

Lenze

Handbuch

IEC 61131-3

inside

***Global Drive
PLC Developer Studio***



Global Drive

Funktionsbibliothek

LenzeMotionControl2.lib

Diese Dokumentation ist gültig für die Funktionsbibliothek **LenzeMotionControl2.lib** ab Version 5.00.

Die Funktionsbibliothek **LenzeMotionControl2.lib** kann für folgende Lenze PLC-Geräte verwendet werden:

	Typenbezeichnung	ab Hardwarestand	ab Softwarestand
9300 Servo PLC	EVS93XX-xT	6A	6.0
ECSxA	ECSxAxxx	1C	7.0

Software-Systemvoraussetzungen:

Drive PLC Developer Studio Professional ab V2.2

Wichtiger Hinweis:

Die Software wird dem Benutzer in der vorliegenden Form zur Verfügung gestellt. Alle Risiken hinsichtlich der Qualität und der durch ihren Einsatz ermittelten Ergebnisse verbleiben beim Benutzer. Entsprechende Sicherheitsvorkehrungen gegen eventuelle Fehlbedienungen sind vom Benutzer vorzusehen.

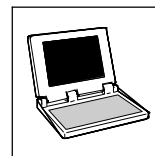
Wir übernehmen keine Verantwortung für direkt oder indirekt entstandene Schäden, z. B. Gewinnverluste, Auftragsverluste oder geschäftliche Beeinträchtigungen jeglicher Art.

© 2008 Lenze Drive Systems GmbH

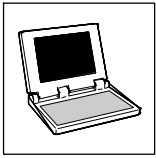
Ohne besondere schriftliche Genehmigung von Lenze Drive Systems GmbH darf kein Teil dieser Dokumentation vervielfältigt oder Dritten zugänglich gemacht werden.

Wir haben alle Angaben in dieser Dokumentation mit größter Sorgfalt zusammengestellt und auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Trotzdem können wir Abweichungen nicht ganz ausschließen. Wir übernehmen keine juristische Verantwortung oder Haftung für Schäden, die dadurch eventuell entstehen. Notwendige Korrekturen werden wir in die nachfolgenden Auflagen einarbeiten.

Alle in dieser Dokumentation aufgeführten Markennamen sind Warenzeichen ihrer jeweiligen Besitzer.

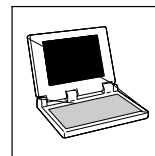


1 Vorwort und Allgemeines	1-1
1.1 Über dieses Handbuch	1-1
1.1.1 Konventionen in diesem Handbuch	1-1
1.1.2 Piktogramme in diesem Handbuch	1-1
1.1.3 Aufbau der Beschreibungen	1-2
1.1.4 Verwendete Begriffe	1-2
1.2 Versionskennung der Funktionsbibliotheken	1-3
2 Funktionsblöcke für Achsensynchronisierungen	2-1
2.1 L_MCInterpolator - Signalinterpolation	2-2
2.2 L_MCPhiDiff - Integration und Differenzbildung von Drehzahlssignalen	2-5
2.3 L_MCPhiIntegrator - Integrator mit TP-Set Funktion	2-7
2.4 L_MCPhiTrim - Winkeltrimmung drehzahlabhängig	2-10
2.5 L_MCPhiTrimTime - Winkeltrimmung zeitbasierend	2-11
2.6 L_MCSignalDerive - Signaldifferentiation	2-12
3 Index	3-1



Funktionsbibliothek LenzeMotionControl2.lib

Inhalt



1 Vorwort und Allgemeines

1.1 Über dieses Handbuch

Dieses Handbuch enthält Informationen zur Funktionsbibliothek **LenzeMotionControl2.lib** für das **Drive PLC Developer Studio**.

- In der Funktionsbibliothek **LenzeMotionControl2.lib** sind Funktionen zur Realisierung von Achsensynchronisierungsaufgaben enthalten.
- In der Funktionsbibliothek **LenzeMotionControl1.lib** sind Funktionen zur Realisierung von Positionieraufgaben enthalten.
- Die Funktionsbibliotheken sind Bestandteile des **Software Package - Positioner**.

1.1.1 Konventionen in diesem Handbuch

Dieses Handbuch verwendet folgende Konventionen zur Unterscheidung von verschiedenen Arten von Informationen:

Variablenbezeichner

sind im erklärenden Text in Kursivschrift dargestellt:

- "Mit *bReset...*"



Tipp!

Informationen zu den Konventionen, die für die Variablenbezeichner von Lenze Systembausteinen, Funktionsblöcken sowie Funktionen verwendet werden, um eine einheitliche und durchgängige Benennung zu gewährleisten und dadurch die Lesbarkeit von SPS-Programmen zu fördern, finden Sie im Anhang der DDS-Online-Dokumentation "Einführung in die IEC 61131-3 Programmierung".

Lenze Funktionen/Funktionsblöcke

erkennen Sie am Namen. Sie beginnen immer mit einem "L_":

- "Mit dem FB **L_MCPProfileGenerator...**"

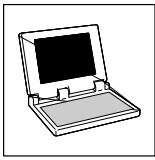
Programmlistings

sind in der Schriftart "Courier" aufgeführt, wobei Schlüsselwörter fett ausgezeichnet sind:

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

1.1.2 Piktogramme in diesem Handbuch

	verwendete Piktogramme	Signalwörter	
Warnung vor Sachschäden		Stop!	Warnt vor möglichen Sachschäden . Mögliche Folgen bei Missachtung: Beschädigung des Antriebsreglers/Antriebssystems oder seiner Umgebung.
Sonstige Hinweise		Tipp! Hinweis!	Kennzeichnet einen Tipp bzw. Hinweis.



Funktionsbibliothek LenzeMotionControl2.lib

Vorwort und Allgemeines Über dieses Handbuch

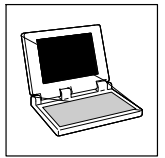
1.1.3 Aufbau der Beschreibungen

Die Beschreibungen der einzelnen Funktionen/Funktionsblöcke sowie Systembausteine in diesem Handbuch sind einheitlich nach folgendem Schema aufgebaut:

	Funktion	Funktionsblock (FB)/ Systembaustein (SB)
①	Überschrift mit Funktion und Bezeichner	
②	Deklaration der Funktion: • Datentyp des Rückgabewertes • Funktionsbezeichner • Liste der Übergabeparameter	-
③	Kurzbeschreibung mit den wichtigsten Eigenschaften	
④	Grafische Darstellung der Funktion mit den zugehörigen Variablen • Übergabeparameter • Rückgabewert	Grafische Darstellung des FB/SB mit den zugehörigen Variablen • Eingangsvariablen • Ausgangsvariablen
⑤	Tabelle mit Informationen zu den Übergabeparametern: • Bezeichner • Datentyp • Einstellmöglichkeiten • Info	Tabelle mit Informationen zu den Ein- und Ausgangsvariablen: • Bezeichner • Datentyp • Variablentyp • Einstellmöglichkeiten • Info
⑥	Tabelle mit Informationen zum Rückgabewert: • Datentyp des Rückgabewertes • Mögliche Rückgabewerte und ihre Bedeutung	-
⑦	Weitere Informationen (Anmerkungen, Tipps, Anwendungsbeispiele, etc.)	

