

Lenze

Manual

IEC 61131-3

inside

***Global Drive
PLC Developer Studio***



Global Drive

Function library

LenzeMotionControl2.lib

This documentation applies to the function library **LenzeMotionControl2.lib** as of version 5.00

The function library **LenzeMotionControl2.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.

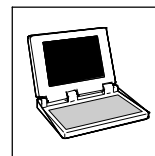
© 2008 Lenze Drive Systems GmbH

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

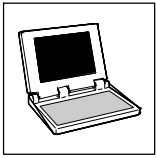
All information given in this documentation has been carefully selected and tested for compliance with the hardware and software described. Nevertheless, discrepancies cannot be ruled out. We do not accept any responsibility or liability for any damage that may occur. Required corrections will be included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

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

Version 1.0 04/2008 - TD29

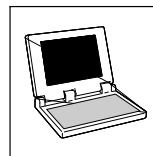


1 Preface and general information	1-1
1.1 About this Manual	1-1
1.1.1 Conventions used in this Manual	1-1
1.1.2 Pictographs used in this Manual	1-1
1.1.3 Layout of the descriptions	1-2
1.1.4 Terminology used	1-2
1.2 Version identifiers of the function libraries	1-3
2 Function blocks for axis synchronisation	2-1
2.1 L_MCInterpolator - signal interpolation	2-2
2.2 L_MCPhiDiff - integration and differential values of speed signals	2-5
2.3 L_MCPhiIntegrator - integrator with TP set function	2-7
2.4 L_MCPhiTrim - speed-dependent phase trimming	2-10
2.5 L_MCPhiTrimTime - time-based phase trimming	2-11
2.6 L_MCSignalDerive - signal differentiation	2-12
3 Index	3-1



Function library LenzeMotionControl2.lib

Contents



1 Preface and general information

1.1 About this Manual

This Manual contains information about the function library **LenzeMotionControl2.lib** for the **Drive PLC Developer Studio**.

- The function library **LenzeMotionControl2.lib** contains functions for implementing axis synchronisation tasks.
- The function library **LenzeMotionControl1.lib** comprises functions necessary for positioning 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**..."

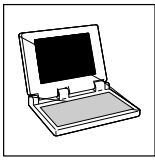
Program listings

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 LenzeMotionControl2.lib

Preface and general information

About this Manual

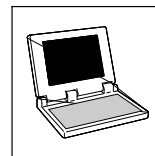
1.1.3 Layout of the descriptions

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")



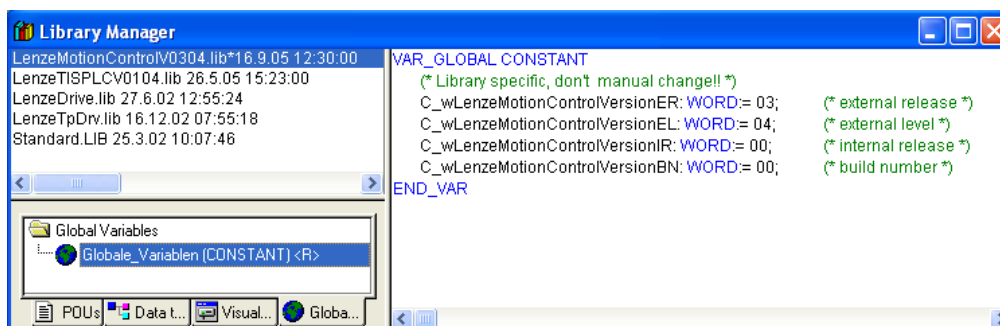
1.2 Version identifiers of the function libraries

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

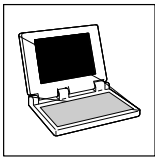
Version: 03 04 00 00

- 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**:

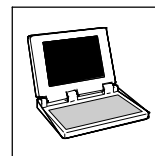
- Use the menu command **Window→Library manager** to open the Library manager in DDS.
- Select the register **Global variables** and open the folder **Global variables**.
- If you highlight the *Global_Variables...*, the version will be indicated in the Library manager.



Function library LenzeMotionControl2.lib

Preface and general information

Version identifiers of the function libraries



2

Function blocks for axis synchronisation

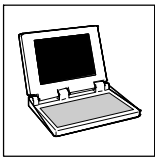
This chapter informs about function blocks which can be used for axis synchronisation.



Note!

Due to their internal structure, the function blocks described in this chapter have 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!



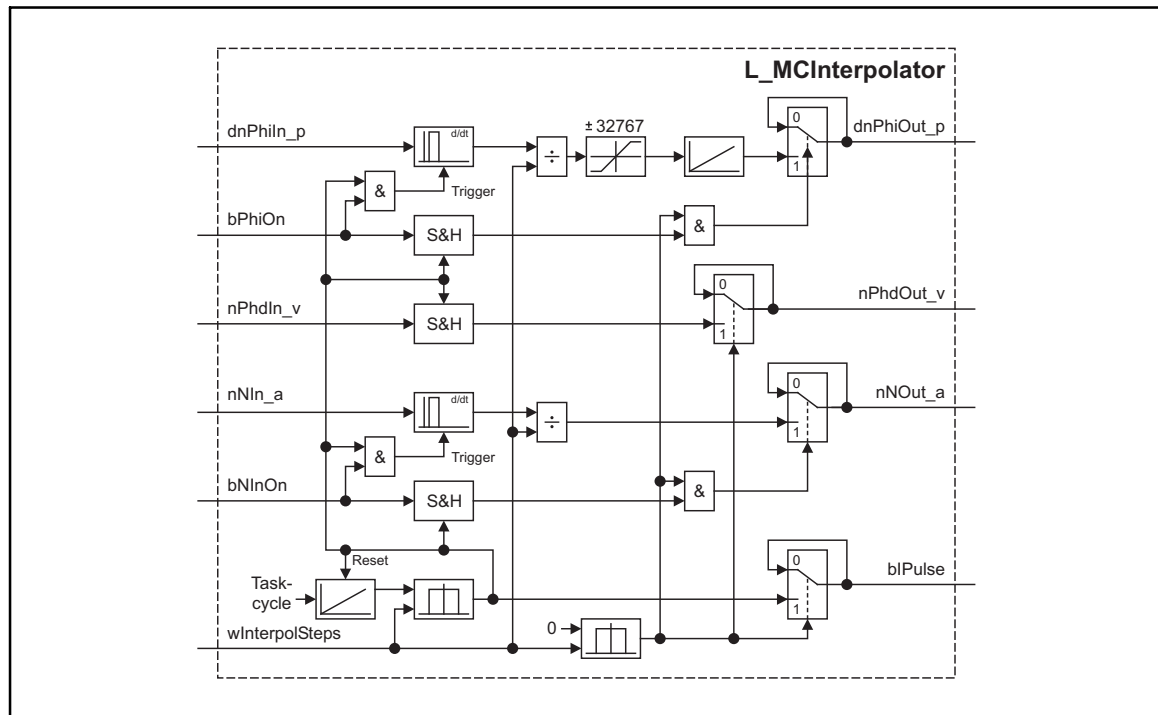
Function library LenzeMotionControl2.lib

