BECKHOFF New Automation Technology

Handbuch | DE

TS4100

TwinCAT 2 | PLC Controller Toolbox

Supplement | Control

28.10.2022 | Version: 1.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

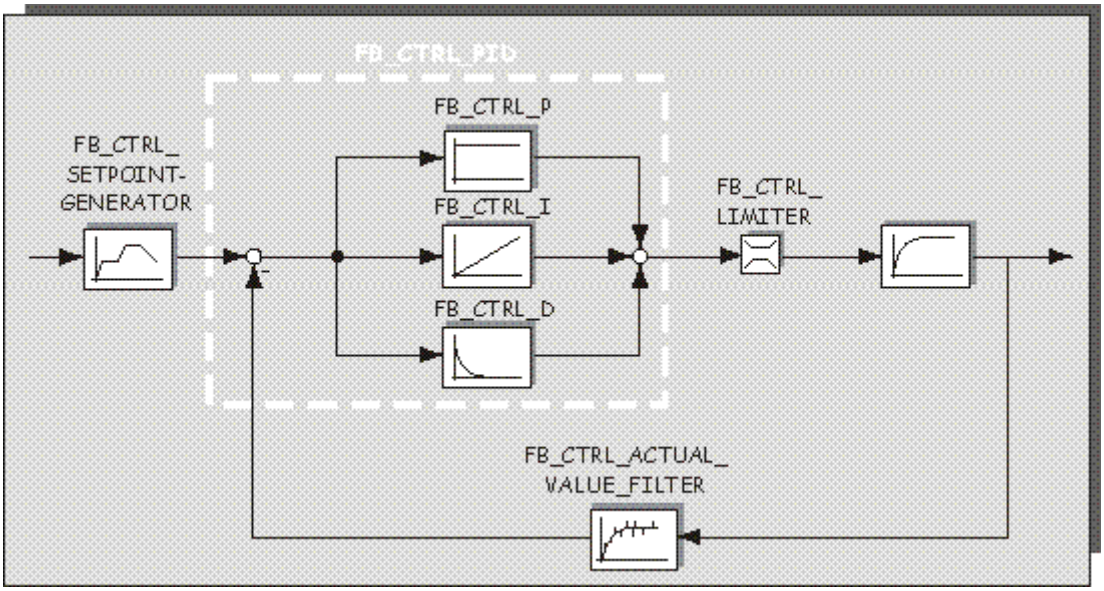
Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

In dieser Bibliothek sind Funktionsbausteine enthalten, die verschiedene regelungstechnische Übertragungsglieder in einem Wirkungsplan repräsentieren. Es sind für viele Anwendung einsetzbare komplexe Regler enthalten, aber auch Basisbausteine, mit denen für spezielle Anwendungen eigene Reglerstrukturen implementiert werden können.



Funktionsbausteine

Name	Beschreibung
FB_CTRL_2POINT [▶ 40]	2-Punkt-Regler
FB_CTRL_2POINT_PWM_ADAPTIVE [▶ 41]	Adaptiver 2-Punkt-Regler mit PWM-Ausgang
FB_CTRL_3PHASE_SETPOINT_GENERATOR [▶ 140]	3 Phasen Sollwertgenerator
FB_CTRL_3POINT [▶ 44]	3-Punkt-Regler
FB_CTRL_3POINT_EXT [▶ 46]	Erweiterter 3-Punkt-Regler
FB_CTRL_ACTUAL_VALUE_FILTER [▶ 78]	Istwertfilter
FB_CTRL_ARITHMETIC_MEAN [▶ 79]	Arithmetisches Mittelwertfilter
FB_CTRL_CHECK_IF_IN_BAND [▶ 113]	Bereichsüberwachung
FB_CTRL_D [▶ 25]	D-Glied
FB_CTRL_DEADBAND [▶ 120]	Totzone
FB_CTRL_DIGITAL_FILTER [▶ 81]	Digitales Filter
FB_CTRL_FLOW_TEMP_SETPOINT_GEN [▶ 147]	Vorgabe der Vorlauftemperatur in Abhängigkeit der Außentemperatur
FB_CTRL_GET_SYSTEM_TIME [▶ 19]	Ausgabe der Windows Systemzeit
FB_CTRL_GET_TASK_CYCLETIME [▶ 20]	Ermittlung der Task-Zykluszeit
FB_CTRL_HYSTERESIS [▶ 26]	Hysterese-Glied
FB_CTRL_I [▶ 28]	I-Glied
FB_CTRL_I_WITH_DRIFTCOMPENSATION [▶ 30]	I-Glied mit Driftkompensation
FB_CTRL_LEAD_LAG [▶ 85]	Lead-Lag-Glied
FB_CTRL_LIMITER [▶ 122]	Stellgrößenbegrenzung
FB_CTRL_LIN_INTERPOLATION [▶ 108]	Lineares Interpolationsglied
FB_CTRL_LOG_DATA [▶ 114]	Datenlogger im *.csv ASCII-Format

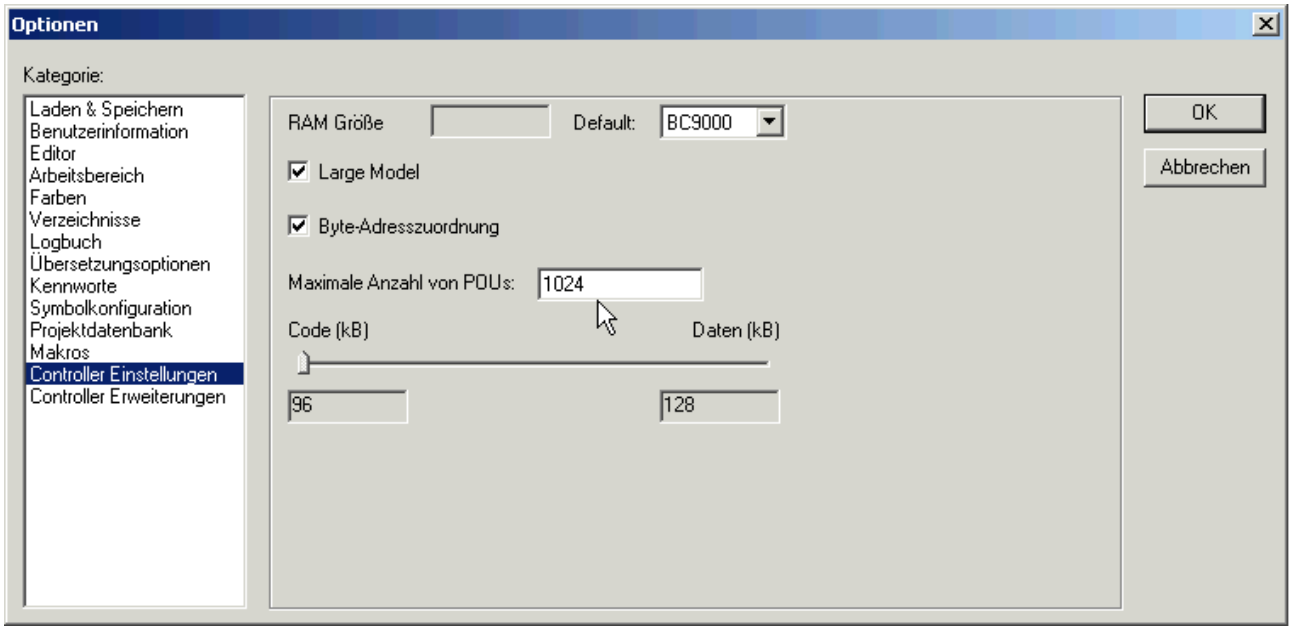
Name	Beschreibung
FB_CTRL_LOG_MAT_FILE [▶ 116]	Datenlogger im Matlab 5 - Format
FB_CTRL_LOOP_SCHEDULER [▶ 22]	Verteilung der Rechenleistung bei mehreren Regelkreisen
FB_CTRL_MOVING_AVERAGE [▶ 84]	Gleitendes Mittelwertfilter
FB_CTRL_MULTIPLE_PWM_OUT [▶ 123]	PWM-Glied mit mehreren Ausgängen
FB_CTRL_NORMALIZE [▶ 110]	Kennlinien - Linearisierung
FB_CTRL_NOISE_GENERATOR [▶ 88]	Rauschgenerator
FB_CTRL_NOTCH_FILTER [▶ 89]	Notch-Filter
FB_CTRL_nPOINT [▶ 48]	n-Punkt-Regler
FB_CTRL_P [▶ 32]	P-Glied
FB_CTRL_PARAMETER_SWITCH [▶ 50]	Parameter-Umschaltalgorithmus für einen Splitrange-Regler
FB_CTRL_PI [▶ 52]	PI-Regler
FB_CTRL_PI_PID [▶ 55]	Kaskadierter PI-PID-Regler
FB_CTRL_PID [▶ 58]	PID-Regler
FB_CTRL_PID_EXT [▶ 67]	Erweiterter PID-Regler
FB_CTRL_PID_EXT_SPLITRANGE [▶ 62]	Erweiterter PID-Regler mit Parametersatzumschaltung
FB_CTRL_PID_SPLITRANGE [▶ 72]	PID-Regler mit Parametersatzumschaltung
FB_CTRL_PT1 [▶ 91]	PT_1-Glied
FB_CTRL_PT2 [▶ 93]	PT_2-Glied
FB_CTRL_PT2oscillation [▶ 95]	Schwingungsfähiges PT_2-Glied
FB_CTRL_PT3 [▶ 97]	PT_3-Glied
FB_CTRL_PTn [▶ 99]	PT_n-Glied
FB_CTRL_PTt [▶ 101]	PT_t-Glied
FB_CTRL_PWM_OUT [▶ 128]	PWM-Glied
FB_CTRL_PWM_OUT_EXT [▶ 129]	Erweitertes PWM-Glied
FB_CTRL_RAMP_GENERATOR [▶ 149]	Rampengenerator
FB_CTRL_RAMP_GENERATOR_EXT [▶ 151]	Erweiterter Rampengenerator
FB_CTRL_SCALE [▶ 132]	Bereichsanpassung
FB_CTRL_SERVO_MOTOR_OUT [▶ 133]	Ansteuerung eines Stellantriebs
FB_CTRL_SERVO_MOTOR_SIMULATION [▶ 103]	Simulation eines Stellantriebs
FB_CTRL_SETPOINT_GENERATOR [▶ 153]	Sollwertgenerator
FB_CTRL_SIGNAL_GENERATOR [▶ 156]	Signalgenerator
FB_CTRL_SPLITRANGE [▶ 135]	Signalzerlegung in den positiven und negativen Anteil.
FB_CTRL_STEPPING_MOTOR_OUT [▶ 137]	Ansteuerung eines Schrittmotors
FB_CTRL_TRANSFERFUNCTION_1 [▶ 33]	Übertragungsfunktion nach der 1. Standardform
FB_CTRL_TRANSFERFUNCTION_2 [▶ 36]	Übertragungsfunktion nach der 2. Standardform
FB_CTRL_TuTg [▶ 104]	TuTg-Glied
FB_CTRL_ZERO_ZONE_DAMPING [▶ 106]	Nullpunkt-Dämpfung

3 Generelle Funktionsweise der FB_CTRL_... Funktionsbausteine

In den folgenden Absätzen wird die generelle Funktionsweise der Bausteine in der Controller Toolbox beschrieben.

Allgemeine Hinweise

Die TcControllerToolbox kann generell sowohl auf einem PC- als auch auf einem BX- oder BC-System eingesetzt werden. Wenn die Toolbox auf einem BC-System eingesetzt wird, ist es erforderlich, im TwinCAT PLC Control unter dem Menüpunkt Projekt, Optionen... die maximale Anzahl der POUs anzupassen.



Diskretisierung

Die kontinuierlichen Übertragungsfunktionen der in dieser Bibliothek zusammengestellten Übertragungsglieder sind mit der Trapezregel (Tustin-Formel) diskretisiert worden.

Tustin-Formel:

$$\frac{1}{s} = \frac{T}{2}\frac{z+1}{z-1}$$

Eingänge der Funktionsbausteine

eMode:

Mit diesem Eingang kann die Betriebsart der meisten Bausteine ausgewählt werden. Hiermit ist es möglich, eine der folgenden Betriebsarten anzuwählen:

eCTRL_MODE_PASSIVE	Der oder die Ausgänge des Bausteins werden auf Null gesetzt, die internen Zustände bleiben aber erhalten.
eCTRL_MODE_ACTIVE	Der Baustein wird entsprechend seiner Beschreibung ausgeführt und entsprechende Ausgangsgrößen werden berechnet (Normalbetrieb).
eCTRL_MODE_RESET	In dieser Betriebsart werden alle internen Zustände zurückgesetzt und das Fehlerbit gelöscht.
eCTRL_MODE_MANUAL	Es wird am Ausgang der Wert der Eingangsgröße **fManSyncValue** ausgegeben (Handbetrieb).

stParams:

Mit dieser Struktur werden dem Funktionsbaustein die benötigten Parameter übergeben. In allen Parameterstrukturen sind die Variablen **tTaskCycleTime** und **tCtrlCycleTime** enthalten. Die Wirkungsweise dieser Parameter ist die Folgende:

Mit dem Parameter **tTaskCycleTime** wird die Zykluszeit angegeben, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird. Wird er nur in jedem 2. Zyklus aufgerufen, muss die Zeit entsprechend verdoppelt werden. Der Parameter **tCtrlCycleTime** gibt die Abtastzeit des Regelkreises an. Diese Zeit muss größer oder gleich des Parameters tTaskCycleTime sein. Wenn die Abtastzeit gleich der tTaskCycleTime gesetzt wird, wird der Baustein bei jedem Aufruf ausgeführt. Wenn sie um den Faktor 5 größer gewählt wird, wird der Baustein nur bei jedem 5. Aufruf bearbeitet. Somit ist es möglich, in einer schnellen Task auch langsame Regelkreise zu implementieren.

Die Parameter **tTaskCycleTime** und **tCtrlCycleTime** sind vom Typ TIME und erlauben somit keine Eingaben, die kleiner als 1ms sind. Um die Regler in einer schnellen SPS-Task mit einer Zykluszeit kleiner 1ms einsetzen zu können, kann eine globale Base-Time angegeben werden, auf die die angegeben Zykluszeiten bezogen werden. Erläuterungen zur Verwendung der Base-Time [▶ 12].

Beispiele:

Es wird angenommen, dass der Baustein in jedem Task-Zyklus aufgerufen wird.

Task Configuration	Parameter: tTaskCycleTime	Parameter: tCtrlCycleTime	Wirkungsweise:
T#10ms	T#10ms	T#10ms	Der Regelkreis wird mit 10ms Abtastzeit bearbeitet.
T#10ms	T#10ms	T#50ms	Der Regelkreis wird mit 50ms Abtastzeit bearbeitet.
T#100ms	T#100ms	T#100ms	Der Regelkreis wird mit 100ms Abtastzeit bearbeitet.
T#100ms	T#100ms	T#50ms	**FEHLER**, eine Ausführung nicht möglich!
T#100ms	T#50ms	T#50ms	**FEHLER**, der Baustein wird zwar ausgeführt, aber es werden falsche Ausgangsgrößen berechnet!

Ausgänge der Funktionsbausteine

eState:

Dieser Ausgang zeigt den aktuellen internen Zustand an, in dem sich der Baustein befindet.

eCTRL_STATE_IDLE	Ein Reset des Bausteins ist erfolgreich ausgeführt worden, der Baustein wartet auf eine Betriebsartenumschaltung.
eCTRL_STATE_PASSIVE	Der Baustein befindet sich im Passive-State, in dem keine Berechnungen ausgeführt werden.
eCTRL_STATE_ACTIVE	Der Baustein befindet sich in Active-State, dieses ist der normale Betriebszustand.
eCTRL_STATE_RESET	Eine Reset-Anforderung wird bearbeitet und der Reset ist noch nicht abgeschlossen.
eCTRL_STATE_MANUAL	Der Baustein befindet sich im Manual-State und die Ausgangsgröße kann manuell an dem entsprechenden Eingang vorgegeben werden.
eCTRL_STATE_...	Eventuell weitere interne Zustände werden bei den entsprechenden Bausteinen beschrieben.
eCTRL_STATE_ERROR	Es ist ein Fehler aufgetreten und der Baustein wird in diesem State nicht ausgeführt. Siehe **eErrorId** für weitere Informationen.

bError:

An diesem boolschen Ausgang wird mit einem TRUE ein Fehler des Bausteins signalisiert.

eErrorId:

An diesem Ausgang wird die Fehlernummer [▶ 16] ausgegeben, wenn der Ausgang bError TRUE ist.

Verwendung der globalen Base-Time (nur auf einem PC-System)

Um die Funktionsbausteine in einer SPS-Task mit einer Zykluszeit kleiner 1ms einsetzen zu können, ist es möglich, die angegebenen Zykluszeiten als Ticks einer Basiszeit zu interpretieren. Bei dieser besonderen Parametrierung wird die Zeiteinheit 1ms als 1 Tick interpretiert. Dieses Vorgehen ist äquivalent zu der Einstellung einer SPS-Zykluszeit kleiner 1ms im TwinCAT Systemmanager.

Die Umschaltung und Angabe der Basiszeit erfolgt mit der **globalen** Struktur **stCtrl_GLOBAL_CycleTimeInterpretation** für alle Funktionsbausteine der Toolbox.

```
VAR_GLOBAL
    stCtrl_GLOBAL_CycleTimeInterpretation : ST_CTRL_CYCLE_TIME_INTERPRETATION;
END_VAR
```

```
TYPE ST_CTRL_CYCLE_TIME_INTERPRETATION :
STRUCT
    bInterpretCycleTimeAsTicks : BOOL;  (* e.g. 2ms -> 2ticks
*)
    fBaseTime                  : FLOAT; (* Base time in seconds,
e.g. 200µs -> 200E-6s *)
END_STRUCT
END_TYPE
```

Um die angegebenen Zykluszeiten als Ticks zu interpretieren, wird die Variable **bInterpretCycleTimeAsTicks** in der globalen Struktur **stCtrl_GLOBAL_CycleTimeInterpretation** auf TRUE gesetzt. In dieser Struktur muss dann auch die Basiszeiteinheit in der Variablen **fBaseTime** gesetzt werden.

Durch das Setzen des Flags **bInterpretCycleTimeAsTicks** wird die Interpretation der Parameter mit den Namen

- tTaskCycleTime
- tCtrlCycleTime

geändert. Alle sonstigen Parameter vom Typ TIME bleiben in ihrer Interpretation und Wirkung unbeeinflusst.

Beispiel:

Die Basiszeiteinheit des TwinCAT-Systems beträgt 200µs. Die SPS-Task und somit die Bausteine der Toolbox werden zyklisch alle 400µs aufgerufen.

Setzen der globalen Struktur:

```
stCtrl_GLOBAL_CycleTimeInterpretation.bInterpretCycleTimeAsTicks := TRUE;
stCtrl_GLOBAL_CycleTimeInterpretation.fBaseTime          := 200E-6;
```

Parametrierung eines Funktionsbausteins aus der Toolbox:

```
stParams.tTaskCycleTime    := T#2ms;   (*2*200µs=400µs *)
stParams.tCtrlCycleTime    := T#4ms;   (*4*200µs=800µs *)
stParams. ...
```

Die an den Bausteinen angegebene TaskCycleTime beträgt 2•200E-6s = 400µs und entspricht somit der eingestellten SPS-Zykluszeit. Die CtrlCycleTime wird auf 800µs = 4•200E-6s gesetzt, so dass der Regelkreis mit einer Abtastzeit von 800µs arbeitet, also in jedem 2. SPS-Zyklus bearbeitet wird.

4 Einstellregeln für die P-, PI- und PID-Regler

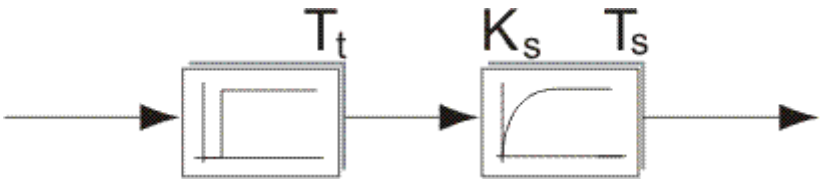
Auf dieser Seite sind einige, aus der regelungstechnischen Literatur bekannten Einstellregeln zusammengestellt. Welche dieser Einstellregeln im speziellen Fall am besten zur Anwendung kommen sollten, muss in Abhängigkeit der vorliegenden Regelstrecke entschieden werden.

Ziegler und Nichols

Ziegler und Nichols

Die Einstellregeln von Ziegler und Nichols können angewendet werden, wenn sich die Reglungstrecke durch ein Totzeitelement und ein Verzögerungselement 1. Ordnung approximieren lässt.

$$G_s(s) = K_s \cdot \frac{e^{-T_t s}}{1 + s \cdot T_s}$$



T_t = Totzeit der Regelstrecke

K_s = Verstärkungsfaktor der Regelstrecke

T_s = Zeitkonstante der Regelstrecke

Regler	K_r	T_n	T_v
P-Regler	$\frac{T_s}{K_s \cdot T_t}$	-	-
PI-Regler	$0.9 \cdot \frac{T_s}{K_s \cdot T_t}$	$3.33 \cdot T_t$	-
PID-Regler	$1.2 \cdot \frac{T_s}{K_s \cdot T_t}$	$2.0 \cdot T_t$	$0.5 \cdot T_t$

Chien, Hrones und Reswick

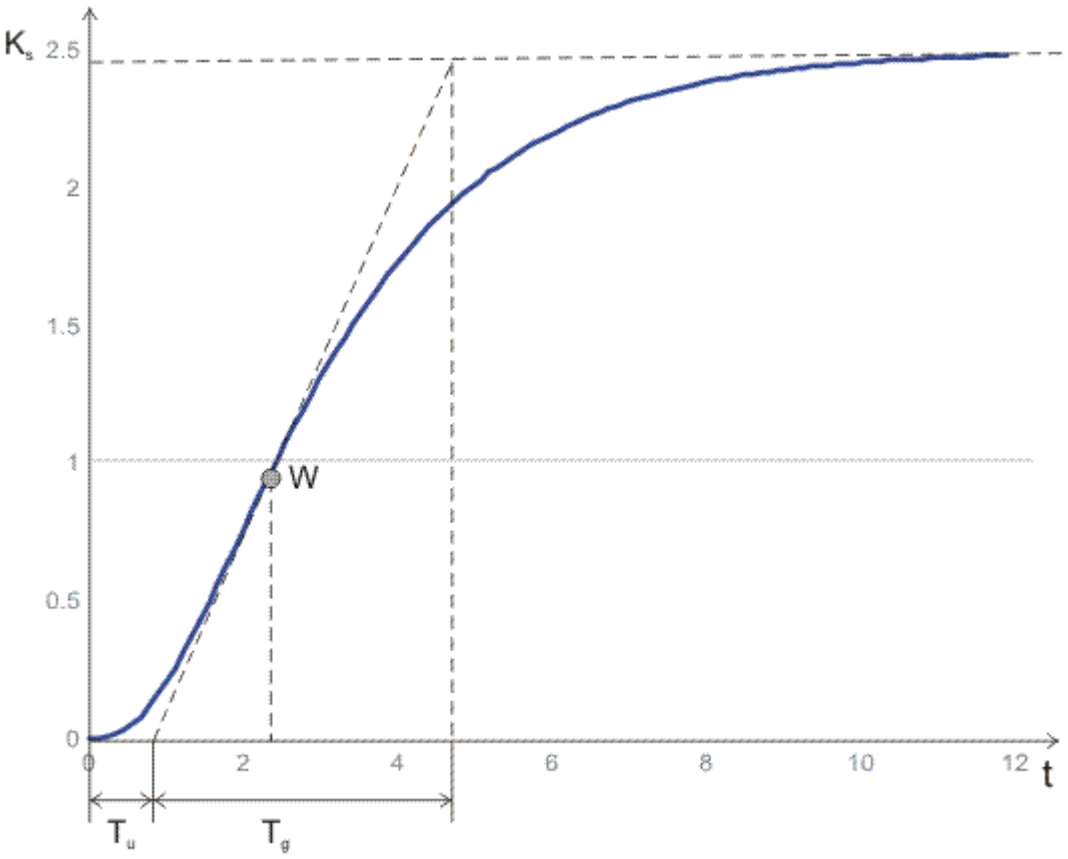
Um dieses Verfahren einsetzen zu können, muss die Sprungantwort der Strecke ein Verzögerungsverhalten zeigen und überschwingungsfrei sein.

Aus der Sprungantwort werden die Verzugszeit T_u, die Ausgleichszeit T_g und die Streckenverstärkung K_s ermittelt.

Die Streckenverstärkung wird aus dem folgenden Quotienten berechnet:

$$K_s = \frac{\text{Step Response Height}}{\text{Controller Output Value}}$$

Die unten aufgeführten Einstellregeln können nur angewendet werden, wenn Tg/Tu > 3 ist!



Optimierung des Störverhaltens:

Regler		Aperiodischer Regelverlauf	20% Überschwingen
P-Regler	K_r	$0.30 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.70 \cdot \frac{T_g}{T_u \cdot K_s}$
PI-Regler	K_r	$0.60 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.70 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$4.00 \cdot T_u$	$2.3 \cdot T_u$
PID-Regler	K_r	$0.95 \cdot \frac{T_g}{T_u \cdot K_s}$	$1.20 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$2.40 \cdot T_u$	$2.00 \cdot T_u$
	T_V	$0.42 \cdot T_u$	$0.42 \cdot T_u$

Optimierung des Führungsverhaltens:

Regler		Aperiodischer Regelverlauf	20% Überschwingen
P-Regler	K_r	$0.30 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.70 \cdot \frac{T_g}{T_u \cdot K_s}$
PI-Regler	K_r	$0.35 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.60 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$1.20 \cdot T_g$	$1.0 \cdot T_g$
PID-Regler	K_r	$0.60 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.95 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$1.00 \cdot T_g$	$1.35 \cdot T_g$
	T_V	$0.50 \cdot T_u$	$0.47 \cdot T_u$

5 PLC-API

5.1 Definition der Strukturen und Enums

In diesem Anhang werden alle in der Controller Toolbox verwendeten Strukturen und Enums beschrieben.

FLOAT:

Die Bibliothek ist so aufgebaut, dass diese sowohl auf einem PC- als auch auf einem BX-, BC-System lauffähig ist. Um diese Portabilität zu erreichen, wird in den Funktionsbausteinen der Bibliothek nur mit dem Datentyp FLOAT gearbeitet. In einer zusätzlichen Bibliothek wird dieser Datentyp als LREAL oder REAL definiert.