1.1.4 Verwendete Begriffe

Begriff	Im folgenden Text verwendet für
Cxxx/y	Subcode y der Codestelle Cxxxx (z. B. C0470/3 = Subcode 3 der Codestelle C0470)
DDS	Drive PLC Developer Studio
FB	Funktionsblock
GDC	Global Drive Control (Parametrier-Programm von Lenze)
Parameter-Codestellen	Codestellen zum Einstellen der Funktionalität eines Funktionsblocks
PLC	9300 Servo PLC - Achsmodul ECSxA "Application"
SB	Systembaustein
[unit]	Platzhalter für die physikalische Einheit des verwendeten externen Maßsystems (z. B. "µm", "mm", "cm")



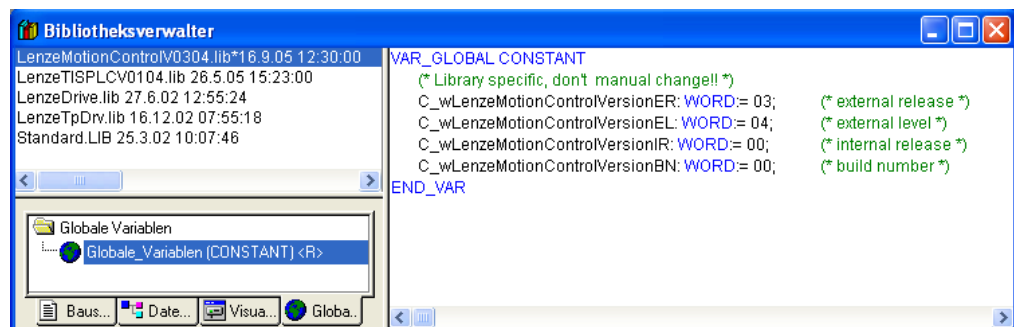
1.2 Versionskennung der Funktionsbibliotheken

Versionskennung **ab Version 3.04:**

Konstante	Bedeutung	Beispiel Wert
C_w[Funktionsbibliotheksname]VersionER	Externes Release	03
C_w[Funktionsbibliotheksname]VersionEL	Externes Level	04
C_w[Funktionsbibliotheksname]VersionIR	Internes Release	00
C_w[Funktionsbibliotheksname]VersionBN	Build-Nummer	00

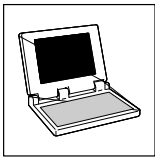
Version: 03 04 00 00

- Im Beispiel bedeutet der Wert "03040000" Version "3.04.00.00".



Im Offline-Modus können Sie sich den Wert dieser Konstanten über die **Bibliotheksverwaltung/Register Globale Variablen** anzeigen lassen:

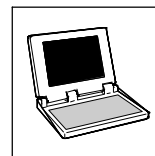
- Die Bibliotheksverwaltung öffnen Sie im DDS über den Menü-Befehl **Fenster→Bibliotheksverwaltung**.
- Wählen Sie die Registerkarte **Globale Variablen** und öffnen Sie den Ordner **Globale Variablen**.
- Wenn Sie die *Globale_Variablen...* markieren, erscheint die Version im Bibliotheksverwalter.



Funktionsbibliothek LenzeMotionControl2.lib

Vorwort und Allgemeines

Versionskennung der Funktionsbibliotheken



2 Funktionsblöcke für Achsensynchronisierungen

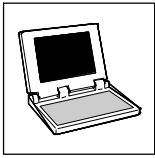
Dieses Kapitel enthält Informationen zu den Funktionsblöcken, die Sie für Achsensynchronisierungen verwenden können.



Hinweis!

Die in diesem Kapitel beschriebenen Funktionsblöcke müssen aufgrund ihrer internen Struktur in einer zeitäquidistanten Task aufgerufen werden (z. B. in einer 5 ms-Task).

Beachten Sie: Die zyklische Task PLC_PRG ist nicht zeitäquidistant!



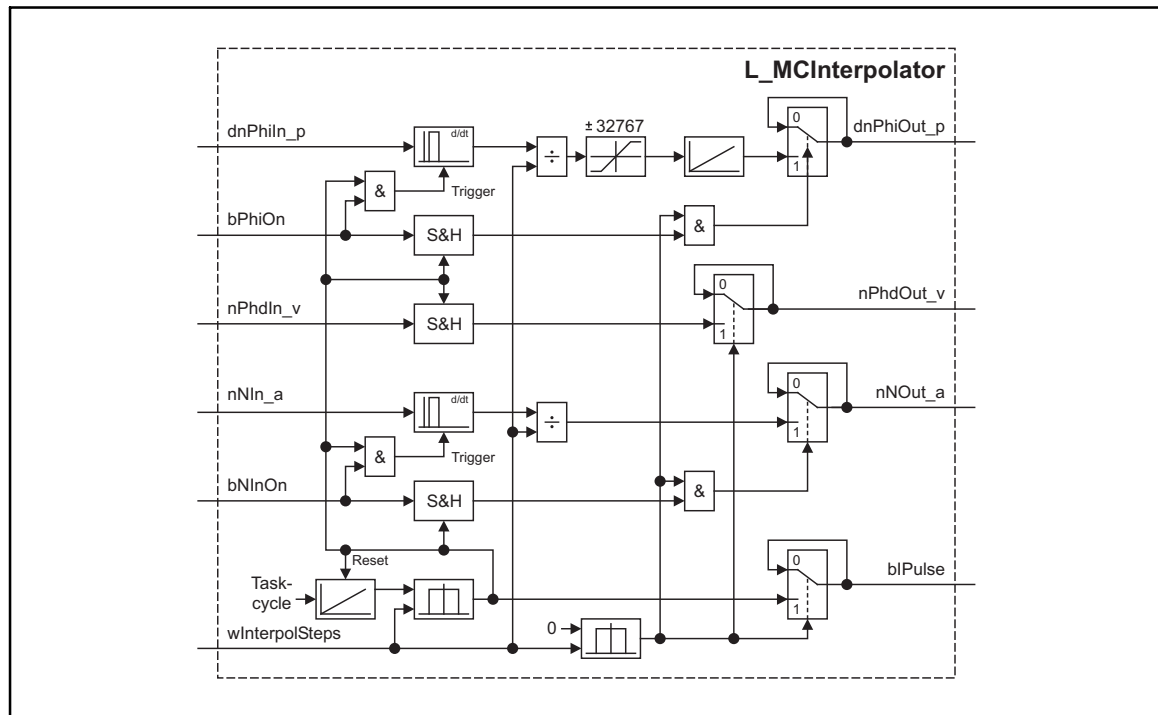
Funktionsbibliothek LenzMotionControl2.lib

Funktionsblöcke für Achsensynchronisierungen

L_MCInterpolator - Signalinterpolation

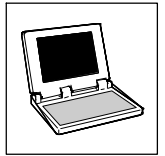
2.1 L_MCInterpolator - Signalinterpolation

Dieser FB interpoliert linear Differenzsignale einer langsameren Task innerhalb einer aufrufenden schnelleren Task, so dass das Ausgangssignal den Pegel des entsprechenden Eingangssignals erst nach der Anzahl Taskzyklen erreicht, die über den Eingang *wInterpolSteps* vorgegeben wurde.



⇒ Eingänge (Variablentyp: VAR_INPUT)				
Bezeichner	Datentyp	Signaltyp	Info/Einstellmöglichkeiten	
dnPhiIn_p	Double Integer	position	Winkelsignal in [inc]	
bPhiOn	Bool	binary	Aktivierung Winkelbearbeitung	
			TRUE	Winkelsignal (<i>dnPhiIn_p</i>) wird interpoliert
			FALSE	Interpolation wird angehalten
nPhdIn_v	Integer	velocity	Drehzahlsignal (Winkeldifferenz) in [inc/ms]	
nNIn_a	Integer	analog	Analoges Drehzahlsignal	
bNInOn	Bool	binary	Aktivierung analoger Drehzahleingang	
			TRUE	Drehzahleingang (<i>nNIn_a</i>) aktivieren
wInterpolSteps	Word	-	Anzahl der L_McInterpolator-Aufrufe und somit Interpolationsschritte - Verhältnis der Taskzykluszeit (Langsame Task/Schnelle Task) - Eingabebereich 0 ... 32767 - Bei Vorgabe einer 0 wird der FB nicht bearbeitet, die aktuellen Ausgangswerte bleiben unverändert.	