Function blocks for axis synchronisation

L_MCInterpolator - signal interpolation

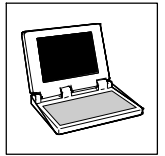
2.1 L_MCInterpolator - signal interpolation

This FB linearly interpolates the difference signals of a slower task within a faster task. In this way, the output signal only reaches the level of the input signal after the number of task cycles selected via the input *wInterpolSteps*.



⇒ Inputs (Variable type: VAR_INPUT)				
Identifiers	Data type	Signal type	Info/possible settings	
dnPhiIn_p	Double integer	position	Phase signal in [inc]	
bPhiOn	Bool	binary	Activation of phase signal interpolation	
			TRUE	Phase signal (<i>dnPhiIn_p</i>) is interpolated
			FALSE	Interpolation is stopped
nPhdIn_v	Integer	velocity	Speed signal (phase difference) in [inc/ms]	
nNIn_a	Integer	analog	Analog speed signal	
bNInOn	Bool	binary	Activation of analog speed input	
			TRUE	Activates speed input (<i>nNIn_a</i>)
wInterpolSteps	Word	-	Number of L_McInterpolator call ups and thus interpolation steps - Ratio of task cycle time (slow task/fast task) - Input range 0 ... 32767 - If 0 is selected the FB will not be processed and the current output values remain unchanged.	

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
dnPhiOut_p	Double integer	position	Interpolated phase
nPhdOut_v	Integer	velocity	Interpolated speed (phase difference)
nNOOut_a	Integer	analog	Interpolated speed (analog)
bIPulse	Bool	binary	Data acceptance pulse (1 clock cycle) TRUE Data is accepted



Application (example)

The FB **L_MCInterpolator** interpolates a setpoint signal coming from a slow task within a faster task and transmits it to the motor control:

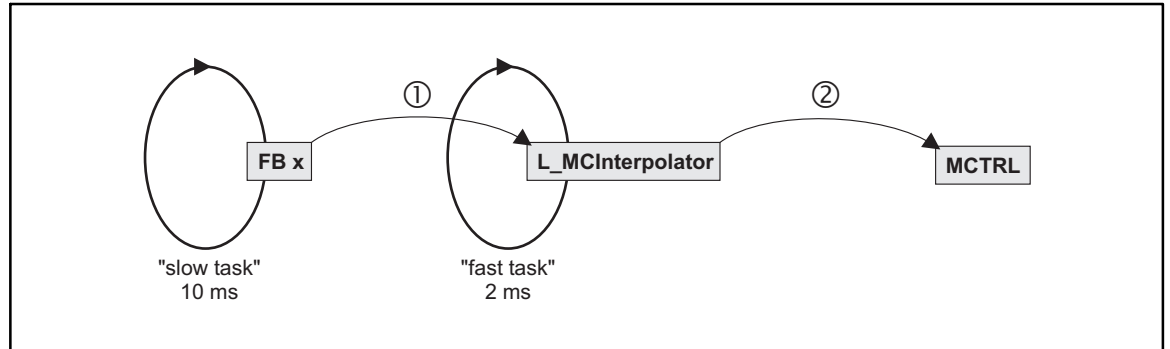


Fig. 2-1

Example: Interpolation of a setpoint signal and transfer to the motor control

- ① Setpoint signal coming from a slow task
- ② Interpolated setpoint signal coming from a faster task in which the FB **L_MCInterpolator** is called up.

After completion of the task cycles selected under *wInterpolSteps*, the output signal reaches the level of the corresponding input signal:

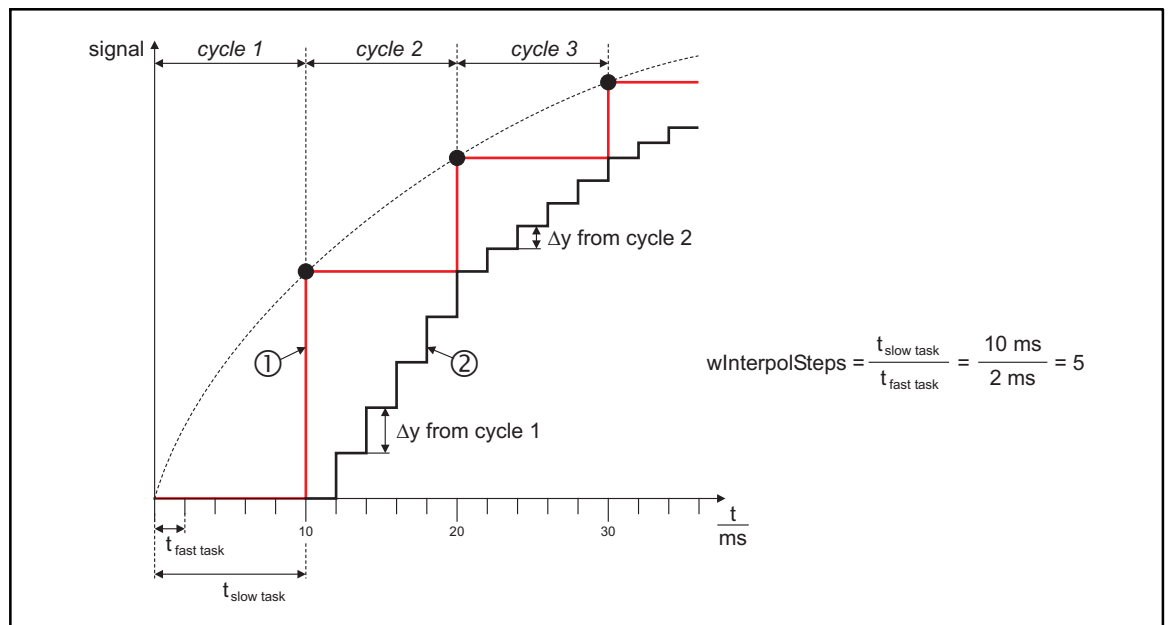


Fig. 2-2

Signal flow

- ① Setpoint signal coming from a slow task
- ② Interpolated setpoint signal coming from a faster task in which the FB **L_MCInterpolator** is called up.

Interpolation steps

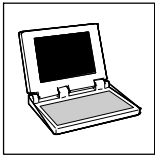
The number of interpolation steps to be selected at the input *wInterpolSteps* is calculated as follows:

$$wInterpolSteps = \frac{\text{Task cycle time (slow task)}}{\text{Task cycle time (fast task)}}$$



Note!

Do not change the number of interpolation steps during operation. Otherwise interpolation errors will occur.



Function library LenzeMotionControl2.lib

Function blocks for axis synchronisation

L_MCInterpolator - signal interpolation

Phase values

- The function is activated by setting *bPhiOn* = TRUE.
- The input value *dnPhiln_p* is differentiated every n-th task cycle (FB **L_MCInterpolator**) divided by n with residual value and integrated.
- The interpolated phase is output at *dnPhiOut_p*.
- n equals the value of *wInterpolSteps*.

Phase difference signals

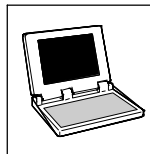
- The function is always active.
- The input value *nPhdIn_v* is copied every n-th task cycle (FB **L_MCInterpolator**) to the output *nPhdOut_v* and kept at a constant value until the next copying procedure.
- This signal type is not interpolated, the data is only transferred in a co-ordinated way.
- n equals the value of *wInterpolSteps*.

Analog signal

- The function is activated by setting *bNlnOn* = TRUE.
- The input value *nNln_a* is differentiated every n-th task cycle (FB **L_MCInterpolator**), divided by n with residual value, integrated and output at *nNOut_a*.
- n equals the value of *wInterpolSteps*.

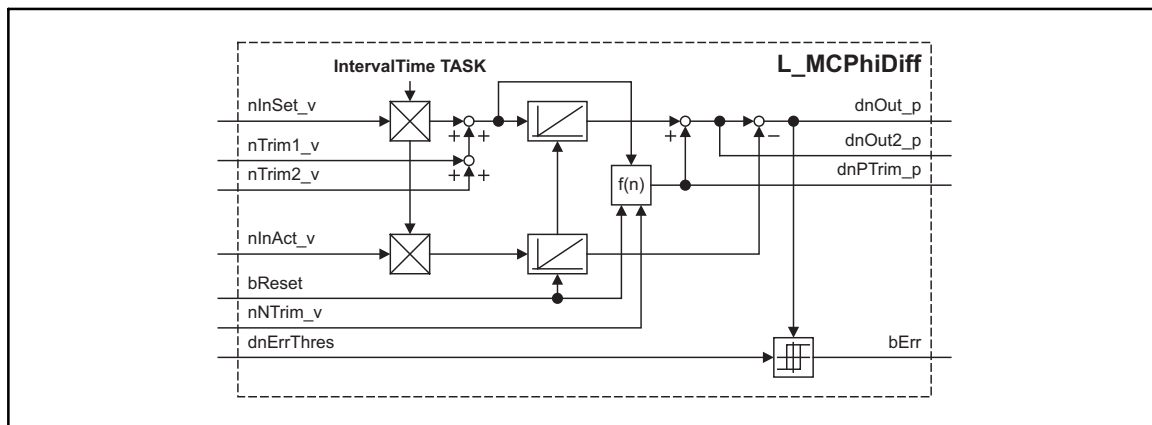
More output signals

- The acceptance of the input values is indicated by *bIPulse* = TRUE (pulse time: 1 task cycle).



2.2 L_MCPHiDiff - integration and differential values of speed signals

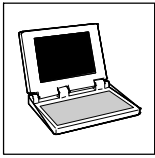
The FB integrates the speed signals $nInSet_v$, $nTrim1_v$, $nTrim2_v$ and $nInAct_v$ as phase values and calculates a differential angle (following error).



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
$nInSet_v$	Integer	velocity	Phase difference setpoint in [inc/ms]
$nTrim1_v$	Integer	velocity	Additive trimming value in [inc/call up cycle]
$nTrim2_v$	Integer	velocity	Additive trimming value in [inc/call up cycle]
$nInAct_v$	Integer	velocity	Actual phase difference value in [inc/ms]
$nNTrim_v$	Integer	velocity	n-depending phase trimming in [inc/ms]
$bReset$	Bool	binary	Reset of position difference TRUE Set position difference = 0
$dnErrThres$	Double integer	-	Following error limit

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
$dnOut_p$	Double integer	position	Following error
$dnOut2_p$	Double integer	position	Integrated setpoint angle
$dnPTrim_p$	Double integer	position	n-depending phase trimming
$bErr$	Bool	binary	Status of limit value monitoring TRUE Limit value has been exceeded ($dnOut > dnErrThres$)

- The value of the following error cannot exceed a certain limit value ($dnErrThres$).
- A phase value $f(n)$ depending linearly on the setpoint can be added.