Auf einem **PC-System** wird automatisch die zusätzliche Bibliothek "**TcFloatPC.lib**" eingebunden.

```
TYPE
     FLOAT : LREAL;
END_TYPE
```

Auf einem **BC-System** wird automatisch die zusätzliche Bibliothek "**TcFloatBC.lb6**" eingebunden.

```
TYPE
     FLOAT : REAL;
END_TYPE
```

Auf einem **BX-System** wird automatisch die zusätzliche Bibliothek "**TcFloatBX.lbx**" eingebunden.

```
TYPE
     FLOAT : REAL;
END_TYPE
```

E_CTRL_MODE:

```
TYPE E_CTRL_MODE :
(
    eCTRL_MODE_IDLE        := 0, (* mode idle *)
    eCTRL_MODE_PASSIVE     := 1, (* mode passive *)
    eCTRL_MODE_ACTIVE      := 2, (* mode active *)
    eCTRL_MODE_RESET       := 3, (* mode reset *)
    eCTRL_MODE_MANUAL      := 4, (* mode manual *)
    eCTRL_MODE_TUNE        := 5, (* mode tuning *)
    eCTRL_MODE_SELFTEST    := 6, (* mode selftest *)
    eCTRL_MODE_SYNC_MOVEMENT   := 7  (* mode synchronize
*)
)
END_TYPE
```

E_CTRL_STATE :

```
TYPE E_CTRL_STATE :
(
    eCTRL_STATE_IDLE        := 0, (* state idle *)
    eCTRL_STATE_PASSIVE     := 1, (* state passive *)
    eCTRL_STATE_ACTIVE      := 2, (* state active *)
    eCTRL_STATE_RESET           := 3, (* state reset *)
    eCTRL_STATE_MANUAL      := 4, (* state manual *)
    eCTRL_STATE_TUNING      := 5, (* state tuning *)
    eCTRL_STATE_TUNED           := 6, (* state tuning ready - tuned *)
    eCTRL_STATE_SELFTEST    := 7, (* state selftest *)
    eCTRL_STATE_ERROR           := 8, (* state error *)
    eCTRL_STATE_SYNC_MOVEMENT       := 9  (* state synchronizing movement *)
);
END_TYPE
```

E_CTRL_ERRORCODES:

```
TYPE E_CTRL_ERRORCODES :
(
    eCTRL_ERROR_NOERROR                     := 0, (* no error *)
    eCTRL_ERROR_INVALIDTASKCYCLETIME                := 1, (* invalid task cycle time *)
    eCTRL_ERROR_INVALIDCTRLCYCLETIME                := 2, (* invalid ctrl cycle time *)
    eCTRL_ERROR_INVALIDPARAM                    := 3, (* invalid parameter *)
```

```
    eCTRL_ERROR_INVALIDPARAM_Tv                 := 4, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td                 := 5, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn                 := 6, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Ti                 := 7, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fHystereisisRange              := 8, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fPosOutOn_Off          := 9, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fNegOutOn_Off          := 10, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_TableDescription               := 11, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_TableData                  := 12, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DataTableADR           := 13, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T0                     := 14, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T1                     := 15, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T2                 := 16, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T3                 := 17, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Theta              := 18, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nOrder                 := 19, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tt                 := 20, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tu                 := 21, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tg                 := 22, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_infinite_slope         := 23, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMaxIsLessThanfMin             := 24, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutMaxLimitIsLessThanfOutMinLimit     := 25, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOuterWindow           := 26, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fInnerWindow           := 27, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOuterWindowIsLessThanfInnerWindow     := 28, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeadBandInput             := 29, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeadBandOutput            := 30, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_PWM_Cycletime          := 31, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_no_Parameterset            := 32, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutOn                 := 33, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutOff                := 34, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fGain              := 35, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOffset                := 36, (* invalid parameter *)
    eCTRL_ERROR_MODE_NOT_SUPPORTED              := 37, (* invalid mode: mode not supported *)
    eCTRL_ERROR_INVALIDPARAM_Tv_heating                 := 38, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td_heating                 := 39, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn_heating                 := 40, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tv_cooling                 := 41, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td_cooling                 := 42, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn_cooling                 := 43, (* invalid parameter *)
    eCTRL_ERROR_RANGE_NOT_SUPPORTED             := 44, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nParameterChangeCycleTicks         := 45, (* invalid parameter *)
    eCTRL_ERROR_ParameterEstimationFailed           := 46, (* invalid parameter *)
    eCTRL_ERROR_NoiseLevelToHigh                := 47, (* invalid parameter *)
    eCTRL_ERROR_INTERNAL_ERROR_0                := 48, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_1                := 49, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_2                := 50, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_3                := 51, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_4                := 52, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_5                := 53, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_6                := 54, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_7                := 55, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_8                := 56, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_9                := 57, (* internal error *)
    eCTRL_ERROR_INVALIDPARAM_WorkArrayADR           := 58, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tOnTiime                   := 59, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tOffTiime                  := 60, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nMaxMovingPulses               := 61, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nAdditionalPulsesAtLimits          := 62, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fCtrlOutMax_Min            := 63, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeltaMax                  := 64, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tMovingTime            := 65, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tDeadTime                  := 66, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tAdditionalMoveTimeAtLimits        := 67, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fThreshold                 := 68, (* invalid parameter *)
    eCTRL_ERROR_MEMCPY                      := 69, (* MEMCPY failed *)
    eCTRL_ERROR_MEMSET                      := 70, (* MEMSET failed *)
    eCTRL_ERROR_INVALIDPARAM_nNumberOfColumns               := 71, (* invalid parameter *)
    eCTRL_ERROR_FileClose                   := 72, (* File Close failed *)
    eCTRL_ERROR_FileOpen                    := 73, (* File Open failed *)
    eCTRL_ERROR_FileWrite                   := 74, (* File Write failed *)
    eCTRL_ERROR_INVALIDPARAM_fVeloNeg                   := 75, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fVeloPos                   := 76, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DeadBandInput          := 77, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DeadBandOutput         := 78, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_CycleDuration          := 79, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tStart                 := 80, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_StepHeigthTuningToLow      := 81, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMinLimitIsLessThanZero        := 82, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMaxLimitIsGreaterThan100          := 83, (* invalid parameter *)
```

```
    eCTRL_ERROR_INVALIDPARAM_fStepSize                  := 84, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOkRangeIsLessOrEqualZero          := 85, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fForceRangeIsLessOrEqualfOkRange       := 86, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPWMPeriod                := 87, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tMinimumPulseTime              := 88, (* invalid parameter *)
    eCTRL_ERROR_FileDelete                  := 89, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumberOfPwmOutputs            := 90, (* File Delete failed *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInputArray_SIZEOF      := 91, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmOutputArray_SIZEOF         := 92, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmWaitTimesConfig_SIZEOF         := 93, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInternalData_SIZEOF        := 94, (* invalid parameter *)
    eCTRL_ERROR_SIZEOF                  := 95, (* SIZEOF failed *)
    eCTRL_ERROR_INVALIDPARAM_nOrderOfTheTransferfunction        := 96, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumeratorArray_SIZEOF         := 97, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDenominatorArray_SIZEOF           := 98, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_a_n_IsZero                 := 99, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_WorkArraySIZEOF            := 100, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_MOVINGRANGE_MIN_MAX            := 101, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_MOVINGTIME                 := 102, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DEADTIME                   := 103, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMinLimitIsGreaterThanfMaxLimit    := 104, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DataTableSIZEOF            := 105, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_OutputVectorDescription        := 106, (* invalid parameter *)
    eCTRL_ERROR_TaskCycleTimeChanged                := 107, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nMinMovingPulses               := 108, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fAcceleration          := 109, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeceleration          := 110, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_StartAndTargetPos              := 111, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fVelocity                  := 112, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fTargetPos                 := 113, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStartPos                  := 114, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMovingLength          := 115, (* invalid parameter *)
    eCTRL_ERROR_NT_GetTime                  := 116, (* internal error NT_GetTime *)
    eCTRL_ERROR_INVALIDPARAM_No3PhaseSolutionPossible           := 117, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStartVelo                 := 118, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fTargetVelo            := 119, (* invalid parameter *)
    eCTRL_ERROR_INVALID_NEW_PARAMETER_TYPE          := 120  (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fBaseTime                  := 121, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nOrderOfTheTransferfunction_SIZEOF     := 122, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nFilterOrder           := 124, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_a_SIZEOF        := 125, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_b_SIZEOF        := 126, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDigitalFiterData_SIZEOF           := 127, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nLogBuffer_SIZEOF              := 128, (* invalid parameter *)
    eCTRL_ERROR_LogBufferOverflow               := 129, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nLogBuffer_ADR             := 130, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_a_ADR           := 131, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_b_ADR           := 132, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInputArray_ADR             := 133, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmOutputArray_ADR            := 134, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmWaitTimesConfig_ADR        := 135, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInternalData_ADR       := 136, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDigitalFiterData_ADR      := 137, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumeratorArray_ADR            := 138, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDenominatorArray_ADR      := 139, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nTransferfunction1Data_ADR         := 140, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nTransferfunction2Data_ADR         := 141, (* invalid parameter *)
    eCTRL_ERROR_FileSeek                    := 142, (* internal error FB_FileSeek *)
    eCTRL_ERROR_INVALIDPARAM_AmbientTempMaxIsLessThanAmbientTempMin := 143, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_ForerunTempMaxIsLessThanForerunTempMin := 144, (* invalid parameter *)
    eCTRL_ERROR_INVALIDLOGCYCLETIME                 := 145, (* invalid parameter *)
    eCTRL_ERROR_INVALIDVERSION_TcControllerToolbox      := 146,
    eCTRL_ERROR_INVALIDPARAM_Bandwidth                  := 147, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_NotchFrequency             := 148, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DampingCoefficient             := 149, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fKpIsLessThanZero              := 150  (* invalid parameter *)
);
END_TYPE
```

E_CTRL_SIGNAL_TYPE:

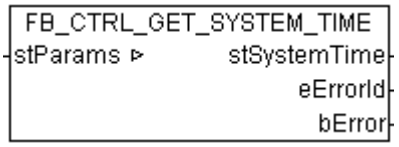
E_CTRL_SIGNAL_TYPE:

```
TYPE E_CTRL_SIGNAL_TYPE :
    (
    eCTRL_TRIANGLE  := 0,
    eCTRL_SINUS     := 1,
```

```
    eCTRL_SAWTOOTH  := 2
    );
END_TYPE
```

5.2 Auxiliary

5.2.1 FB_CTRL_GET_SYSTEM_TIME (nur auf einem PC-System)



Der Funktionsbaustein liest die aktuelle Windows-Systemzeit und stellt sie in dem SystemTimeStruct zur Verfügung.

Beschreibung:

Dieser Funktionsbaustein stellt in der Ausgangsstruktur die aktuelle Systemzeit zur Verfügung. Die Auflösung wird mit dem Parameter tCtrlCycleTime bestimmt, wobei die maximale Auflösung 10 ms beträgt und die Bedingung tCtrlCycleTime > 2 • tTaskCycleTime eingehalten werden muss. Anderenfalls reduziert sich die Auflösung auf 2 • tCtrlCycleTime.

VAR_OUTPUT

```
VAR_OUTPUT
    stSystemTime    : TIMESTRUCT;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

```
TYPE TIMESTRUCT
STRUCT
    wYear       : WORD;
    wMonth      : WORD;
    wDayOfWeek    : WORD;
    wDay    : WORD;
    wHour       : WORD;
    wMinute       : WORD;
    wSecond       : WORD;
    wMilliseconds : WORD;
END_STRUCT
END_TYPE
```

stSystemTime : Struktur, in der die Systemzeit ausgegeben wird.

wYear : Das Jahr: 1970 ~ 2106;

wMonth : Der Monat: 1 ~ 12 (Januar = 1, Februar = 2 usw.);

wDayOfWeek : Der Wochentag: 0 ~ 6 (Sonntag = 0, Montag = 1 usw.);

wDay : Tag des Monats: 1 ~ 31;

wHour : Stunde: 0 ~ 23;

wMinute : Minute: 0 ~ 59;

wSecond : Sekunde: 0 ~ 59;

wMilliseconds : Millisekunde: 0 ~ 999;

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_GET_SYSTEM_TIME;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_GET_SYSTEM_TIME:
STRUCT
    tTaskCycleTime   : TIME;  (* task cycle time [TIME]
*)
    tCtrlCycleTime   : TIME;  (* controller cycle time [TIME]
*)
END_STRUCT
END_TYPE
```

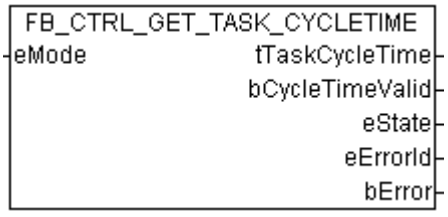
tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Task-Zyklus aufgerufen wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

5.2.2 FB_CTRL_GET_TASK_CYCLETIME (nur auf einem PC-System)



Mit diesem Funktionsbaustein kann die Task-Zykluszeit eines Programms mit einer Auflösung von 1 ms bestimmt werden.

Die TaskCycleTime kann nur dann richtig bestimmt werden, wenn kein Breakpoint im Programm aktiv ist.

Die Task-Zykluszeit wird nur einmal bestimmt. Wenn einer der Ausgänge bCycleTimeValid oder bError TRUE ist, werden keine weiteren Messungen mehr durchgeführt.

Wenn Zykluszeiten verwendet werden, die kleiner 1 ms sind oder die kein Vielfaches von 1 ms sind, sollte dieser Baustein nicht verwendet werden.

VAR_INPUT

```
VAR_INPUT
    eMode    : E_CTRL_MODE;
END_VAR
```

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    tTaskCycleTime  : TIME;             (* resolution: 1ms *)
    bCycleTimeValid     : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

tTaskCycleTime : Dieser Ausgang gibt die aktuelle Task-Zykluszeit mit einer Auflösung von 1ms an.

bCycleTimeValid : Wenn dieser Ausgang TRUE ist, ist die am Ausgang tTaskCycleTime angegebene Zeit gültig.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

Beispiel:

```
PROGRAM PRG_GET_TASK_CYCLETIME_TEST
VAR
    tTaskCycleTime          : TIME;
    bCycleTimeValid         : BOOL;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbCTRL_GET_TASK_CYCLETIME   : FB_CTRL_GET_TASK_CYCLETIME;
    (* control loop *)
    bInit           : BOOL := TRUE;
    fSetpointValue      : FLOAT := 45.0;
    fActualValue        : FLOAT;
    fbCTRL_PI       : FB_CTRL_PI;
    stCTRL_PI_Params    : ST_CTRL_PI_PARAMS;
    fbCTRL_PT1      : FB_CTRL_PT1;
    stCTRL_PT1_Params : ST_CTRL_PT1_PARAMS;
END_VAR
```

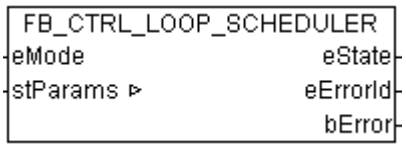
```
(* call fb to get the task cycle time *)
fbCTRL_GET_TASK_CYCLETIME( eMode       :=
eCTRL_MODE_ACTIVE,
                tTaskCycleTime  =>
tTaskCycleTime,
                bCycleTimeValid =>
bCycleTimeValid,
                eState      =>
eCTRL_MODE_ACTIVE,
                eErrorId    => eErrorId,
                bError      => bError
               );
(* call control loop if the cycle time is valid *)
IF fbCTRL_GET_TASK_CYCLETIME.bCycleTimeValid
THEN
    IF bInit
    THEN
    stCTRL_PT1_Params.tTaskCycleTime :=
fbCTRL_GET_TASK_CYCLETIME.tTaskCycleTime;
    stCTRL_PT1_Params.tCtrlCycleTime := T#100ms;
    stCTRL_PT1_Params.fKp        := 1.0;
    stCTRL_PT1_Params.tT1        := T#10s;
    stCTRL_PI_Params.tTaskCycleTime  :=
fbCTRL_GET_TASK_CYCLETIME.tTaskCycleTime;
    stCTRL_PI_Params.tCtrlCycleTime  := T#100ms;
    stCTRL_PI_Params.fKp         := 0.5;
    stCTRL_PI_Params.tTn         := T#5s;
    stCTRL_PI_Params.fOutMaxLimit    := 100.0;
    stCTRL_PI_Params.fOutMinLimit    := 0.0;
    bInit := FALSE;
    END_IF
    (* call controller *)
    fbCTRL_PI( fActualValue   := fbCTRL_PT1.fOut,
           fSetpointValue := fSetpointValue,
           eMode      := eCTRL_MODE_ACTIVE,
           stParams       := stCTRL_PI_Params
          );
```

```
    (* call PT1 *)
    fbCTRL_PT1( fIn      := fbCTRL_PI.fOut,
        eMode    := eCTRL_MODE_ACTIVE,
        stParams := stCTRL_PT1_Params,
        fOut     => fActualValue
          );
END_IF
```

Voraussetzungen

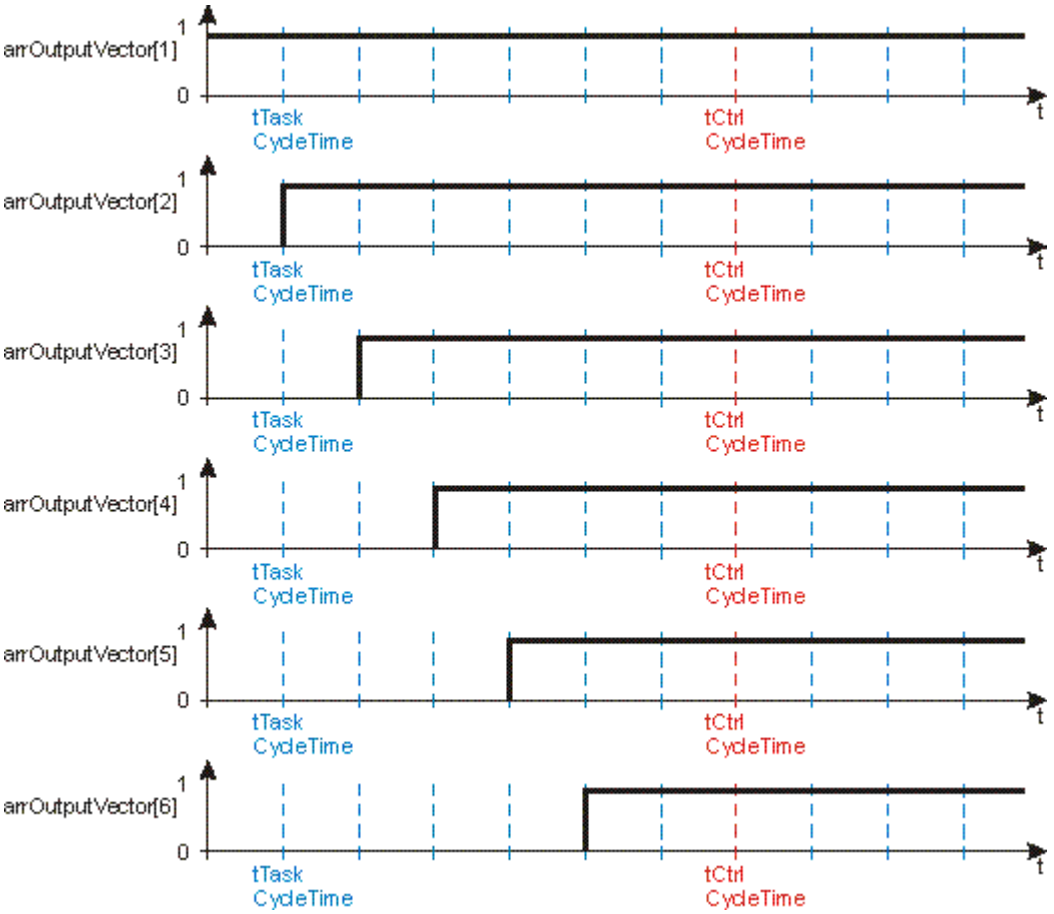
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

5.2.3 FB_CTRL_LOOP_SCHEDULER



Mit diesem Funktionsbaustein kann die Systemlast von mehreren Regelkreisen, welche mit der gleichen tCtrlCycleTime parametriert sind und für die die Bedingung tCtrlCycleTime > tTaskCycleTime gilt, verteilt werden. Mit dem Ausgangsvektor, der von diesem Baustein berechnet wird, werden die einzelnen Regelkreise zeitlich versetzt gestartet, so dass sich eine Verteilung der Systemlast ergibt.

Verhalten des Ausgangsvektors:



In diesem Bild werden 6 Regelkreise verwaltet. Bei diesen gilt tCtrlCycleTime = 7 · tTaskCycleTime.

Um den Funktionsbaustein nutzen zu können, muss vom Programmierer in der SPS das folgende Array angelegt werden:

```
arrOutputVector     :
ARRAY[1..nNumberOfControlLoops] OF BOOL;
```

Die Bits in diesem Vektor werden von dem Funktionsbaustein auf TRUE oder FALSE gesetzt. Die Regelkreise, die mit dem Loop-Scheduler verwaltet werden, sollten dann in den eCTRL_MODE_ACTIVE geschaltet werden, wenn das entsprechende Bit im Ausgangsvektor TRUE ist. Siehe Beispielcode unten.

VAR_INPUT

```
VAR_INPUT
    nManValue : DWORD;
    eMode     : E_CTRL_MODE;
END_VAR
```

nManValue : Mit diesem Eingang können im eCTRL_MODE_MANUAL die ersten 32 Bit in dem Ausgangsvektor gesetzt werden. Eine 1 setzt das erste Bit, eine 2 das zweite Bit, eine 3 das erste und zweite Bit, ...

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOOP_SCHEDULER_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Loop-Schedulers. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_LOOP_SCHEDULER_PARAMS:
STRUCT
    tCtrlCycleTime        : TIME     := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime        : TIME     := T#0ms; (* task
cycle time [TIME] *)
    nNumberOfControlLoops     : UINT;
    pOutputVector_ADR     : POINTER TO BOOL := 0;
    nOutputVector_SIZEOF      : UINT := 0;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der die Regelkreise bearbeitet werden, die mit dem Loop-Scheduler verwaltet werden. Diese muss größer oder gleich der TaskCycleTime sein.

tTaskCycleTime : Zykluszeit, mit der der Loop-Scheduler und die Funktionsbausteine der Regelkreise aufgerufen werden. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

nNumberOfControlLoops : Anzahl der Regelkreise die verwaltet werden.

pOutputVector_ADR :.Adresse des Ausgangsvektors.

nOutputVector_SIZEOF :Größe des Ausgangsvektors in Byte.

Beispiel:

```
PROGRAM PRG_LoopScheduler
VAR
    arrOutputVector    : ARRAY[1..5] OF BOOL;
```

```
    eMode           : E_CTRL_MODE;
    stParams            : ST_CTRL_LOOP_SCHEDULER_PARAMS;
    eErrorId            : E_CTRL_ERRORCODES;
    bError          : BOOL;
    fbCTRL_LoopScheduler    : FB_CTRL_LOOP_SCHEDULER;
    bInit           : BOOL := TRUE;
    (* modes of the control loops *)
    eMode_CtrlLoop_1        : E_CTRL_MODE;
    eMode_CtrlLoop_2        : E_CTRL_MODE;
    eMode_CtrlLoop_3        : E_CTRL_MODE;
    eMode_CtrlLoop_4        : E_CTRL_MODE;
    eMode_CtrlLoop_5        : E_CTRL_MODE;
END_VAR
```

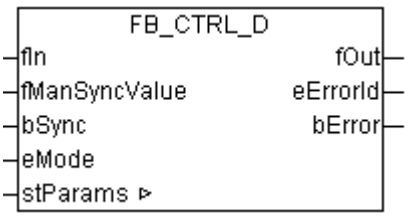
```
IF bInit
THEN
    stParams.tCtrlCycleTime     := T#10ms;
    stParams.tTaskCycleTime     := T#2ms;
    stParams.nNumberOfControlLoops  := 5;
    bInit               := FALSE;
END_IF
(* set addresses *)
stParams.nOutputVector_SIZEOF   := SIZEOF(arrOutputVector);
stParams.pOutputVector_ADR      := ADR(arrOutputVector);
(* call scheduler *)
fbCTRL_LoopScheduler( eMode     := eMode,
              stParams  := stParams,
              eErrorId  => eErrorId,
              bError    => bError);
IF arrOutputVector[ 1 ]
THEN
    (* activate control loop 1 *)
    eMode_CtrlLoop_1 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[ 2 ]
THEN
    (* activate control loop 2 *)
    eMode_CtrlLoop_2 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[ 3 ]
THEN
    (* activate control loop 3 *)
    eMode_CtrlLoop_3 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[ 4 ]
THEN
    (* activate control loop 4 *)
    eMode_CtrlLoop_4 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[ 5 ]
THEN
    (* activate control loop 5 *)
    eMode_CtrlLoop_5 := eCTRL_MODE_ACTIVE;
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3 Base

5.3.1 FB_CTRL_D



Der Funktionsbaustein stellt ein DT_1-Übertragungsglied (reales D-Glied) in einem Wirkungsplan dar.

Übertragungsfunktion (kontinuierlich):

$$G(s) = \frac{T_v s}{1 + T_d s}$$

VAR_INPUT

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

fIn : Eingang des D-Glieds.

fManSyncValue : Eingang, auf den das D-Glied synchronisiert werden kann, oder dessen Wert im Manual-Mode am Ausgang anliegt.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das D-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des D-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_D_PARAMS;
END_VAR
```

stParams : Parameterstruktur des D-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_D_PARAMS
:
STRUCT
    tCtrlCycleTime      : TIME     :=  T#0ms; (* controller
cycle time [TIME] *)
    tTaskCycleTime      : TIME     :=  T#0ms; (* task cycle time
[TIME] *)
    tTv         : TIME     :=  T#0ms; (* derivative
action time Tv *)
    tTd         : TIME     :=  T#0ms; (* derivative
damping time Td *)
    fOutMaxLimit        : FLOAT    :=  1E38;  (* maximum output
limit *)
    fOutMinLimit        : FLOAT    := -1E38;  (* minimum output
limit *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tTv : Differenzierzeitkonstante

tTd : Dämpfungszeitkonstante

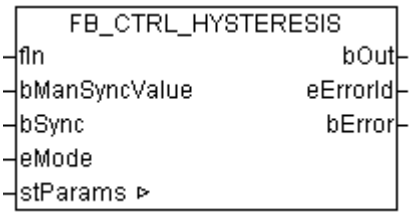
fOutMaxLimit : Oberes Limit, an dem der Ausgang des D-Glieds begrenzt wird.

fOutMinLimit : Unteres Limit, an dem der Ausgang des D-Glieds begrenzt wird.

Voraussetzungen

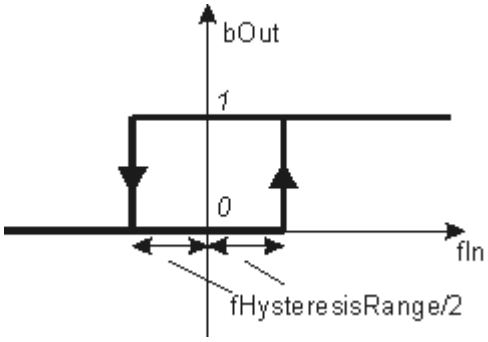
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3.2 FB_CTRL_HYSTERESIS



Der Funktionsbaustein stellt ein Hysterese-Übertragungsglied in einem Wirkungsplan dar.

Übertragungsfunktion:



VAR_INPUT

```
VAR_INPUT
    fIn             : FLOAT;
    bManSyncValue   : BOOL;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

fIn : Eingang des Hysterese-Glieds.

bManSyncValue : Eingang, mit dem das Hysterese-Glied auf einen der beiden Zweige gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das Hysterese-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    bOut        : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bOut : Ausgang des Hysterese-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_HYSTERESIS_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Hysterese-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_HYSTERESIS_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME := T#0ms; (* task cycle time
[TIME] *)
    fHysteresisRange   : FLOAT; (* range of the hysteresis loop
*)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

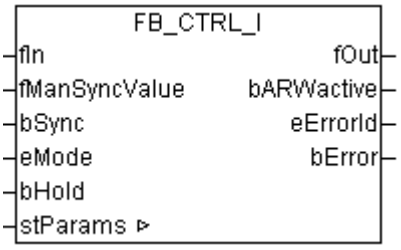
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fHysteresisRange : Hysterese-Bereich, siehe Bild oben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3.3 FB_CTRL_I



Der Funktionsbaustein stellt ein I-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = \frac{1}{T_I s}$$

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fIn : Eingang des I-Glieds.

fManSyncValue : Eingang, auf den das I-Glied synchronisiert werden kann, oder dessen Wert im Manual-Mode am Ausgang anliegt.

bSync : Mit einer steigenden Flanke an diesem Eingang wird der Integrator auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den Integrator unabhängig von dem Eingang fIn konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    bARWactive      : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des I-Glieds.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich der Integrator in der Begrenzung befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_I_PARAMS;
END_VAR
```

stParams : Parameterstruktur des I-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_I_PARAMS
:
STRUCT
    tCtrlCycleTime : TIME  :=  T#0ms; (* controller cycle time
*)
    tTaskCycleTime : TIME  :=  T#0ms; (* task cycle time
*)
    tTi        : TIME  :=  T#0ms; (* integral action time
Ti *)
    fOutMaxLimit   : FLOAT :=  1E38;  (* maximum output limit
*)
    fOutMinLimit   : FLOAT := -1E38;  (* minimum output limit
*)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tTi : Integrationszeit des I-Glieds.

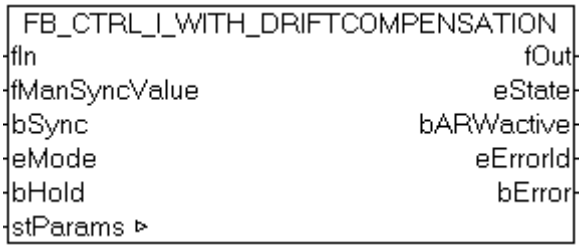
fOutMaxLimit : Oberes Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fOutMinLimit : Unteres Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

Voraussetzungen

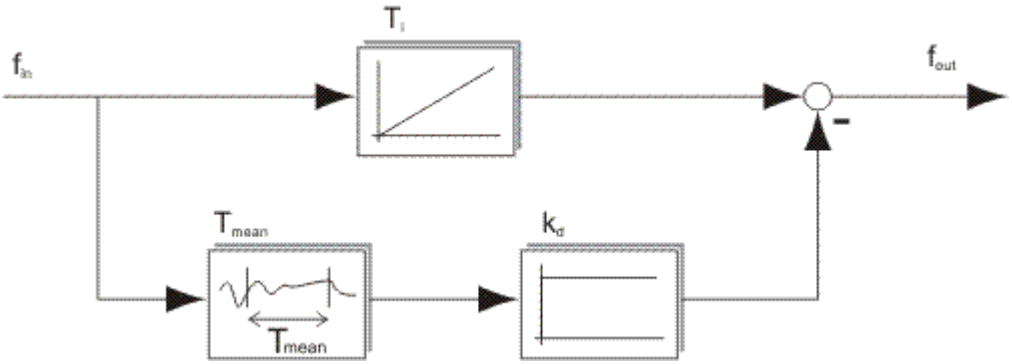
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3.4 FB_CTRL_I_WITH_DRIFTCOMPENSATION



Der Funktionsbaustein stellt ein I-Übertragungsglied mit einer Driftkompensation dar.

Wirkungsplan:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fIn : Eingang des I-Glieds.

fManSyncValue : Eingang, auf den das I-Glied synchronisiert werden kann, oder dessen Wert im Manual-Mode am Ausgang anliegt.

bSync : Mit einer steigenden Flanke an diesem Eingang wird der Integrator auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den Integrator unabhängig von dem Eingang fIn konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    bARWactive      : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des I-Glieds.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich der Integrator in der Begrenzung befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS;
END_VAR
```

stParams : Parameterstruktur des I-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS:
STRUCT
     tCtrlCycleTime      : TIME := T#0ms; (* controller cycle
time [TIME] *)
     tTaskCycleTime      : TIME := T#0ms; (* task cycle time
[TIME] *)
     tTi         : TIME := T#0ms; (* integral action
time Ti *)
     fOutMaxLimit    : FLOAT := 1E38; (* maximum output
limit *)
     fOutMinLimit    : FLOAT := -1E38; (* minimum output
limit *)
     fDampingCoefficient : FLOAT := 0.0;
     tAveragingTime      : TIME := T#0ms; (* averaging time
*)
     pWorkArray_ADR      : POINTER TO FLOAT := 0;
     nWorkArray_SIZEOF   : UINT := 0;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tTi : Integrationszeit des I-Glieds.

fOutMaxLimit : Oberes Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fOutMinLimit : Unteres Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fDampingCoefficient : Faktor k_d im Wirkungsplan.

tAveragingTime : Zeit, über die das gleitende Mittelwertfilter den Mittelwert bildet.

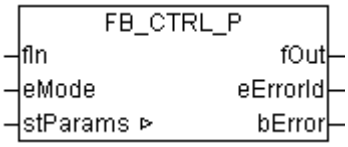
pWorkArray_ADR : Adresse eines Array of FLOAT der Größe tAveragingTime / tCtrlCycleTime. (Siehe Beschreibung des Bausteins: FB_CTRL_MOVING_AVERAGE.)

nWorkArray_SIZEOF : Größe eines Array of FLOAT der Größe tAveragingTime / tCtrlCycleTime. (Siehe Beschreibung des Bausteins: FB_CTRL_MOVING_AVERAGE.)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3.5 FUNCTION_BLOCKFB_CTRL_P



Der Funktionsbaustein stellt ein P-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p$$

VAR_INPUT

```
VAR_INPUT
    fIn         :FLOAT;
    eMode       :E_CTRL_MODE;
END_VAR
```

fIn : Eingang des P-Glieds.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        :FLOAT;
    eState      :E_CTRL_STATE;
    eErrorId        :E_CTRL_ERRORCODES;
    bError      :BOOL;
END_VAR
```

fOut : Ausgang des P-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        :ST_CTRL_P_PARAMS;
END_VAR
```

stParams : Parameterstruktur des P-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_P_PARAMS
:
STRUCT
    tCtrlCycleTime : TIME    := T#0ms;  (* controller cycle
time *)
    tTaskCycleTime : TIME    := T#0ms;  (* task cycle time
*)
    fKp        : FLOAT   := 0.0;    (* proportional gain Kp
*)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

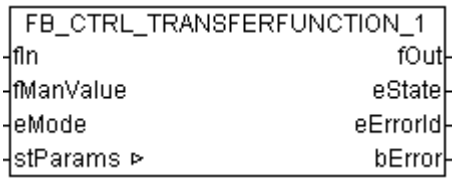
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Proportionalverstärkung des P-Glieds.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3.6 FB_CTRL_TRANSFERFUNCTION_1



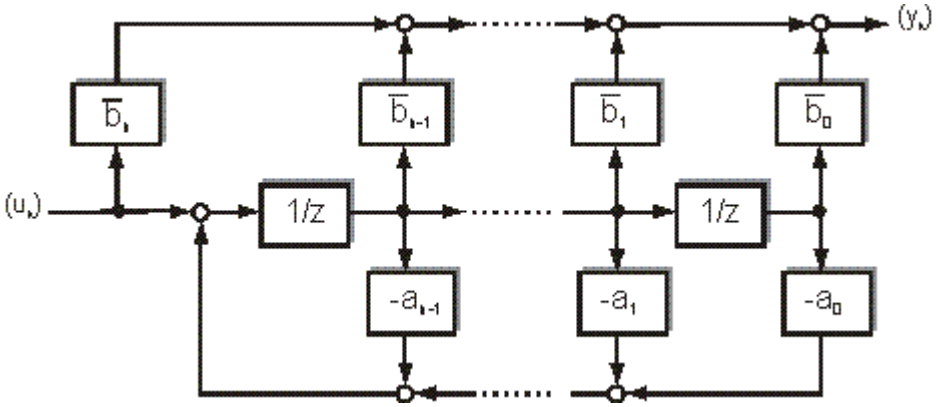
Dieser Baustein berechnet eine diskrete Übertragungsfunktion mit der unten dargestellten 1. Standardform. Die Übertragungsfunktion kann hierbei von beliebiger Ordnung n sein.

In den Parameter-Arrays werden die Koeffizienten der folgenden Übertragungsfunktion abgelegt:

$$G(z) = \frac{b_n z^n + b_{(n-1)} z^{(n-1)} + \cdots + b_1 z + b_0}{z^n + a_{(n-1)} z^{(n-1)} + \cdots + a_1 z + a_0}$$

Beschreibung des Übertragungsverhalten:

Die obige Übertragungsfunktion wird nach einigen Umformungen in jedem Abtastschritt mit der 1. Standardform berechnet.



Um diesen Funktionsbaustein nutzen zu können, müssen vom Programmierer in der SPS die folgenden Arrays angelegt werden:

```
ar_fNumeratorArray     :
ARRAY[0..nOrderOfTheTransferfunction] OF FLOAT;
ar_DenominatorArray    : ARRAY[0..nOrderOfTheTransferfunction]
OF FLOAT;
ar_stTransferfunction1Data : ARRAY[0..nOrderOfTheTransferfunction]
OF ST_CTRL_TRANSFERFUNCTION_1_DATA;
```

In dem Array ar_fNumeratorArray werden die Koeffizienten b_0 bis b_n abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
ar_fNumeratorArray[
0 ]     := b0;
ar_fNumeratorArray[ 1 ]     := b1;
```

```
...
ar_fNumeratorArray[ n-1 ]   := bn-1;
ar_fNumeratorArray[ n ]     := bn;
```

In dem Array ar_DenominatorArray werden die Koeffizienten a_0 bis a_n abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
ar_DenominatorArray
[ 0 ]    := a0;
ar_DenominatorArray [ 1 ]    := a1;
...
ar_DenominatorArray [ n-1 ]  := an-1;
ar_DenomiantorArray [ n ]    := an;
```

Die internen Daten, die der Baustein benötigt, werden in dem Array ar_stTransferfunction1Data abgelegt. Diese Daten dürfen innerhalb des SPS-Programms keinesfalls geändert werden. Diese Vorgehensweise wird auch in dem unten aufgeführten Beispielprogramm dargestellt.

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingang der Übertragungsfunktion.

fManValue : Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    bError      : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

fOut : Ausgang der Übertragungsfunktion.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_TRANSFERFUNCTION_1_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_TRANSFERFUNCTION_1_PARAMS:
STRUCT
    tTaskCycleTime        : TIME;      (* task
cycle time in seconds *)
    tCtrlCycleTime        : TIME := T#0ms; (*
controller cycle time *)
    nOrderOfTheTransferfunction   : USINT;
    pNumeratorArray_ADR       : POINTER TO FLOAT :=
0;
    nNumeratorArray_SIZEOF    : UINT;
    pDenominatorArray_ADR     : POINTER TO FLOAT :=
0;
    nDenomiantorArray_SIZEOF      : UINT;
    pTransferfunction1Data_ADR    : POINTER TO
ST_CTRL_TRANSFERFUNCTION_1_DATA;
    nTransferfunction1Data_SIZEOF : UINT;
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

nOrderOfTheTransferfunction : Ordnung der Übertragungsfunktion [0...]

pNumeratorArray_ADR : Adresse des Arrays mit den Zählerkoeffizienten.

nNumeratorArray_SIZEOF : Größe des Arrays mit den Zählerkoeffizienten in Byte.

pDenominatorArray_ADR : Adresse des Arrays mit den Nennerkoeffizienten.

nDenominatorArray_SIZEOF : Größe des Arrays mit den Nennerkoeffizienten in Byte.

pTransferfunction1Data_ADR : Adresse des Daten-Arrays.

nTransferfunction1Data_SIZEOF : Größe des Daten-Arrays in Byte.