Ausgänge ⇨ (Variablentyp: VAR_OUTPUT)			
Bezeichner	Datentyp	Signaltyp	Info
dnPhiOut_p	Double Integer	position	Interpolierter Winkel
nPhdOut_v	Integer	velocity	Interpolierte Drehzahl (Winkeldifferenz)
nNOOut_a	Integer	analog	Interpolierte Drehzahl (analog)
bIPulse	Bool	binary	Datenübernahme-Impuls (1 Taktzyklus) TRUE Datenübernahme erfolgt



Anwendung (Beispiel)

Ein in einer langsamen Task erzeugtes Sollwertsignal wird vom FB **L_MCInterpolator** innerhalb einer schnelleren Task interpoliert und dann der Motorsteuerung zur Verfügung gestellt:

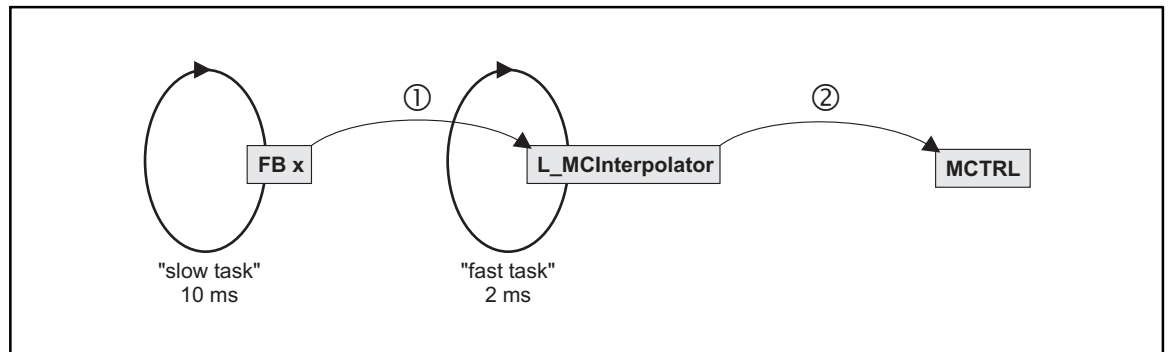


Abb. 2-1

Beispiel: Aufbereitung eines Sollwertsignals für die Motorsteuerung

- ① Sollwertsignal aus einer langsamen Task
- ② Interpoliertes Sollwertsignal aus einer schnelleren Task, in der der FB **L_MCInterpolator** aufgerufen wird.

Das Ausgangssignal erreicht den Pegel des entsprechenden Eingangssignals erst nach Ablauf von *wInterpolSteps* Taskzyklen der schnelleren Task:

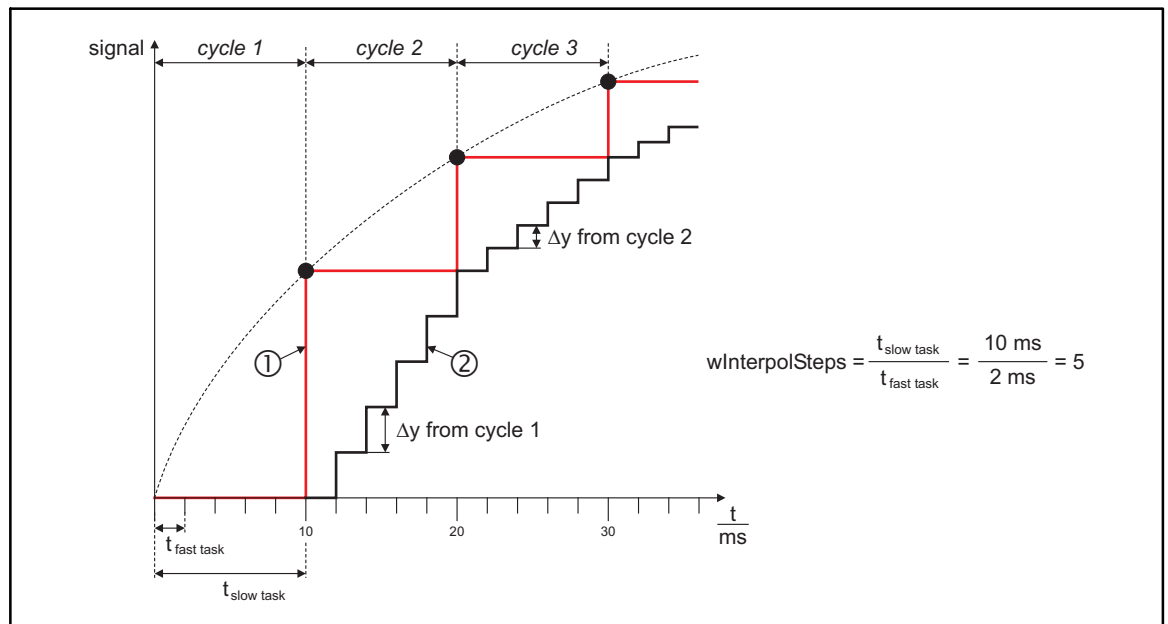
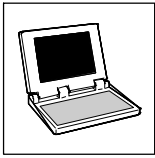


Abb. 2-2

Signalverlauf

- ① Sollwertsignal aus einer langsamen Task
- ② Interpoliertes Sollwertsignal aus einer schnelleren Task, in der der FB **L_MCInterpolator** aufgerufen wird.



Funktionsbibliothek *LenzeMotionControl2.lib*

Funktionsblöcke für Achsensynchronisierungen

L_MCInterpolator - Signalinterpolation

Interpolationsschritte

Die Anzahl der Interpolationsschritte, die am Eingang *wInterpolSteps* vorzugeben ist, berechnet sich wie folgt:

$$wInterpolSteps = \frac{\text{Taskzykluszeit (langsameTask)}}{\text{Taskzykluszeit (schnelleTask)}}$$



Hinweis!

Ändern Sie die Anzahl der Interpolationsschritte nicht während des Betriebs, da sonst Fehler bei der Interpolation entstehen.

Winkelwerte

- Die Funktion wird mit *bPhiOn* = TRUE aktiviert.
- Der Eingangswert *dnPhIn_p* wird jeden n-ten Taskzyklus (Aufruf des FB **L_MCInterpolator**) differenziert, durch n mit Restwertbehandlung geteilt und integriert.
- Der so gebildete interpolierte Winkel wird an *dnPhiOut_p* ausgegeben.
- n entspricht dem Wert von *wInterpolSteps*.

Winkeldifferenzsignale

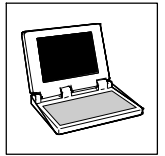
- Die Funktion ist ständig aktiv.
- Der Eingangswert *nPhdIn_v* wird jeden n-ten Taskzyklus (Aufruf des FB **L_MCInterpolator**) auf den Ausgang *nPhdOut_v* kopiert und bis zum nächsten Kopiervorgang konstant gehalten.
- Bei diesem Signaltyp erfolgt keine Interpolation, sondern lediglich eine koordinierte Datenübergabe.
- n entspricht dem Wert von *wInterpolSteps*.

