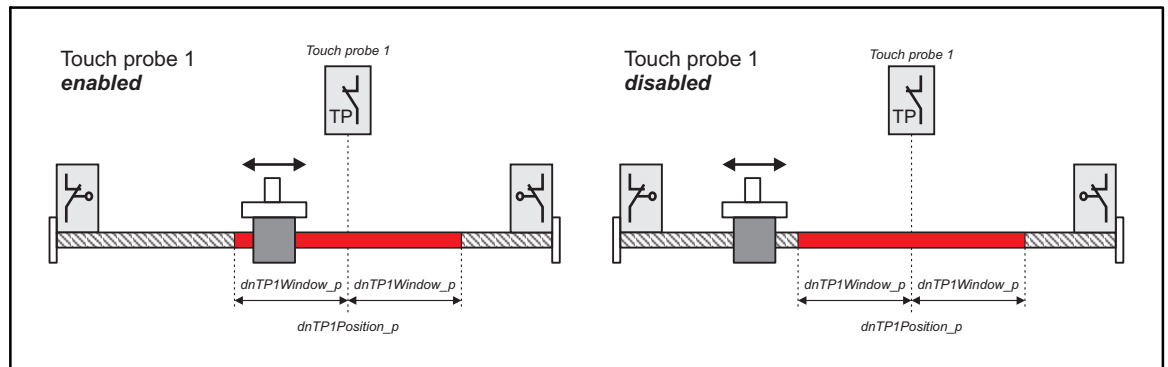


Fig. 2-18

Example: Position-dependent enabling of touch probe 1

It is possible to define a separate enable window for every touch probe channel:

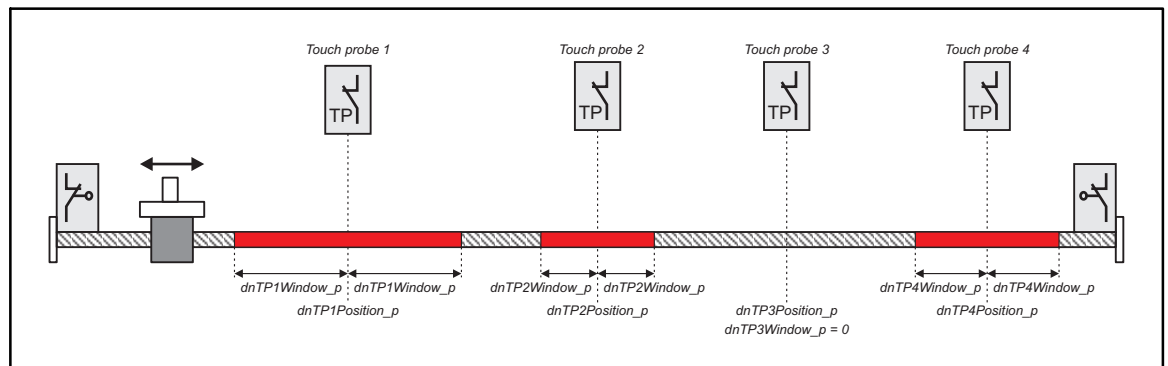


Fig. 2-19

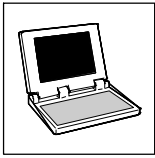
Example: Definition of the enable windows for touch probe 1 ... touch probe 4

- The position of the touch probe sensor referred to the reference position is entered via *dnTPxPosition_p*.
- The size of the window before and after the touch probe sensor position is defined via *dnTPxWindow_p*.



Note!

- The individual windows must not overlap each other. Otherwise the touch probe channels cannot be correctly assigned.
 - If the window size is set to 0 [inc] the touch probe channel will be processed independently of the actual position. (See Fig. 2-19, touch probe 3)
 - The position value and the window size must not exceed the range "double integer".
-
- *bTPxEnable* only becomes TRUE if a touch probe channel has been activated via *byTPxSelect* and the actual position is in the defined enable window or, at least, one window value is set to 0 [inc].



Function library *LenzeMotionControl1.lib*

Function blocks for positioning tasks

***L_MCTouchProbeCtrl* - touch probe preprocessing**

Output of touch probe input signals

If the actual position is within the enable window of an activated touch probe channel and the corresponding touch probe signal has been detected the inputs of the channel will be switched through to the corresponding outputs.

Example:

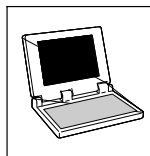
Touch probe channel 2 is activated (*byTPxSelect* - bit1 = "1"), the actual position (*dnActPos_p*) is within the enable window of touch probe channel 2 (*dnTP2Window_p*) and *bTP2Received* is TRUE (TP detected):

- Input *TP2ProfileDataInternal* → output *TPxProfileDataInternal*
- Input *bTP2Received* → output *bTPxReceived*
- Input *dnTP2IncActLastScan_p* → output *dnTPxIncActLastScan_p*

Signal outputs for detected touch probes

The outputs *bTP1Detected*... *bTP5Detected* are set to TRUE if the corresponding touch probe has been detected in the enable window or, with window size = 0 [inc], the touch probe has been detected independently of the actual position.

- This is only possible if the corresponding touch probe channel has been selected via *byTPxSelect*.
- The signal outputs are reset by setting *byTPxSelect* to "0".