Function library *LenzeMotionControl2.lib*

Function blocks for axis synchronisation

L_MCPHiDiff - integration and differential values of speed signals

Speed-dependent phase trimming

- The phase-angle shift enables a leading or lagging of the phase with rising speed.
- Via *nNTrim_v* a corresponding adjustment can be entered in increments which will be reached at 15000 rpm of the drive (linear relationship):

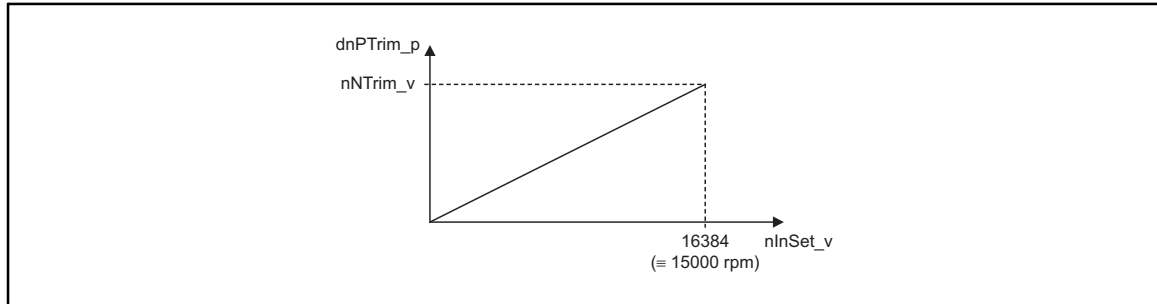
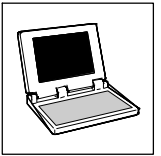


Fig. 2-3

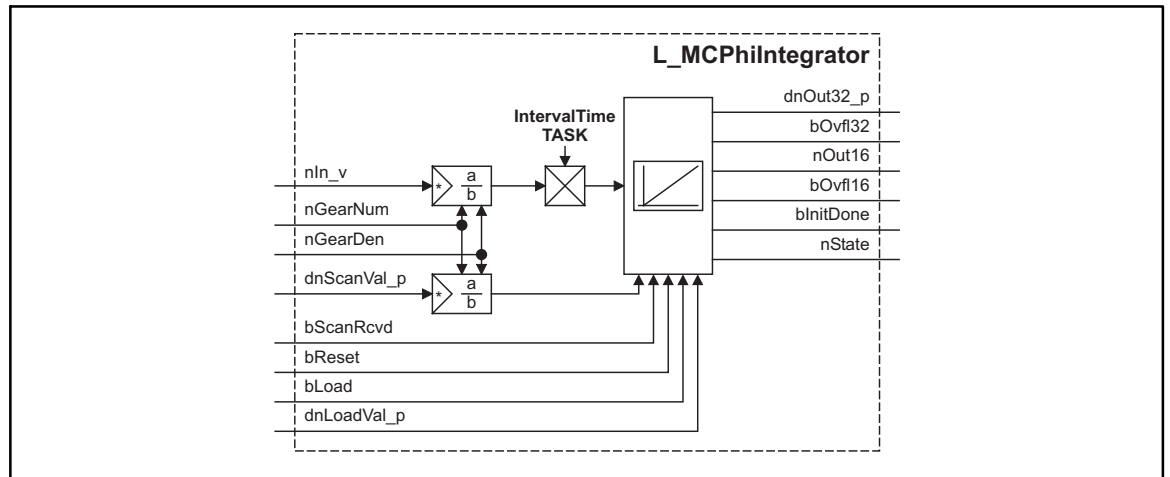
Linear relationship with speed-dependent phase trimming



2.3 L_MCPHiIntegrator - integrator with TP set function

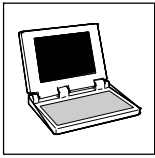
This FB evaluates the input value nIn_v with a gear factor and calculates the sum in an integrator.

- The integrator output has a width of 16 bits and 32 bits.
- It indicates if an output overflows.
- The integrator can be loaded and initialised (e.g. by means of a touch probe signal).



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
nIn_v	Integer	velocity	Speed input (phase difference)
nGearNum	Integer	-	Gear factor numerator
nGearDen	Integer	-	Gear factor denominator
dnScanVal_p	Double integer	position	Initialisation value
Integrator functions			
bScanRcvd	Bool	binary	FALSE → TRUE Integrator initialisation
bReset	Bool	binary	TRUE Integrator reset • The integrator is not added.
bLoad	Bool	binary	TRUE Loading the integrator with dnLoadVal_p • The integrator is not added. • The output <i>bInitDone</i> will be reset to FALSE with a delay of one call up cycle after <i>bLoad</i> has been reset to FALSE and a FALSE-TRUE transition at <i>bScanRcvd</i>.
dnLoadVal_p	Double integer	position	Value used to load the integrator.