Analogsignal

- Die Funktion wird mit *bNInOn* = TRUE aktiviert.
- Der Eingangswert *nNIn_a* wird jeden n-ten Taskzyklus (Aufruf des FB **L_MCInterpolator**) differenziert, durch n mit Restwertbehandlung geteilt, integriert und an *nNOut_a* ausgegeben.
- n entspricht dem Wert von *wInterpolSteps*.

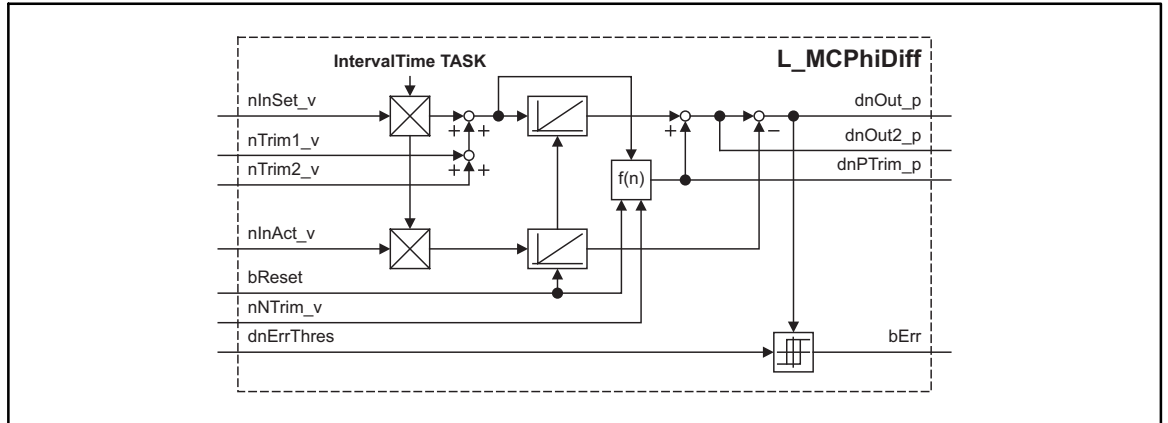
Weitere Ausgangssignale

- Die Übernahme der Eingangswerte wird durch *bIPulse* = TRUE signalisiert (Impulsdauer: 1 Taskzyklus).



2.2 L_MCPHiDiff - Integration und Differenzbildung von Drehzahlsignalen

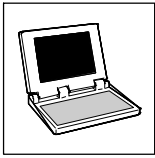
Der FB integriert die Drehzahlsignale $nInSet_v$, $nTrim1_v$, $nTrim2_v$ sowie $nInAct_v$ zu Winkelwerten und bildet einen Differenzwinkel (Schleppfehler).



⇒ Eingänge (Variablentyp: VAR_INPUT)			
Bezeichner	Datentyp	Signaltyp	Info/Einstellmöglichkeiten
$nInSet_v$	Integer	velocity	Winkeldifferenz-Sollwert in [inc/ms]
$nTrim1_v$	Integer	velocity	Additiver Trimmwert in [Inc/Aufrufzyklus]
$nTrim2_v$	Integer	velocity	Additiver Trimmwert in [Inc/Aufrufzyklus]
$nInAct_v$	Integer	velocity	Winkeldifferenz-Istwert in [inc/ms]
$nNTrim_v$	Integer	velocity	n-abhängige Winkeltrimmung in [inc/ms]
$bReset$	Bool	binary	Lagedifferenz zurücksetzen
			TRUE Lagedifferenz = 0 setzen
$dnErrThres$	Double Integer	-	Schleppfehlergrenze

Ausgänge ⇒ (Variablentyp: VAR_OUTPUT)			
Bezeichner	Datentyp	Signaltyp	Info
$dnOut_p$	Double Integer	position	Schleppfehler
$dnOut2_p$	Double Integer	position	Integrierter Sollwinkel
$dnPTrim_p$	Double Integer	position	n-abhängige Winkeltrimmung
$bErr$	Bool	binary	Status Grenzwert-Überwachung
			TRUE Grenzwert überschritten ($dnOut > dnErrThres$)

- Der Betrag des Schleppfehlers wird auf einen Grenzwert ($dnErrThres$) überwacht.
- Die Addition eines linear vom Sollwert abhängigen Winkelwertes $f(n)$ ist möglich.



Funktionsbibliothek *LenzeMotionControl2.lib*

Funktionsblöcke für Achsensynchronisierungen

L_MCPHiDiff - Integration und Differenzbildung von Drehzahlsignalen

Drehzahl-abhängige Winkeltrimmung

- Die Winkelverstellung ermöglicht ein Vor- oder Nacheilen des Winkels mit steigender Drehzahl.
- Über *nNTrim_v* wird eine entsprechende Verstellung in Inkrementen vorgegeben, die bei 15000 rpm des Antriebs erreicht wird (linearer Zusammenhang):

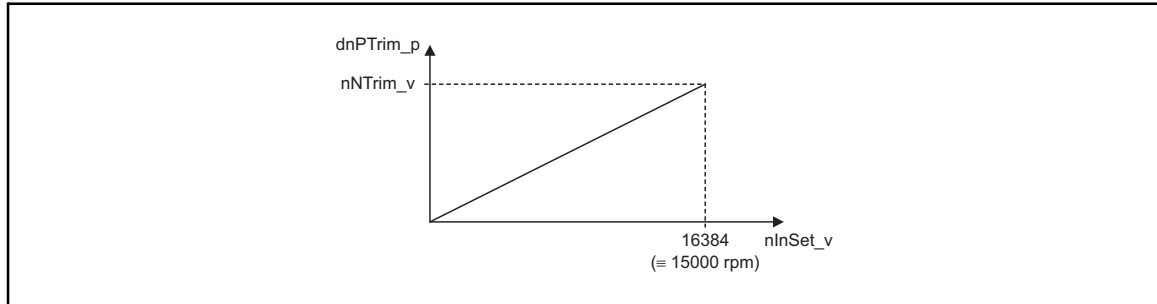
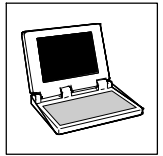


Abb. 2-3

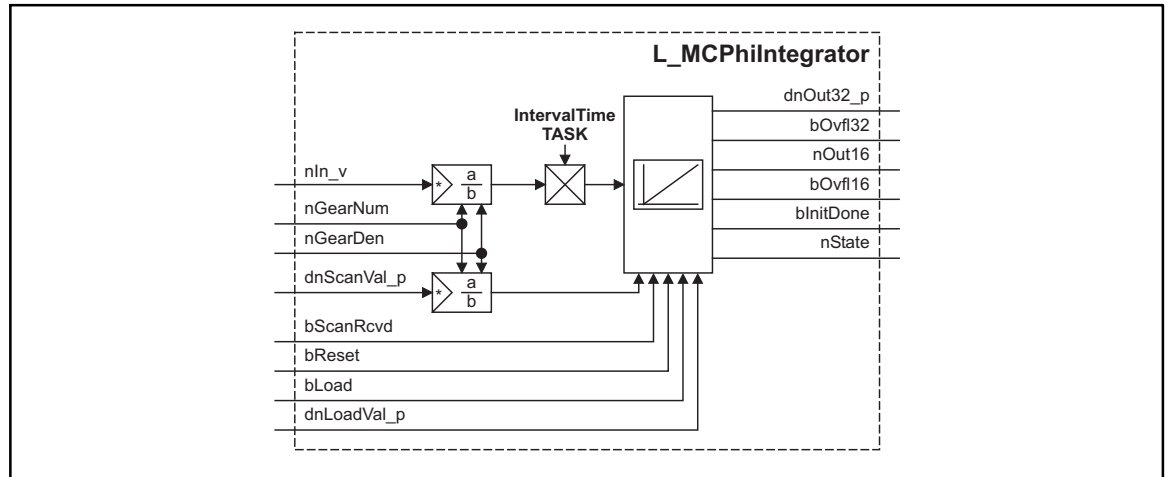
Linearer Zusammenhang bei der Drehzahl-abhängigen Winkeltrimmung



2.3 L_MCPHiIntegrator - Integrator mit TP-Set Funktion

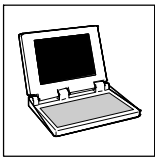
Dieser FB bewertet den Eingangswert *nIn_v* mit einem Getriebefaktor und summiert ihn in einem Integrator.