3

Function blocks for converting units

This chapter informs about function blocks used for the conversion of units:

Function block	Conversion	from $\Rightarrow$ into*	Input	Output	Info
L_MCGetPosition	Position	internal $\Rightarrow$ external	inc	unit	3-2
L_MCGetVelocity	Speed	internal $\Rightarrow$ external	inc/ms	unit/s	3-3
L_MCSetAcceleration	Acceleration	external $\Rightarrow$ internal	unit/s ²	16384/A _{max}	3-4
L_MCSetDeceleration	Deceleration	external $\Rightarrow$ internal	unit/s ²	16384/A _{max}	3-5
L_MCSetProfileConverter	All profile values • Input and output values are available in form of a structure.	external $\Rightarrow$ internal	Combination: unit, unit/s, unit/s ²	Combination: inc, 16384/V _{max} 16384/A _{max}	3-6
L_MCSetProfileValues	All profile values • Input values can be assigned individually, output values are available in form of a structure.	external $\Rightarrow$ internal	Combination: unit, unit/s, unit/s ²	Combination: inc, 16384/V _{max} 16384/A _{max}	3-7
L_MCSetPosition	Position	external $\Rightarrow$ internal	unit	inc	3-9
L_MCSetVelocity	Traversing speed	external $\Rightarrow$ internal	unit/s	16384/V _{max}	3-10

* internal = internal unit, external = physical unit



Tip!

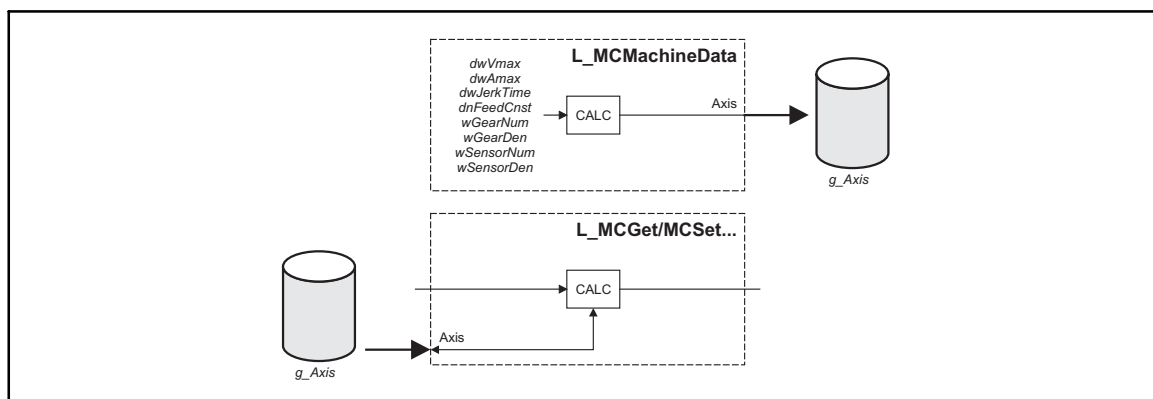
Every input value must be described as fixed comma.

- I.e. the last four numerical values are decimal positions.
- Example: Enter the numerical value “10000” for input value “1.0000”.

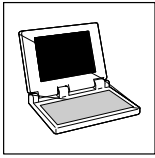
Transfer of machine parameters

The function blocks described here do not only need the value to be converted but also machine parameters at the input *Axis*.

- For this, connect the input *Axis* with a global variable (e.g. *g_Axis*), which will be written by the FB **L_MCMachineData**:



- *g_Axis* is a data structure (data type: MC_AXIS_REF) with elements which include machine parameters in the internal measuring system. See chapter 4, “Structured data types”. (4-1)



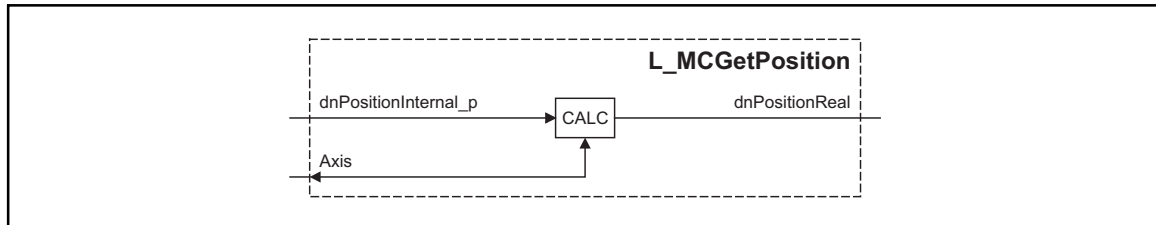
Function library LenzeMotionControl1.lib

Function blocks for converting units

L_MCGetPosition

3.1 L_MCGetPosition

This FB converts internal actual position units into physical units.



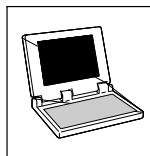
⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
dnPositionInternal_p	Double integer	position	Position in [inc] 65536 = 1 motor revolution

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
dnPositionReal	Double integer	-	Position in [unit] 10000 = 1.0000 unit - Highest possible value: 214748.3647 unit

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters - Data structure including the machine parameters in the internal measuring system. - Connect this input with a global variable which is written by FB L_MCMachineData.