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
dnOut32_p	Double integer	position	Integrator value (32 bits)
bOvf132	Bool	binary	Overflow (32 bits) TRUE Overflow • The output remains TRUE for one call-up cycle.
nOut16	Integer	-	Integrator value (16 bits)
bOvf16	Bool	binary	Overflow (16 bits) TRUE Overflow • The output remains TRUE for one call-up cycle.
bInitDone	Bool	binary	Status of initialisation TRUE Integrator has been initialised



Function library LenzeMotionControl2.lib

Function blocks for axis synchronisation

L_MCPHiIntegrator - integrator with TP set function

Outputs ⇨ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nState	Integer	-	Status
			0 Input parameters permissible, integrator is calculated.
			10 Function "Reset integrator" (<i>bReset</i>) is active.
			20 Function "Load integrator" (<i>bLoad</i>) is active.
			-10 Calculation was not carried out because gear factor denominator = 0 or gear factor numerator = -32768

Function

The input value nIn_v is evaluated with the gear factors $nGearNum$ and $nGearDen$, integrated and output as 16-bit ($nOut16$) or 32-bit ($dnOut32_p$) width.

$$nOut16 = nOut16 + nIn_v \cdot \frac{nGearNum}{nGearDen}$$

$$dnOut32_p = dnOut32_p + nIn_v \cdot \frac{nGearNum}{nGearDen}$$

- The gear factor evaluation uses residual values.
- A positive or negative integrator overflow is indicated by:
 - $bOvfl16 = \text{TRUE}$ for output $nOut16$,
 - $bOvfl32 = \text{TRUE}$ for output $dnOut32_p$.
 - The signal time is 1 task cycle.

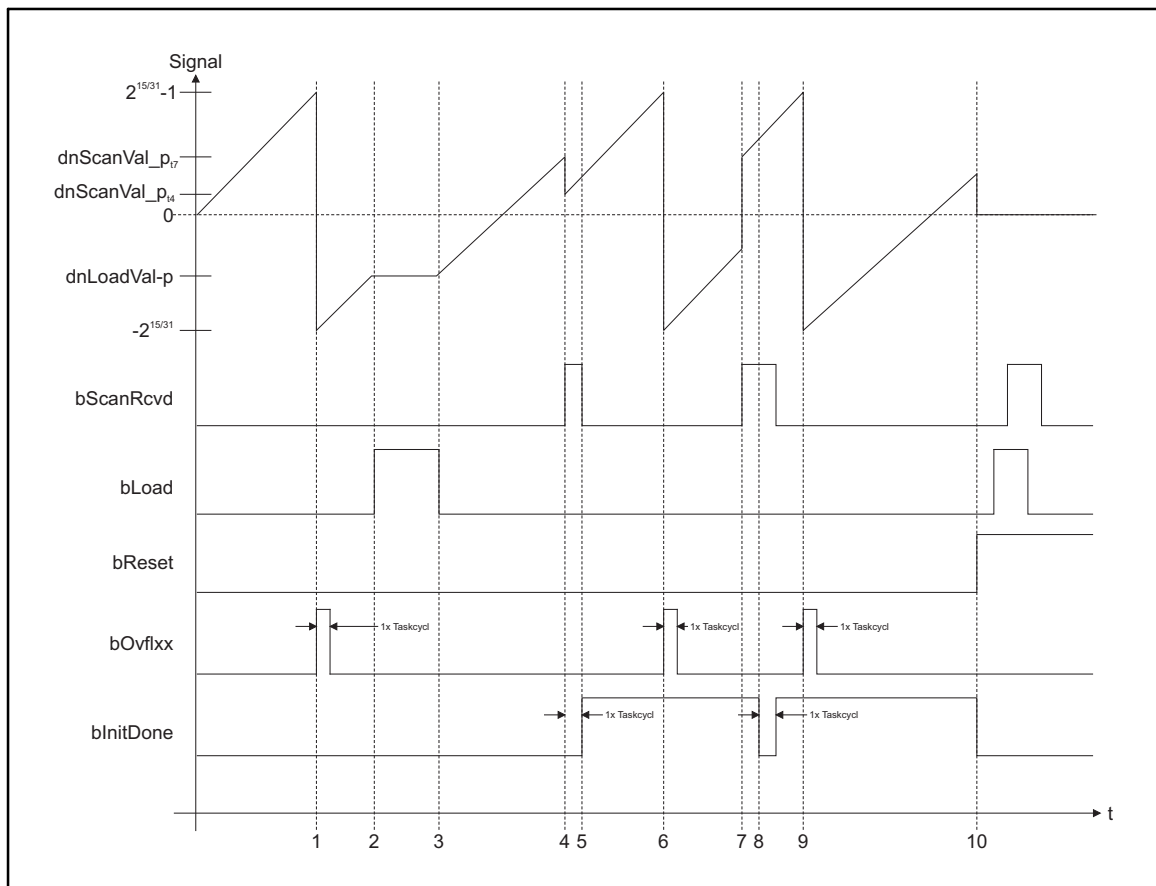
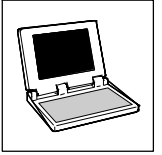


Fig. 2-4

Signal flow



Example

Use the current speed at *nIn_v* and the gear factors to build up a specific machine measuring system from the motor measuring system.

- 2^{16} inc in the motor measuring system should correspond to 1 motor revolution.
- 2^{16} inc in the machine measuring system (*nOut16*) should correspond to 1 machine shaft revolution.

1. Integrator initialisation

- If a FALSE-TRUE transition occurs at *bScanRcvd* the integrator is initialised, i.e. compensation with *dnScanVal_p* and *dnLoadVal_p*:

$$nOut16 = (INT)(dnLoadVal_p + dnScanVal_p)$$

$$dnOut32_p = dnLoadVal_p + dnScanVal_p$$

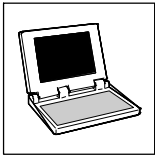
- After successful initialisation, the output *blnitDone* will be set to TRUE in the next call up cycle.
 - If the initialisation is repeated without previous reset (*bReset* = TRUE) the output *blnitDone* will be set to FALSE for 1 task cycle.
- #### 2. Reset integrator (*bReset* = TRUE)
- The integrator is set to 0, calculations are not carried out.
 - The Boolean outputs are set to FALSE.
 - The input *bReset* has the highest priority.
- #### 3. Load integrator (*bLoad* = TRUE)
- The integrator is loaded with the value of *dnLoadVal_p* and the outputs are set.
 - As long as *bLoad* is set to TRUE the input is not added and the drive does not check if an overflow occurred.
 - If *bLoad* is reset to FALSE the integrator continues operation from the loaded value on. The output *blnitDone* will be reset to FALSE with a delay of one call up cycle after a FALSE-TRUE transition at *bScanRcvd*.