- Der Ausgang des Integrator hat eine Breite von 16 Bit und 32 Bit.
- Ein Überlauf des jeweiligen Ausgangs wird signalisiert.
- Der Integrator ist ladbar und lässt sich initialisieren (z. B. mit einem Touch-Probe-Signal).



⇒ Eingänge (Variablentyp: VAR_INPUT)			
Bezeichner	Datentyp	Signaltyp	Info/Einstellmöglichkeiten
nIn_v	Integer	velocity	Drehzahleingang (Winkeldifferenz)
nGearNum	Integer	-	Getriebefaktor Zähler
nGearDen	Integer	-	Getriebefaktor Nenner
dnScanVal_p	Double Integer	position	Initialisierungswert
Integratorfunktionen			
bScanRcvd	Bool	binary	FALSE → TRUE Integrator initialisieren
bReset	Bool	binary	
bLoad	Bool	binary	
dnLoadVal_p	Double Integer	position	TRUE Integrator zurücksetzen • Der Integrator wird nicht aufintegriert. TRUE Integrator laden mit <i>dnLoadVal_p</i> • Der Integrator wird nicht aufintegriert. • Der Ausgang <i>bInitDone</i> wird nach Rücksetzen von <i>bLoad</i> auf FALSE und anschließender FALSE-TRUE-Flanke an <i>bScanRcvd</i> mit einer Verzögerung von einem Aufrufzyklus auf FALSE zurückgesetzt.
			Wert, mit dem der Integrator geladen werden soll.

Ausgänge ⇒ (Variablentyp: VAR_OUTPUT)			
Bezeichner	Datentyp	Signaltyp	Info
dnOut32_p	Double Integer	position	Integratorwert (32 Bit)
bOvf32	Bool	binary	Überlauf (32 Bit)
			TRUE Überlauf • Der Ausgang bleibt für einen Aufrufzyklus gesetzt.
nOut16	Integer	-	Integratorwert (16 Bit)
bOvf16	Bool	binary	Überlauf (16 Bit)
			TRUE Überlauf • Der Ausgang bleibt für einen Aufrufzyklus gesetzt.
bInitDone	Bool	binary	Status Initialisierung
			TRUE Integrator ist initialisiert



Funktionsbibliothek LenzMotionControl2.lib

Funktionsblöcke für Achsensynchronisierungen

L_MCPHilIntegrator - Integrator mit TP-Set Funktion

Ausgänge ⇒ (Variablentyp: VAR_OUTPUT)			
Bezeichner	Datentyp	Signaltyp	Info
nState	Integer	-	Status
			0 Eingangsparameter zulässig, Integratorberechnung wird durchgeführt.
			10 Funktion "Integrator zurücksetzen" (<i>bReset</i>) ist aktiv.
			20 Funktion "Integrator laden" (<i>bLoad</i>) ist aktiv.
			-10 Berechnung wurde nicht durchgeführt, da Getriebefaktoren-nenner = 0 oder Getriebefaktor-zähler = -32768

Funktion

Der Eingangswert *nIn_v* wird mit den Getriebefaktoren *nGearNum* und *nGearDen* bewertet, aufintegriert und in 16 Bit (*nOut16*) bzw. 32 Bit (*dnOut32_p*) Breite ausgegeben.

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

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

- Die Bewertung des Getriebefaktors erfolgt mit Restwertbehandlung.
- Ein positiver oder negativer Überlauf des Integrators wird signalisiert durch:
 - *bOvfl16* = TRUE für den Ausgang *nOut16*,
 - *bOvfl32* = TRUE für den Ausgang *dnOut32_p*.
 - Die Dauer des Signals beträgt 1 Taskzyklus.

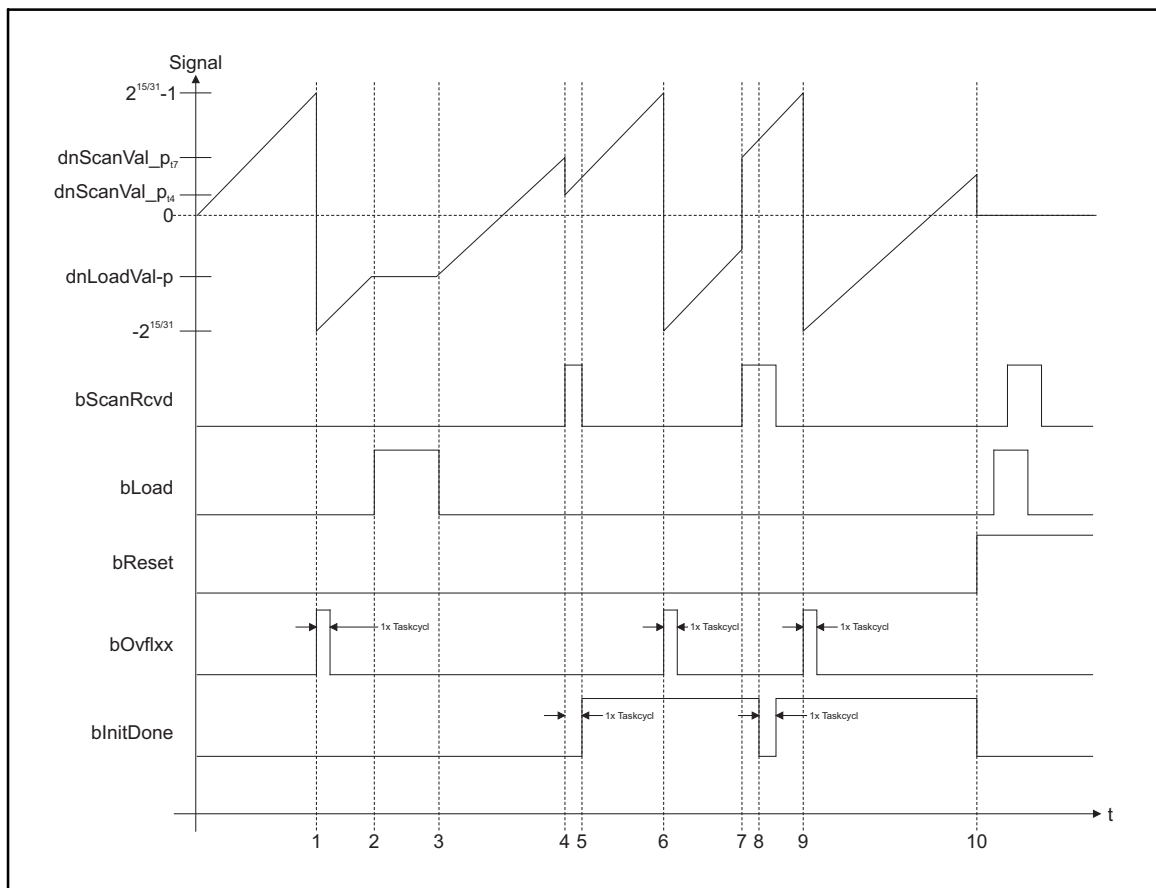
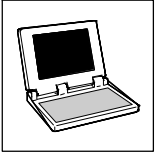


Abb. 2-4

Signalverlauf



Beispiel

Mit der aktuellen Drehzahl an *nIn_v* und den Getriebefaktoren können Sie aus dem Motormaßsystem ein spezifisches Maschinenmaßsystem bilden.