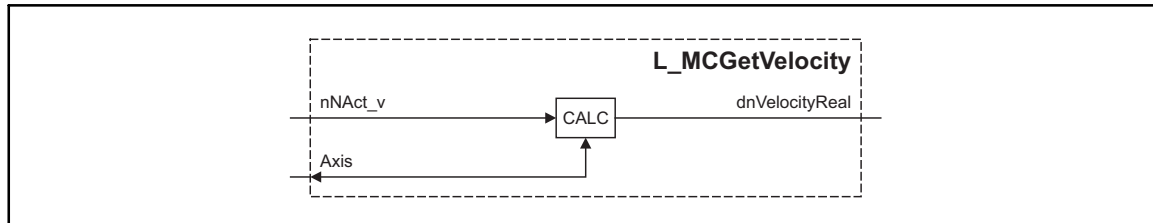
Formula

$$\text{dnPositionReal [unit]} = \text{dnPositionInternal_p [inc]} \cdot \frac{10000 \cdot \text{Feed constant [unit/rev.]} \cdot \text{Gear factor denominator} \cdot \text{Sensor factor numerator}}{65536 \text{ [inc/rev.]} \cdot \text{Gear factor numerator} \cdot \text{Sensor factor denominator}}$$



3.2 L_MCGetVelocity

This FB converts internal actual speed units into physical units.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
nNAct_v	Integer	velocity	Speed in [inc/ms] • 16384 [inc/ms] = 15000 rev./min

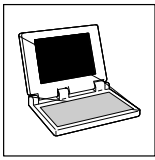
Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
dnVelocityReal	Double integer	-	Speed in [unit/s] • 10000 = 1.0000 unit/s • Highest possible value: 214748.3647 unit/s

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters • Data structure including the machine parameters in the internal measuring system. • Connect this input with a global variable which is written by FB L_MCMachineData .

Formula

$$\text{dnVelocityReal [unit/s]} = \text{nNAct_v [inc/ms]} \cdot 1000 [\text{ms/s}] \cdot X$$

$$\text{mit } X = \frac{10000 \cdot \text{Feed constant [unit/rev.]} \cdot \text{Gear factor denominator} \cdot \text{Sensor factor numerator}}{65536 [\text{inc/rev.}] \cdot \text{Gear factor numerator} \cdot \text{Sensor factor denominator}}$$



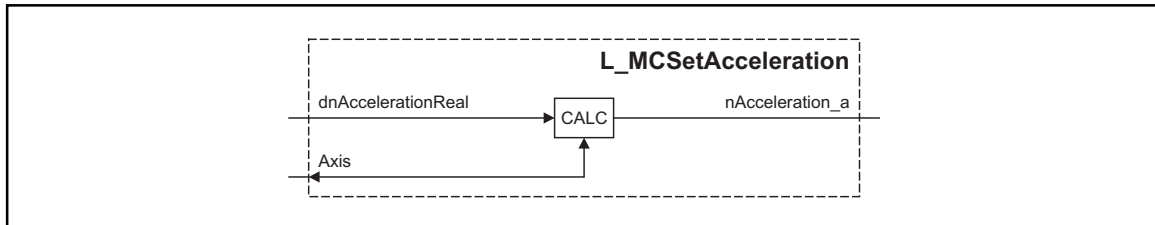
Function library LenzeMotionControl1.lib

Function blocks for converting units

L_MCSetAcceleration

3.3 L_MCSetAcceleration

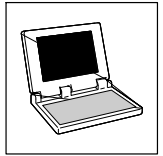
This FB converts the physical unit of the acceleration into an internal unit.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
dnAccelerationReal	Double integer	-	Acceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²

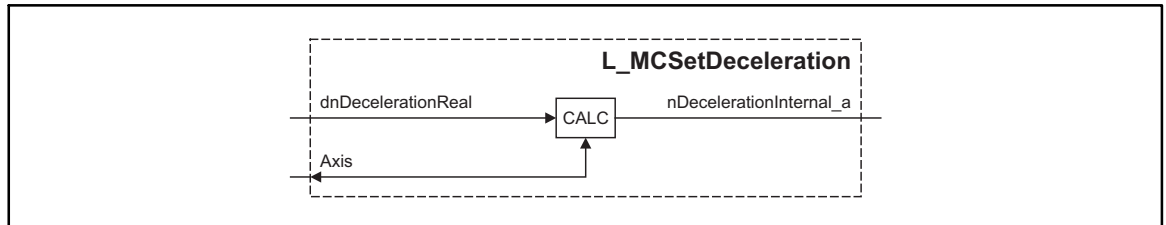
Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nAcceleration_a	Integer	-	Acceleration • Value is the relative reference for the maximum acceleration which has been defined via the FB L_MCMachineData (16384/A _{max}). • Value is limited to ±16384. • Value = "0" if reference value = "0". (Axis.dwMaxAcceleration of L_MCMachineData)

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters • Data structures which include machine parameters in the internal measuring system. • Connect this input with a global variable which is written by the FB L_MCMachineData .



3.4 L_MCSetDeceleration

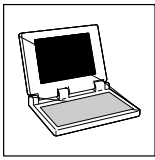
This FB converts the physical unit of the deceleration into an internal unit.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
dnDecelerationReal	Double integer	-	Deceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nDecelerationInternal_a	Integer	-	Deceleration • Value is the relative reference for the maximum acceleration which has been defined via the FB L_MCMachineData (16384/A _{max}). • Value is limited to ±16384. • Value = "0" if reference value = "0". (Axis.dwMaxAcceleration of L_MCMachineData)

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters • Data structures which include machine parameters in the internal measuring system. • Connect this input with a global variable which is written by the FB L_MCMachineData .



Function library LenzeMotionControl1.lib

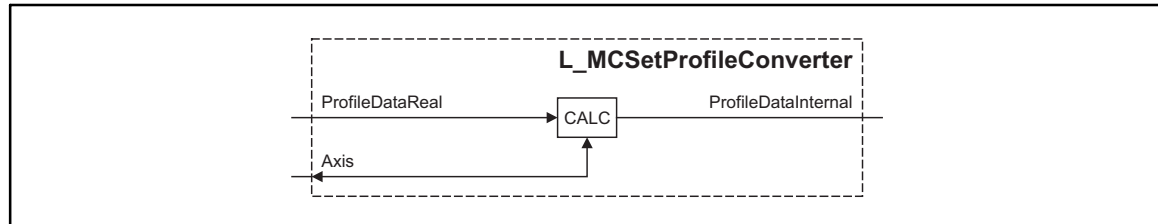
Function blocks for converting units

L_MCSetProfileConverter

3.5 L_MCSetProfileConverter

This FB converts the physical units of a complete profile into internal units.