```
TYPE
ST_CTRL_TRANSFERFUNCTION_1_DATA:
STRUCT
    Interne Struktur. Auf diese darf nicht schreibend
zugegriffen werden.
END_STRUCT
END_TYPE
```

Beispiel:

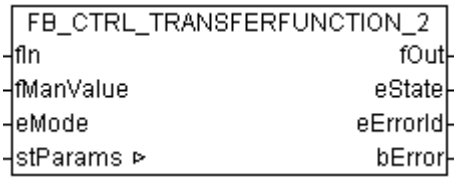
```
PROGRAM PRG_TRANSFERFUNCTION_1_TEST
VAR CONSTANT
    nOrderOfTheTransferfunction : USINT := 2;
END_VAR
VAR
    ar_fNumeratorArray     :
ARRAY[0..nOrderOfTheTransferfunction] OF FLOAT;
    ar_DenominatorArray    :
ARRAY[0..nOrderOfTheTransferfunction] OF FLOAT;
    ar_stTransferfunction1Data :
ARRAY[0..nOrderOfTheTransferfunction] OF
ST_CTRL_TRANSFERFUNCTION_1_DATA;
    eMode     : E_CTRL_MODE;
    stParams      : ST_CTRL_TRANSFERFUNCTION_1_PARAMS;
    eErrorId      : E_CTRL_ERRORCODES;
    bError    : BOOL;
    fbTansferfunction : FB_CTRL_TRANSFERFUNCTION_1;
    bInit     : BOOL := TRUE;
    fIn       : FLOAT := 0;
    fOut      : FLOAT;
    b_0, b_1, b_2   : FLOAT;
    a_0,a_1,a_2     : FLOAT;
END_VAR
IF bInit
THEN
    (* set values in the local arrays *)
    ar_fNumeratorArray[0] := 1.24906304658218E-007;
    ar_fNumeratorArray[1] := 2.49812609316437E-007;
    ar_fNumeratorArray[2] := 1.24906304658218E-007;
    ar_DenominatorArray[0] := 0.998501124344101;
    ar_DenominatorArray[1] := -1.99850062471888;
    ar_DenominatorArray[2] := 1.0;
    (* set values in the parameter struct *)
    stParams.tTaskCycleTime        := T#2ms;
    stParams.tCtrlCycleTime        := T#2ms;
    stParams.nOrderOfTheTransferfunction   :=
nOrderOfTheTransferfunction;
    (* set the mode *)
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
(* set the addresses *)
stParams.pNumeratorArray_ADR       := ADR(
ar_fNumeratorArray);
stParams.nNumeratorArray_SIZEOF    := SIZEOF(
ar_fNumeratorArray);
```

```
stParams.pDenominatorArray_ADR     := ADR( ar_DenominatorArray
);
stParams.nDenominatorArray_SIZEOF      := SIZEOF(
ar_DenominatorArray );
stParams.pTransferfunction1Data_ADR    := ADR(
ar_stTransferfunction2Data );
stParams.nTransferfunction1Data_SIZEOF := SIZEOF(
ar_stTransferfunction2Data );
(* call the function block *)
fbTansferfunction ( fIn       := fIn,
            eMode     := eMode,
            stParams  := stParams,
            fOut      => fOut,
            eErrorId  => eErrorId,
            bError    => bError
            );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.3.7 FB_CTRL_TRANSFERFUNCTION_2



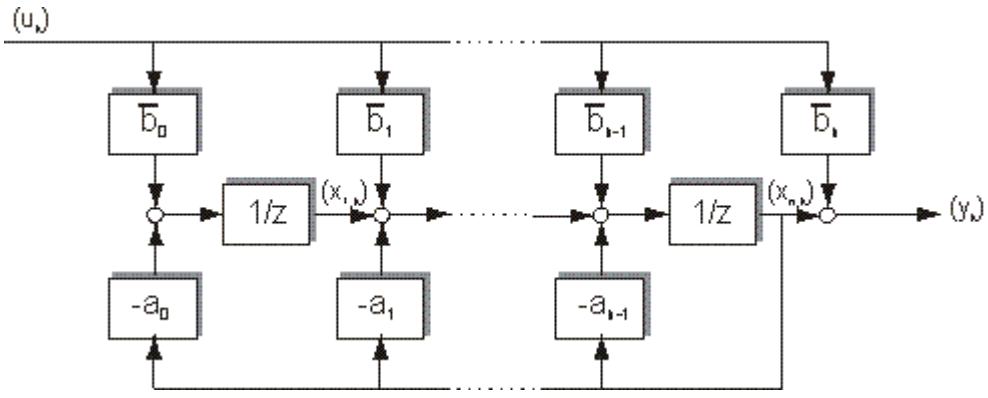
Dieser Baustein berechnet eine diskrete Übertragungsfunktion mit der unten dargestellten 2. Standardform. Die Übertragungsfunktion kann hierbei von beliebiger Ordnung n sein.

In den Parameter-Arrays werden die Koeffizienten der folgenden Übertragungsfunktion abgelegt:

$$G(z) = \frac{b_n z^n + b_{(n-1)} z^{(n-1)} + \cdots + b_1 z + b_0}{z^n + a_{(n-1)} z^{(n-1)} + \cdots + a_1 z + a_0}$$

Beschreibung des Übertragungsverhaltens:

Die interne Berechnung erfolgt in jedem Abtastschritt nach der 2. Standardform, für die das folgende Blockschaltbild gilt:



Durch einige Umformungen lässt sich die Übertragungsfunktion auf die im Blockschaltbild dargestellte Form bringen:

$$G(z) = \widetilde{b}_n + \frac{\widetilde{b}_{(n-1)}z^{-1} + \cdots + \widetilde{b}_1 z^{-(n-1)} + \widetilde{b}_0 z^{-n}}{1 + a_{(n-1)}z^{-1} + \cdots + a_1 z^{-(n-1)} + a_0 z^{-n}}$$

Abb. 1: FB_CTRL_TRANSFERFUNCTION_2_G2

Die Koeffizienten des Zählerpolynoms berechnen sich dabei nach folgender Vorschrift:

$$\begin{aligned} \widetilde{b}_i &= b_i - b_n a_i \quad \forall\, 0 \leq i < n \\ \widetilde{b}_n &= b_n \end{aligned}$$

Abb. 2: FB_CTRL_TRANSFERFUNCTION_2_bi

Um diesen Funktionsbaustein nutzen zu können, müssen vom Programmierer in der SPS die folgenden Arrays angelegt werden:

```
ar_fNumeratorArray     :
ARRAY[0..nOrderOfTheTransferfunction] OF FLOAT;
ar_DenominatorArray    : ARRAY[0..nOrderOfTheTransferfunction]
OF FLOAT;
ar_stTransferfunction2Data : ARRAY[0..nOrderOfTheTransferfunction]
OF ST_CTRL_TRANSFERFUNCTION_2_DATA;
```

In dem Array ar_fNumeratorArray werden die Koeffizienten b_0 bis b_n abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
ar_fNumeratorArray[
0 ]      := b0;
ar_fNumeratorArray[ 1 ]      := b1;
...
ar_fNumeratorArray[ n-1 ]    := bn-1;
ar_fNumeratorArray[ n ]      := bn;
```

In dem Array ar_DenominatorArray werden die Koeffizienten a_0 bis a_n abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
ar_DenominatorArray
[ 0 ]     := a0;
ar_DenominatorArray [ 1 ]     := a1;
...
ar_DenominatorArray [ n-1 ]   := an-1;
ar_DenomiantorArray [ n ]     := an;
```

Die internen Daten, die der Baustein benötigt, werden in dem Array ar_stTransferfunction2Data abgelegt. Diese Daten dürfen innerhalb des SPS-Programms keinesfalls geändert werden. Diese Vorgehensweise wird auch in dem unten aufgeführten Beispielprogramm dargestellt.

VAR_INPUT

```
VAR_INPUT
    fIn          : FLOAT;
    fManValue        : FLOAT;
    eMode        : E_CTRL_MODE;
END_VAR
```

fIn : Eingang der Übertragungsfunktion.

fManValue : Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    bError       : BOOL;
    eErrorId         : E_CTRL_ERRORCODES;
END_VAR
```

fOut : Ausgang der Übertragungsfunktion.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_TRANSFERFUNCTION_2_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_TRANSFERFUNCTION_2_PARAMS:
STRUCT
    tTaskCycleTime        : TIME;      (* task
cycle time in seconds *)
    tCtrlCycleTime        : TIME := T#0ms; (*
controller cycle time *)
    nOrderOfTheTransferfunction   : USINT;
    pNumeratorArray_ADR       : POINTER TO FLOAT :=
0;
    nNumeratorArray_SIZEOF    : UINT;
    pDenominatorArray_ADR     : POINTER TO FLOAT :=
0;
    nDenomiantorArray_SIZEOF      : UINT;
    pTransferfunction2Data_ADR    : POINTER TO
ST_CTRL_TRANSFERFUNCTION_2_DATA;
    nTransferfunction2Data_SIZEOF : UINT;
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

nOrderOfTheTransferfunction : Ordnung der Übertragungsfunktion [0...]

pNumeratorArray_ADR : Adresse des Arrays mit den Zählerkoeffizienten.

nNumeratorArray_SIZEOF : Größe des Arrays mit den Zählerkoeffizienten in Byte.

pDenominatorArray_ADR : Adresse des Arrays mit den Nennerkoeffizienten.

nDenominatorArray_SIZEOF : Größe des Arrays mit den Nennerkoeffizienten in Byte.

pTransferfunction2Data_ADR : Adresse des Daten-Arrays.

nTransferfunction2Data_SIZEOF : Größe des Daten-Arrays in Byte.

```
TYPE
ST_CTRL_TRANSFERFUNCTION_2_DATA:
STRUCT
    Interne Struktur. Auf diese darf nicht schreibend
zugegriffen werden.
END_STRUCT
END_TYPE
```

Beispiel:

```
PROGRAM PRG_TRANSFERFUNCTION_2_TEST
VAR CONSTANT
    nOrderOfTheTransferfunction : USINT := 2;
END_VAR
VAR
    ar_fNumeratorArray     :
ARRAY[0..nOrderOfTheTransferfunction] OF FLOAT;
    ar_DenominatorArray    :
ARRAY[0..nOrderOfTheTransferfunction] OF FLOAT;
    ar_stTransferfunction2Data :
```

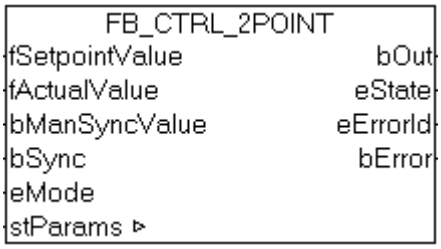
```
ARRAY[0..nOrderOfTheTransferfunction] OF
ST_CTRL_TRANSFERFUNCTION_2_DATA;
    eMode     : E_CTRL_MODE;
    stParams      : ST_CTRL_TRANSFERFUNCTION_2_PARAMS;
    eErrorId      : E_CTRL_ERRORCODES;
    bError    : BOOL;
    fbTansferfunction : FB_CTRL_TRANSFERFUNCTION_2;
    bInit     : BOOL := TRUE;
    fIn       : FLOAT := 0;
    fOut      : FLOAT;
    b_0, b_1, b_2   : FLOAT;
    a_0,a_1,a_2     : FLOAT;
END_VAR
IF bInit
THEN
    (* set values in the local arrays *)
    ar_fNumeratorArray[0] := 1.24906304658218E-007;
    ar_fNumeratorArray[1] := 2.49812609316437E-007;
    ar_fNumeratorArray[2] := 1.24906304658218E-007;
    ar_DenominatorArray[0] := 0.998501124344101;
    ar_DenominatorArray[1] := -1.99850062471888;
    ar_DenominatorArray[2] := 1.0;
    (* set values in the parameter struct *)
    stParams.tTaskCycleTime := T#2ms;
    stParams.tCtrlCycleTime := T#2ms;
    stParams.nOrderOfTheTransferfunction :=
nOrderOfTheTransferfunction;
    (* set the mode *)
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
(* set the addresses *)
stParams.pNumeratorArray_ADR := ADR( ar_fNumeratorArray);
stParams.nNumeratorArray_SIZEOF := SIZEOF(
ar_fNumeratorArray);
stParams.pDenominatorArray_ADR := ADR( ar_DenominatorArray );
stParams.nDenominatorArray_SIZEOF := SIZEOF( ar_DenominatorArray
);
stParams.pTransferfunction2Data_ADR := ADR(
ar_stTransferfunction2Data );
stParams.nTransferfunction2Data_SIZEOF := SIZEOF(
ar_stTransferfunction2Data );
fbTansferfunction ( fIn       := fIn,
            eMode     := eMode,
            stParams  := stParams,
            fOut      => fOut,
            eErrorId  => eErrorId,
            bError    => bError
            );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

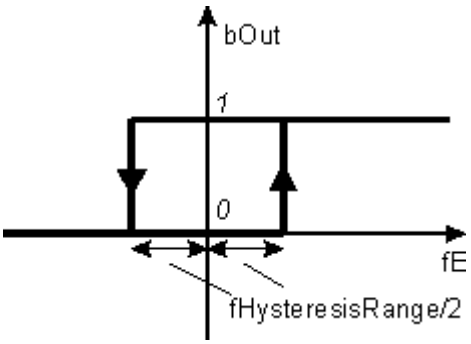
5.4 Controller

5.4.1 FB_CTRL_2POINT



Der Funktionsbaustein stellt ein 2Punkt-Übertragungsglied im Wirkungsplan dar.

Verhalten des Ausgangs:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    bManSyncValue   : BOOL;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

bManSyncValue : Eingang, mit dem das 2Punkt-Glied auf einen der beiden Zweige gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das 2Punkt-Glied auf den Wert bManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    bOut        : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bOut : Ausgang des 2Punkt-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_2POINT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des 2Punkt-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fHysteresisRange   : FLOAT;         (* range of the
hysteresis loop *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

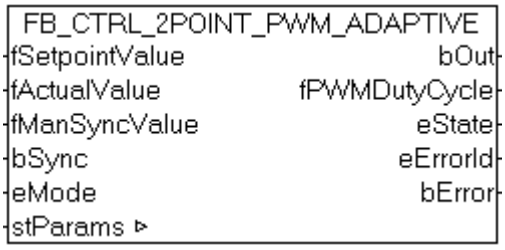
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fHysteresisRange: Hysterese-Bereich, siehe Bild oben.

Voraussetzungen

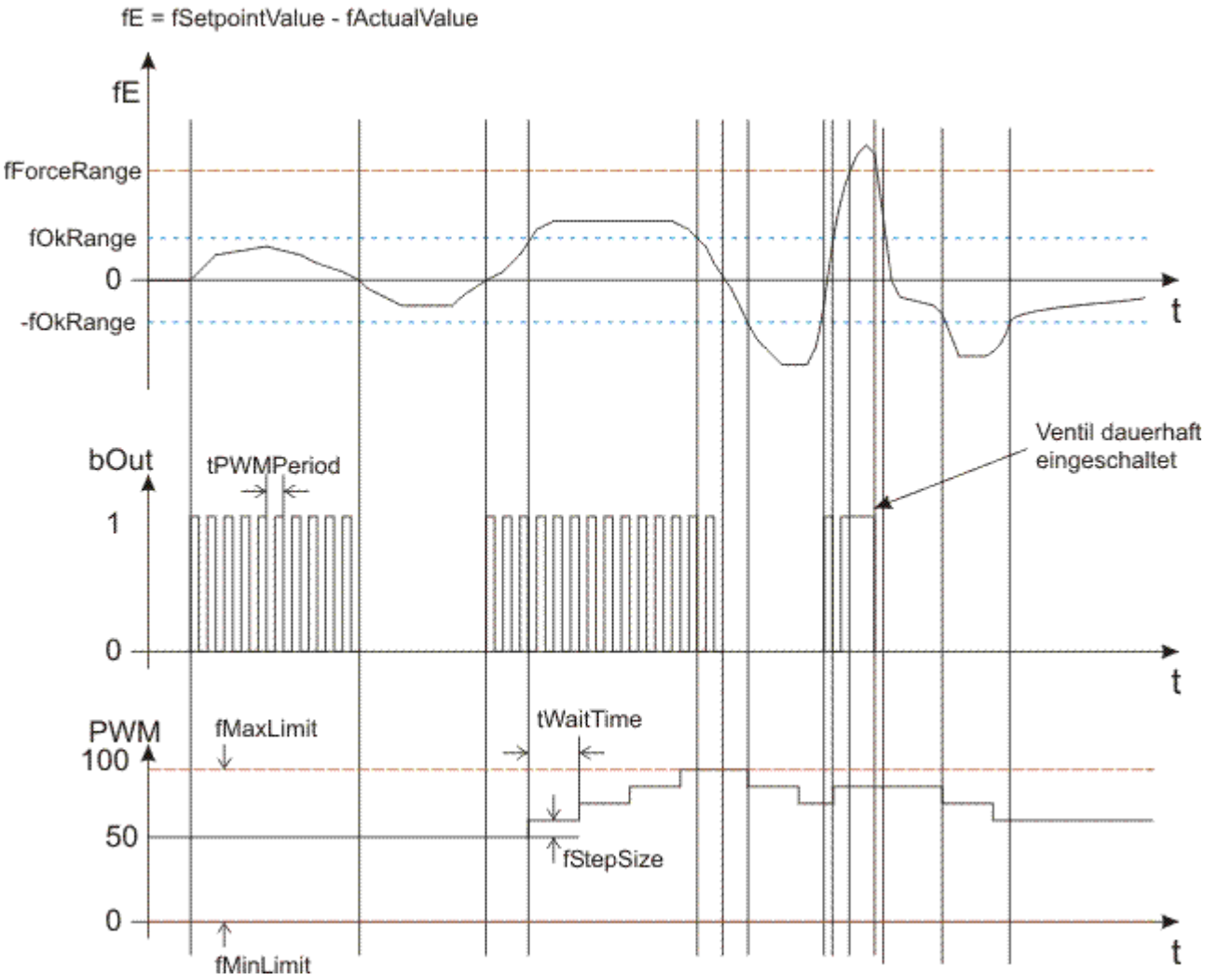
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.2 FB_CTRL_2POINT_PWM_ADAPTIVE



Der Funktionsbaustein stellt einen adaptiven Zweipunkt-Regler dar, der insbesondere für Einzelraumregelungen geeignet ist, bei denen hohe Vorlauftemperaturen vorhanden sind und die ein thermisches Stellglied einsetzen.

Verhalten des Ausgangs:



Beschreibung der Funktionsweise:

Der Regler nutz intern einen PWM-Baustein, der zu Ansteuerung des thermischen Stellglieds verwendet wird. Das Puls-Pausen-Verhältnis des PWM-Bausteins wird adaptiv an das Verhalten der Regelstrecke angepasst. Der PWM-Ausgang wird eingeschaltet, sobald die Regelabweichung fE = Sollwert - Istwert größer Null ist und abgeschaltet, wenn die Regelabweichung kleiner Null ist. Solange sich die Regelabweichung innerhalb des Intervalls [-fOkRange ... fOkRange] befindet, wird das Puls-Pausen-Verhältnis nicht verändert. Wenn fE > fOkRange ist, wird das Puls-Pausen-Verhältnis um fStepSize erhöht. Nach diesem Erhöhen wird die Zeit tWaitTime abgewartet, bevor das Puls-Pausen-Verhältnis eventuell erneut verändert wird. Bei unterschreiten von -fOkRange wird das Puls-Pausen-Verhältnis um fStepSize reduziert. Das Puls-Pausen-Verhältnis wir nur innerhalb des Intervalls [fMinLimit ... fMaxLimit] variiert. Die Periodendauer des PWM-Signals wird mit dem Parameter tPWMPeriod angegeben.

VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

fManSyncValue : Eingang, auf den das Puls-Pausen-Verhältnis des Reglers gesetzt werden kann, oder mit dem der Ausgang im Manual-Mode gesetzt werden kann. Im Manual-Mode wird der Ausgang gesetzt, wenn fManSyncValue > 0.0 ist.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das Puls-Pausen-Verhältnis des internen PWM-Bausteins auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    bOut        : BOOL;
    fPWMDutyCycle    : FLOAT;            (* debug only *)
    eState      : E_CTRL_STATE;
    eErrorId         : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bOut : Ausgang des Reglers.

fPWMDutyCycle : Aktuelles Puls-Pausen-Verhältnis des internen PWM-Bausteins.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des 2Punkt-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms; (* controller cycle
[TIME] *)
    tTaskCycleTime     : TIME     := T#0ms; (* task cycle time
[TIME] *)
    tPWMPeriod     : TIME
    fOkRange       : FLOAT
    fForceRange    : FLOAT
    fStepSize      : FLOAT
    fMinLimit      : FLOAT           (* [0% ... 100%]
*)
    fMaxLimit      : FLOAT           (* [0% ... 100%]
*)
    tWaitTime      : TIME
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tPMWPeriod :Periodendauer des PWM-Signals.

fOkRange :Bereich von fE, in dem das Puls-Pausen-Verhältnis nicht verändert wird.

fForceRange :Wenn fE diesen Bereich überschreitet, wird das Ausgang dauerhaft auf TRUE gesetzt.

fStepSize :Wert, um den das Puls-Pausen-Verhältnis bei jeder Adaption variiert wird. [0% ... 100%]

fMaxLimit :Maximales Puls-Pausen-Verhältnis in Prozent [0% ... 100%].

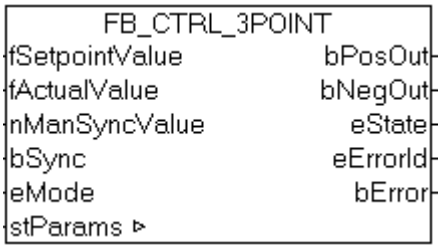
fMinLimit :Minimales Puls-Pausen-Verhältnis in Prozent [0% ... 100%].

tWaiTimet :Wartezeit zwischen den einzelnen Variationen des Puls-Pausen-Verhältnisses.

Voraussetzungen

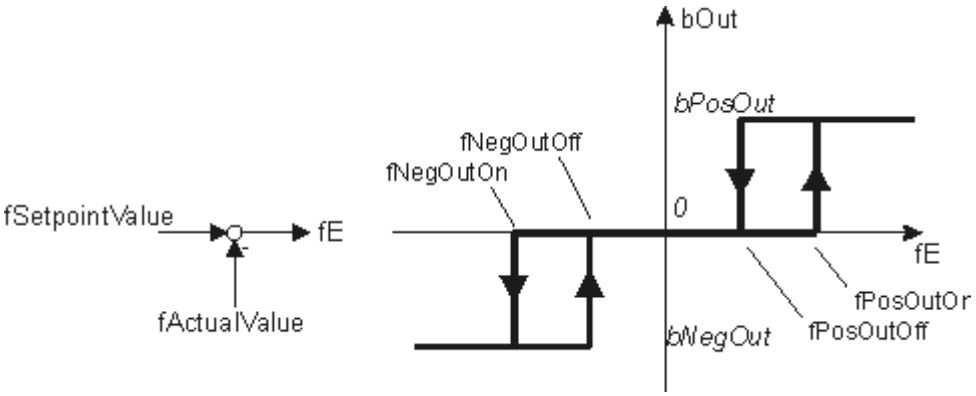
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.3 FB_CTRL_3POINT



Der Funktionsbaustein stellt ein 3-Punkt-Übertragungsglied im Wirkungsplan dar.

Verhalten des Ausgangs:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    nManSyncValue   : INT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

nManSyncValue : Eingang, mit dem das 3-Punkt-Glied auf einen der drei Zweige gesetzt werden kann.

nManSyncValue >= 1 → bPosOut = TRUE, bNegOut = FALSE
nManSyncValue <= -1 → bPosOut = FALSE, bNegOut = TRUE
sonst → bPosOut = FALSE, bNegOut = FALSE

bSync : Mit einer steigenden Flanke an diesem Eingang wird das 3-Punkt-Glied auf den Wert bManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    bPosOut         : BOOL;
    bNegOut         : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bPosOut : Dieser Ausgang des 3Punkt-Glieds ist TRUE, wenn der obere Zweig der Kennlinie aktiv ist.

bNegOut : Dieser Ausgang des 3Punkt-Glieds ist TRUE, wenn der untere Zweig der Kennlinie aktiv ist.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_3POINT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des 3Punkt-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_3POINT_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fPosOutOn      : FLOAT;         (* switch to
bPosOut on *)
    fPosOutOff     : FLOAT;         (* switch to
bPosOut off *)
    fNegOutOn      : FLOAT;         (* switch to
bNegOut on *)
    fNegOutOff     : FLOAT;         (* switch to
bNegOut off *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fPosOutOn : Regelabweichung, bei der von bPosOut = FALSE zu bPosOut = TRUE geschaltet wird (bNegOut = FALSE).

fPosOutOff : Regelabweichung, bei der von bPosOut = TRUE zu bPosOut = FALSE geschaltet wird (bNegOut = FALSE).

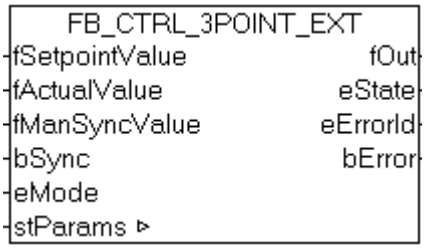
fNegOutOn : Regelabweichung, bei der von bNegOut = FALSE zu bNegOut = TRUE geschaltet wird (bPosOut = FALSE).

fNegOutOff : Regelabweichung, bei der von bNegOut = TRUE zu bNegOut = FALSE geschaltet wird (bPosOut = FALSE).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

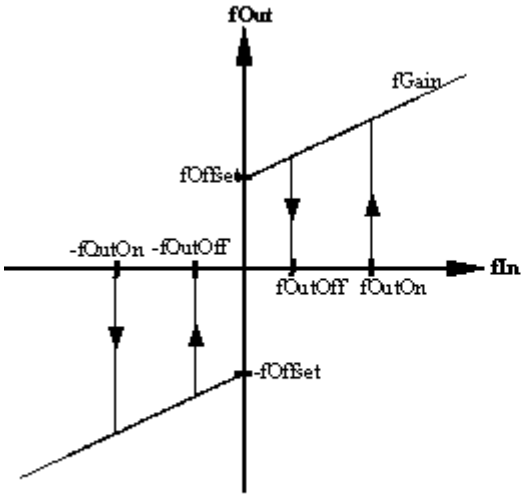
5.4.4 FB_CTRL_3POINT_EXT

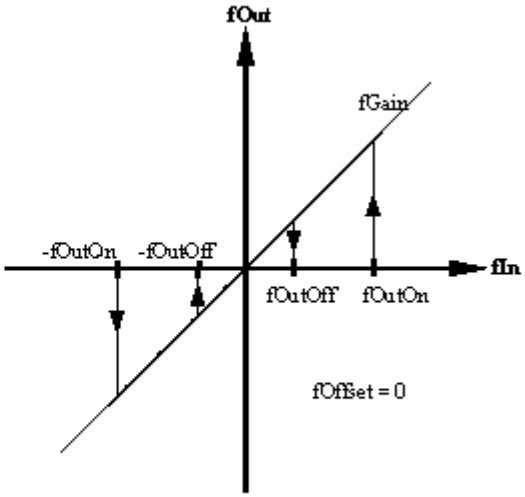


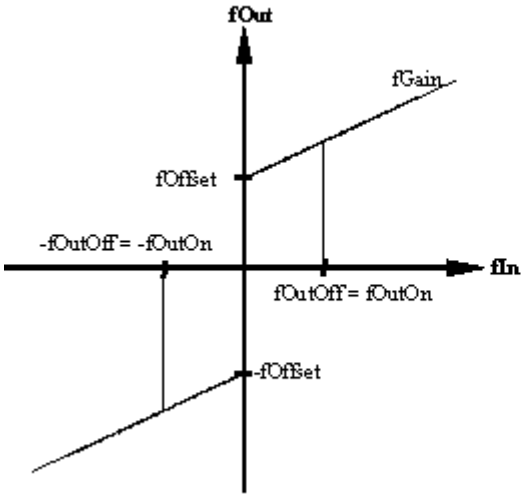
Der Funktionsbaustein stellt ein erweitertes 3-Punkt-Glied im Wirkungsplan dar.