- 2^{16} inc im Motormaßsystem sollen 1 Umdrehung des Motors entsprechen.
- 2^{16} inc im Maschinenmaßsystem (*nOut16*) sollen 1 Umdrehung der Maschinenwelle entsprechen.

1. Integrator initialisieren

- Mit einer FALSE-TRUE-Flanke an *bScanRcvd* wird der Integrator initialisiert, d. h. es erfolgt eine Verrechnung mit *dnScanVal_p* und *dnLoadVal_p*:

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

$$dnOut32_p = dnLoadVal_p + dnScanVal_p$$

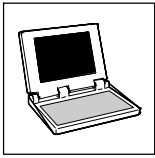
- Nach erfolgter Initialisierung wird der Ausgang *blnitDone* im nächsten Aufrufzyklus auf TRUE gesetzt.
 - Bei einer erneuten Initialisierung ohne vorherigen Reset (*bReset* = TRUE) wird für einen Taskzyklus der Ausgang *blnitDone* = FALSE gesetzt.
- #### 2. Integrator zurücksetzen (*bReset* = TRUE)
- Der Integrator wird auf 0 gesetzt, es werden keine Berechnungen durchgeführt.
 - Die boolschen Ausgänge werden auf FALSE gesetzt.
 - Der Eingang *bReset* hat die höchste Priorität.
- #### 3. Integrator laden (*bLoad* = TRUE)
- Der Integrator wird mit dem Wert an *dnLoadVal_p* geladen und die Ausgänge werden gesetzt.
 - Während *bLoad* auf TRUE gesetzt ist, erfolgt keine Aufintegration des Eingangs und es findet keine Überprüfung auf Überlauf statt.
 - Wird *bLoad* auf FALSE zurückgesetzt, läuft der Integrator von dem geladenen Wert aus weiter. Der Ausgang *blnitDone* wird nach einer FALSE-TRUE-Flanke an *bScanRcvd* mit einer Verzögerung von einem Aufrufzyklus auf FALSE zurückgesetzt.



Tipp!

Bei 9300 Servo PLC oder ECS:

- Aktuelle Drehzahl: *MCTRL_nAct_v*
- Winkelwert als Analogsignal: *MCTRL_nPos_a*



Funktionsbibliothek LenzeMotionControl2.lib

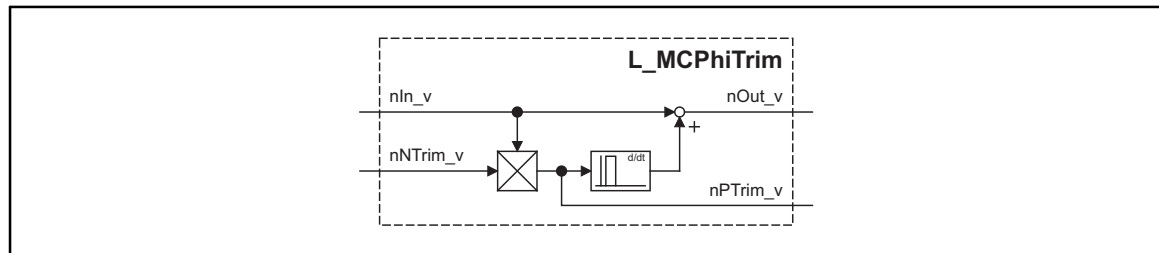
Funktionsblöcke für Achsensynchronisierungen

L_MCPHiTrim - Winkeltrimmung drehzahlabhängig

2.4

L_MCPHiTrim - Winkeltrimmung drehzahlabhängig

Dieser FB berechnet einen linear geschwindigkeitsabhängigen Winkelwert. Dieser wird zum aktuellen Geschwindigkeitswert nIn_v addiert.



⇒ Eingänge (Variablentyp: VAR_INPUT)			
Bezeichner	Datentyp	Signaltyp	Info/Einstellmöglichkeiten
nIn_v	Integer	velocity	Eingangswert
nNTrim_v	Integer	velocity	Endwert Winkeltrimmung bei 15000 rpm

Ausgänge ⇒ (Variablentyp: VAR_OUTPUT)			
Bezeichner	Datentyp	Signaltyp	Info
nOut_v	Integer	velocity	Ausgangswert Trimmung in [inc/ms] • Wert wird auf ±32767 [inc/ms] begrenzt
nPTrim_v	Integer	velocity	Aktueller Winkeltrimmwert

Der Ausgangswert $nOut_v$ berechnet sich wie folgt:

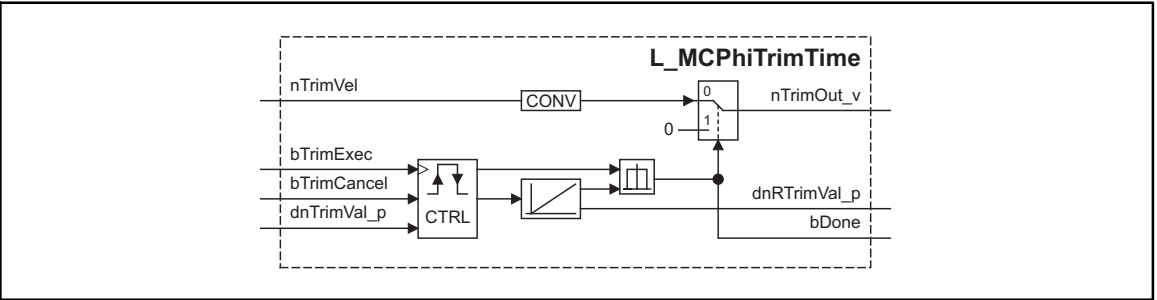
$$nOut_v = nIn_v + \frac{d}{dt} \cdot \left(\frac{nIn_v \cdot nNTrim_v}{16384} \right)$$

- Der Wert des Summanden $\frac{d}{dt} \cdot \left(\frac{nIn_v \cdot nNTrim_v}{16384} \right)$ wird auf ±32767 [inc/ms] begrenzt.
- Der Ausgangswert $nOut_v$ wird auf ±32767 [inc/ms] begrenzt.



2.5 L_MCPHiTrimTime - Winkeltrimmung zeitbasierend

Dieser FB führt eine relative Winkeltrimmung aus. Nach dem Start wird der am Eingang *dnTrimVal_p* eingestellte Trimmweg mit der Trimmgeschwindigkeit *nTrimVel* abgefahren.