Tip!

For 9300 Servo PLC or ECS:

- Current speed: *MCTRL_nAct_v*
- Actual phase value as analog signal: *MCTRL_nPos_a*



Function library LenzeMotionControl2.lib

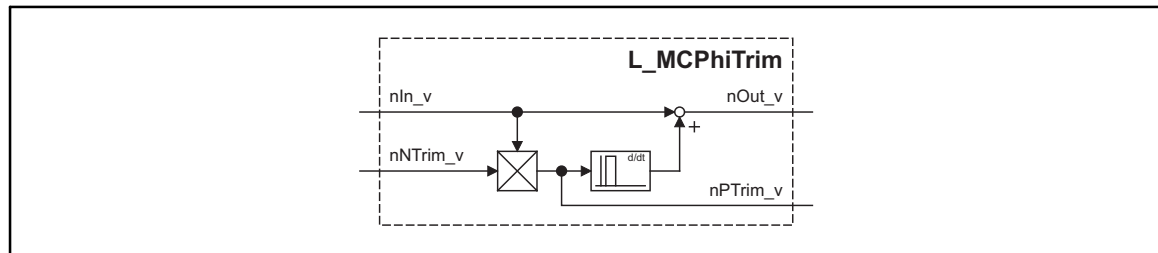
Function blocks for axis synchronisation

L_MCPHiTrim - speed-dependent phase trimming

2.4

L_MCPHiTrim - speed-dependent phase trimming

This FB calculates a speed-dependent phase value. This value is added to the current speed value nIn_v .



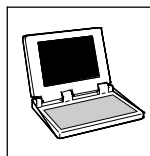
⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
nIn_v	Integer	Velocity	Input value
$nNTrim_v$	Integer	Velocity	Phase trimming limit value at 15000 rpm

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
$nOut_v$	Integer	Velocity	Trimming output value in [inc/ms] • Value is limited to ± 32767 [inc/ms]
$nPTrim_v$	Integer	Velocity	Current phase trimming value

The output value $nOut_v$ is calculated as follows:

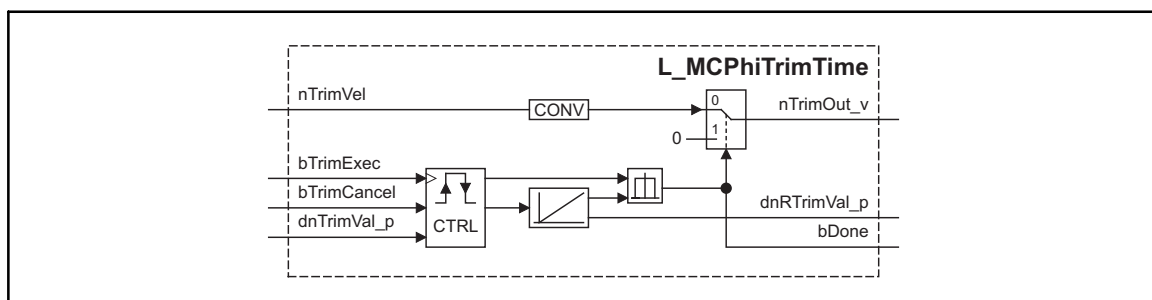
$$nOut_v = nIn_v + \frac{d}{dt} \cdot \left(\frac{nIn_v \cdot nNTrim_v}{16384} \right)$$

- The value of the summand $\frac{d}{dt} \cdot \left(\frac{nIn_v \cdot nNTrim_v}{16384} \right)$ is limited to ± 32767 [inc/ms].
- The output value $nOut_v$ is limited to ± 32767 [inc/ms].



2.5 L_MCPHiTrimTime - time-based phase trimming

This FB is used for relative phase trimming. After the start, the trimming path selected at input *dnTrimVal_p* will be traversed at trimming speed *nTrimVel*.



⇒ Inputs (Variable type: VAR_INPUT)							
Identifiers	Data type	Signal type	Info/possible settings				
nTrimVel	Integer	velocity	Trimming speed in [inc/s] A new value will be accepted after a FALSE-TRUE or TRUE-FALSE transition at the input <i>bTrimExec</i>.				
bTrimExec	Bool	binary	Start trimming <table><tr><td>FALSE ↗ TRUE</td><td>Start trimming</td></tr><tr><td>TRUE ↘ FALSE</td><td> A running trimming process is stopped. </td></tr></table>	FALSE ↗ TRUE	Start trimming	TRUE ↘ FALSE	A running trimming process is stopped.
FALSE ↗ TRUE	Start trimming						
TRUE ↘ FALSE	A running trimming process is stopped.						
bTrimCancel	Bool	binary	Cancel trimming <table><tr><td>TRUE</td><td>Cancel trimming</td></tr></table>	TRUE	Cancel trimming		
TRUE	Cancel trimming						
dnTrimVal_p	Double integer	position	Trimming path in [inc] 65536 inc = 1 motor revolution				

Outputs ⇒ (Variable type: VAR_OUTPUT)					
Identifiers	Data type	Signal type	Info		
nTrimOut_v	Integer	velocity	Phase trimming value in [inc/ms]		
dnRTrimVal_p	Double integer	position	Trimming path still to be traversed		
bDone	Bool	binary	Status of trimming <table><tr><td>TRUE</td><td>Trimming has been completed</td></tr></table>	TRUE	Trimming has been completed
TRUE	Trimming has been completed				