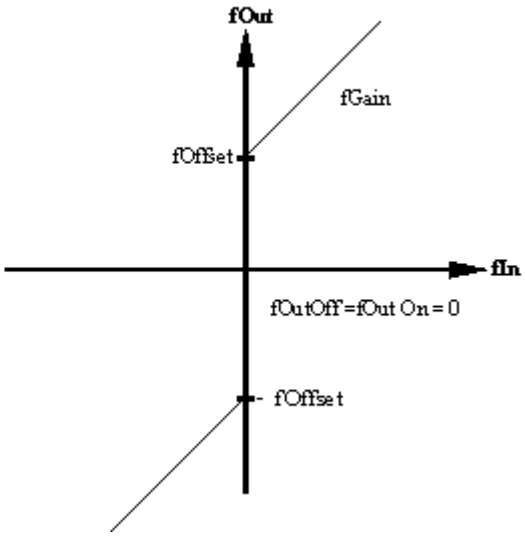
Verhalten des Ausgangs:

fIn := fSetpointValue - fActualValue;









VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

fManSyncValue : Eingang, mit dem das erweiterte 3-Punkt-Glied auf einen der Ausgangszweige gesetzt werden kann.

| fManSyncValue | < 1 → fOut = 0.0
| fManSyncValue | >= 1 → fOut = fE * fGain + fOffset

bSync : Mit einer steigenden Flanke an diesem Eingang wird das 3-Punkt-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut     : FLOAT;
    eState   : E_CTRL_STATE;
    eErrorId : E_CTRL_ERRORCODES;
    bError   : BOOL;
END_VAR
```

fOut : Ausgang des erweiterten 3-Punkt-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_3POINT_EXT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des erweiterten 3-Punkt-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_3POINT_EXT_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME := T#0ms; (* task cycle time
[TIME] *)
    fOutOff         : FLOAT;     (* x-parameter for
threshold OFF *)
    fOutOn          : FLOAT;     (* x-parameter for
threshold ON *)
    fGain           : FLOAT;     (* y-parameter gain
*)
    fOffset         : FLOAT;     (* y-parameter offset
*)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fOutOff : Wenn die Regelabweichung diesen Wert unterschreitet, wird der Ausgang abgeschaltet (auf Null gesetzt).

fOutOn : Wenn die Regelabweichung diesen Wert überschreitet, wird der Ausgang eingeschaltet.

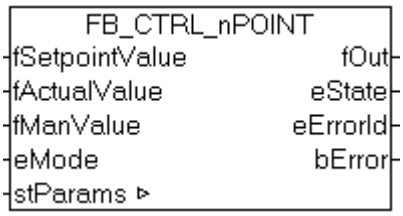
fGain : Verstärkungsfaktor.

fOffset : Offset.

Voraussetzungen

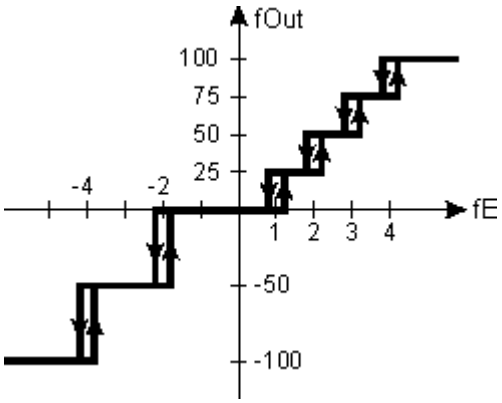
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.5 FB_CTRL_nPOINT



Der Funktionsbaustein stellt ein nPunkt-Übertragungsglied im Wirkungsplan dar.

Verhalten des Ausgangs:



Daten-Array des Beispiels:

fE	fOut
xx	-100
-4	-50
-2	0
1	25
2	50
3	75
4	100

Der Wert des Arrays mit dem Index (1,1), also der linke Wert in der 1. Zeile kann frei gewählt werden, da er nicht ausgewertet wird.

VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
```

```
    fManValue       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

fManValue : Eingang, der im Manual-Mode ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : nPOINT_CTRL_TABLE_ELEMENT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des nPunkt-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_nPOINT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des nPunkt-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_nPOINT_PARAMS :
STRUCT
    tCtrlCycleTime        : TIME     := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime        : TIME     := T#0ms; (* task
cycle time [TIME] *)
    pDataTable_ADR        : POINTER TO
nPOINT_CTRL_TABLE_ELEMENT := 0;
    nDataTable_SIZEOF     : UINT := 0;
    nDataTable_NumberOfRows   : UINT := 0;
    fHysteresisRange      : FLOAT;         (* range of
the hysteresis loop *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

pDataTable_ADR :Adresse der Daten-Tabelle.

nDataTable_SIZEOF :Größe der Daten-Tabelle in Bytes.

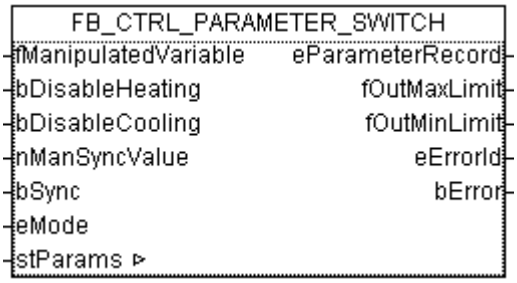
nDataTable_NumberOfRows :Zeilenanzahl der Daten-Tabelle.

fHysteresisRange :Hysterese-Bereich, siehe Bild oben. Der Hysterese-Bereich wirkt wie bei dem FB_CTRL_2POINT [▶ 40] beschrieben.

Voraussetzungen

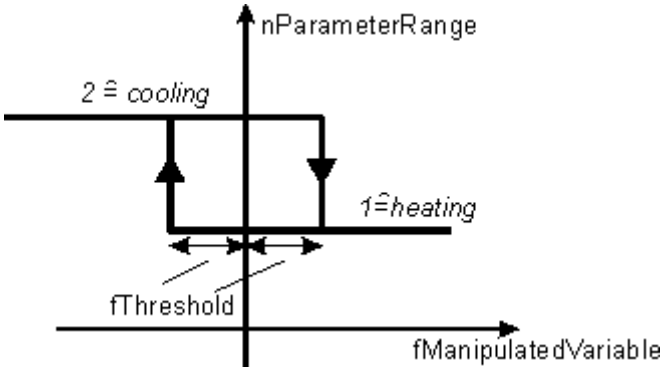
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.6 FB_CTRL_PARAMETER_SWITCH



Mit diesem Baustein kann der Parametersatz des FB_CTRL_PID_SPLITRANGE umgeschaltet werden.

Verhalten des Ausgangs:



Beschreibung des Funktionsbausteins:

Dieser Baustein dient zur Umschaltung des Parametersatzes bei dem **FB_CTRL_PID_SPLITRANGE**. Insbesondere ist dieser Baustein dazu gedacht, bei Regelungen, die mit zwei Stellgliedern heizen und kühlen können, die Parametersätze umzuschalten und die Begrenzungen des Reglers zu setzen. Als Eingangsparameter wird die Zeit **tMinWaitTime** angegeben, die bei einer Umschaltanforderung mindestens vergehen muss, damit der Parameterbereich gewechselt wird und die Reglerbegrenzungen so gesetzt werden, damit vom Heizbetrieb auf Kühlbetrieb umgeschaltet wird. Durch diese Maßnahmen soll verhindert werden, dass bei einem leichten Überschwingen des Reglers sofort die Betriebsart gewechselt wird.

Für den Heizbetrieb wird der Parameterbereich **eCTRL_PARAMETER_RECORD_HEATING = Heizen** angewählt und für den Kühlbetrieb der Parameterbereich **eCTRL_PARAMETER_RECORD_COOLING = Kühlen**. Die Reglerparametersätze müssen entsprechend dieser Vorgabe parametriert werden.

Die eigentliche Umschaltanforderung wird mit einem 2-Punkt-Glied bestimmt (vergleiche Bild). Als Eingangsgröße für die gezeigte Hysterese-Kennlinie sollte die Reglerausgangsgröße, also die Stellgröße, verwendet werden. Eine Umschaltanforderung, die das Hysterese-Glied erzeugt, muss mindestens für die angegebene Wartezeit anliegen, damit der Parameterbereich gewechselt wird.

Mit den Eingängen **bDisableRange1** und **bDisableRange2** ist es möglich, die Umschaltung in einen der beiden Bereiche zu verhindern. So kann beispielsweise im Sommer der Heizbetrieb deaktiviert werden und im Winter der Kühlbetrieb. Denkbar ist es auch, den Wechsel der Betriebsart von der aktuellen

Regeldifferenz abhängig zu machen. Im Sommer muss es zum Beispiel 2°C zu warm sein, damit in den Kühlbetrieb geschaltet wird. Auch dieses kann durch eine entsprechende Beschaltung der Eingänge erreicht werden.

Zusätzlich zu der Ausgabe des Parameterbereiches werden Max- und Min-Limits ausgeben, die in den Parametersatz des PID-Reglers kopiert werden können. Wenn sich der **FB_CTRL_PARAMETER_SWITCH** in der Betriebsart **Heizen** befindet, werden die Limits folgendermaßen gesetzt:

$$fOutMinLimit = -1.0 \cdot stParams.fThreshold;$$
$$fOutMaxLimit = stParams.fOutMaxLimit;$$

In der Betriebsart **Kühlen** werden die Limits folgendermaßen gesetzt:

$$fOutMinLimit = stParams.fOutMaxLimit;$$
$$fOutMaxLimit = stParams.fThreshold;$$

VAR_INPUT

```
VAR_INPUT
    fManipulatedVariable    : FLOAT; (* fOut from the FB_CTRL_PID_SPLITRANGE *)
    nManSyncValue       : eCTRL_PARAMETER_RECORD_HEATING;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

fManipulatedVariable : Eingangsgröße des FB_Parameter_Switch. Diese sollte gleich der Ausgangsgröße des Reglers sein.

nManSyncValue : Eingang, mit dem der Funktionsbaustein auf einen der beiden Parameterbereiche gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird der Funktionsbaustein auf den Wert nManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eParameterRecord        : E_CTRL_PARAMETER_RECORD;
    fOutMaxLimit        : FLOAT;
    fOutMinLimit        : FLOAT;
    eState          : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

eParameterRecord : Ausgang des Funktionsbausteins, der den Parameterbereich angibt.

fOutMaxLimit : Maximale Ausgangsgröße, mit der der Regler begrenzt wird. (Sollte in die Parameter-Struktur des Reglers kopiert werden.)

fOutMinLimit : Minimale Ausgangsgröße, mit der der Regler begrenzt wird. (Sollte in die Parameter-Struktur des Reglers kopiert werden.)

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PARAMETER_SWITCH_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tTaskCycleTime     : TIME; (* task cycle time [TIME]
*)
    tCtrlCycleTime     : TIME; (* controller cycle time [TIME]
*)
    fThreshold     : FLOAT;
    fOutMaxLimit       : FLOAT; (* max limit for heating
*)
    fOutMinLimit       : FLOAT; (* min limit for heating
*)
    tMinWaitTime       : TIME;
END_STRUCT
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

fThreshold : Schaltschwelle, siehe Bild oben.

fOutMaxLimit : Max-Limit, welches an den Regler weitergegeben wird.

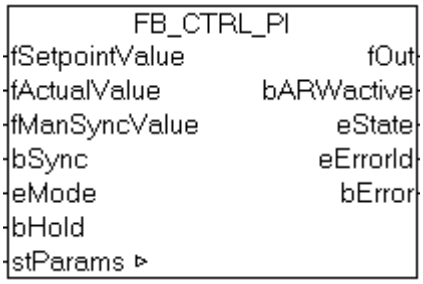
fOutMinLimit : Min-Limit, welches an den Regler weitergegeben wird.

tMinWaitTime : Wartezeit (siehe Beschreibung oben)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.7 FB_CTRL_PI

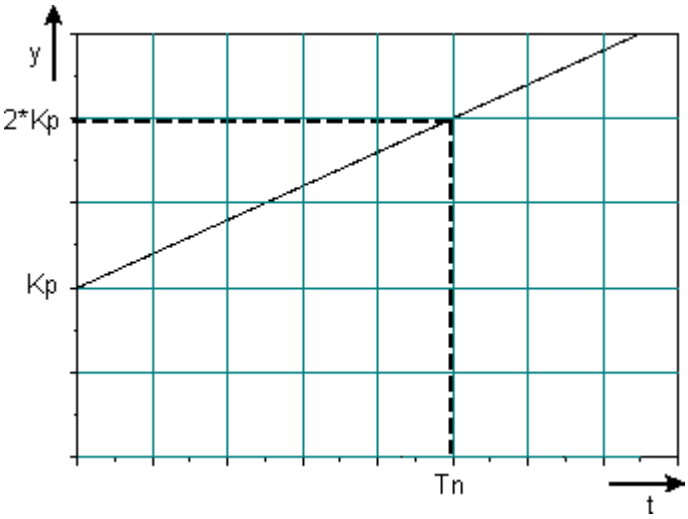


Der Funktionsbaustein stellt ein PI-Übertragungsglied im Wirkungsplan dar.

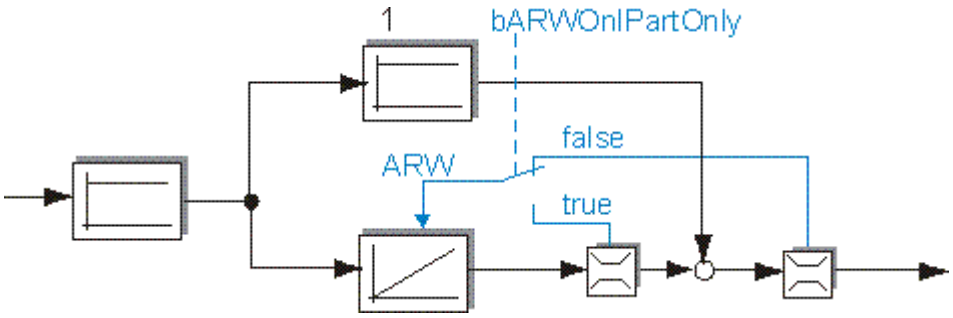
Verhalten des Ausgangs:

$$G(s) = K_p(1 + \frac{1}{T_n s})$$

Sprungantwort:



ARW:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

fManSyncValue : Eingang, mit dem das PI-Glied gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Regelabweichung konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    bARWactive      : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des PI-Glieds.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich das PI-Glied in der Begrenzung befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PI_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PI-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_PI_PARAMS
:
STRUCT
    tCtrlCycleTime     : TIME     :=  T#0ms; (* controller
cycle time [TIME] *)
    tTaskCycleTime     : TIME     :=  T#0ms; (* task cycle time
[TIME] *)
    tTn        : TIME     :=  T#0ms; (* integral action
time Tn *)
    fKp        : FLOAT    :=  0;     (* proportional
gain *)
    fOutMaxLimit       : FLOAT    :=  1E38;  (* maximum output
limit *)
    fOutMinLimit       : FLOAT    := -1E38;  (* minimum output
limit *)
    bARWOnIPartOnly    : BOOL     := FALSE;  (* FALSE: Hold the
I-part if the entire control output reaches a limit. *)
                         (* TRUE: Hold the
I-part if only the I-part reaches a limit. *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tTn : Nachstellzeit

fKp : Reglerverstärkung / Übertragungsbeiwert

fOutMaxLimit : Oberes Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

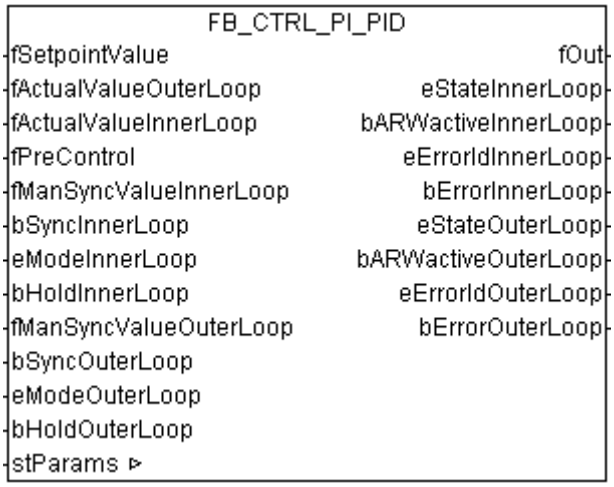
fOutMinLimit : Unteres Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

bARWOnIPartOnly: Wenn dieser Parameter FALSE ist (Standardeinstellung), wird die Integration des I-Anteil dann angehalten, wenn der gesamte Reglerausgang das obere oder untere Limit erreicht. Wenn dieser TRUE ist, wird die Integration dann angehalten, wenn der I-Anteil (der Integratorausgang) ein Limit erreicht. (Vgl. Wirkungsplan)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.8 FB_CTRL_PI_PID



Der Funktionsbaustein stellt einen kaskadierten PI - PID-Regler im Wirkungsplan dar. Intern nutzt dieser Baustein die Übertragungsglieder FB_CTRL_PI, FB_CTRL_LIMITER und den FB_CTRL_PID.

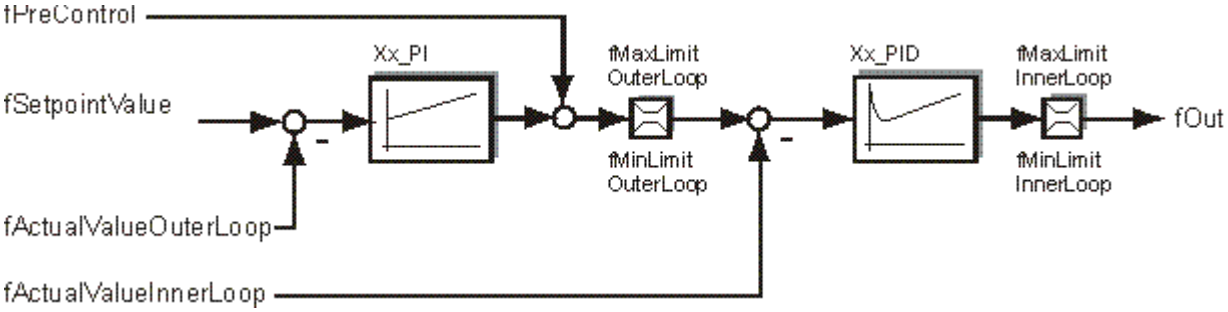
Übertragungsfunktion des PI-Glieds:

$$G(s) = K_p(1 + \frac{1}{T_n s})$$

Übertragungsfunktion des PID-Glieds:

$$G(s) = K_p(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s})$$

Wirkungsplan des kaskadierten Übertragungsglieds:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue      : FLOAT;        (* setpoint value *)
    fActualValueOuterLoop   : FLOAT;        (* actual value from the process to the PI-controller *)
    fActualValueInnerLoop   : FLOAT;        (* actual value from the process to the PID-
controller *)
    fPreControl         : FLOAT;        (* pre control value *)

    fManSyncValueInnerLoop  : FLOAT;        (* input value for the manual mode or the sync request *
)
    bSyncInnerLoop      : BOOL;         (* rising edge set the output to the fManSyncValue *)
    eModeInnerLoop      : E_CTRL_MODE;  (* sets the mode of the function block *)
    bHoldInnerLoop      : BOOL;         (* hold the internal integrator *)

    fManSyncValueOuterLoop  : FLOAT;        (* input value for the manual mode or the sync request *
```

```
)
    bSyncOuterLoop      : BOOL;         (* rising edge set the output to the fManSyncValue *)
    eModeOuterLoop      : E_CTRL_MODE;  (* sets the mode of the function block *)
    bHoldOuterLoop      : BOOL;         (* hold the internal integrator *)
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValueOuterLoop : Istwert der Regelgröße, die auf den PI-Regler des äußeren Regelkreises zurückgeführt wird.

fActualValueInnerLoop : Istwert der Regelgröße, die auf den PID-Regler des inneren Regelkreises zurückgeführt wird.

fPreControl : Vorsteuerung, die hinter dem PI-Regler aufgeschaltet wird.

fManSyncValueInnerLoop : Eingang, auf dessen Wert der interne Zustand des PID-Glieds (innerer Regelkreises) gesetzt werden kann.

bSyncInnerLoop : Mit einer steigenden Flanke an diesem Eingang wird das PID-Glied (innerer Regelkreises) auf den Wert fManSyncValueInnerLoop gesetzt.

eModeInnerLoop : Eingang, der die Betriebsart [▶ 16] des PID-Glieds (innerer Regelkreises) festlegt.

bHoldInnerLoop : Ein TRUE an diesem Eingang hält den internen Zustand des PID-Glieds (innerer Regelkreises) konstant auf dem aktuellen Wert.

fManSyncValueOuterLoop : Eingang, auf dessen Wert der interne Zustand des PI-Glieds (äußerer Regelkreises) gesetzt werden kann.

bSyncOuterLoop : Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied (äußerer Regelkreises) auf den Wert fManSyncValueOuterLoop gesetzt.

eModeOuterLoop : Eingang, der die Betriebsart [▶ 16] des PI-Glieds (äußerer Regelkreises) festlegt.

bHoldOuterLoop : Ein TRUE an diesem Eingang hält den internen Zustand des PI-Glieds (äußerer Regelkreises) konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut            : FLOAT;            (* manipulated value to the process *)

    eStateInnerLoop         : E_CTRL_STATE;
    bARWactiveInnerLoop     : BOOL;
    eErrorIdInnerLoop           : E_CTRL_ERRORCODES;    (* error code *)
    bErrorInnerLoop         : BOOL;             (* TRUE if an error situation exists *)