⇒ Eingänge (Variablentyp: VAR_INPUT)							
Bezeichner	Datentyp	Signaltyp	Info/Einstellmöglichkeiten				
nTrimVel	Integer	velocity	Trimmgeschwindigkeit in [inc/s] Ein neuer Wert wird nur nach einer FALSE-TRUE- oder TRUE-FALSE-Flanke am Eingang <i>bTrimExec</i> übernommen.				
bTrimExec	Bool	binary	Trimmung starten <table><tr><td>FALSE ↗ TRUE</td><td>Trimmung starten</td></tr><tr><td>TRUE ↘ FALSE</td><td> Eine laufende Trimmung wird abgebrochen. </td></tr></table>	FALSE ↗ TRUE	Trimmung starten	TRUE ↘ FALSE	Eine laufende Trimmung wird abgebrochen.
FALSE ↗ TRUE	Trimmung starten						
TRUE ↘ FALSE	Eine laufende Trimmung wird abgebrochen.						
bTrimCancel	Bool	binary	Trimmung abbrechen <table><tr><td>TRUE</td><td>Trimmung abbrechen</td></tr></table>	TRUE	Trimmung abbrechen		
TRUE	Trimmung abbrechen						
dnTrimVal_p	Double Integer	position	Trimmweg in [inc] 65536 inc = 1 Motorumdrehung				

Ausgänge ⇒ (Variablentyp: VAR_OUTPUT)					
Bezeichner	Datentyp	Signaltyp	Info		
nTrimOut_v	Integer	velocity	Winkeltrimmwert in [inc/ms]		
dnRTrimVal_p	Double Integer	position	Noch zu verfahrens Rest-Trimmweg		
bDone	Bool	binary	Status Trimmung <table><tr><td>TRUE</td><td>Trimmung erfolgt</td></tr></table>	TRUE	Trimmung erfolgt
TRUE	Trimmung erfolgt				

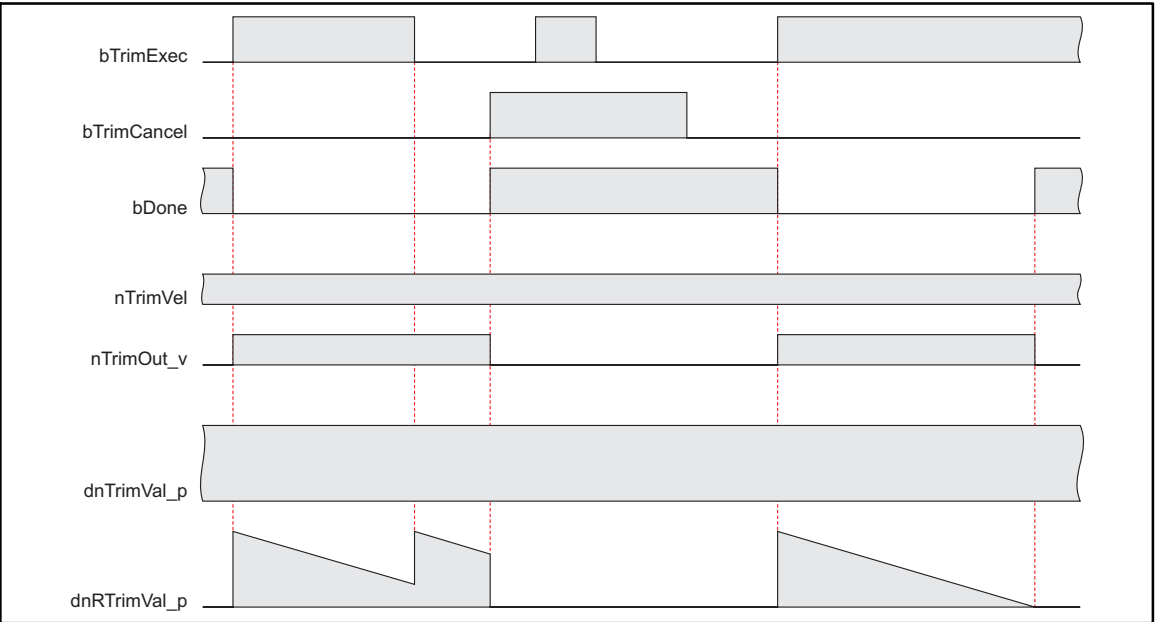
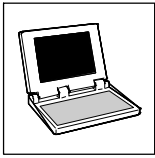


Abb. 2-5 Signalverlauf



Funktionsbibliothek LenzeMotionControl2.lib

Funktionsblöcke für Achsensynchronisierungen

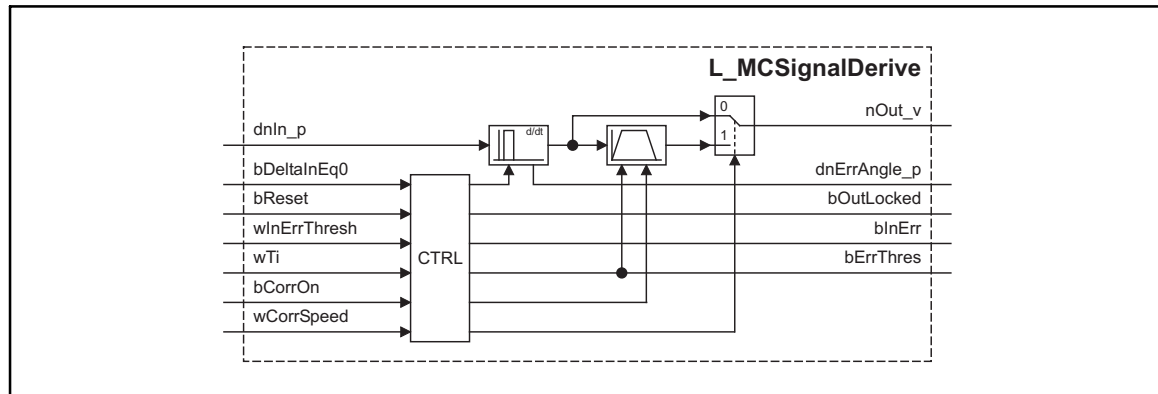
L_MCSignalDerive - Signaldifferentiation

2.6

L_MCSignalDerive - Signaldifferentiation

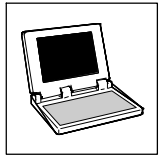
Dieser FB verarbeitet und überwacht absolute Winkelinformationen, die z. B. über ein Bussystem übertragen werden.

- Der absolute Winkel am Eingang *dnIn_p* wird differenziert und als Winkeldifferenz an *nOut_v* ausgegeben.
- Zur Überwachung des Empfangs wird ein binäres Signal (Drehzahl = 0) der Signalquelle benötigt.



⇒ Eingänge (Variablentyp: VAR_INPUT)				
Bezeichner	Datentyp	Signaltyp	Info/Einstellmöglichkeiten	
dnIn_p	Double Integer	position	Winkleingang	
bDeltaInEq0	Bool	binary	Konfiguration Überwachung	
			TRUE	n-Quelle = 0 (Überwachung AUS)
			FALSE	n-Quelle ≠ 0
bReset	Bool	binary	Fehler-Reset	
			TRUE	dnErrAngle_p = 0
			TRUE ➡ FALSE	bErrThres = FALSE (Fehler-Reset nach zweiten FB-Aufruf)
wInErrThresh	Word	-	Grenzwert Fehlerzyklen	
wTi	Word	-	Rampenzeit • 100 = 1 s (Bezug 15000 rpm) ⇒ Innerhalb 1 s erfolgt eine Änderung um 16384 inc/ms ≙ 15000 rpm • 0 = Rampe ausgeschaltet	
bCorrOn	Bool	binary	Winkelfehlerkorrektur	
			TRUE	Winkelfehlerkorrektur aktiv
wCorrSpeed	Word	-	Max. Korrekturgeschwindigkeit als relativer Offset in [inc/Aufrufzyklus] • Wert wird intern begrenzt auf 0 ... 16384	

Ausgänge → (Variablentyp: VAR_OUTPUT)				
Bezeichner	Datentyp	Signaltyp	Info	
nOut_v	Integer	velocity	Drehzahlausgang (Winkeldifferenz) • Wert wird intern begrenzt auf $\pm 16384 \equiv 15000$ rpm	
dnErrAngle_p	Double Integer	position	Aufintegrierter Winkelfehler	
bOutLocked	Bool	binary	Rampen-Status	
			FALSE	• Ein Hoch-/Ablauf der Rampe oder Korrekturvorgang ist aktiv. • Zu viele fehlerhafte Taskzyklen ($bErrThres = TRUE$).
			TRUE	sonst
			Fehler-Status	
bInErr	Bool	binary	TRUE Fehler (siehe folgende Tabelle)	
bErrThresh	Bool	binary	TRUE Fehler (Fehleranzahl $> wInErrThresh$) • Der aufintegrierte Winkelfehler $dnErrAngle_p$ wird auf 0 zurückgesetzt.	