- Whereas the FB **L_MCSetProfileValues** needs individual input values, this FB can be fed with structured input values.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
ProfileDataReal	MC_PROFILE_REAL_MEASURING	-	Positioning profile - Data structure with elements which describe the entire profile. - The values are available as physical units.

More information on the data type MC-PROFILE_REAL_MEASURING can be found in the chapter "Structured data types" (□ 4-3).

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
ProfileDataInternal	MC_PROFILE_INTERNAL_MEASURING	-	Positioning profile - Data structure with elements which describe the entire profile. - The values are stored in the internal measuring system. - The values of the elements <i>nTraversingVelocity</i>, <i>nAcceleration</i> and <i>nDeceleration</i> are limited to ± 16384. - If speed (<i>dwVmax</i>) or acceleration (<i>dwAmax</i>) = "0" the elements <i>nTraversingVelocity</i>, <i>nAcceleration</i> and <i>nDeceleration</i> also have the value "0".

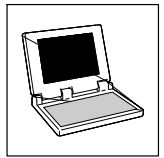
More information on the data type MC-PROFILE_INTERNAL_MEASURING can be found in the chapter "Structured data types" (□ 4-4).

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters - Data structure including the machine parameters in the internal measuring system. - Connect this input with a global variable which is written by FB L_MCMachineData.



Tip!

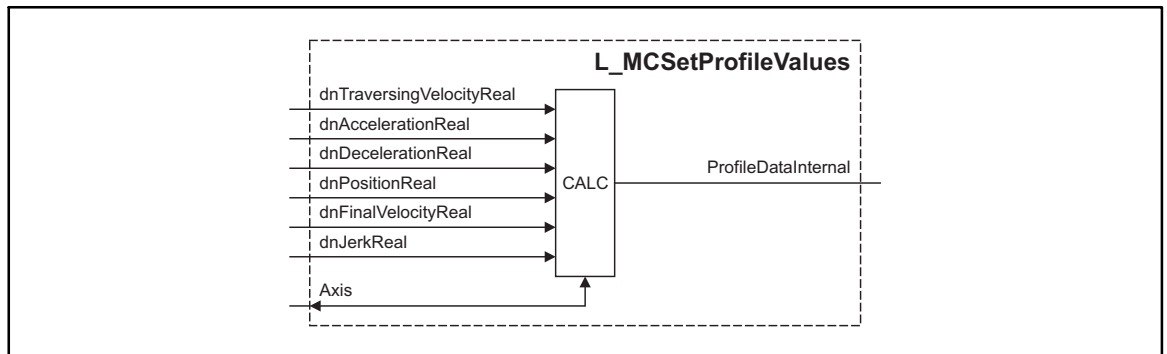
- The position is converted with a residual value. A residual value will be considered in the next calculation. For correct residual value handling, the FB must always be called before a new profile is started.
- This FB summarises the FBs **L_MCSetVelocity**, **L_MCSetAcceleration**, **L_MCSetDeceleration** and **L_MCSetPosition**.



3.6 L_MCSetProfileValues

This FB converts the physical units of a complete profile into internal units.

- Whereas the FB **L_MCSetProfileConverter** needs structured input values, this FB can be fed with individual input values.

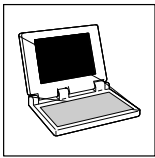


⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
dnTraversingVelocityReal	Double integer	-	Traversing speed in [unit/s] • 10000 = 1.0000 unit/s
dnAccelerationReal	Double integer	-	Acceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²
dnDecelerationReal	Double integer	-	Deceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²
dnPositionReal	Double integer	-	Position in [unit] • 10000 = 1.0000 unit
dnFinalVelocityReal	Double integer	-	Final speed in [unit/s] • 10000 = 1.0000 unit/s
dnJerkReal	Double integer	-	Reserved

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
ProfileDataInternal	MC_PROFILE_INTERNAL_MEASURING	-	Positioning profile - Data structure with elements which describe the entire profile. - The values are stored in the internal measuring system. - The values of the elements <i>nTraversingVelocity</i>, <i>nAcceleration</i> and <i>nDeceleration</i> are limited to ±16384. - If speed (dwVmax) or acceleration (dwAmax) = "0" the elements <i>nTraversingVelocity</i>, <i>nAcceleration</i> and <i>nDeceleration</i> also have the value "0".

More information on the data type MC_PROFILE_INTERNAL_MEASURING can be found in the chapter "Structured data types" (4-4).

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters - Data structure including the machine parameters in the internal measuring system. - Connect this input with a global variable which is written by FB L_MCMachineData.