    eStateOuterLoop         : E_CTRL_STATE;
    bARWactiveOuterLoop     : BOOL;
    eErrorIdOuterLoop           : E_CTRL_ERRORCODES;    (* error code *)
    bErrorOuterLoop         : BOOL;             (* TRUE if an error situation exists *)
END_VAR
```

fOut : Ausgang des PI-PID-Glieds.

eStateInnerLoop : State des internen PID-Glieds (innerer Regelkreis).

bARWactiveInnerLoop : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied (innerer Regelkreis) in der Begrenzung befindet.

eErrorIdInnerLoop : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16] des PID-Glieds (innerer Regelkreis).

bErrorInnerLoop : Wird TRUE, sobald ein Fehler in dem PID-Glied (innerer Regelkreis) eintritt.
eStateOuterLoop : State des internen PI-Glieds (äußerer Regelkreis).

bARWactiveOuterLoop : Ein TRUE an diesem Ausgang signalisiert, dass sich das PI-Glied (äußerer Regelkreis) in der Begrenzung befindet.

eErrorIdOuterLoop : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16] des PI-Glieds (äußerer Regelkreis).

bErrorOuterLoop : Wird TRUE, sobald ein Fehler in dem PI-Glied (äußerer Regelkreis) eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PI_PID_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PI-PID-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PI_PID_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME  := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime    : TIME  := T#0ms; (* task cycle time
[TIME] *)
    fKp_OuterLoop     : FLOAT := 0; (* proportional gain
*)
    tTn_OuterLoop     : TIME  := T#0s; (* Tn *)
    fMaxLimit_OuterLoop   : FLOAT := 1E38;
    fMinLimit_OuterLoop   : FLOAT := -1E38;
    fKp_InnerLoop     : FLOAT := 0; (* proportional gain
*)
    tTn_InnerLoop     : TIME  := T#0ms; (* Tn *)
    tTv_InnerLoop     : TIME  := T#0ms; (* Tv *)
    tTd_InnerLoop     : TIME  := T#0ms; (* Td *)
    fMaxLimit_InnerLoop   : FLOAT := 1E38;
    fMinLimit_InnerLoop   : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp_OuterLoop : Reglerverstärkung / Reglerbeiwert des PI-Glieds (äußerer Regelkreis).

tTn_OuterLoop : Nachstellzeit des PI-Glieds (äußerer Regelkreis). Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

fMaxLimit_OuterLoop : Oberes Limit, an dem die Integration des PID-Glieds angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWactiveOuterLoop signalisiert.

fMinLimit_OuterLoop : Unteres Limit, an dem die Integration des PID-Glieds angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWactiveOuterLoop signalisiert.

fKp_InnerLoop : Reglerverstärkung / Reglerbeiwert des PID-Glieds (innerer Regelkreis).

tTn_InnerLoop : Nachstellzeit des PID-Glieds (innerer Regelkreis). Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv_InnerLoop : Vorhaltzeit des PID-Glieds (innerer Regelkreis). Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd_InnerLoop : Dämpfungszeit des PID-Glieds (innerer Regelkreis).

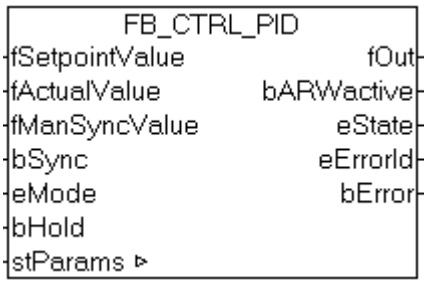
fMaxLimit_InnerLoop: Oberes Limit, an dem die Integration des PID-Glieds angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWactiveInnerLoop signalisiert.

fMinLimit_InnerLoop : Unteres Limit, an dem die Integration des PID-Glieds angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWactiveInnerLoop signalisiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.9 FB_CTRL_PID



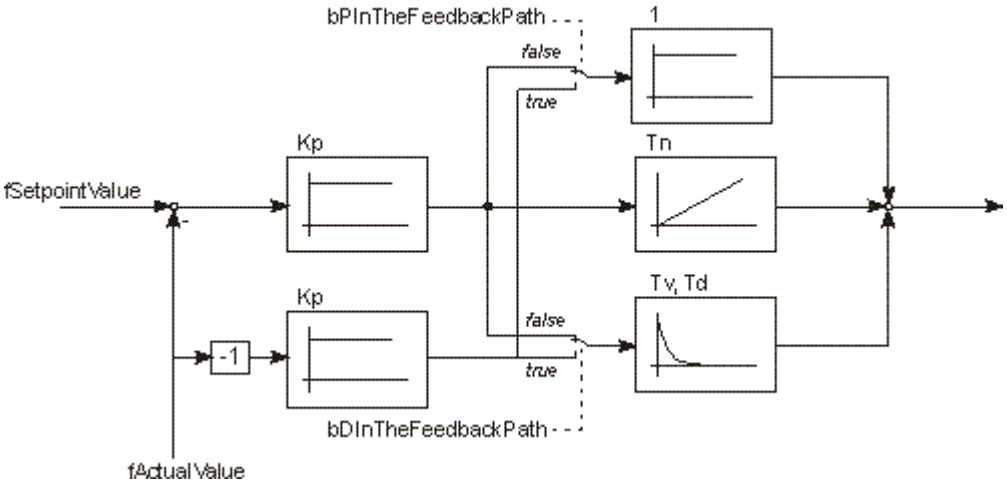
Der Funktionsbaustein stellt ein PID-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** FALSE sind, andernfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blockes:

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Wirkungsplan:



Der Standard Wirkungsplan eines PID-Reglers in additiver Form ist um die beiden als "Schalter" wirkenden boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** erweitert worden, so dass ein modifizierter Wirkungsplan aktiviert werden kann.

Der Regelungstechnische Hintergrund ist der, dass bei Sollwertsprüngen durch den Differential-Anteil des Regelalgorithmus große Stellgröße entstehen, die die Stellelemente belasten und das Regelsystem zum Schwingen anregen können. Ein Regelalgorithmus, dessen Differential-Anteil nur auf der Regelgröße angewendet wird (bDInTheFeedbackPath := TRUE), vermeidet dieses Problem.

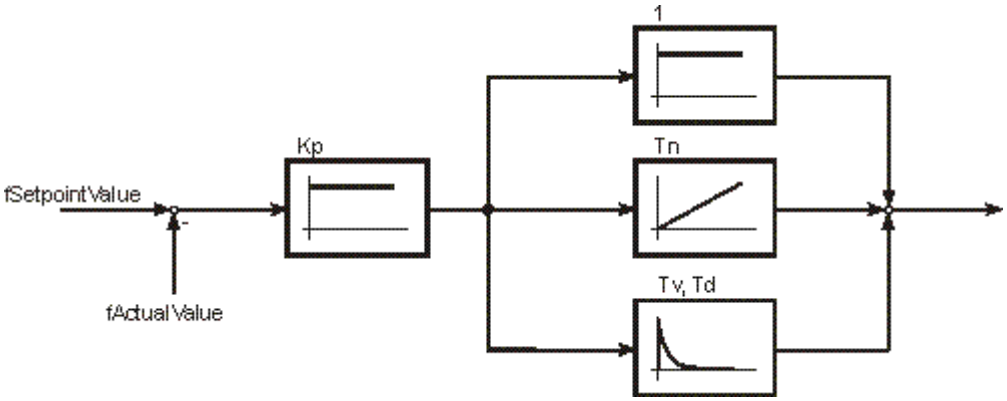
Mit den Eingängen bPInTheFeedbackPath und bDInTheFeedbackPath können die folgenden Übertragungsfunktionen des geschlossenen Regelkreises realisiert werden:

bPInTheFeedbackPath	bDInTheFeedbackPath	$G(s)$
false	false	$G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \frac{G_I \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$

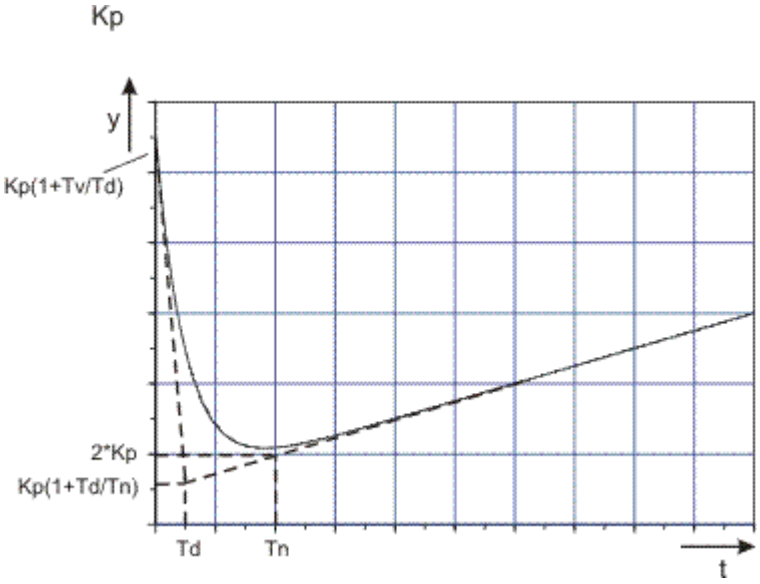
mit:

$$G_{DI}(s) = K_p(\frac{1}{T_n s} + \frac{T_v s}{1+T_d s})$$
$$G_{PI}(s) = K_p(1 + \frac{1}{T_n s})$$
$$G_I(s) = K_p(\frac{1}{T_n s})$$

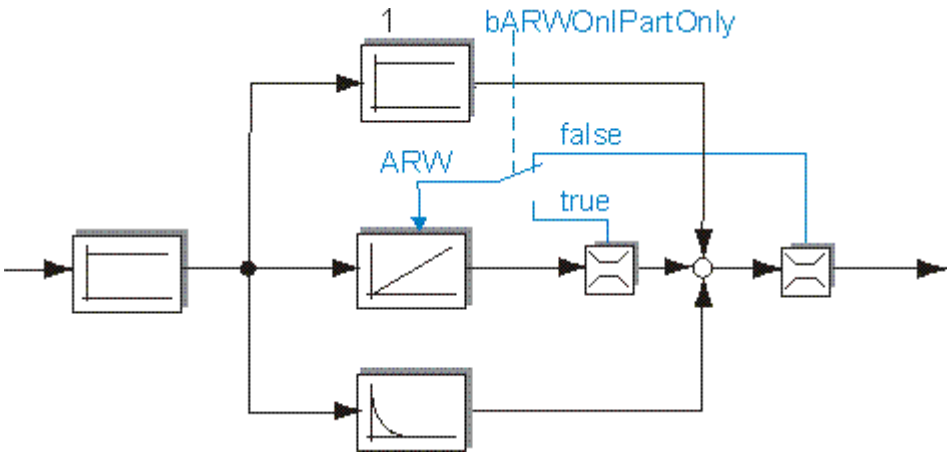
! Die Standardeinstellung für die beiden Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** ist **FALSE**. Der PID-Regler entspricht dann einem Standard PID-Regler in additiver Form.



Sprungantwort:



ARW:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    fManSyncValue       : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

fManSyncValue : Eingang, auf dessen Wert der interne Zustand des PID-Glieds gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PID-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Regelabweichung konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    bARWactive      : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des PID-Glieds.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PID-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms;     (* controller
cycle time [TIME] *)
    tTaskCycleTime     : TIME     := T#0ms;     (* task cycle
time [TIME] *)
    fKp        : FLOAT    := 0;     (* proportional
gain *)
    tTn        : TIME     := T#0ms;     (* Tn *)
    tTv        : TIME     := T#0ms;     (* Tv *)
    tTd        : TIME     := T#0ms;     (* Td *)
    fOutMaxLimit       : FLOAT    :=  1E38;     (* maximum
output limit *)
    fOutMinLimit       : FLOAT    := -1E38;     (* minimum
output limit *)
    bPInTheFeedbackPath : BOOL;
    bDInTheFeedbackPath : BOOL;
    bARWOnIPartOnly    : BOOL;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Reglerverstärkung / Reglerbeiwert

tTn : Nachstellzeit. Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv : Vorhaltzeit. Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd : Dämpfungszeit

fOutMaxLimit : Oberes Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fOutMinLimit : Unteres Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

bPInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des internen P-Glieds ausgewählt werden kann (siehe Wirkungsplan). Standardeinstellung: FALSE

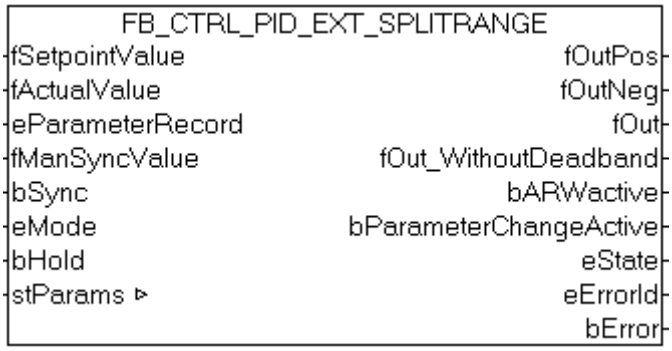
bDInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des internen D-Glieds ausgewählt werden kann (siehe Wirkungsplan). Standardeinstellung: FALSE

bARWOnIPartOnly: Wenn dieser Parameter FALSE ist (Standardeinstellung), wird die Integration des I-Anteil dann angehalten, wenn der gesamte Reglerausgang das obere oder untere Limit erreicht. Wenn dieser TRUE ist, wird die Integration dann angehalten, wenn der I-Anteil (der Integratorausgang) ein Limit erreicht. (Vgl. Wirkungsplan)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.10 FB_CTRL_PID_EXT_SPLITRANGE



Der Funktionsbaustein stellt ein erweitertes PID-Übertragungsglied im Wirkungsplan dar. Bei diesem Regler ist es möglich, dass bei aktiver Regelung zwischen zwei Parametersätzen umgeschaltet werden kann. Zusätzlich stehen die Funktionalitäten des Inner- und Outer-Window's sowie des Input- und Output-Deadband's zur Verfügung.

Beschreibung:

Bei diesem Funktionsbaustein handelt es sich um eine Erweiterung des FB_CTRL_PID_EXT, so dass der Regler für eine Strecke mit zwei Stelleinrichtungen eingesetzt werden kann, dessen Übertragungsverhalten unterschiedlich ist. Ein typischer Anwendungsfall ist eine Strecke mit einem Stellglied zum Heizen und einem Stellglied zum Kühlen. Um eine optimale Regelung einer solchen Anordnung zu ermöglichen kann zwischen zwei PID-Parametersätzen umgeschaltet werden. Die Parametersatzumschaltung erfolgt dabei so, dass sich auch während der Umschaltung eine stetige Stellgröße ergibt.

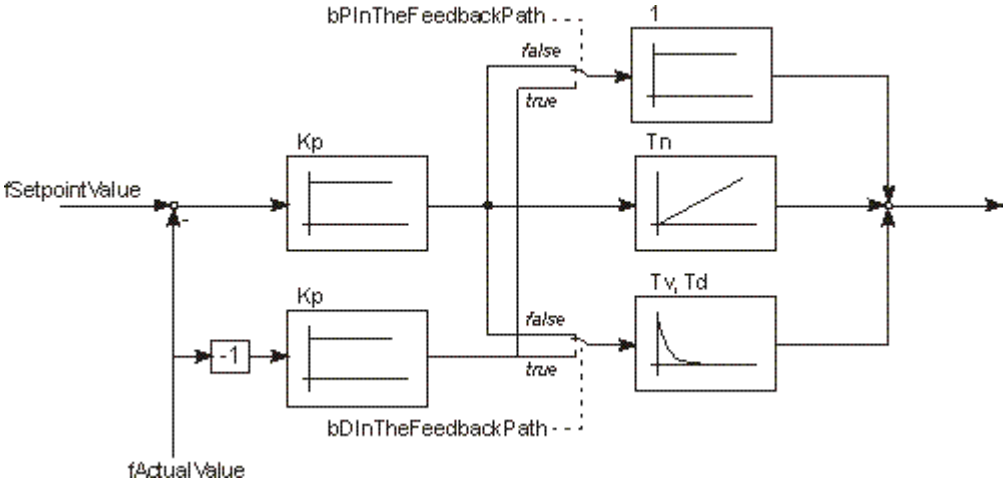
Der Umschaltalgorithmus berechnet eine lineare zeitabhängige Transition zwischen den beiden Parametersätzen. Mit dem Parameter nParameterChangeCycleTicks kann die Anzahl der Task-Zyklen vorgegeben werden, in denen stetig zwischen den Parametersätzen umgeschaltet wird.

Übertragungsfunktion:

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** FALSE sind, andernfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blockes:

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Wirkungsplan:



Der Standard Wirkungsplan eines PID-Reglers in additiver Form ist um die beiden als "Schalter" wirkenden boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** erweitert worden, so dass ein modifizierter Wirkungsplan aktiviert werden kann.

Der Regelungstechnische Hintergrund ist der, dass bei Sollwertsprüngen durch den Differential-Anteil des Regelalgorithmus große Stellgröße entstehen, die die Stellelemente belasten und das Regelsystem zum Schwingen anregen können. Ein Regelalgorithmus, dessen Differential-Anteil nur auf der Regelgröße angewendet wird (bDInTheFeedbackPath := TRUE), vermeidet dieses Problem.

Mit den Eingängen bPInTheFeedbackPath und bDInTheFeedbackPath können die folgenden Übertragungsfunktionen des geschlossenen Regelkreises realisiert werden:

bPInTheFeedbackPath	bDInTheFeedbackPath	$G(s)$
false	false	$G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \frac{G_I \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$

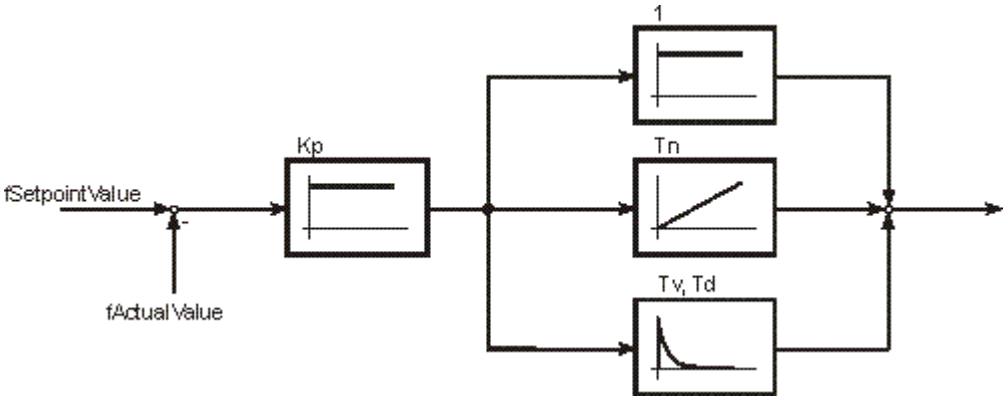
mit:

$$G_{DI}(s) = K_p(\frac{1}{T_n s} + \frac{T_v s}{1 + T_d s})$$

$$G_{PI}(s) = K_p(1 + \frac{1}{T_n s})$$

$$G_I(s) = K_p(\frac{1}{T_n s})$$

! Die Standardeinstellung für die beiden Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** ist **FALSE**. Der PID-Regler entspricht dann einem Standard PID-Regler in additiver Form.



Zusätzliche Funktionen:

Abschaltung des I-Anteils in dem OuterWindow

Wenn die Regelabweichung größer als der Parameter fOuterWindow ist, wird die Integration der Regelabweichung angehalten. Hiermit kann verhindert werden, dass sich bei einer großen Regelabweichung ein sehr großer I-Anteil aufbaut, der eventuell zu einem zu starken Überschwingen führt. Wenn diese Funktion nicht gewünscht ist, kann sie mit fOuterWindow:= 0 deaktiviert werden.

Lineare Reduzierung des I-Anteils in dem InnerWindow

Mit dieser Funktion ist es möglich, den I-Anteil in dem mit dem Parameter fInnerWindow festgelegten Bereich linear zu Null zu führen. Wenn diese Funktion nicht gewünscht ist, kann sie mit fInnerWindow := 0 deaktiviert werden.

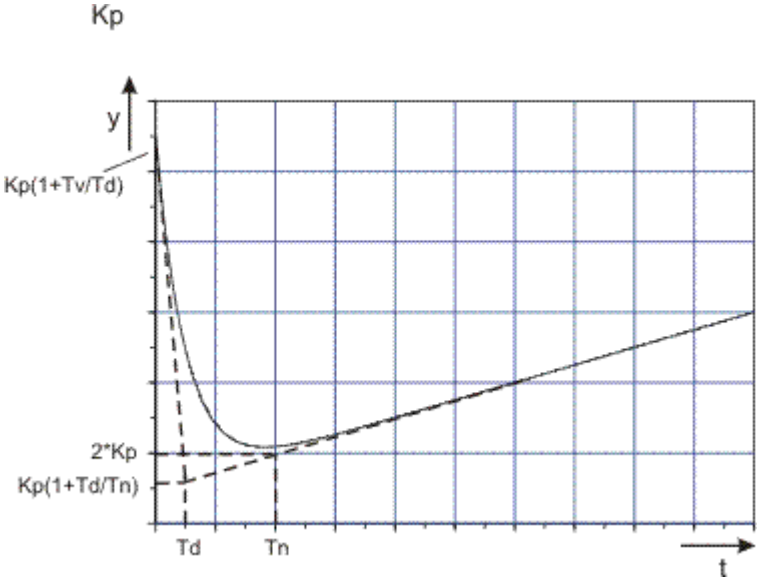
Totzone des Ausgangs

Wenn der Parameter fDeadBandOutput > 0 gesetzt wird, wird der Ausgang in dem Intervall [-fDeadBandOutput ... fDeadBandOutput] zu Null gesetzt.

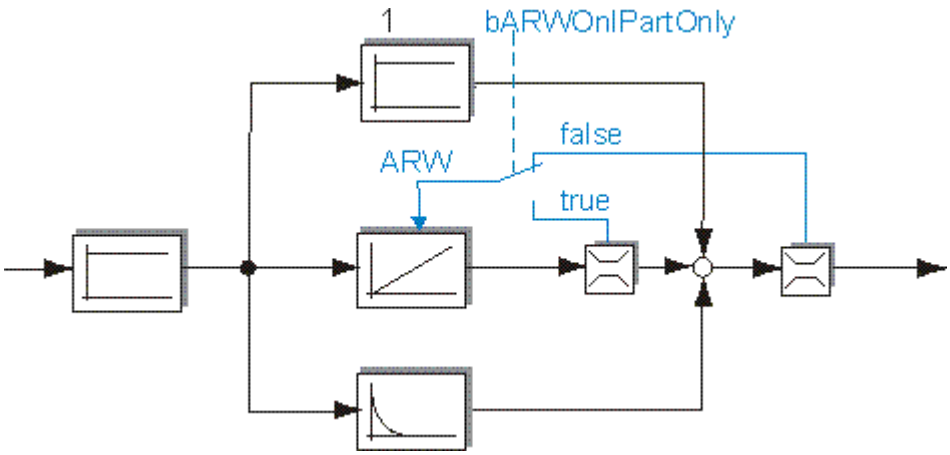
Totzone des Eingangs

Wenn der Parameter fDeadBandInput > 0 gesetzt wird, wird der Ausgang konstant gehalten, solange sich die Regelabweichung in dem Intervall [-fDeadBandInput ... fDeadBandInput] befindet.

Sprungantwort:



ARW:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    eParameterRecord    : E_CTRL_PARAMETER_RECORD;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

eParameterRecord : Index des aktiven Parametersatzes

fManSyncValue : Eingang, mit dem das PI-Glied gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Regelabweichung konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOutPos             : FLOAT;
    fOutNeg             : FLOAT;
    fOut            : FLOAT;

    bARWActive          : BOOL  := FALSE;       (* ARW active ? [TRUE/FALSE] -> freeze I-part *)
    bParameterChangeActive  : BOOL;

    bError          : BOOL;             (* error flag *)
    eErrorId            : E_CTRL_ERRORCODES;    (* error code *)

END_VAR
```

fOutPos : Ausgang des PID-Glieds, wenn die Stellgröße positiv ist. Anderenfalls wird eine Null ausgegeben.

fOutNeg : Ausgang des PID-Glieds, wenn die Stellgröße negativ ist. Anderenfalls wird eine Null ausgegeben.

fOut : Ausgang des PID-Glieds.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

bParameterChangeActive : Ein TRUE an diesem Ausgang signalisiert, dass ein Wechsel von einem zum anderen Parametersatz erfolgt.

fCtrlDerivation : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_SPLITRANGE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PID-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME     := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME     := T#0ms; (* task
cycle time [TIME] *)
    fKp_heating         : FLOAT    := 0;      (*
proportional gain *)
    tTn_heating         : TIME     := T#0ms; (* Tn
*)
    tTv_heating         : TIME     := T#0ms; (* Tv
*)
    tTd_heating         : TIME     := T#0ms; (* Td
*)
    fKp_cooling         : FLOAT    := 0;      (*
proportional gain *)
    tTn_cooling         : TIME     := T#0ms; (* Tn
*)
    tTv_cooling         : TIME     := T#0ms; (* Tv
*)
    tTd_cooling         : TIME     := T#0ms; (* Td
*)
    nParameterChangeCycleTicks : INT;
    fDeadBandInput          : REAL     := 0.0; (* ctrl
deviation dead band/neutral zone for const output *)
    fDeadBandOutput         : REAL     := 0.0; (* ctrl
output dead band/neutral zone for zero output *)
    fInnerWindow            : REAL     := 0.0; (* inner
window for reduced I-part (dE-window) *)
    fOuterWindow            : REAL     := 0.0; (* outer
window for disabling I-part (dE-window) *)
    fOutMaxLimit            : FLOAT    :=  1E38; (* maximum
output limit *)
    fOutMinLimit            : FLOAT    := -1E38; (* minimum
output limit *)
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

Bereich eCTRL_PARAMETER_RECORD_**HEATING**:

fKp_heating : Reglerverstärkung / Reglerbeiwert

tTn_heating : Nachstellzeit. Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv_heating : Vorhaltzeit. Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd_heating : Dämpfungszeit

Bereich eCTRL_PARAMETER_RECORD_**COOLING:**

fKp_cooling : Reglerverstärkung / Reglerbeiwert

tTn_cooling : Nachstellzeit. Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv_cooling : Vorhaltzeit. Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd_cooling : Dämpfungszeit

nParameterChangeCycleTicks: Anzahl der Task-Zyklen, in denen von einem Parametersatz zum anderen gewechselt wird.

fDeadBandInput : siehe obige Beschreibung

fDeadBandOutput: siehe obige Beschreibung

fInnerWindow : siehe obige Beschreibung

fOuterWindow : siehe obige Beschreibung

fOutMaxLimit : Oberes Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fOutMinLimit : Unteres Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

bPInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des P-Glieds ausgewählt werden kann (siehe Wirkungsplan).

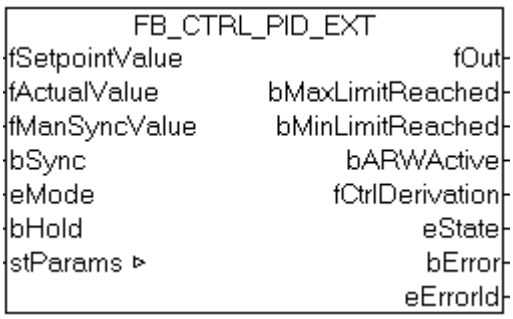
bDInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des D-Glieds ausgewählt werden kann (siehe Wirkungsplan).

bARWOnIPartOnly: Wenn dieser Parameter FALSE ist (Standardeinstellung), wird die Integration des I-Anteil dann angehalten, wenn der gesamte Reglerausgang das obere oder untere Limit erreicht.
Wenn dieser TRUE ist, wird die Integration dann angehalten, wenn der I-Anteil (der Integratorausgang) ein Limit erreicht. (Vgl. Wirkungsplan)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.11 FB_CTRL_PID_EXT



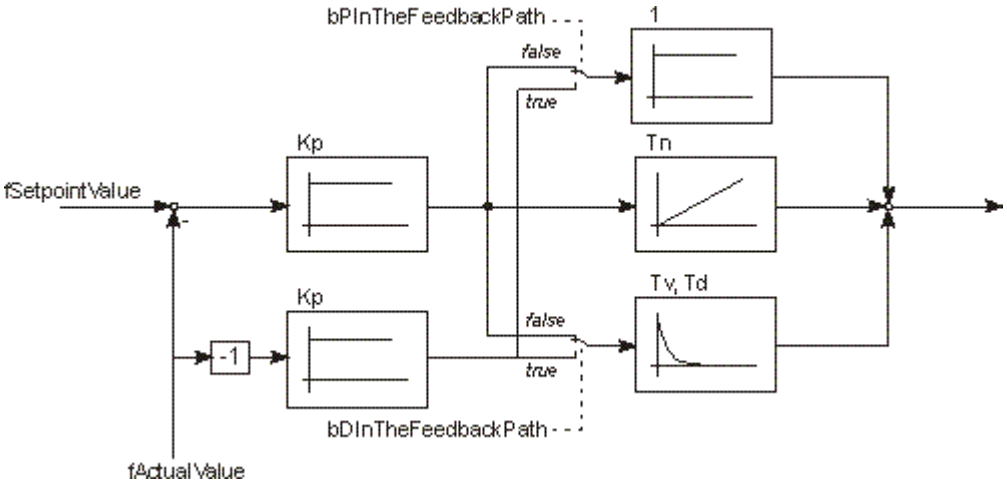
Der Funktionsbaustein stellt ein erweitertes PID-Glied im Wirkungsplan dar.

Übertragungsfunktion:

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** FALSE sind, andernfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blockes:

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Wirkungsplan:



Der Standard Wirkungsplan eines PID-Reglers in additiver Form ist um die beiden als "Schalter" wirkenden boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** erweitert worden, so dass ein modifizierter Wirkungsplan aktiviert werden kann.

Der Regelungstechnische Hintergrund ist der, dass bei Sollwertsprüngen durch den Differential-Anteil des Regelalgorithmus große Stellgröße entstehen, die die Stellelemente belasten und das Regelsystem zum Schwingen anregen können. Ein Regelalgorithmus, dessen Differential-Anteil nur auf der Regelgröße angewendet wird (bDInTheFeedbackPath := TRUE), vermeidet dieses Problem.

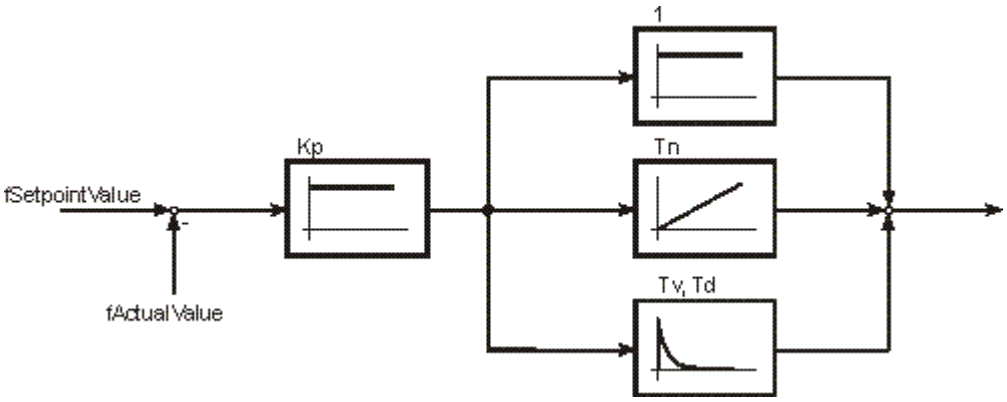
Mit den Eingängen bPInTheFeedbackPath und bDInTheFeedbackPath können die folgenden Übertragungsfunktionen des geschlossenen Regelkreises realisiert werden:

bPInTheFeedbackPath	bDInTheFeedbackPath	$G(s)$
false	false	$G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \frac{G_I \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$

mit:

$$G_{DI}(s) = K_p(\frac{1}{T_n s} + \frac{T_v s}{1 + T_d s})$$
$$G_{PI}(s) = K_p(1 + \frac{1}{T_n s})$$
$$G_I(s) = K_p(\frac{1}{T_n s})$$

! Die Standardeinstellung für die beiden Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** ist **FALSE**. Der PID-Regler entspricht dann einem Standard PID-Regler in additiver Form.



Zusätzliche Funktionen:

Abschaltung des I-Anteils in dem OuterWindow

Wenn die Regelabweichung größer als der Parameter fOuterWindow ist, wird die Integration der Regelabweichung angehalten. Hiermit kann verhindert werden, dass sich bei einer großen Regelabweichung ein sehr großer I-Anteil aufbaut, der eventuell zu einem zu starken Überschwingen führt. Wenn diese Funktion nicht gewünscht ist, kann sie mit fOuterWindow := 0 deaktiviert werden.

Lineare Reduzierung des I-Anteils in dem InnerWindow

Mit dieser Funktion ist es möglich, den I-Anteil in dem mit dem Parameter fInnerWindow festgelegten Bereich linear zu Null zu führen. Wenn diese Funktion nicht gewünscht ist, kann sie mit fInnerWindow := 0 deaktiviert werden.

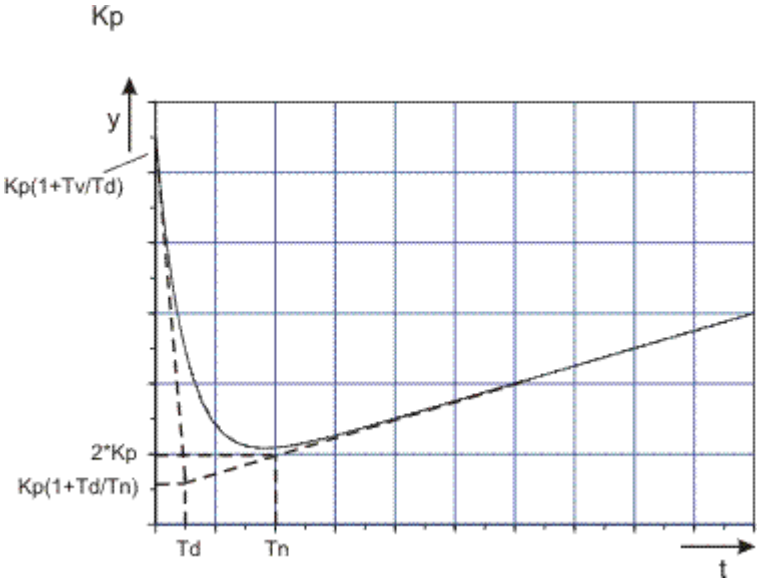
Totzone des Ausgangs

Wenn der Parameter fDeadBandOutput > 0 gesetzt wird, wird der Ausgang in dem Intervall [-fDeadBandOutput ... fDeadBandOutput] zu Null gesetzt.

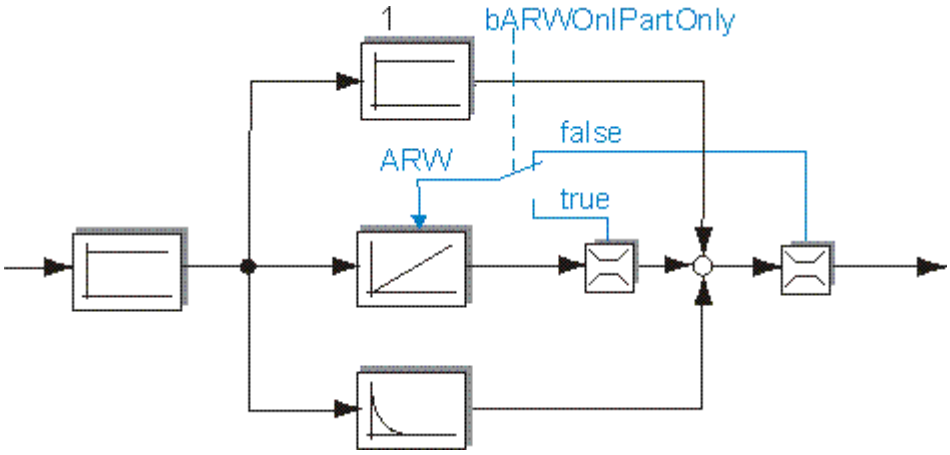
Totzone des Eingangs

Wenn der Parameter fDeadBandInput > 0 gesetzt wird, wird der Ausgang konstant gehalten, solange sich die Regelabweichung in dem Intervall [-fDeadBandInput ... fDeadBandInput] befindet.