Überwachungsfunktion

Die Überwachung erfolgt auf Basis des Winkelfortschritts von $dnIn_p$ (Drehzahl) und der Konfiguration des Eingangs $bDeltaInEq0$:

Winkelfortschritt ($dnIn_p_{t0} - dnIn_p_{t0-1} = 0$)	Konfiguration Überwachung ($bDeltaInEq0_b$)	Fehlerausgang ($bInErr_b$)
FALSE	beliebig	FALSE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

- Ist die Bedingung für $bInErr = TRUE$ erfüllt, wird der letzte gültige Ausgangswert $nOut_v$ gehalten (extrapoliert). Der zurückgelegte Winkel wird in $dnErrAngle_p$ aufintegriert. (Siehe Abb. 2-6, $bInErr$, Signal zum Zeitpunkt $t2$ und $t3$.)
- Überschreitet die Anzahl der fehlerhaften Taskzyklen ($bInErr = TRUE$) den Wert in $wInErrThresh$, wird zusätzlich der Ausgang $bErrThresh = TRUE$ gesetzt und $dnErrAngle_p$ wird auf "0" zurückgesetzt. (Siehe Abb. 2-6, Signal zum Zeitpunkt $t4$.)
 - Das Ausgangssignal $nOut_v$ wird mit der Rampe wTi auf 0 gefahren.
 - Bei einer Rampenzeit $wTi = 0$ s ist die Rampenbildung ausgeschaltet. Der Ausgangswert $nOut_v$ springt auf 0.
 - Mit einer TRUE-FALSE-Flanke wird der Fehler an $bReset$ zurückgesetzt, sofern $bInErr = FALSE$ ist. Es werden hierbei folgende zwei Fälle unterschieden:

Geschwindigkeitswert = 0 (zum Zeitpunkt des Reset)	Geschwindigkeitswert $\neq 0$ (zum Zeitpunkt des Reset)
Am Ausgang $nOut_v$ wird die Geschwindigkeit, die $dnIn_p$ entspricht, ohne Rampe ausgegeben. • Siehe Abb. 2-6, Signal zum Zeitpunkt $t6$.	$nOut_v$ wird mit der Rampenzeit wTi auf den Geschwindigkeitswert gefahren, der $dnIn_p$ entspricht. Wenn dieser erreicht ist, wird der Rampengenerator ausgeschaltet und die Geschwindigkeit direkt ausgegeben.

- Wenn innerhalb von $wInErrThresh$ Taskzyklen wieder fehlerfreie Signale an $dnIn_p$ und $bDeltaInEq0$ festgestellt werden ($bErr = FALSE$), wird wie folgt verfahren:
 - Berechnung des zurückzulegenden Sollwinkels aus dem Winkelwert $dnIn_p$ vor und nach der Unterbrechung ($X_{soll} = dnIn_p_{t0} - dnIn_p_{t0-Errorcycles}$).
 - Subtraktion des an $nOut_v$ ausgegebenen Winkels ($X_{diff} = X_{soll} - X_{out}$).
 - Übertragung der Differenz in $dnErrAngle_p$. Bei $bCorrOn = TRUE$ wird der Fehlerwinkel durch Aufschaltung des Drehzahloffset $wCorrSpeed$ auf den Ausgabewert $nOut_v$ abgearbeitet (siehe Abb. 2-6, Signal zum Zeitpunkt $t3$).
 - Der Ausgang $bInErr$ wird FALSE gesetzt.

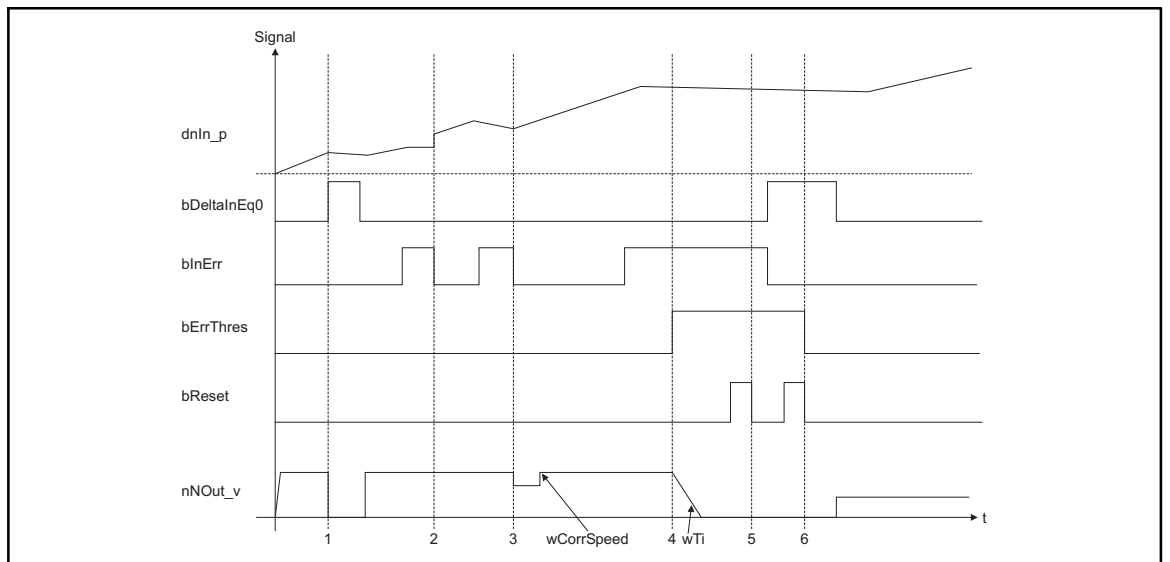
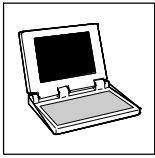


Abb. 2-6

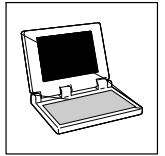
Signalverlauf



Funktionsbibliothek LenzeMotionControl2.lib

Funktionsblöcke für Achsensynchronisierungen

L_MCSignalDerive - Signaldifferentiation



3 Index

B

Begriffsdefinitionen, 1-2

D

Drehzahl-abhängige Winkeltrimmung, 2-6

I

Integration und Differenzbildung von Drehzahlsignalen (L_MCPhiDiff), 2-5

Drehzahl-abhängige Winkeltrimmung, 2-6

Integrator mit TP-Set Funktion (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

Sicherheitshinweise, Gestaltung

Sonstige Hinweise, 1-1

Warnung vor Sachschäden, 1-1

Signaldifferentiation (L_MCSignalDerive), 2-12

Signalinterpolation (L_MCInterpolator), 2-2

U

unit, 1-2

W

Winkeltrimmung drehzahlabhängig (L_MCPhiTrim), 2-10

Winkeltrimmung zeitbasierend (L_MCPhiTrimTime), 2-11