Function library LenzeMotionControl1.lib

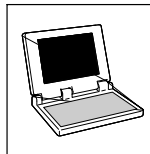
Function blocks for converting units

L_MCSetProfileValues



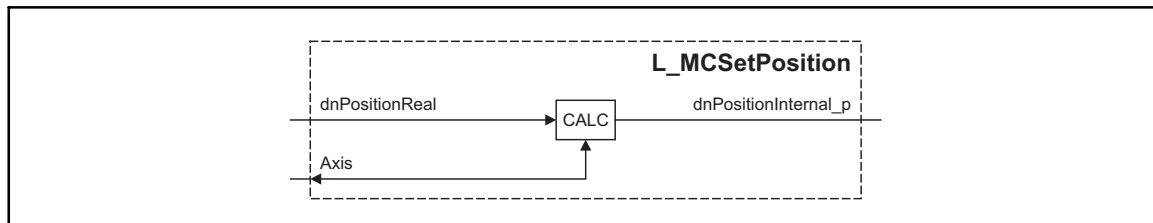
Tip!

- The position is converted with a residual value. A residual value will be considered in the next calculation. For correct residual value handling, the FB must always be called before a new profile is started.
- This FB summarises the FBs **L_MCSetVelocity**, **L_MCSetAcceleration**, **L_MCSetDeceleration** and **L_MCSetPosition**.



3.7 L_MCSetPosition

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



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
dnPositionReal	Double integer	position	Position in [unit] • 10000 = 1.0000 unit

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
dnPositionInternal_p	Double integer	position	Position in [inc] • 65536 = 1 motor revolution

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters • Data structure including the machine parameters in the internal measuring system. • Connect this input with a global variable which is written by FB L_MCMachineData.

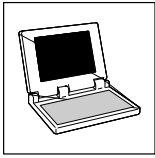
Formula

$$dnPositionInternal_p \text{ [inc]} = \frac{dnPositionReal \text{ [unit]}}{10000} \cdot \frac{65536 \text{ [inc/rev]} \cdot \text{Gear factor numerator} \cdot \text{Sensor factor denominator}}{\text{Feed constant [unit/rev]} \cdot \text{Gear factor denominator} \cdot \text{Sensor factor numerator}}$$



Tip!

The position is converted with a residual value. A residual value will be considered in the next calculation. For correct residual value handling, the FB must always be called before a new profile is started.



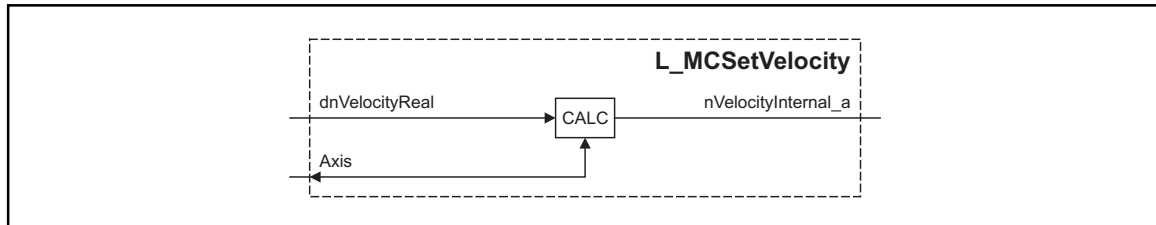
Function library LenzeMotionControl1.lib

Function blocks for converting units

L_MCSetVelocity

3.8 L_MCSetVelocity

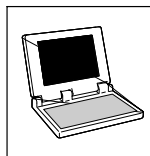
This FB converts the physical unit of the traversing speed into an internal unit.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info
dnVelocityReal	Double integer	-	Traversing speed in [unit/s] • 10000 = 1.0000 unit/s

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nVelocityInternal_a	Integer	-	Traversing speed • Value is the relative reference for the maximum speed which has been defined via the FB L_MCMachineData ($16384/V_{max}$). • Value is limited to ± 16384 . • Value = "0" if reference value = "0". (<i>Axis.dwMaxVelocity</i> of L_MCMachineData)

⇒ Inputs/outputs ⇒ (Variable type: VAR_IN_OUT)			
Identifiers	Data type	Signal type	Info
Axis	MC_AXIS_REF	-	Machine parameters • Data structures which include machine parameters in the internal measuring system. • Connect this input with a global variable which is written by the FB L_MCMachineData .



4 Structured data types

This chapter informs about structured data types defined in the function library “LenzeMotionControl.Lib1”. They use machine parameters and positioning profile data.

4.1 MC_AXIS_REF

The data structure MC_AXIS_REF is used to transfer machine parameters to other MotionControl function blocks.

The following table describes the elements included in the data structure MC_AXIS_REF:

Element identifier	Data type	Info
dwMaxVelocity	Double word	Copy of the max. traversing speed in [unit/s] • 10000 = 1.0000 unit/s
dwMaxAcceleration	Double word	Copy of the max. acceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²
dnFeedConstant	Double integer	Copy of the feed constant • Machine feed in units per revolution of the gearbox load side. • 10000 = 1.0000 unit/s
wRatioNumerator	Word	Copy gear factor numerator
wRatioDenominator	Word	Copy gear factor denominator
wSensorRatioNumerator	Word	Copy position encoder gear factor numerator
wSensorRatioDenominator	Word	Copy position encoder gear factor denominator
dnIntMaxVelocity	Double integer	Internal max. speed for profile generator • ⇒ Formula 1
dnIntMaxAcceleration	Double integer	Internal max. acceleration for profile generator • ⇒ Formula 2
nVnorm	Integer	Internal value to calculate the speed setpoint for the speed encoder selected under C0011 (formula 4) • ⇒ Formula 3 • Selected speed ⇒ Formula 4
dnIntMaxJerk	Double integer	Reserved

Formula 1

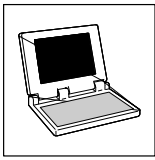
$$\text{dnIntMaxVelocity} = \frac{\text{dwMaxVelocity}}{10000} \cdot \frac{16384}{1000} \cdot \text{PosResolution}$$