Sprungantwort:



ARW:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

fManSyncValue : Eingang, mit dem das PID-Glied gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PID-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Regelabweichung konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;

    bMaxLimitReached    : BOOL  := FALSE;       (* minimum limitation active ? [TRUE/FALSE] -
> ARW *)
    bMinLimitReached    : BOOL  := FALSE;       (* maximum limitation active ? [TRUE/FALSE] -
> ARW *)
    bARWActive      : BOOL  := FALSE;       (* ARW active ? [TRUE/FALSE] -> freeze I-part *)

    fCtrlDerivation     : FLOAT;            (* controller command derivation dy/dt *)

    eState      : E_CTRL_STATE;
    bError      : BOOL;             (* error flag *)
    eErrorId        : E_CTRL_ERRORCODES;    (* error code *)

END_VAR
```

fOut : Ausgang des PID-Glieds.

bMaxLimitReached : Der Ausgang ist TRUE, wenn sich der Baustein in der oberen Begrenzung befindet.

bMinLimitReached : Der Ausgang ist TRUE, wenn sich der Baustein in der unteren Begrenzung befindet.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

fCtrlDerivation : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_EXT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PID-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_EXT_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME      := T#0ms;       (* controller
cycle time [TIME] *)
    tTaskCycleTime      : TIME      := T#0ms;       (* task cycle
time [TIME] *)
    fKp         : FLOAT     := 0;       (* proportional
gain *)
    tTn         : TIME      := T#0ms;       (* Tn *)
    tTv         : TIME      := T#0ms;       (* Tv *)
    tTd         : TIME      := T#0ms;       (* Td *)
    fDeadBandInput      : REAL      := 0.0;         (* ctrl
deviation dead band/neutral zone for const output *)
    fDeadBandOutput     : REAL      := 0.0;         (* ctrl output
dead band/neutral zone for zero output *)
    fInnerWindow        : REAL      := 0.0;         (* inner window
for reduced I-part (dE-window) *)
    fOuterWindow        : REAL      := 0.0;         (* outer window
for disabling I-part (dE-window) *)
    fOutMaxLimit        : FLOAT     :=  1E38;       (* maximum
output limit *)
    fOutMinLimit        : FLOAT     := -1E38;       (* minimum
output limit *)
    bPInTheFeedbackPath : BOOL;
    bDInTheFeedbackPath : BOOL;
    bARWOnIPartOnly     : BOOL;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Reglerverstärkung / Reglerbeiwert

tTn : Nachstellzeit. Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv : Vorhaltzeit. Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd : Dämpfungszeit

fDeadBandInput : siehe obige Beschreibung

fDeadBandOutput: siehe obige Beschreibung

fInnerWindow : siehe obige Beschreibung

fOuterWindow : siehe obige Beschreibung

fOutMaxLimit : Oberes Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fOutMinLimit : Unteres Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

bPInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des P-Glieds ausgewählt werden kann (siehe Wirkungsplan).

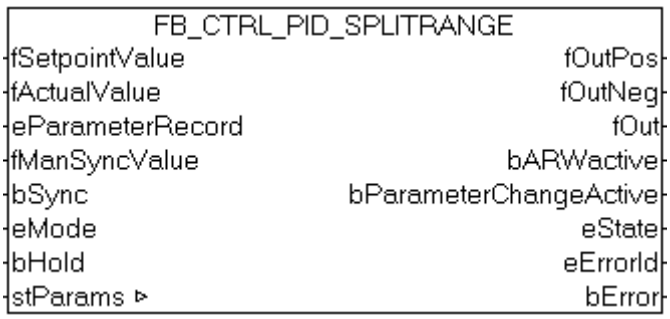
bDInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des D-Glieds ausgewählt werden kann (siehe Wirkungsplan).

bARWOnIPartOnly: Wenn dieser Parameter FALSE ist (Standardeinstellung), wird die Integration des I-Anteil dann angehalten, wenn der gesamte Reglerausgang das obere oder untere Limit erreicht.
Wenn dieser TRUE ist, wird die Integration dann angehalten, wenn der I-Anteil (der Integratorausgang) ein Limit erreicht. (Vgl. Wirkungsplan)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.4.12 FB_CTRL_PID_SPLITRANGE



Der Funktionsbaustein stellt ein erweitertes PID-Übertragungsglied im Wirkungsplan dar. Bei diesem Regler ist es möglich, dass bei aktiver Regelung zwischen zwei Parametersätzen umgeschaltet werden kann.

Beschreibung:

Bei diesem Funktionsbaustein handelt es sich um eine Erweiterung des FB_CTRL_PID, so dass der Regler für eine Strecke mit zwei Stelleinrichtungen eingesetzt werden kann, dessen Übertragungsverhalten unterschiedlich ist. Ein typischer Anwendungsfall ist eine Strecke mit einem Stellglied zum Heizen und einem Stellglied zum Kühlen. Um eine optimale Regelung einer solchen Anordnung zu ermöglichen kann zwischen zwei PID-Parametersätzen umgeschaltet werden. Die Parametersatzumschaltung erfolgt dabei so, dass sich auch während der Umschaltung eine stetige Stellgröße ergibt.

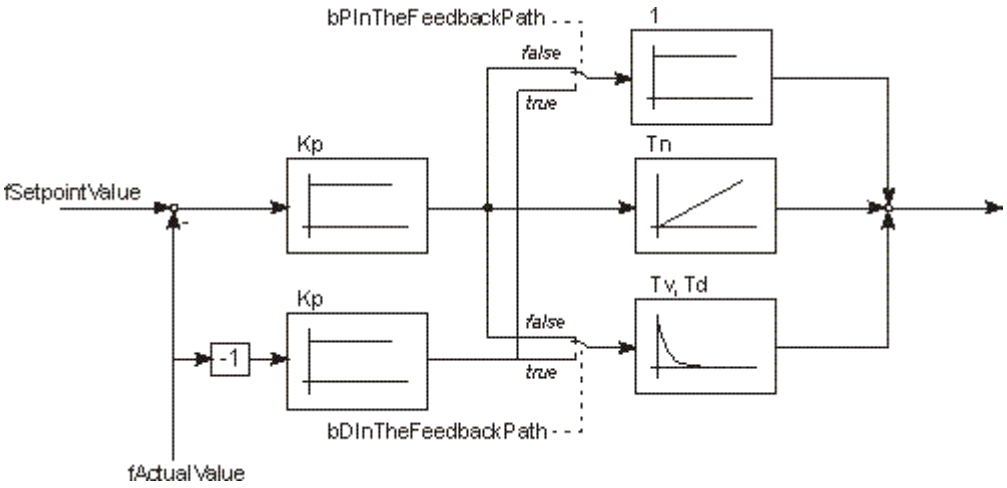
Der Umschaltalgorithmus berechnet eine lineare zeitabhängige Transition zwischen den beiden Parametersätzen. Mit dem Parameter nParameterChangeCycleTicks kann die Anzahl der Task-Zyklen vorgegeben werden, in denen stetig zwischen den Parametersätzen umgeschaltet wird.

Übertragungsfunktion:

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** FALSE sind, andernfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blockes:

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Wirkungsplan:



Der Standard Wirkungsplan eines PID-Reglers in additiver Form ist um die beiden als "Schalter" wirkenden boolschen Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** erweitert worden, so dass ein modifizierter Wirkungsplan aktiviert werden kann.

Der Regelungstechnische Hintergrund ist der, dass bei Sollwertsprüngen durch den Differential-Anteil des Regelalgorithmus große Stellgröße entstehen, die die Stellelemente belasten und das Regelsystem zum Schwingen anregen können. Ein Regelalgorithmus, dessen Differential-Anteil nur auf der Regelgröße angewendet wird (bDInTheFeedbackPath := TRUE), vermeidet dieses Problem.

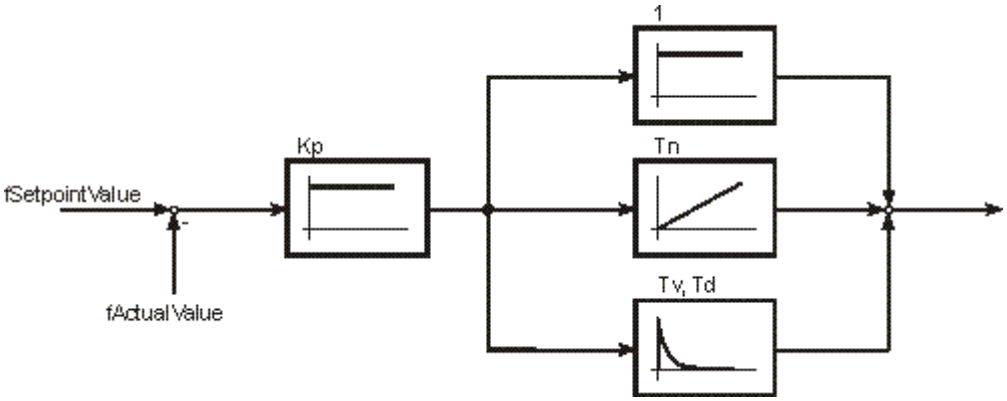
Mit den Eingängen bPInTheFeedbackPath und bDInTheFeedbackPath können die folgenden Übertragungsfunktionen des geschlossenen Regelkreises realisiert werden:

bPInTheFeedbackPath	bDInTheFeedbackPath	$G(s)$
false	false	$G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \frac{G_I \cdot G_S(s)}{1+G_{PID}(s) \cdot G_S(s)}$

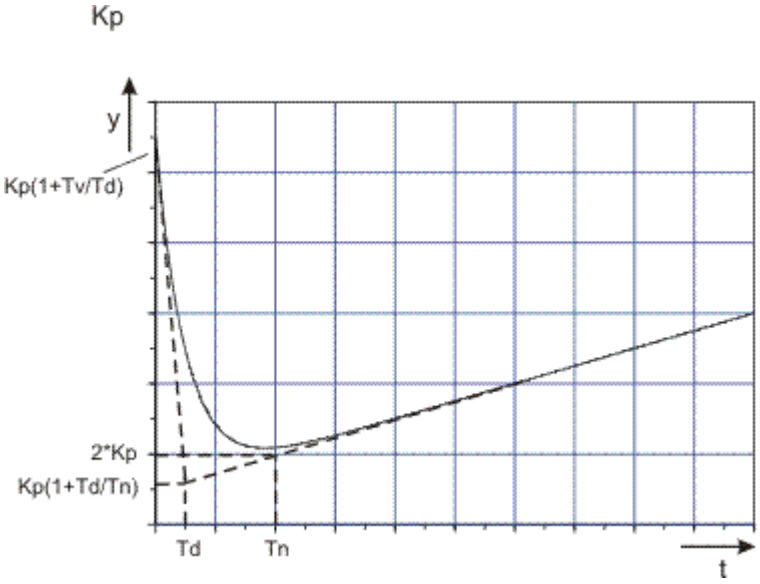
mit:

$$G_{DI}(s) = K_p(\frac{1}{T_n s} + \frac{T_v s}{1+T_d s})$$
$$G_{PI}(s) = K_p(1 + \frac{1}{T_n s})$$
$$G_I(s) = K_p(\frac{1}{T_n s})$$

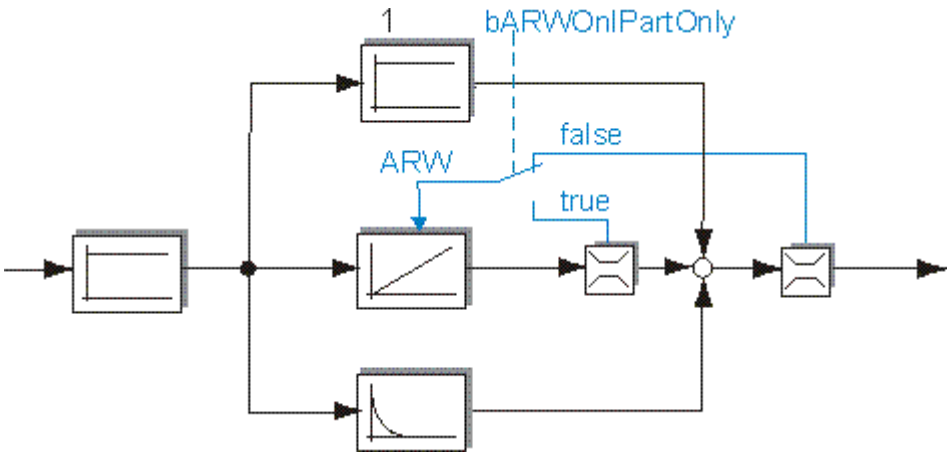
! Die Standardeinstellung für die beiden Eingänge **bPInTheFeedbackPath** und **bDInTheFeedbackPath** ist **FALSE**. Der PID-Regler entspricht dann einem Standard PID-Regler in additiver Form.



Sprungantwort:



ARW:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    eParameterRecord        : E_CTRL_PARAMETER_RECORD;
    fManSyncValue       : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

eParameterRecord : Index des aktiven Parametersatzes

fManSyncValue : Eingang, mit dem das PI-Glied gesetzt werden kann.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang, unabhängig von der Regelabweichung, konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOutPos             : FLOAT;
    fOutNeg             : FLOAT;
    fOut            : FLOAT;

    bARWActive          : BOOL  := FALSE;       (* ARW active ? [TRUE/FALSE] -> freeze I-part *)
    bParameterChangeActive  : BOOL;

    eState          : E_CTRL_STATE;
    bError          : BOOL;             (* error flag *)
    eErrorId            : E_CTRL_ERRORCODES;    (* error code *)

END_VAR
```

fOutPos : Ausgang des PID-Glieds, wenn die Stellgröße positiv ist. Anderenfalls wird eine Null ausgegeben.

fOutNeg : Ausgang des PID-Glieds, wenn die Stellgröße negativ ist. Anderenfalls wird eine Null ausgegeben.

fOut : Ausgang des PID-Glieds.

bARWactive : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

bParameterChangeActive : Ein TRUE an diesem Ausgang signalisiert, dass ein Wechsel von einem zum anderen Parametersatz erfolgt.

fCtrlDerivation : Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_SPLITRANGE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PID-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME      := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME      := T#0ms; (* task
cycle time [TIME] *)
    fKp_heating         : FLOAT     := 0;       (*
proportional gain *)
    tTn_heating         : TIME      := T#0ms; (* Tn
*)
    tTv_heating         : TIME      := T#0ms; (* Tv
*)
    tTd_heating         : TIME      := T#0ms; (* Td
*)
    fKp_cooling         : FLOAT     := 0;       (*
proportional gain *)
    tTn_cooling         : TIME      := T#0ms; (* Tn
*)
    tTv_cooling         : TIME      := T#0ms; (* Tv
*)
    tTd_cooling         : TIME      := T#0ms; (* Td
*)
    nParameterChangeCycleTicks : INT;
    fOutMaxLimit            : FLOAT     :=  1E38; (* maximum
output limit *)
    fOutMinLimit            : FLOAT     := -1E38; (* minimum
output limit *)
    bPInTheFeedbackPath     : BOOL;
```

```
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

Bereich eCTRL_PARAMETER_RECORD_**HEATING**:

fKp_heating : Reglerverstärkung / Reglerbeiwert

tTn_heating : Nachstellzeit. Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv_heating : Vorhaltzeit. Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd_heating : Dämpfungszeit

Bereich eCTRL_PARAMETER_RECORD_**COOLING**:

fKp_cooling : Reglerverstärkung / Reglerbeiwert

tTn_cooling : Nachstellzeit. Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.

tTv_cooling : Vorhaltzeit. Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.

tTd_cooling : Dämpfungszeit

nParameterChangeCycleTicks: Anzahl der Task-Zyklen, in denen von einem zum anderen Parametersatz gewechselt wird.

fOutMaxLimit : Oberes Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

fOutMinLimit : Unteres Limit, an dem die Integration angehalten und der Ausgang begrenzt wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang bARWActive signalisiert.

bPInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des P-Glieds ausgewählt werden kann (siehe Wirkungsplan).

bDInTheFeedbackPath : Eingang, mit dem die Eingangsgröße des D-Glieds ausgewählt werden kann (siehe Wirkungsplan).

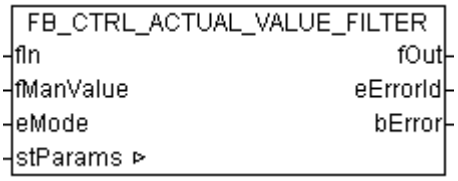
bARWOnIPartOnly: Wenn dieser Parameter FALSE ist (Standardeinstellung), wird die Integration des I-Anteil dann angehalten, wenn der gesamte Reglerausgang das obere oder untere Limit erreicht. Wenn dieser TRUE ist, wird die Integration dann angehalten, wenn der I-Anteil (der Integratorausgang) ein Limit erreicht. (Vgl. Wirkungsplan)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

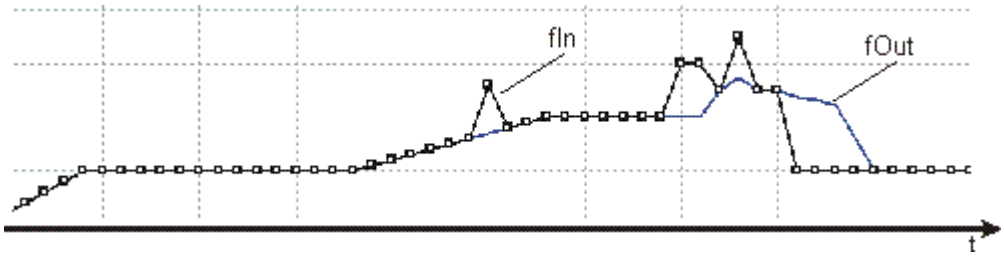
5.5 Filter / ControlledSystemSimulation

5.5.1 FB_CTRL_ACTUAL_VALUE_FILTER



Der Funktionsbaustein ermöglicht eine Plausibilitätskontrolle und Filterung einer gemessenen Eingangsgröße.

Verhalten des Ausgangs:



Mit diesem Baustein kann eine Plausibilitätskontrolle einer gemessenen Eingangsgröße vorgenommen werden. Wenn die Differenz zweier aufeinander folgender Abtastwerte (Messwerte) größer als die vorgegebene Schranke **fDeltaMax** ist, wird der aktuelle Eingangswert für maximal drei Zyklen unterdrückt. Während dieser Zeit wird der Ausgangswert aus den zurückliegenden Eingangswerten extrapoliert. Wenn die vorgegebene Schranke länger als 3 Zyklen überschritten wird, folgt der Ausgang wieder der Eingangsgröße. Das Verhalten des Ausgangs ist in dem obigen Bild dargestellt.

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingangsgröße des Filters.

fManValue : Eingangsgröße, dessen Wert im Manual-Mode ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des Funktionsbausteins.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Actual-Value-Filter-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
FB_CTRL_ACTUAL_VALUE_FILTER:
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fDeltaMax      :FLOAT;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

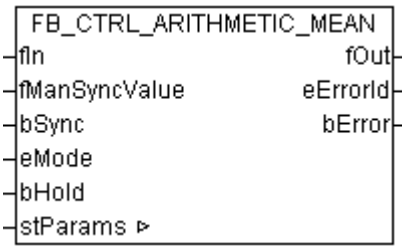
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fDeltaMax : Maximale Differenz zweier aufeinander folgender Eingangswerte. Siehe Beschreibung oben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.2 FB_CTRL_ARITHMETIC_MEAN



Berechnung des Mittelwertes:

$$\bar{x} = \frac{1}{n} \sum_{n} x_n$$

VAR_INPUT

```
VAR_INPUT
    fIn          : FLOAT;
    fManSyncValue    : FLOAT;
    bSync        : BOOL;
    eMode        : E_CTRL_MODE;
    bHold        : BOOL;
END_VAR
```

fIn : Eingangsgröße des Mittelwert-Filters.

fManSyncValue : Eingangsgröße, auf die das Mittelwert-Filter gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das Mittelwert-Filter auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut          : FLOAT;
    eState        : E_CTRL_STATE;
    eErrorId          : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

fOut : Ausgang des Mittelwert-Filters.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams         : ST_CTRL_ARITHMETIC_MEAN_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Mittelwertfilters. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_ARITHMETIC_MEAN_PARAMS:
STRUCT
    tCtrlCycleTime       : TIME     := T#0ms; (* controller
cycle time [TIME] *)
    tTaskCycleTime       : TIME     := T#0ms; (* task cycle
time [TIME] *)
END_STRUCT
END_TYPE
```

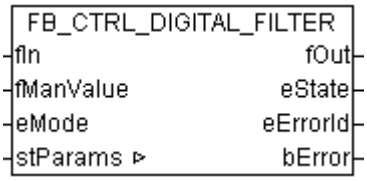
tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

Voraussetzungen

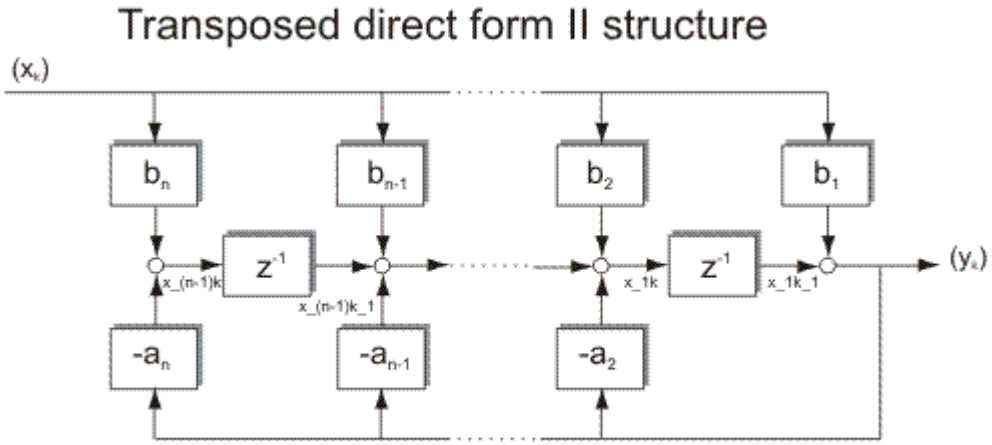
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.3 FB_CTRL_DIGITAL_FILTER



Dieser Baustein berechnet eine diskrete Übertragungsfunktion mit der unten dargestellten Struktur. Mit dieser Struktur ist sowohl die Realisierung eines FIR-Filters (**F**inite **I**mpulse **R**esponse) als auch die Realisierung eines IIR-Filters (**I**nfinite **I**mpulse **R**esponse) möglich. Die Übertragungsfunktion kann hierbei von beliebiger Ordnung n sein.

In den Parameter-Arrays werden die Koeffizienten der folgenden Übertragungsstruktur abgelegt:



order: n-1

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + \cdots + b_n z^{-(n-1)}}{1 + a_2 z^{-1} + a_3 z^{-2} + \cdots + a_n z^{-(n-1)}}$$

Um diesen Funktionsbaustein nutzen zu können, müssen vom Programmierer in der SPS die folgenden Arrays angelegt werden:

```
ar_CoefficientsArray_a     :
ARRAY[1..nFilterOrder+1] OF FLOAT;
ar_CoefficientsArray_b     : ARRAY[1..nFilterOrder+1] OF
FLOAT;
ar_stDigitalFilterData     : ARRAY[1..nFilterOrder  ] OF
ST_CTRL_DIGITAL_FILTER_DATA;
```

In dem Array ar_CoefficientsArray_b werden die Koeffizienten b_1 bis b_n abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
ar_CoefficientsArray_b[ 1 ]     := b1;
ar_CoefficientsArray_b[ 2 ]     := b2;
...
ar_CoefficientsArray_b[ n-1 ]   := bn-1;
ar_CoefficientsArray_b[ n ]     := bn;
```

In dem Array ar_CoefficientsArray_a werden die Koeffizienten a_1 bis a_n abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
ar_CoefficientsArray_a[ 1 ]    := xxx;    (* wird nicht ausgewertet
*)
ar_CoefficientsArray_a[ 2 ]    := a2;
...
ar_CoefficientsArray_a[ n-1 ]  := an-1;
ar_CoefficientsArray_a[ n ]    := an;
```

Die internen Daten, die der Baustein benötigt, werden in dem Array ar_stDigitalFilterData abgelegt. Diese Daten dürfen innerhalb des SPS-Programms keinesfalls geändert werden. Diese Vorgehensweise wird auch in dem unten aufgeführten Beispielprogramm dargestellt.

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingang des Digitalfilters.

fManValue : Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    bError      : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

fOut : Ausgang des Digitalfilters

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_DIGITAL_FILTER_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_DIGITAL_FILTER_PARAMS :
STRUCT
    tTaskCycleTime          : TIME;      (* task cycle
time        *)
    tCtrlCycleTime          : TIME := T#0ms; (* controller
cycle time *)
    nFilterOrder        : USINT;
    pCoefficientsArray_a_ADR    : POINTER TO FLOAT := 0;
    nCoefficientsArray_a_SIZEOF : UINT;
    pCoefficientsArray_b_ADR    : POINTER TO FLOAT := 0;
    nCoefficientsArray_b_SIZEOF : UINT;
    pDigitalFilterData_ADR      : POINTER TO
ST_CTRL_DIGITAL_FILTER_DATA;
    nDigitalFilterData_SIZEOF   : UINT;
END_STRUCT
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

nFilterOrder : Ordnung des Digitalfilters [0...]

pCoefficientsArray_a_ADR : Adresse des Arrays mit den Koeffizienten a.

nCoefficientsArray_a_SIZEOF :Größe des Arrays mit den Koeffizienten a in Byte.

pCoefficientsArray_b_ADR : Adresse des Arrays mit den Koeffizienten b.

nCoefficientsArray_b_SIZEOF : Größe des Arrays mit den Koeffizienten b in Byte.

pDigitalFilterData_ADR : Adresse des Daten-Arrays.

nDigitalFilterData_SIZEOF : Größe des Daten-Arrays in Byte.

```
TYPE
ST_CTRL_DIGITAL_FILTER_DATA
STRUCT
    Interne Struktur. Auf diese darf nicht schreibend
zugegriffen werden.
END_STRUCT
END_TYPE
```

Beispiel:

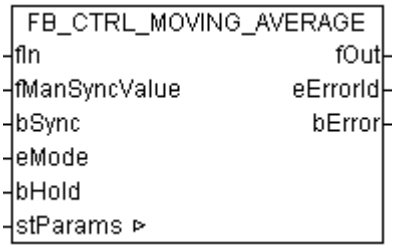
```
PROGRAM PRG_DIGITAL_FILTER_TEST
VAR
    fbDigitalFilter : FB_CTRL_DIGITAL_FILTER;
    ar_CoefficientsArray_a : ARRAY[1..3 ] OF FLOAT;
    ar_CoefficientsArray_b : ARRAY[1..3 ] OF FLOAT;
    ar_stDigitalFilterData : ARRAY[1..2 ] OF
ST_CTRL_DIGITAL_FILTER_DATA;
    eMode     : E_CTRL_MODE;
    stParams  : ST_CTRL_DIGITAL_FILTER_PARAMS;
    eErrorId  : E_CTRL_ERRORCODES;
    bError    : BOOL;
    fIn       : FLOAT := 0;
    fOut      : FLOAT;
    bInit     : BOOL := TRUE;
END_VAR
```

```
IF bInit
THEN
    (* set values in the local arrays *)
    ar_CoefficientsArray_a[1] := 0.0;    (* not used *)
    ar_CoefficientsArray_a[2] := 0.2;
    ar_CoefficientsArray_a[3] := 0.1;
    ar_CoefficientsArray_b[1] := 0.6;
    ar_CoefficientsArray_b[2] := 0.4;
    ar_CoefficientsArray_b[3] := 0.2;
    (* set values in the parameter struct *)
    stParams.tTaskCycleTime := T#2ms;
    stParams.tCtrlCycleTime := T#2ms;
    stParams.nFilterOrder   := 2;
    (* set the mode *)
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
(* set addresses *)
stParams.pCoefficientsArray_a_ADR    := ADR(
ar_CoefficientsArray_a);
stParams.nCoefficientsArray_a_SIZEOF := SIZEOF(
ar_CoefficientsArray_a);
stParams.pCoefficientsArray_b_ADR    := ADR(
ar_CoefficientsArray_b);
stParams.nCoefficientsArray_b_SIZEOF := SIZEOF(
ar_CoefficientsArray_b);
stParams.pDigitalFilterData_ADR      := ADR( ar_stDigitalFilterData
);
stParams.nDigitalFilterData_SIZEOF   := SIZEOF(
ar_stDigitalFilterData );
fbDigitalFilter ( fIn       := fIn,
          eMode     := eMode,
          stParams  := stParams,
          fOut      => fOut,
          eErrorId  => eErrorId,
          bError    => bError);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.4 FB_CTRL_MOVING_AVERAGE



Der Funktionsbaustein stellt ein gleitendes Mittelwertfilter im Wirkungsplan dar.

Es wird der arithmetische Mittelwert aus den letzten n Werten gebildet.

$$\bar{x}_k^n = \frac{1}{n} \sum_{i=k-n}^{k} x_i$$

Von dem Programmierer muss ein Array: ARRAY [1.. n] of FLOAT angelegt werden, in dem der Funktionsbaustein die intern benötigten Daten ablegen kann.

VAR_INPUT

```
VAR_INPUT
    fIn          : FLOAT;
    fManSyncValue    : FLOAT;
    bSync        : BOOL;
    eMode        : E_CTRL_MODE;
    bHold        : BOOL;
END_VAR
```

fIn : Eingangsgröße des Moving-Average-Filters.

fManSyncValue : Eingangsgröße, auf die das Moving-Average-Filter gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das Moving-Average-Filter auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    eErrorId         : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

fOut : Ausgang des Moving-Average-Filters.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_MOVING_AVERAGE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Moving-Average-Filters. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_MOVING_AVERAGE_PARAMS:
STRUCT
    tCtrlCycleTime       : TIME     := T#0ms; (* controller
cycle time [TIME] *)
    tTaskCycleTime       : TIME     := T#0ms; (* task cycle
time [TIME] *)
    nSamplesToFilter     : UINT;
    pWorkArray_ADR       : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF    : UINT         := 0;
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

nSamplesToFilter : Anzahl der Werte n, über die der arithmetische Mittelwert gebildet wird.

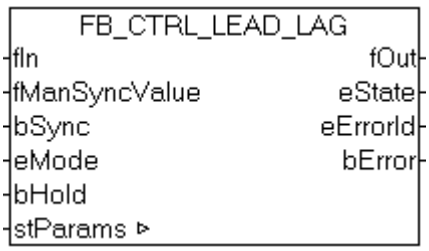
pWorkArray_ADR : Adresse des Arrays, indem die Eingangswerte zwischengespeichert werden.

nWorkArray_SIZEOF: Größe des WorkArrays.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.5 FB_CTRL_LEAD_LAG

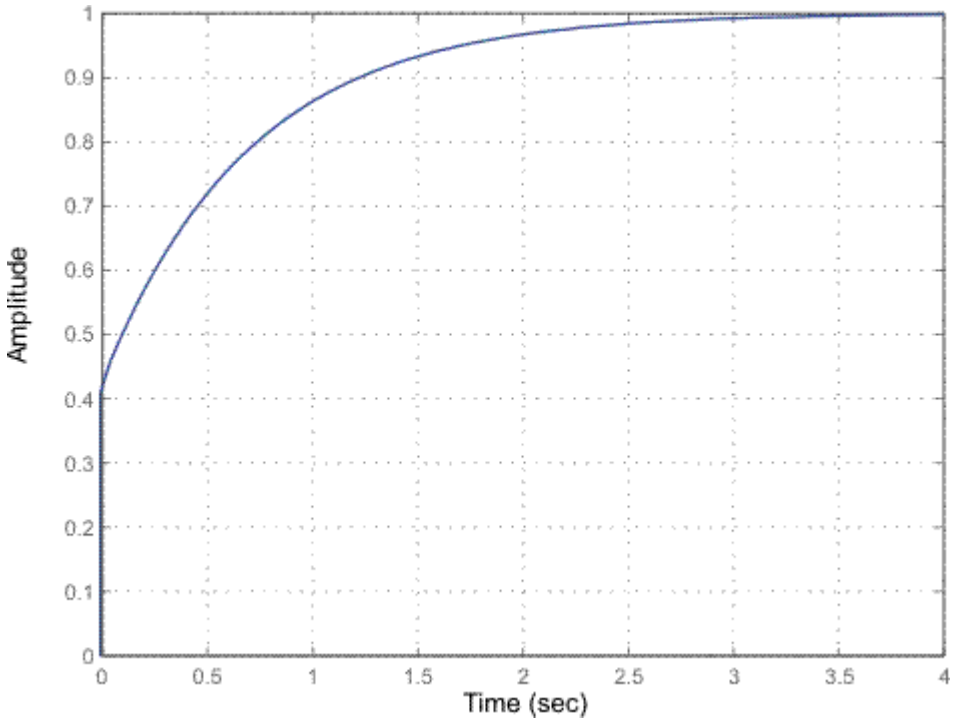


Der Funktionsbaustein stellt ein digitales Lead-Lag-Filter dar.