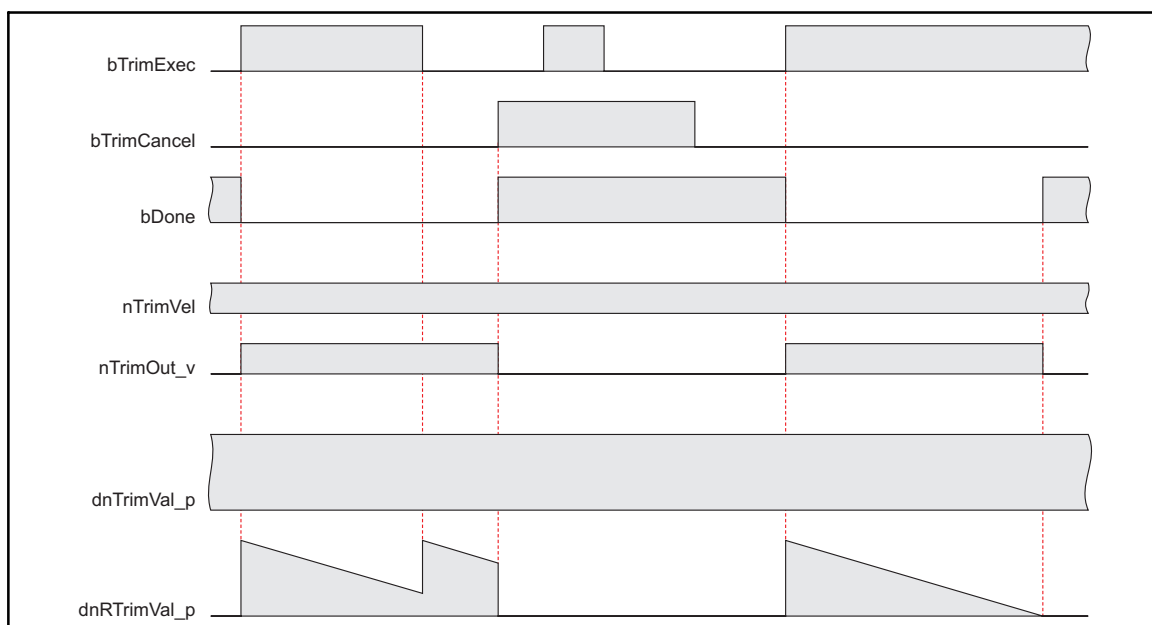
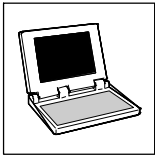


Fig. 2-5

Signal flow



Function library LenzeMotionControl2.lib

Function blocks for axis synchronisation

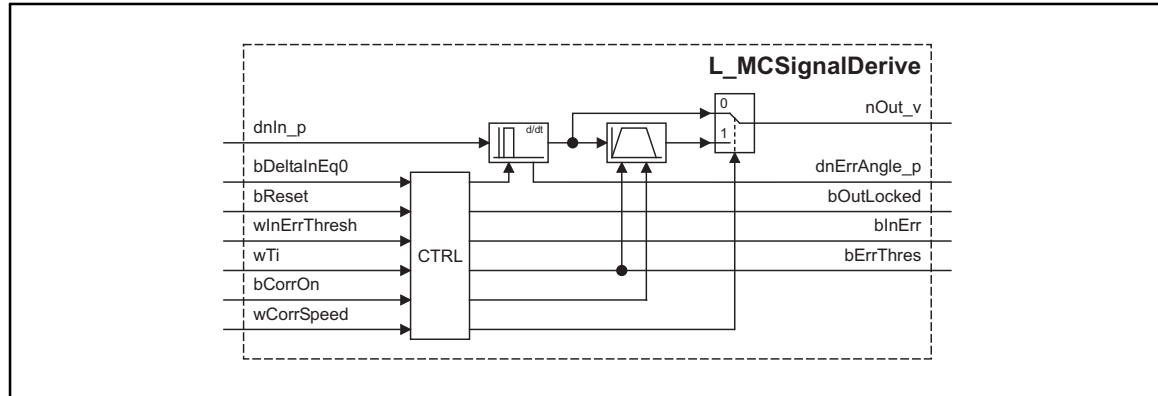
L_MCSignalDerive - signal differentiation

2.6

L_MCSignalDerive - signal differentiation

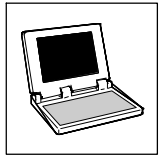
This FB processes and monitors absolute phase information, for instance transferred via a bus system.

- The absolute phase at the input *dnIn_p* is differentiated and output as phase difference to *nOut_v*.
- The receipt can only be monitored by a binary signal (speed = 0) of the signal source.



⇒ Inputs (Variable type: VAR_INPUT)			
Identifiers	Data type	Signal type	Info/possible settings
dnIn_p	Double integer	position	Phase input
bDeltalnEq0	Bool	binary	Configuration monitoring
			TRUE n-source = 0 (monitoring OFF)
			FALSE n-source ≠ 0
bReset	Bool	binary	Error reset
			TRUE dnErrAngle_p = 0
			TRUE ↗ FALSE bErrThres = FALSE (error reset after the second FB call)
wlnErrThresh	Word	-	Limit value for error cycles
wTi	Word	-	Ramp time
			• 100 = 1 s (reference 15000 rpm)
			⇒ in 1 s the speed changes by 16384 inc/ms = 15000 rpm
			• 0 = ramp switched off
bCorrOn	Bool	binary	Phase error correction
			TRUE Phase error correction active
wCorrSpeed	Word	-	Max. correction speed as relative offset in [inc/call-up cycle]
			• Value is internally limited to 0 ... 16384

Outputs ⇒ (Variable type: VAR_OUTPUT)			
Identifiers	Data type	Signal type	Info
nOut_v	Integer	velocity	Speed output (phase difference)
			• Value is internally limited to ± 16384 = 15000 rpm
dnErrAngle_p	Double integer	position	Added phase error
bOutLocked	Bool	binary	Ramp status
			FALSE
			• Acceleration/deceleration along the ramps or a correction in process.
			• Too many faulty task cycles (bErrThres = TRUE).
			TRUE otherwise
			Error status
bInErr	Bool	binary	TRUE Error (see the following table)
bErrThresh	Bool	binary	TRUE Error (number of errors > wlnErrThresh)
			• The added phase error dnErrAngle_p is reset to 0.