Formula 2

$$\text{dnIntMaxAcceleration} = \frac{\text{dwMaxAcceleration}}{10000} \cdot \frac{16384}{1000000} \cdot \text{PosResolution}$$

Intermediate calculation for formula 1 and 2

$$\text{PosResolution} = 65535 \cdot \frac{\text{wRatioNumerator}}{\text{wRatioDenominator}} \cdot \frac{\text{wSensorRatioDenominator}}{\text{wSensorRatioNumerator}} \cdot \frac{10000}{\text{dnFeedConst}}$$



Function library *LenzeMotionControl1.lib*

Structured data types

MC_AXIS_REF

Formula 3

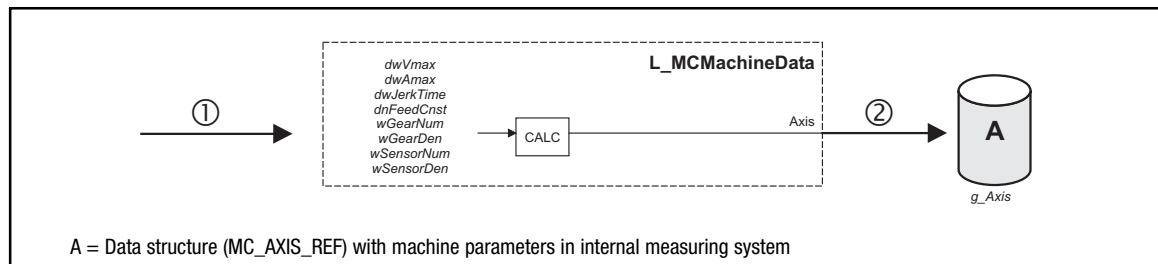
$$n_{\text{Vnorm}} = n_{\text{max}} \cdot \frac{65536}{60000} \cdot \frac{w_{\text{SensorRatioDenominator}}}{w_{\text{SensorRatioNumerator}}}$$

$$n_{\text{max}} = C0011$$

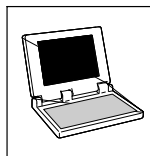
Formula 4

$$\text{Normalised speed for controller} = \frac{\text{current traversing speed}}{n_{\text{Vnorm}}}$$

Typical application



- The machine parameters are preselected for the machine as physical units. ①
- Machine parameters are converted for the internal measuring system of the profile generator with the FB **L_MCMachineData**. (2-4)
- The converted data are available at the output *Axis* as data structure MC_AXIS_REF. For further processing by other MotionControl FBs, normally a global variable (here: *g_Axis*) is assigned. ②



4.2 MC_PROFILE_REAL_MEASURING

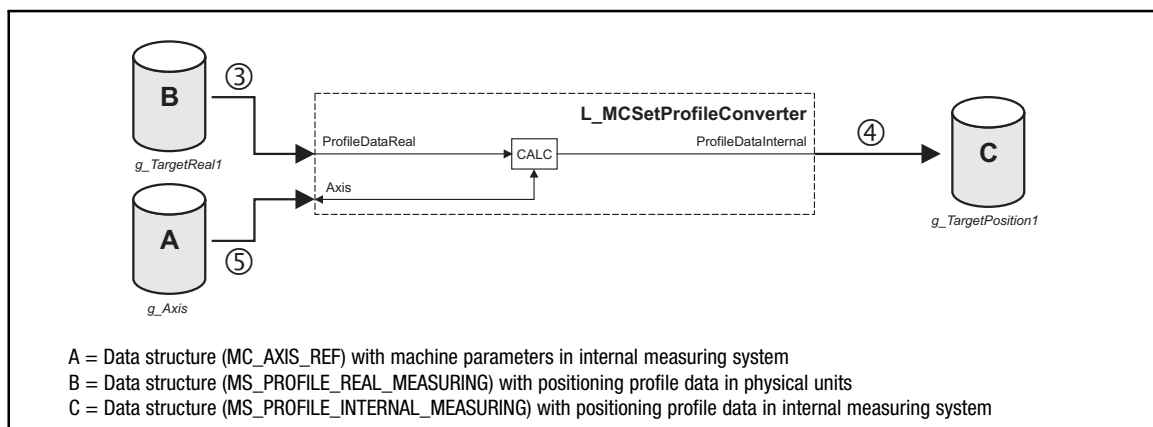
The data structure MC_PROFILE_REAL_MEASURING describes a positioning profile in physical units (real measuring system).

The following table describes the elements included in the data structure MC_PROFILE_REAL_MEASURING:

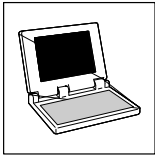
Element identifier	Data type	Info
dnTraversingVelocity	Double integer	Traversing speed in [unit/s] • 10000 = 1.0000 unit/s
dnAcceleration	Double integer	Acceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²
dnDeceleration	Double integer	Deceleration in [unit/s ²] • 10000 = 1.0000 unit/s ²
dnPosition	Double integer	Position in [unit] • 10000 = 1.0000 unit
dnFinalVelocity	Double integer	Final speed in [unit/s] • 10000 = 1.0000 unit/s
dnJerk	Double integer	Reserved

Typical application

The positioning profiles are also entered as physical units and converted for the internal measuring system using the corresponding FBs of the **LenzeMotionControl1.lib**.