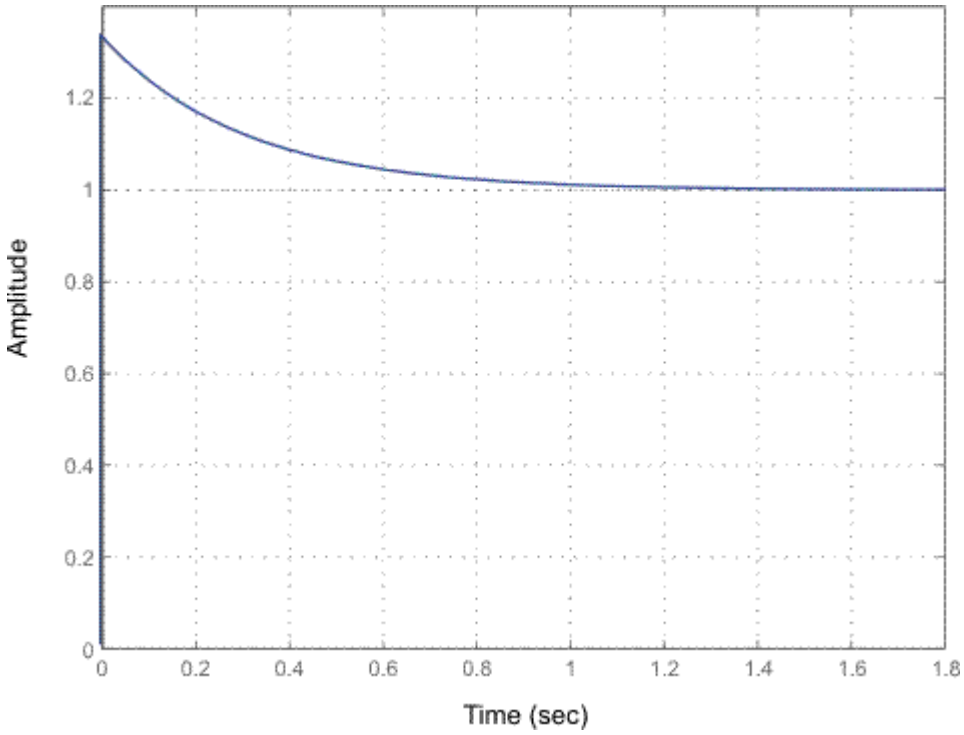
Übertragungsfunktion:

$$G(s) = \frac{1 + T_1 s}{1 + T_2 s}$$

Sprungantwort mit T2 > T1:



Sprungantwort mit T1 > T2:



Die Höhe der Sprungantwort zum Zeitpunkt T=0 ergibt sich zu T1 / T2.

VAR_INPUT

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : FLOAT;
    fManValue       : FLOAT;
    eMode           : E_CTRL_MODE;
END_VAR
```

fIn : Eingangsgröße des Notch-Filters.

fManSyncValue : Eingangsgröße die im Manual-Mode am Ausgang ausgegeben wird.

bSync : Reserve.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut          : FLOAT;
    eState        : E_CTRL_STATE;
    eErrorId          : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

fOut : Ausgang des Lead-Lag-Filters.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].p>

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams         : ST_CTRL_LEAD_LAG_FILTER_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Lead-Lag-Filters. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_LEAD_LAG_FILTER_PARAMS:STRUCT    tCtrlCycleTime : TIME :=
T#0ms; (* controller cycle time [TIME] *)    tTaskCycleTime : TIME :=
T#0ms; (* task cycle time [TIME] *)    tT1         : TIME :=
T#0ms; (* T1 *)

    tT2         : TIME := T#0ms; (* T2 *)END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

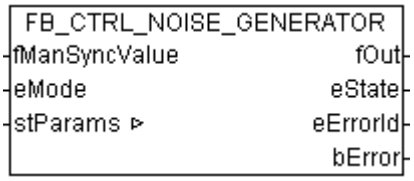
tT1 : T1, siehe G(s)

tT2 : T2, siehe G(s)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.6 FB_CTRL_NOISE_GENERATOR (nur auf einem PC-System)



Der Funktionsbaustein generiert ein Rauschsignal auf der Basis einer Pseudo-Zufallszahl im Intervall [-fAmplitude ... fAmplitude] .

Ausgangssignal:

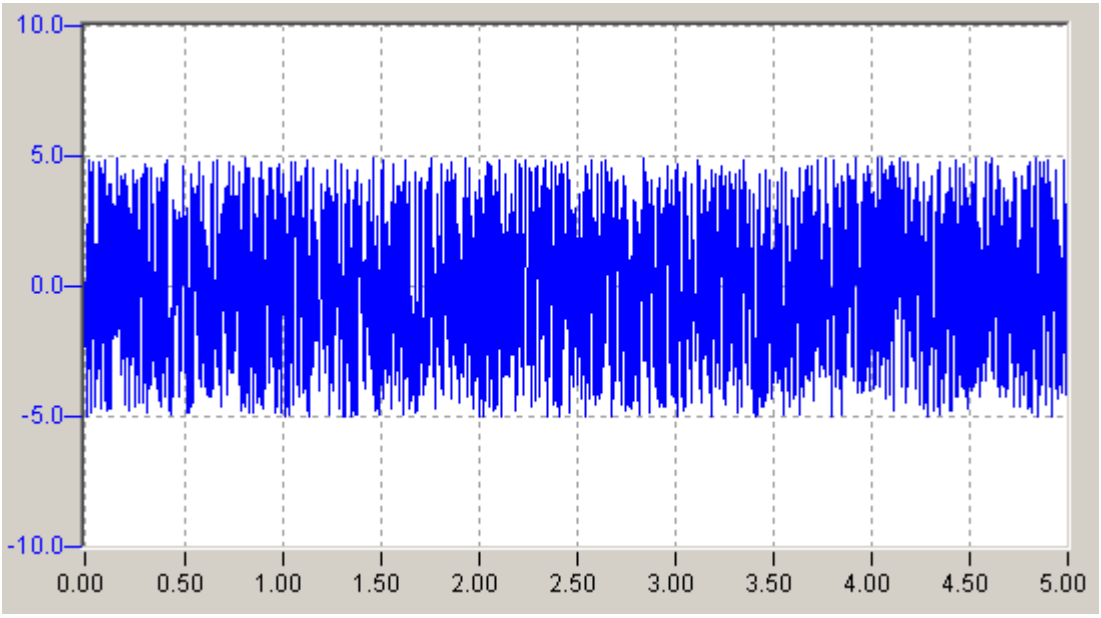


Abb. 3: FB_CTRL_NOISE_GENERATOR_OUT

Ausgangssignal bei einer Amplitude von 5.0.

VAR_INPUT

```
VAR_INPUT
    fManSyncValue    : FLOAT;
    eMode        : E_CTRL_MODE;
END_VAR
```

fManSyncValue : Eingangsgröße, die im Manual-Mode am Ausgang ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    eErrorId         : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

fOut : Ausgang des Rauschgenerators.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_NOISE_GENERATOR_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Rauschgenerators. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_NOISE_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime : TIME     := T#0ms; (* task cycle time
[TIME]       *)
    fAmplitude     : FLOAT    := 0;
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

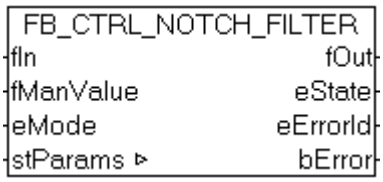
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fAmplitude : Amplitude des Ausgangssignals. Es wird am Ausgang des Funktionsbausteins ein Rauschsignal im Intervall [-fAmplitude/2.0 ... fAmplitude/2.0] erzeugt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

5.5.7 FB_CTRL_NOTCH_FILTER

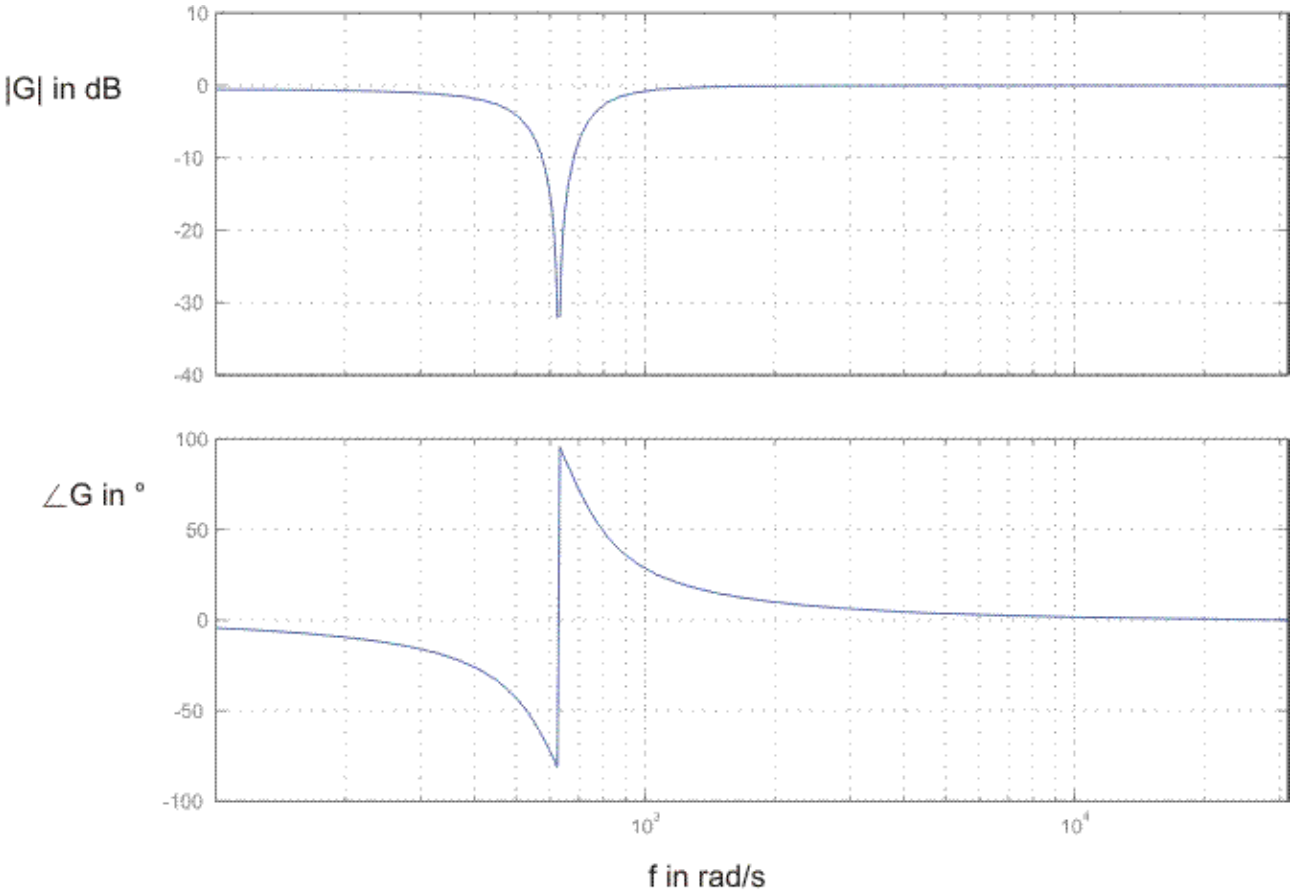


Der Funktionsbaustein stellt ein digitales Notch-Filter dar.

Übertragungsfunktion:

$$G(s) = \frac{s^2 - \omega_0^2}{(s - i\epsilon\omega_0)^2 - \omega_0^2}$$

Bodediagramm:



mit:

$$f_0 = 100\text{Hz}$$
$$\epsilon = 0.25$$

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingangsgröße des Notch-Filters.

fManValue : Eingangsgröße die im Manual-Mode am Ausgang ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut          : FLOAT;
    eState        : E_CTRL_STATE;
    eErrorId          : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

fOut : Ausgang des Notch-Filters.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_NOTCH_FILTER_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Notch-Filters. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_NOTCH_FILTER_PARAMS:STRUCT    tCtrlCycleTime : TIME :=
T#0ms; (* controller cycle time [TIME] *)    tTaskCycleTime : TIME :=
T#0ms; (* task cycle time [TIME] *)    fNotchFreq : FLOAT := 0;
(* [Hz] *)    fBandwidth : FLOAT := 0;
(* Bandwidth in Hz = fBandwidth*fNotchFreq *)END_STRUCTEND_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

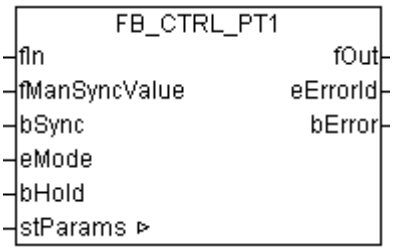
fNotchFreq : Notch-Frequenz in Hz

fBandwidth : Bandbreite bezogen auf die Notchfrequenz: Bandbreite in Hz = fNotchFreq * fBandwidth

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.8 FB_CTRL_PT1

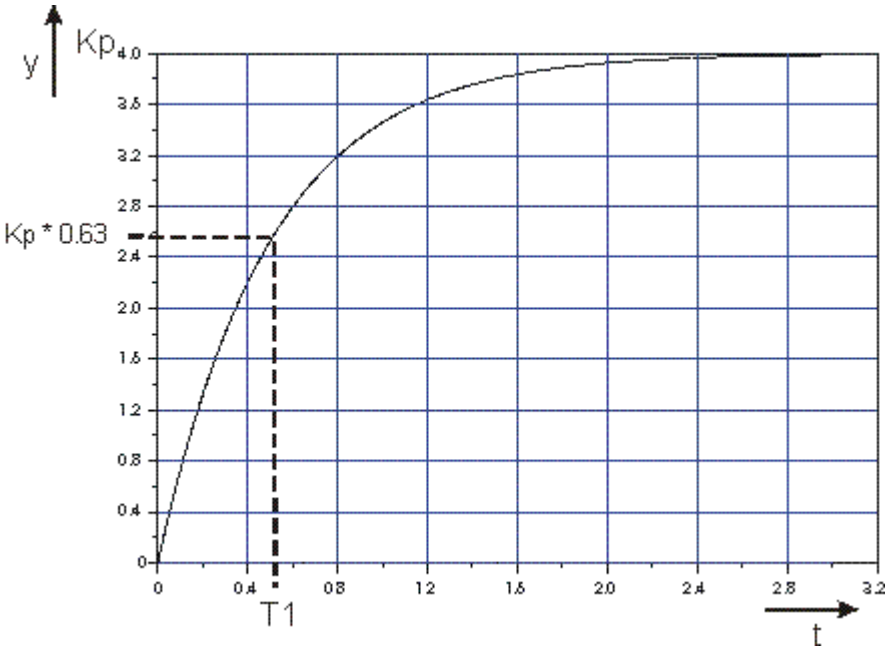


Der Funktionsbaustein stellt ein PT1-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \frac{1}{1 + T_1 s}$$

Sprungantwort:



VAR_INPUT

```
VAR_INPUT
    fIn          : FLOAT;
    fManSyncValue    : FLOAT;
    bSync        : BOOL;
    eMode        : E_CTRL_MODE;
    bHold        : BOOL;
END_VAR
```

fIn : Eingangsgröße des PT1-Glieds.

fManSyncValue : Eingangsgröße, auf die das PT1-Glied gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PT1-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    eErrorId         : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

fOut : Ausgang des PT1-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT1_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PT1-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT1_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime  : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fKp         : FLOAT    := 0;     (* proportional gain
*)
    tT1         : TIME     := T#0ms; (* T1 *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

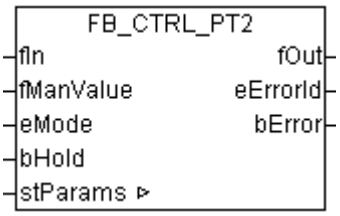
fKp : Reglerverstärkung / Übertragungsbeiwert

tT1 : Zeitkonstante

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.9 FB_CTRL_PT2

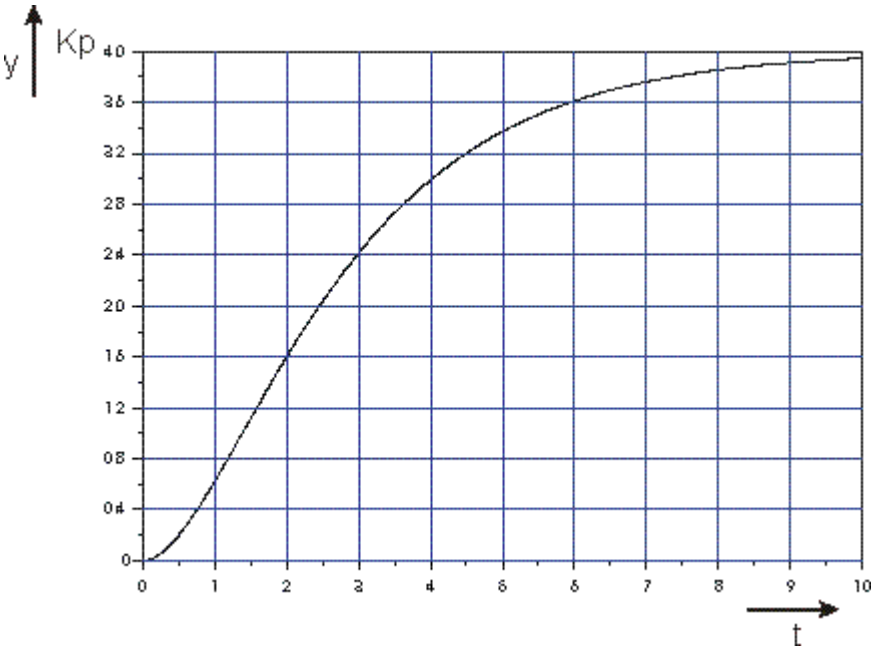


Der Funktionsbaustein stellt ein nicht schwingungsfähiges PT2-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \frac{1}{1+T_1 s} \frac{1}{1+T_2 s}$$

Sprungantwort:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fIn : Eingangsgröße des PT2-Glieds.

fManValue : Eingangsgröße, die im Manual-Mode ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut     : FLOAT;
    eState   : E_CTRL_STATE;
    eErrorId : E_CTRL_ERRORCODES;
    bError   : BOOL;
END_VAR
```

fOut : Ausgang des PT2-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT2_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PT2-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT2_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME     := T#0ms; (* controller cycle
```

```
time [TIME] *)
    tTaskCycleTime  : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fKp         : FLOAT    := 0;     (* proportional gain
*)
    tT1         : TIME     := T#0ms; (* T1 *)
    tT2         : TIME     := T#0ms; (* T2 *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Reglerverstärkung / Übertragungsbeiwert

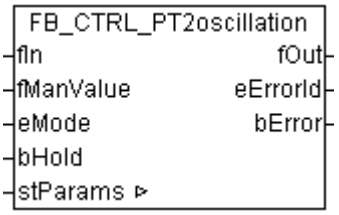
tT1 : Zeitkonstante T1

tT2 : Zeitkonstante T2

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.10 FB_CTRL_PT2oscillation

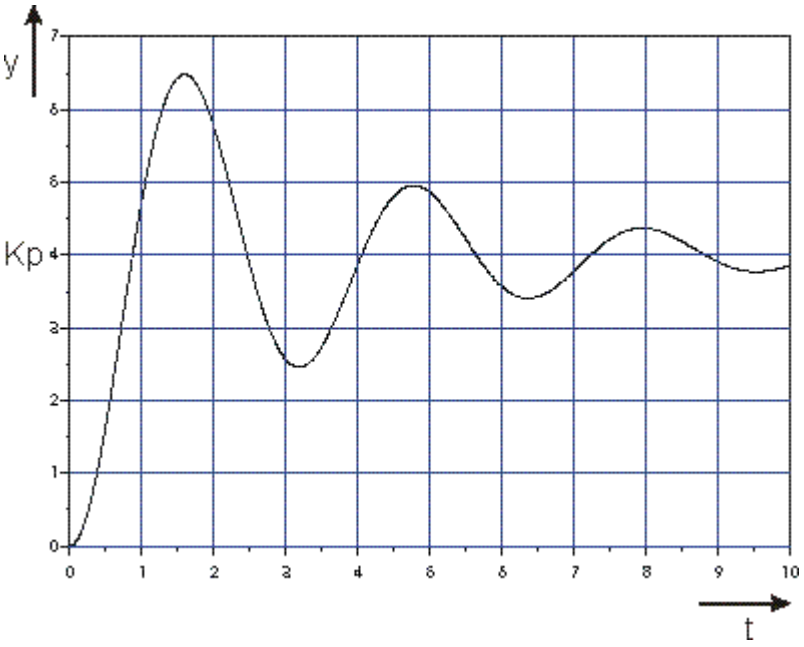


Der Funktionsbaustein stellt ein schwingungsfähiges PT2-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \frac{1}{1 + 2\vartheta T_0 s + T_0^2 s^2}$$

Sprungantwort:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fIn : Eingangsgröße des schwingungsfähigen PT2-Glieds.

fManValue : Eingangsgröße, die im Manual-Mode ausgegeben wird.

eState : State des Funktionsbausteins.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut    : FLOAT;
    eState  : E_CTRL_STATE;
    eErrorId : E_CTRL_ERRORCODES;
    bError  : BOOL;
END_VAR
```

fOut : Ausgang des PT2-Glieds.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT2oscillation_PARAMS;
END_VAR
```

stParams : Parameterstruktur des schwingungsfähigen PT2-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT2oscillation_PARAMS :
STRUCT
```

```
    tCtrlCycleTime      : TIME      := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime      : TIME      := T#0ms; (* task cycle time
[TIME] *)
    fKp         : FLOAT     := 0;      (* proportional
gain *)
    fTheta          : FLOAT     := 0;      (* damping ratio
*)
    tT0         : TIME      := T#0ms; (* T0 *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Proportionalbeiwert

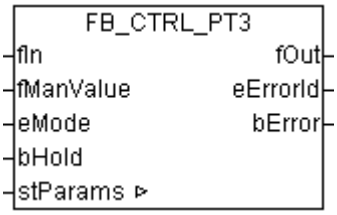
fTheta : Dämpfungsgrad

tT0 : Kennzeit

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.11 FB_CTRL_PT3

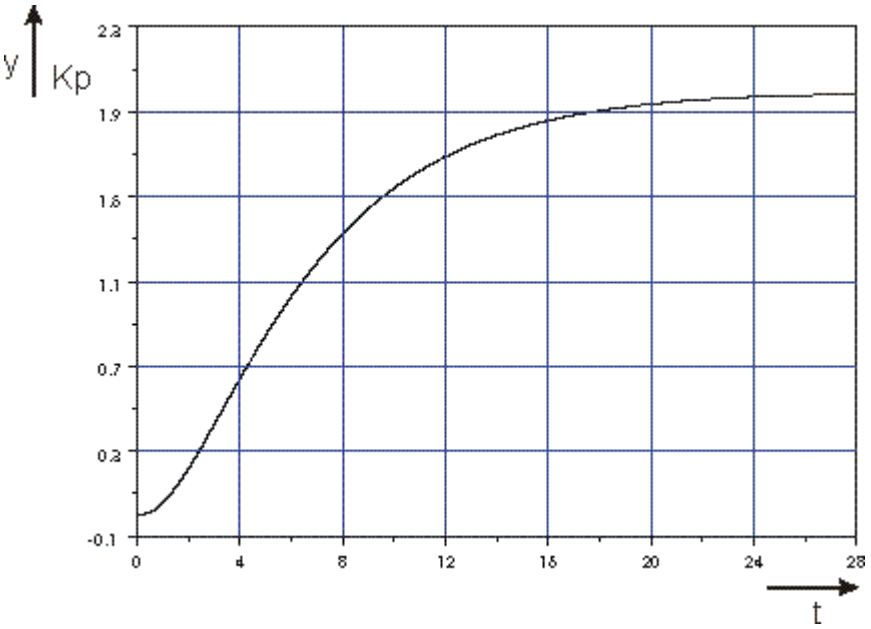


Der Funktionsbaustein stellt ein nicht schwingungsfähiges PT3-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \frac{1}{1+T_1 s} \frac{1}{1+T_2 s} \frac{1}{1+T_3 s}$$

Sprungantwort:



VAR_INPUT

```
VAR_INPUT
    fIn          : FLOAT;
    fManValue       : FLOAT;
    eMode        : E_CTRL_MODE;
    bHold        : BOOL;
END_VAR
```

fIn : Eingangsgröße des PT3-Glieds.

fManValue : Eingangsgröße, die im Manual-Mode ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut    : FLOAT;
    eState  : E_CTRL_STATE;
    eErrorId : E_CTRL_ERRORCODES;
    bError  : BOOL;
END_VAR
```

fOut : Ausgang des PT3-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT3_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PT3-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT3_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME     := T#0ms; (* controller cycle
time [TIME] *)
```

```
    tTaskCycleTime  : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fKp         : FLOAT    := 0;     (* proportional gain
*)
    tT1         : TIME     := T#0ms; (* T1 *)
    tT2         : TIME     := T#0ms; (* T2 *)
    tT3         : TIME     := T#0ms; (* T3 *)
END_STRUCT
END_TYPE>
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Reglerverstärkung / Übertragungsbeiwert

tT1 : Zeitkonstante T1

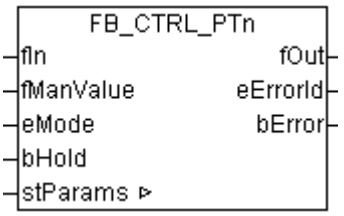
tT2 : Zeitkonstante T2p

tT3 : Zeitkonstante T3

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.12 FB_CTRL_PTn

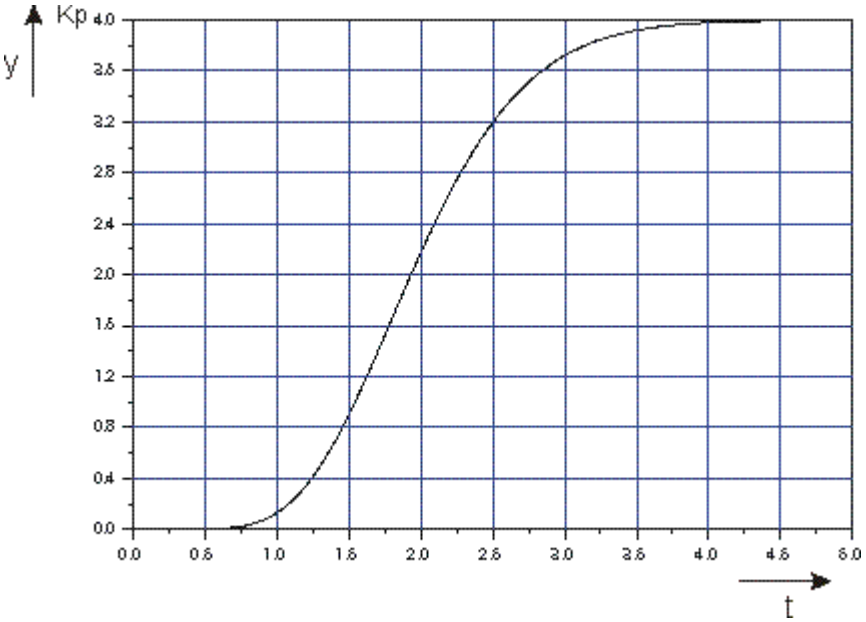


Der Funktionsbaustein stellt ein nicht schwingungsfähiges PTn-Übertragungsglied mit (n<= 10) und gleichen Zeitkonstanten im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \frac{1}{(1 + T_1 s)^n}$$

Sprungantwort bei n=10:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fIn : Eingangsgröße des PTn-Glieds.

fManValue : Eingangsgröße, die im Manual-Mode ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut    : FLOAT;
    eState  : E_CTRL_STATE;
    eErrorId : E_CTRL_ERRORCODES;
    bError  : BOOL;
END_VAR
```

fOut : Ausgang des PTn-Glieds.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PTn_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PTn-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PTn_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME     := T#0ms; (* controller cycle
time [TIME] *)
```

```
    tTaskCycleTime  : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fKp         : FLOAT    := 0;     (* proportional gain
*)
    tT1         : TIME     := T#0ms; (* T1 *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

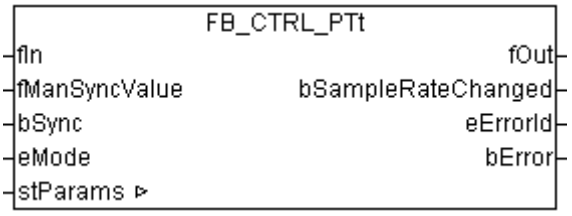
fKp : Reglerverstärkung / Übertragungsbeiwert

tT1 : Zeitkonstante T1

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.13 FB_CTRL_PTt



Der Funktionsbaustein stellt ein PTt-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \cdot e^{-T_t s}$$

Der Funktionsbaustein besitzt intern ein Array mit 500 Elementen, mit dem die Eingangswerte verzögert werden. Bei Verwendung der tCtrlCycleTime ergibt sich somit eine maximale Verzögerung von 500 * tCtrlCycleTime. Wenn diese maximale Verzögerung nicht ausreicht, wird intern die Abtastzeit vergrößert, so dass es möglich ist, die geforderte Totzeit zu erreichen. Bei dieser Vorgehensweise ist aber zu beachten, dass sich die zeitlichen Diskretisierungsstufen vergrößern. Wenn intern eine neue Abtastzeit berechnet wird, so wird dies mit einem TRUE an dem Ausgang bSampleRateChanged signalisiert.

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingangsgröße des PTt-Glieds.

fManSyncValue : Eingangsgröße, auf die das PTt-Glied gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das PTt-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut          : FLOAT;
    bSampleRateChanged       : BOOL;
    eErrorId          : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

fOut : Ausgang des PTt-Glieds.

bSampleRateChanged : Ausgang der anzeigt, ob der Baustein intern die Abtastrate reduziert hat, da das Array zur Verzögerung des Eingangssignals sonst nicht genügend Platz bietet.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams         : ST_CTRL_PTt_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PTt-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PTt_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime  : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fKp          : FLOAT    := 0;     (* proportional gain
*)
    tTt          : TIME     := T#0ms; (* Tt *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

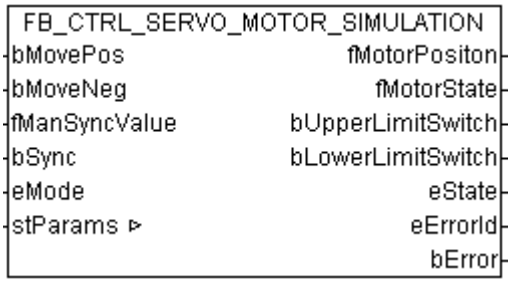
fKp : Reglerverstärkung / Übertragungsbeiwert

tTt : Totzeit

Voraussetzungen

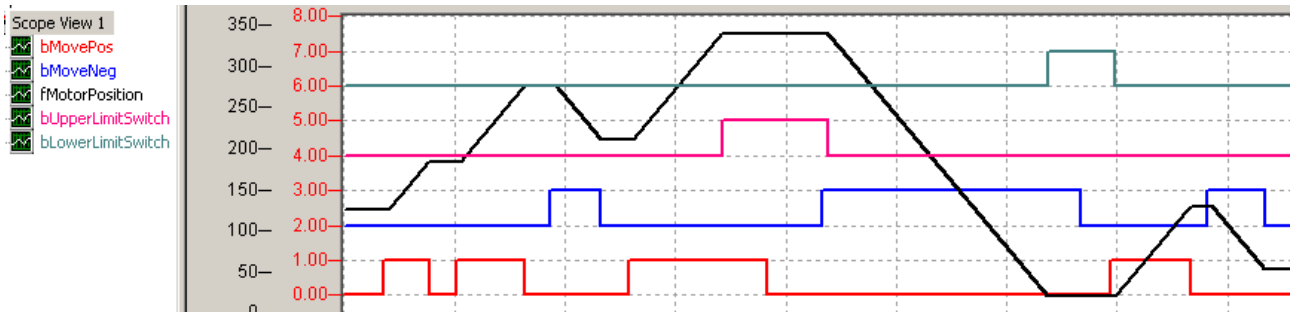
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.14 FB_CTRL_SERVO_MOTOR_SIMULATION (nur auf einem PC-System)



MIt diesem Funktionsbaustein kann das Verhalten eines Stellantriebs simuliert werden.