Monitoring function

Monitoring is based on the phase progress of $dnIn_p$ (speed) and the configuration of the input $bDeltaInEq0$:

Phase progress ($dnIn_p_{t0} - dnIn_p_{t0-1} = 0$)	Configuration monitoring ($bDeltaInEq0_b$)	Error output ($bInErr_b$)
FALSE	Any	FALSE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

1. If the condition for $bInErr = TRUE$ is met, the last valid output value $nOut_v$ will be extrapolated. The phase already passed is added to $dnErrAngle_p$. (See Fig. 2-6, $bInErr$, signal at $t2$ and $t3$.)
2. If the number of faulty task cycles ($bInErr = TRUE$) exceeds the value under $wInErrThresh$ the output $bErrThresh$ will also be set to TRUE and $dnErrAngle_p$ will be reset to "0". (See Fig. 2-6, signal at $t4$.)
 - The output signal $nOut_v$ is decelerated to 0 along the ramp wTi .
 - With a ramp time $wTi = 0$ s, ramp creation is switched off. The output value $nOut_v$ jumps to 0.
 - A TRUE-FALSE transition resets the error at $bReset$ if $bInErr = FALSE$. We distinguish between:

Speed value = 0 (at the time of reset)	Speed value $\neq 0$ (at the time of reset)
Output $nOut_v$ outputs the speed which equals $dnIn_p$ without ramp. • See Fig. 2-6, signal at $t6$.	$nOut_v$ is set to the speed value with the ramp time wTi which equals $dnIn_p$. If this speed value is reached, the ramp generator is switched off and the speed can be directly output.

3. If correct signals are detected within $wInErrThresh$ task cycles at $dnIn_p$ and $bDeltaInEq0$ ($bErr = FALSE$), the procedure is as follows:
 - Calculation of the setpoint angle from the angle value $dnIn_p$ before and after the interruption ($Xset = dnIn_p_{t0} - dnIn_p_{t0-errorcycles}$).
 - Subtraction of the angle output at $nOut_v$ ($Xdiff = Xset - Xout$).
 - Transfer of the difference to $dnErrAngle_p$. With $bCorrOn = TRUE$ the error angle is compensated by adding the speed offset $wCorrSpeed$ to the output value $nOut_v$ (see Fig. 2-6, signal at $t3$).
 - The output $bInErr$ is set to FALSE.

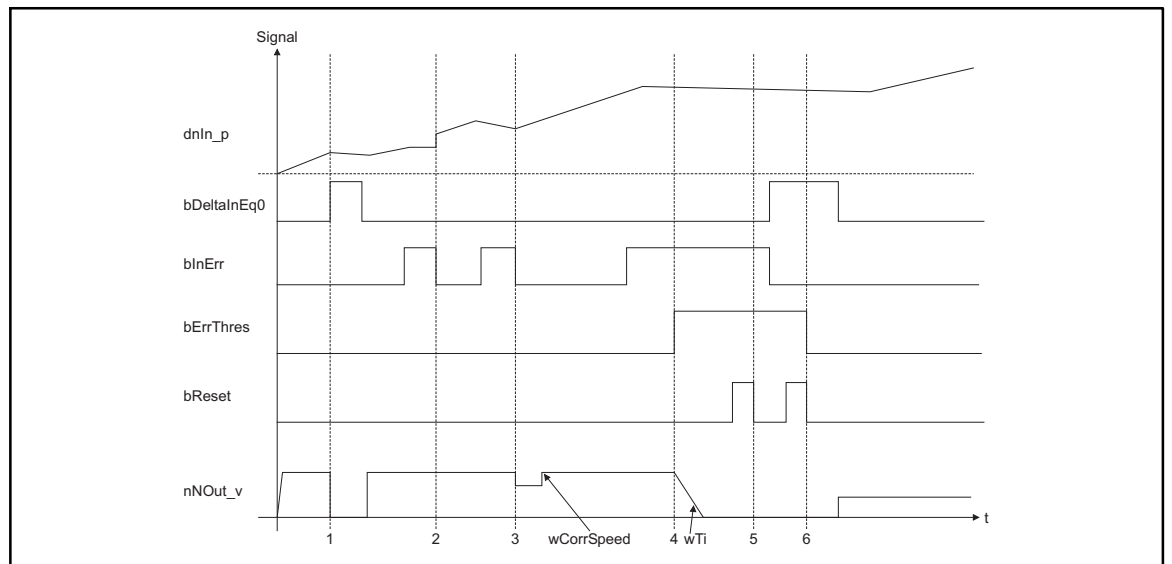
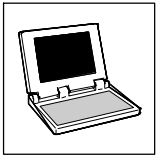


Fig. 2-6

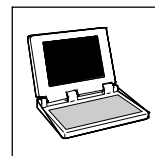
Signal flow



Function library LenzeMotionControl2.lib

Function blocks for axis synchronisation

L_MCSignalDerive - signal differentiation



3 Index

I

Integration and differential values of speed signals
(L_MCPhiDiff), 2-5

Speed-dependent phase trimming, 2-6

Integrator with TP set function (L_MCPhiIntegrator), 2-7

L

L_MCInterpolator, 2-2

L_MCPhiDiff, 2-5

L_MCPhiIntegrator, 2-7

L_MCPhiTrim, 2-10

L_MCPhiTrimTime, 2-11

L_MCSignalDerive, 2-12

S

Safety information, Layout

Other notes, 1-1

Warning of material damage, 1-1

Signal differentiation (L_MCSignalDerive), 2-12

Signal interpolation (L_MCInterpolator), 2-2

Speed-dependent phase trimming, 2-6

Speed-dependent phase trimming (L_MCPhiTrim), 2-10

T

Term definitions, 1-2

Time-based phase trimming (L_MCPhiTrimTime), 2-11

U

unit, 1-2