- You can transfer a complete positioning profile to the FB **L_MCSetProfileConverter** via the input *ProfileDataReal*. The profile is transferred in physical units in form of the data structure MC_PROFILE_REAL_MEASURING to ensure that it can be converted into the internal measuring system. ③ (□ 3-6)
- The converted data are available at the output *ProfileDataInternal* as data structure MC_PROFILE_INTERNAL_MEASURING. For further processing by other MotionControl FBs, normally a global variable (here: *g_TargetPosition1*) is assigned. ④
- For the conversion this FB also needs the machine parameters in the internal measuring system via the input *Axis*. ⑤



Function library LenzeMotionControl1.lib

Structured data types

MC_PROFILE_INTERNAL_MEASURING

4.3 MC_PROFILE_INTERNAL_MEASURING

The data structure MC_PROFILE_INTERNAL_MEASURING describes a positioning profile in internal units in the internal measuring system as required by the profile generators.

The following table describes the elements included in the data structure MC_PROFILE_INTERNAL_MEASURING:

Element identifier	Data type	Info
nTraversingVelocity	Integer	Traversing speed Value is the relative reference for the maximum speed which has been defined via the FB L_MCMachineData (16384/V_{max}).
nAcceleration	Integer	Acceleration Value is the relative reference for the maximum acceleration which has been defined via the FB L_MCMachineData (16384/A_{max}).
nDeceleration	Integer	Deceleration Value is the relative reference for the maximum acceleration which has been defined via the FB L_MCMachineData (16384/A_{max}).
dnPosition	Double integer	Position in [inc] 65536 = 1 motor revolution ⇒ Formula 1
nFinalVelocity	Integer	Final speed Value is the relative reference for the maximum speed which has been defined via the FB L_MCMachineData (16384/V_{max}).
nJerk	Integer	Reserved

Formula 1

$$dnPosition [inc] = \frac{dnPositionReal [unit]}{10000} \cdot \frac{65536 [inc/rev.]}{dnFeedConstant [unit/rev.]} \cdot \frac{wRatioNumerator}{wRatioDenominator} \cdot \frac{wSensorRatioDenominator}{wSensorRatioNumerator}$$

Typical application

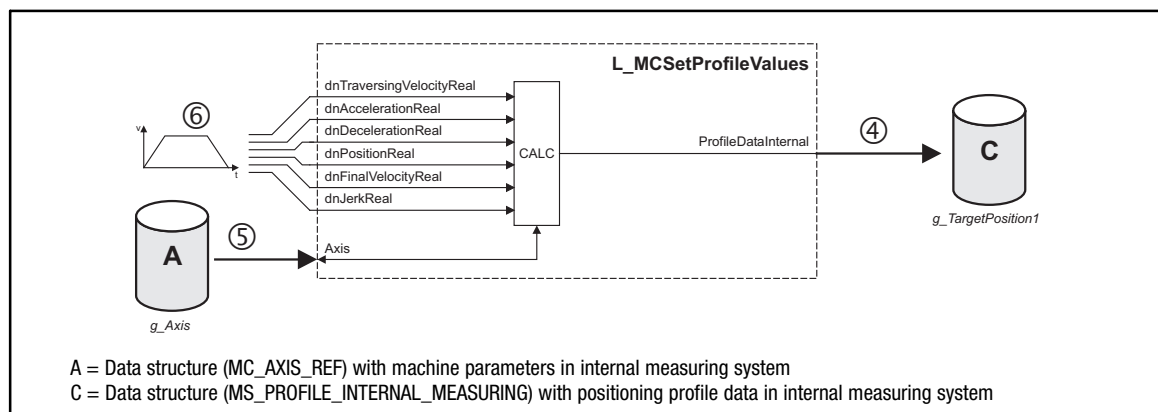
The positioning profile data are entered in physical units and converted for the internal measuring system using the FB **L_MCSetProfileConverter** or **L_MCSetProfileValues**.

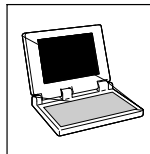


Tip!

Unlike the FB **L_MCSetProfileConverter** which expects the positioning profile data in physical units in form of data structure MC_PROFILE_REAL_MEASURING, data can be transferred as individual values to the FB **L_MCSetProfileValues**.

- For a typical application for the FB **L_MCSetProfileValues** see chapter 4.2. (4-3)





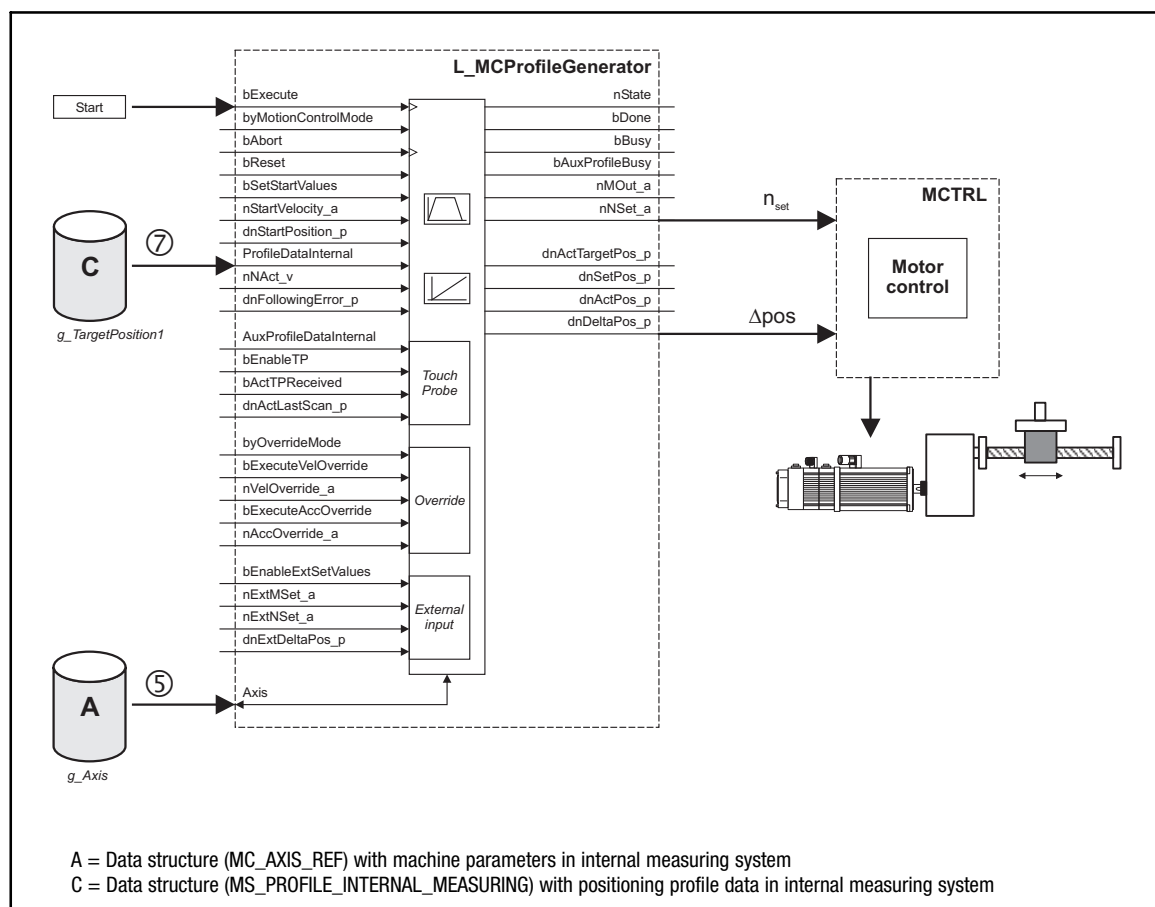
- The converted data are available at the output *ProfileDataInternal* as data structure *MC_PROFILE_INTERNAL_MEASURING*. For further processing by other MotionControl FBs, normally a global variable (here: *g_TargetPosition1*) is assigned. ④
- This FB needs the machine parameters in the internal measuring system via the input *Axis*. ⑤



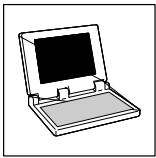
Tip!

Information about the function blocks which convert physical units from the real measuring system into information that can be processed by the internal measuring system of the profile generator can be found in chapter 3, "Function blocks for unit conversion". (3-1)

Data transfer to the profile generator (here: **L_MCPProfileGenerator**):



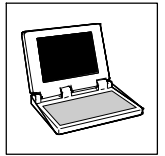
- If the positioning profiles are stored in the internal measuring system, they can be transferred to the profile generator for further processing via global variables (here: *g_TargetPosition1*). ⑦
- The profile generator also needs the machine parameters in the internal measuring system via the input *Axis*. ⑤



Function library LenzeMotionControl1.lib

Structured data types

MC_PROFILE_INTERNAL_MEASURING



5 Index

C

Calculation of correction values, 2-13

D

Display of setpoint positions on keypads, 2-14

F

Feed correction, 2-10

Formulas for positioning, 2-20

I

Input of machine parameters, 2-6

Interpolation method for calculation of correction values, 2-13

L

L_MCAutoResetInterface, 2-2

L_MCGetPosition, 3-2

L_MCGetVelocity, 3-3

L_MCMachineData, 2-4

L_MCPositionCorrection, 2-10

L_MCProfileGenerator, 2-15

L_MCResidualDistance, 2-28

L_MCSetAcceleration, 3-4

L_MCSetDeceleration, 3-5

L_MCSetPosition, 3-9

L_MCSetProfileConverter, 3-6

L_MCSetProfileValues, 3-7

L_MCSetVelocity, 3-10

L_MCSwitchCam, 2-31

L_MCSwitchPoint, 2-35

L_MCTouchProbeCtrl, 2-38

M

Machine parameters (L_MCMachineData), 2-4

Example, 2-7

Input, 2-6

S-profiles, 2-8

Transfer to other FBs, 3-1

Measured-value table, 2-11

O

Overchange, 2-27

override, 2-23

P

Profile generator (L_MCProfileGenerator), 2-15

Formulas for positioning, 2-20

Modes for remaining path positioning, 2-23

Override, 2-23

Profile change during positioning, 2-23

Profile generator modes, 2-21

Profile linkage with overchange, 2-27

Remaining path positioning, 2-22

Selection of the positioning profile, 2-20

Transfer of machine parameters, 2-20

Profile linkage with overchange, 2-27

R

Remaining path calculation, 2-28

Calculation of the shortest remaining path, 2-29

Definition of the message window "Target position reached", 2-29

Formulas for the calculation, 2-30

Remaining path positioning, 2-22

S

S-shaped ramps (S-profiles), 2-8

Safety information, Layout

Other notes, 1-1

Warning of material damage, 1-1

Switching points (L_MCSwitchPoint), 2-35

Dead time compensation, 2-33, 2-36

Definition of the switching range, 2-32, 2-36

Switching hysteresis, 2-33, 2-37

Switching points rotary axis (L_MCSwitchCam), 2-31

T

Term definitions, 1-2

Touch probe preprocessing, 2-38

Activation/deactivation of the touch probe channels, 2-40

Output of touch probe input signals, 2-42

Position-dependent touch probe enable, 2-41

Signal outputs for detected touch probes, 2-42

U

unit, 1-2