Verhalten des Ausgangs:



VAR_INPUT

```
VAR_INPUT
    bMovePos        : BOOL;
    bMoveNeg        : BOOL;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

bMovePos : Eingang, der den simulierten Stellantrieb in positive Richtung fährt.

bMoveNeg : Eingang, der den simulierten Stellantrieb in negative Richtung fährt.

fManSyncValue : Eingang, mit dem die simulierte Motorstellung gesetzt werden kann, oder auf dessen Wert im Manual-Mode gefahren wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird die simulierte Motorposition auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fMotorPositon   : FLOAT; (* [ fMovingRangeMin ... fMovingRangeMax ] *)
    fMotorState     : FLOAT; (* [ 0 ... 100 ] *)
    bUpperLimitSwitch       : BOOL;
    bLowerLimitSwitch       : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fMotorPosition : Simulierte Motorstellung im Intervall [fMovingRangeMin ... fMovingRangeMax].

fMotorState : Simulierte Motorstellung im Intervall [0 ... 100.0].

bUpperLimitSwitch : Simulierter Endschalter am positiven Anschlag des Stellantriebs.

bLowerLimitSwitch : Simulierter Endschalter am negativen Anschlag des Stellantriebs.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME  := T#0ms;     (* controller
cycle time [TIME] *)
    tTaskCycleTime    : TIME  := T#0ms;     (* task cycle
time [TIME] *)
    fMovingRangeMin       : FLOAT := 0;     (* min position
in the moving range, e.g. 0° *)
    fMovingRangeMax       : FLOAT := 0;     (* max position
in the moving range, e.g. 360° *)
    tMovingTime       : TIME := T#0ms;      (* time to move
from the min to the max *)
    tDeadTime         : TIME := T#0ms;      (* wait dead
time before starting the movement *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen in aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fMovingRangeMin : Minimale Position des simulierten Stellantriebs.

fMovingRangeMax : Maximale Position des simulierten Stellantriebs.

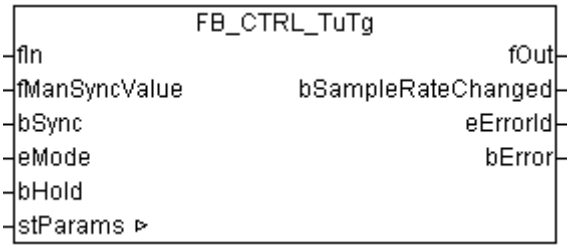
tMovingTime : Die Zeit, die benötigt wird, um den simulierten Stellantrieb von einem Anschlag zum anderen zu fahren.

tDeadTime : Totzeit des simulierten Stellantriebs.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

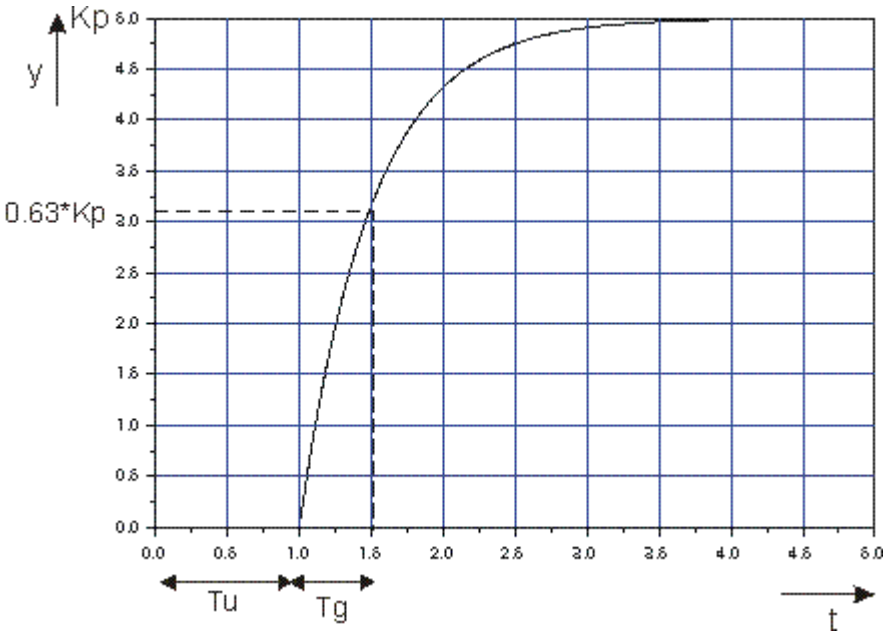
5.5.15 FB_CTRL_TuTg



Der Funktionsbaustein stellt ein TuTg-Übertragungsglied (Totzeit-Verzögerungs-Glied) im Wirkungsplan dar.

Übertragungsfunktion:

$$G(s) = K_p \frac{1}{1 + T_g s} \cdot e^{-T_u s}$$



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManSyncValue    : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

fIn : Eingangsgröße des TuTg-Glieds.

fManSyncValue : Eingangsgröße, auf die das TuTg-Glied gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird das TuTg-Glied auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

bHold : Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang unabhängig von der Eingangsgröße konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut         : FLOAT;
    bSampleRateChanged       : BOOL;
    eState       : E_CTRL_STATE;
    eErrorId         : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

fOut : Ausgang des TuTg-Glieds.

bSmpleRateChanged : Ausgang der anzeigt, ob der Baustein intern die Abtastrate reduziert hat, da das Array zur Verzögerung des Eingangssignals sonst nicht genügend Platz bietet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_TuTg_PARAMS;
END_VAR
```

stParams : Parameterstruktur des TuTg-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_TuTg_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime  : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fKp         : FLOAT    := 0;     (* proportional gain
*)
    tTu         : TIME     := T#0ms; (* dead time *)
    tTg         : TIME     := T#0ms; (* delay time
*)END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen in aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fKp : Reglerverstärkung / Übertragungsbeiwert

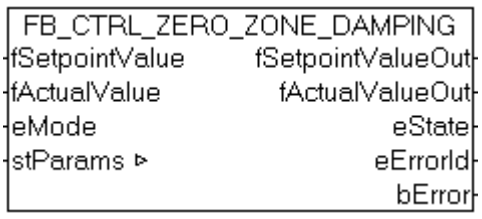
tTu : Totzeit

tTg : Zeitkonstante

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.5.16 FB_CTRL_ZERO_ZONE_DAMPING

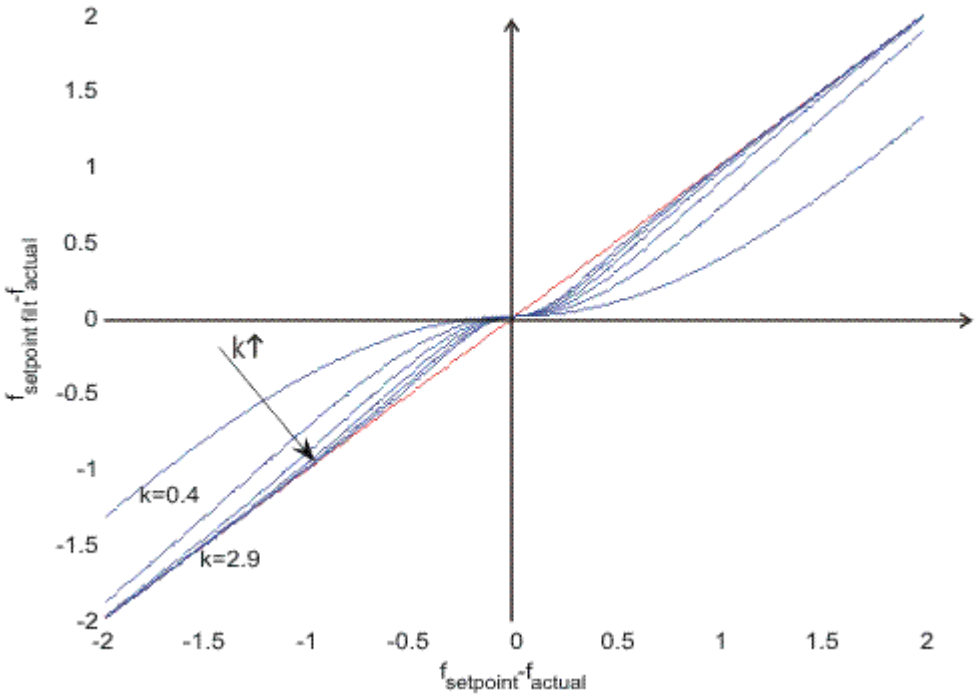


Mit diesem Funktionsbaustein kann eine Nullpunktdämpfung realisiert werden, um Regeleingriffe im Bereich | Istwert.-Sollwert | < ε zu minimieren.

Übertragungsverhalten im Zeitbereich:

$$f_{setpoint\_out} = (f_{setpoint\_in} - f_{actual\_in}) \cdot \tanh(|f_{setpoint\_in} - f_{actual\_in}| \cdot k_{damping}) + f_{actual\_in}$$

$$f_{actual\_out} = f_{setpoint\_in}$$



VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

fSetpointValue : Sollwert der Regelgröße.

fActualValue : Istwert der Regelgröße.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fSetpointValueOut : FLOAT;
    fActualValueOut     : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fSetpointValueOut : Gefilterter Sollwert zum Regler.

fActualValueOut : Istwert zum Regler.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_ZERO_ZONE_DAMPING_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Übertragungselementes. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PI_PST_CTRL_ZERO_ZONE_DAMPING_PARAMSARAMS :
STRUCT
```

```
    tCtrlCycleTime      : TIME      :=  T#0ms; (* controller
cycle time [TIME] *)
    tTaskCycleTime      : TIME      :=  T#0ms; (* task cycle time
[TIME] *)
    fDampingCoefficient: FLOAT     :=  0.0; (*damping
coefficient *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fDampingCoefficient : Der Parameter entspricht $k_{damping}$ in der Übertragungsfunktion.

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

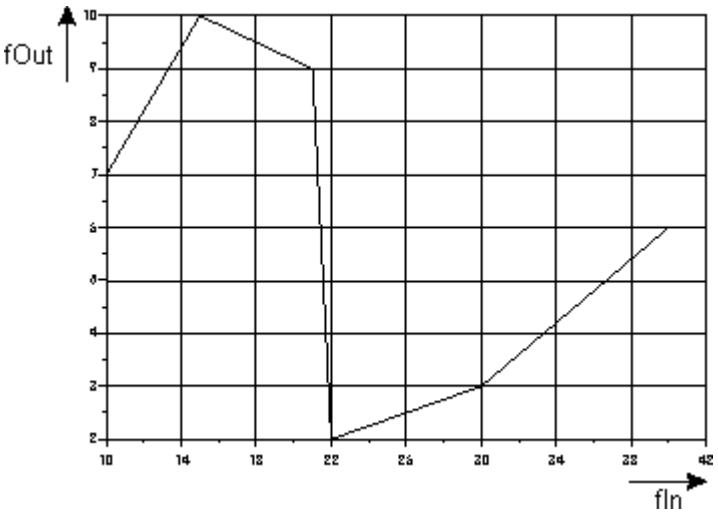
5.6 Interpolation

5.6.1 FB_CTRL_LIN_INTERPOLATION

Dieser Baustein gibt aus einer Stützstellentabelle den linear interpolierten Wert aus.

Verhalten des Ausgangs:

fIn	fOut
arrTable[1,1] := 10;	arrTable[1,2] := 7;
arrTable[2,1] := 15;	arrTable[2,2] := 10;
arrTable[3,1] := 21;	arrTable[3,2] := 9;
arrTable[4,1] := 22;	arrTable[4,2] := 2;
arrTable[5,1] := 30;	arrTable[5,2] := 3;
arrTable[6,1] := 40;	arrTable[6,2] := 6;



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    bExtrapolate    : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingangsgröße des Interpolationsbausteins.

bManValue : Eingangsgröße, dessen Wert im Manual-Mode ausgegeben wird.

bExtrapolate : Wenn dieser Eingang FALSE ist, wird bei Überschreiten der Tabellengrenze der Wert der letzten Stützstelle ausgegeben. Wenn ein TRUE anliegt, wird mit den letzten zwei Stützstellen extrapoliert.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut            : FLOAT;
    bInIsGreaterThanMaxElement      : BOOL;
    bInIsLessThanMinElement     : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

fOut : Linear interpolierter Tabellenwert.

bInIsGreaterThanMaxElement : Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße größer ist als die größte Stützestelle.

bInIsLessThanMinElement : Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße kleiner ist als die kleinste Stützestelle.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIN_INTERPOLATION_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Interpolations-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
```

```
    tCtrlCycleTime          : TIME      := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME      := T#0ms; (* task
cycle time [TIME] *)
    pDataTable_ADR          : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF       : UINT          := 0;
    nDataTable_NumberOfRows     : UINT          := 0;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

pDataTable_ADR : Adresse des n x 2 - Arrays, welches linear interpoliert wird

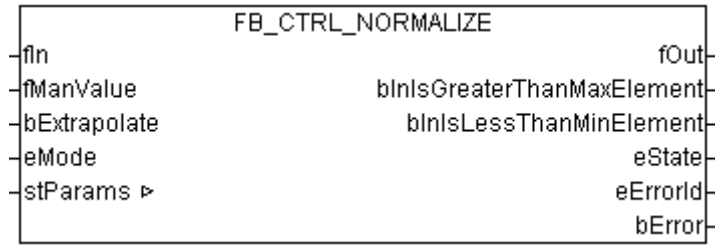
pDataTable_SIZEOF : Größe des n x 2 - Arrays.

pDataTable_NumberOfRows : Anzahl der Zeilen des Arrays.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.6.2 FB_CTRL_NORMALIZE

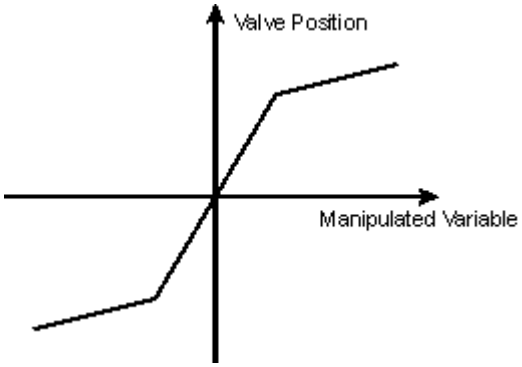


Mit diesem Funktionsbaustein kann ein nichtlineares Übertragungsglied mit Hilfe einer inversen Kennlinie linearisiert werden.

In der Tabelle dieses Bausteins wird die Kennlinie des zu linearisierenden Übertragungsglieds hinterlegt. Der Funktionsbaustein berechnet aus dieser die inverse Kennlinie, mit der die Liniearisierung erfolgen kann.

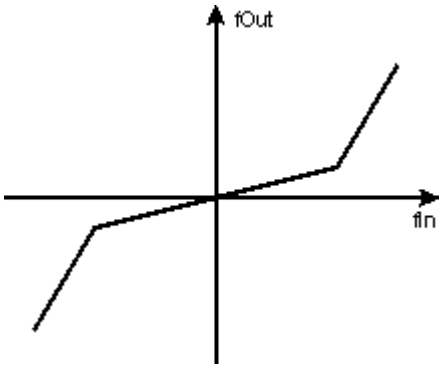
Beispiel:

Die folgende Ventilkennlinie wird mit 4 Stützstellen in der Tabelle abgelegt.

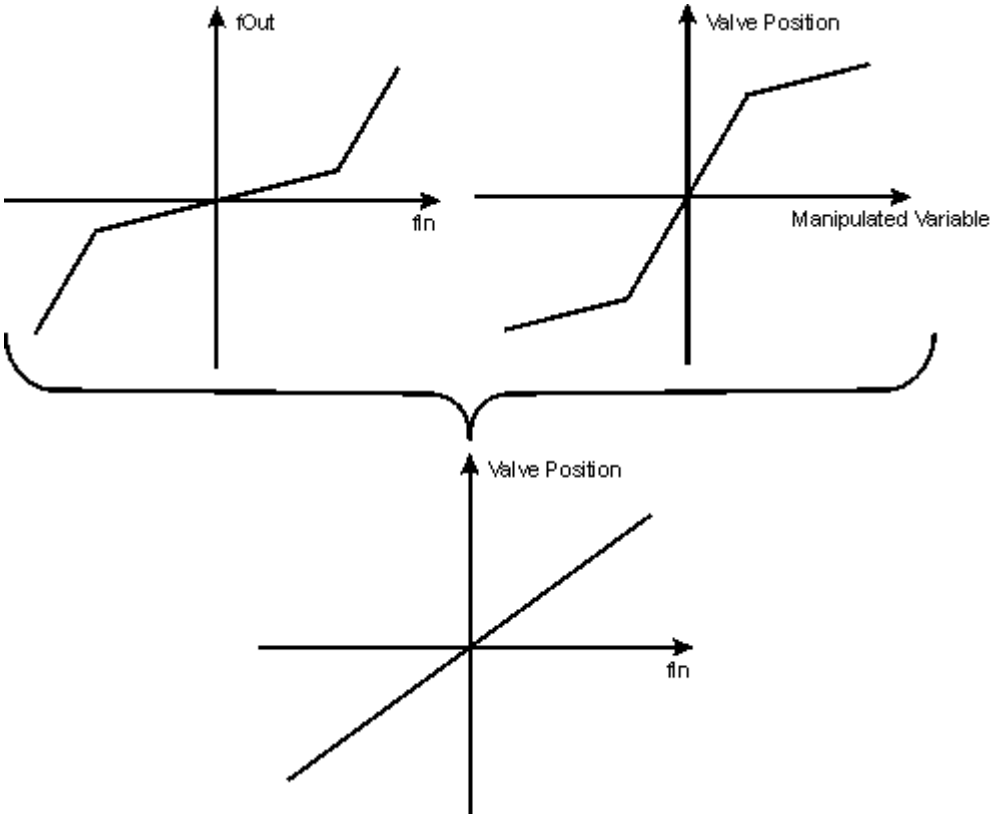


Stellgröße	Ventilstellung
arrTable[1,1] := -6;	arrTable[1,2] := -100;
arrTable[2,1] := -1;	arrTable[2,2] := -70;
arrTable[3,1] := 1;	arrTable[3,2] := 70;
arrTable[4,1] := 6;	arrTable[4,2] := 100;

Aus dieser Kennlinie wird die inverse Kennlinie berechnet:



Durch die Reihenschaltung dieser beiden Kennlinien ergibt sich im Idealfall ein lineares Übertragungsverhalten.



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue       : FLOAT;
    bExtrapolate    : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Eingangsgröße.

bManValue : Eingangsgröße, dessen Wert im Manual-Mode ausgegeben wird.

bExtrapolate : Wenn dieser Eingang FALSE ist, wird bei Überschreiten der Tabellengrenze der Wert der letzten Stützstelle ausgegeben. Wenn ein TRUE anliegt, wird mit den letzten zwei Stützstellen extrapoliert.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut            : FLOAT;
    bInIsGreaterThanMaxElement      : BOOL;
    bInIsLessThanMinElement     : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

fOut : Linear interpolierter Tabellenwert.

bInIsGreaterThanMaxElement : Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße größer ist als die größte Stützestelle.

bInIsLessThanMinElement : Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße kleiner ist als die kleinste Stützestelle.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_NORMALIZE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_NORMALIZE_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME      := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME      := T#0ms; (* task
cycle time [TIME] *)
    pDataTable_ADR          : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF       : UINT          := 0;
    nDataTable_NumberOfRows     : UINT          := 0;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

pDataTable_ADR : Adresse des n x 2 - Arrays, welches linear interpoliert wird

pDataTable_SIZEOF : Größe des n x 2 - Arrays.

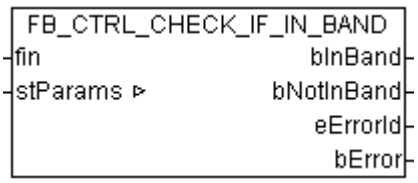
pDataTable_NumberOfRows : Anzahl der Zeilen des Arrays.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.7 Monitoring / Alarming

5.7.1 FB_CTRL_CHECK_IF_IN_BAND



Der Funktionsbaustein überwacht, ob sich die Eingangsgröße in dem Intervall [fMin ... fMax] befindet, also ob die Ungleichung

$$fMin \leq fIn \leq fMax$$

erfüllt wird.

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

fIn : Die zu überwachende Eingangsgröße.

VAR_OUTPUT

```
VAR_OUTPUT
    bInBand         : BOOL;
    bNotInBand      : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bInBand : Ein TRUE an diesem Ausgang signalisiert, dass sich die Eingangsgröße in dem angegebenen Intervall befindet.

bNotInBand : Ein TRUE an diesem Ausgang signalisiert, dass sich die Eingangsgröße **nicht** in dem angegebenen Intervall befindet.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_CHECK_IF_IN_BAND_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_CHECK_IF_IN_BAND_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fMin           : FLOAT;
    fMax           : FLOAT;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

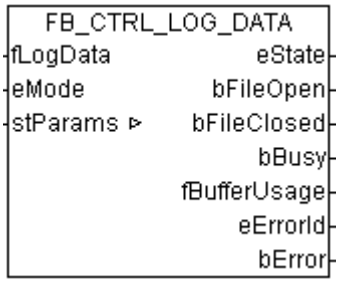
fMin : Untere Grenze des Intervalls.

fMax : Obere Grenze des Intervalls.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.7.2 FB_CTRL_LOG_DATA (nur auf einem PC-System)



Der Funktionsbaustein ermöglicht das Erstellen eines Log-Files im *.csv Format (comma separated values), in dem maximal 10 Größen aufgezeichnet werden können. In die erste Zeile dieser Datei werden die vom Benutzer angegebenen Spaltenüberschriften geschrieben. In die folgenden Zeilen werden die Eingangsdaten in zeitlich äquidistanten Abständen geschrieben. Die einzelnen Einträg werden durch ein Komma getrennt. Mit dem Parameter **tLogCycleTime** wird der zeitliche Abstand der Eintragungen festgelegt. Wenn zum Beispiel tLogCycleTime := T\#2s gewählt wird, wird alle 2s ein Eintrag in die Datei geschrieben. Die erzeugten Dateien können beispielsweise mit einem Tabellenkalkulationsprogramm ausgewertet werden.

In die erste Spalte der Datei wird der Zeitstempel des Log-Eintrags in s gespeichert. In den weiteren Spalten folgen die Daten des Bausteineingangs **fLogData**.

Wenn der Mode auf **eCTRL_MODE_ACTIVE** gesetzt wird, wird die Log-Datei geöffnet und es werden Einträge in die Datei geschrieben. Diese Datei bleibt solange geöffnet, bis der Mode des Bausteins auf eCTRL_MODE_PASSIVE gesetzt wird. Bevor die Log-Datei ausgewertet werden kann, muss sie unbedingt durch ein Umschalten in den **eCTRL_MODE_PASSIVE** geschlossen werden. Anderenfalls ist es möglich, dass noch nicht alle Einträge in die Datei geschrieben worden sind.

Der Baustein ermöglicht es, mit und ohne einen externen Puffer zu arbeiten. Wenn eine Pufferadresse und eine Puffergröße ungleich Null parametriert werden, wird der externe Puffer genutzt. Ohne einen externen Puffer wird ein interner Puffer mit einer Größe von 255 Byte verwendet.

Betrieb ohne externen Puffer:	Wenn das Loggen einer Zeile gestartet ist, wird der Ausgang bBusy TRUE. Der nächste Datensatz wird erst dann geloggt, wenn der Ausgang bBusy wieder FALSE ist. Der Füllstand des internen Puffers wird am Ausgang **fBufferUsage** angezeigt.
Betrieb mit externem Puffer:	Der Benutzer muss einen Puffer des Typs *ARRAY OF BYTES* anlegen, der größer als 255 Byte ist. Die einzelnen Meldungen werden temporär in dem externen Puffer abgelegt und

	dieser Puffer wird so schnell wie möglich in die Datei geschrieben. Der Füllstand des Puffers wird am Ausgang **fBufferUsage** angezeigt. Wenn es zu einem Pufferüberlauf kommt, wird der Baustein gestoppt und ein Fehler ausgegeben.

VAR_INPUT

```
VAR_INPUT
    fLogData    : T_CTRL_LOGGER_DATA;
    eMode   : E_CTRL_MODE;
END_VAR
```

```
VAR_GLOBAL CONSTANT
    nCTRL_LOGGER_DATA_ARRAY_SIZE     :UINT := 10;
END_VAR
```

```
TYPE
    T_CTRL_LOGGER_DATA     :ARRAY [1..nCTRL_LOGGER_DATA_ARRAY_SIZE]
OF FLOAT;
END_TYPE
```

fLogData : Array mit den Werten, die in das Log-File geschrieben werden.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eState      : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT     (* Buffer fill level in percent [0 ... 100] *)
    bBusy       : BOOL
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

eState : State des Funktionsbausteins.

bFileOpen : Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geöffnet wurde.

bFileClosed : Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geschlossen wurde.

fBufferUsage : Aktueller Füllstand des externen Puffers in Prozent.

bBusy : Ein TRUE an diesem Ausgang signalisiert, dass das Loggen einer Zeile aktiv ist.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_DATA_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Log-Bausteins. Diese besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_LOG_DATA_PARAMS
:STRUCT
tLogCycleTime        : TIME := T#0ms; (* controller cycle time
[TIME] *)
tTaskCycleTime       : TIME := T#0ms; (* task cycle time [TIME]
*)
sFileName        : STRING;
sNetId           : T_AmsNetId := ''; (* ams net id
*)
tFileOperationTimeout    : TIME := T#3s; (* file operation timeout
*)
nNumberOfColumns     : INT(1..10);
arColumnHeadings     : ARRAY [1..10] OF STRING;
bAppendData          : BOOL := FALSE;
bWriteTimeStamps     : BOOL := TRUE;
bWriteColumnHeadings     : BOOL := TRUE;
bWriteAbsoluteTimeStamps : BOOL := FALSE; (* Set to true if the
NT-time form the local machine should logged in the first column.
```

```
*)
(* The log cycle time must than be greater or equal T#5s!!!
*)
pLogBuffer_ADR       : POINTER TO BYTE;
nLogBuffer_SIZEOF    : UDINT;END_STRUCTEND_TYPE
```

tLogCycleTime : Zykluszeit, mit der Einträge in das Log-File geschrieben werden. Diese muss größer oder gleich der TaskCycleTime sein.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

sFileName : Name und Pfad des Log-Files, z.B.: d:\Logfile.csv.

sNetId : Das LogFile wird auf das System mit der hier angegebenen NetId geschrieben.

tFileOperationTimeout : Timeout für alle Dateioperationen.

nNumberOfColums: Anzahl der Spalten, die in die Datei geschrieben werden (maximal 10).

arColumnHeadings: Array aus Strings, die die Spaltenüberschriften enthalten.

bAppendData : Wenn dieser Parameter auf TRUE gesetzt ist, wird beim erneuten Öffnen einer Datei der neue Datensatz angehängt, anderenfalls wird die Datei **ohne Rückfrage** überschrieben und somit der bereits bestehende Inhalt gelöscht..

bWriteTimeStamps : Wenn dieser Parameter auf TRUE gesetzt ist, wird in die 1. Spalte der Datei der Zeitstempel der Messung geschrieben.

bWriteColumnHeadings : Wenn dieser Parameter auf TRUE gesetzt ist, werden in die 1. Zeile der Datei die Spaltenüberschriften geschrieben.

bWriteAbsoluteTimeStamps : Wenn dieser Parameter auf TRUE gesetzt wird, wir als Timestamp die lokale NT-Zeit verwendet. In diesem Fall beträgt die minimale LogCycleTime 5s!

pLogBuffer_ADR : Adresse des externen LogBuffers. Um einen Puffer zu erkennen muss die Adresse ungleich 0 sein.

nLogBuffer_SIZEOF: Größe des LogBuffers. Der Puffer muss ein ARRAY OF BYTE mit mindestens 256 Elementen sein. Die Größe des Puffers kann mit Hilfe des Ausgangs fBufferUsage optimiert werden.

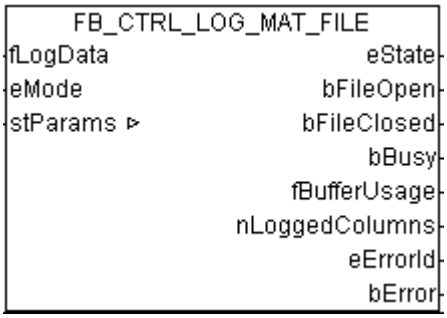
HINWEIS

Der Parametersatz darf nur dann geändert werden, wenn die Datei geschlossen ist (bFileClosed = TRUE)! Anderenfalls kann es zu Fehlern bei dem Dateihandling kommen!

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

5.7.3 FB_CTRL_LOG_MAT_FILE(nur auf einem PC-System)



Der Funktionsbaustein ermöglicht das Erstellen eines Log-Files im Matlab 5 - Format (*.mat), in dem maximal 10 Größen aufgezeichnet werden können.

In der Datei werden zwei Variablen angelegt, ein double-Array und ein cell-Array. In dem double-Array werden die aufgezeichneten Größen zeilenweise aufgezeichnet. In dem cell-Array werden die Bezeichnungen der einzelnen Zeilen abgelegt. Der Benutzer kann den Namen des double-Arrays in der Parameterstruktur des Funktionsbausteins angeben. Der Name des cell-Arrays wird aus dem Namen des double-Arrays gebildet, indem "_Info" an den Variablen-Namen angehängt wird.

In die Spalten des Daten-Arrays werden die Eingangsdaten in zeitlich äquidistanten Abständen geschrieben. In der ersten Spalte kann der Timestamp des jeweiligen Eintrags in s abgelegt werden. Mit dem Parameter **tLogCycleTime** wird der zeitliche Abstand der Eintragungen festgelegt. Wenn zum Beispiel tLogCycleTime := T\#2s gewählt wird, wird alle 2s ein Eintrag in die Datei geschrieben.

Wenn der Mode auf **eCTRL_MODE_ACTIVE** gesetzt wird, wird die Log-Datei geöffnet und es werden Einträge in die Datei geschrieben. Diese Datei bleibt solange geöffnet, bis der Mode des Bausteins auf eCTRL_MODE_PASSIVE gesetzt wird. Bevor die Log-Datei ausgewertet werden kann, muss sie unbedingt durch ein Umschalten in den **eCTRL_MODE_PASSIVE** geschlossen werden. Anderenfalls ist es möglich, dass noch nicht alle Einträge in die Datei geschrieben worden sind und dass diese nicht konsistent ist.

Der Baustein ermöglicht es, mit und ohne einem externen Buffer zu arbeiten. Wenn eine Bufferadresse und eine Buffergröße ungleich Null parametriert wird, wird der externe Buffer genutzt. Ohne einen externen Buffer wird ein interner Buffer mit einer Größe von 1500 Byte verwendet.

Betrieb ohne externen Buffer:	Wenn das Loggen einer Zeile gestartet ist, wird der Ausgang bBusy TRUE. Der nächste Datensatz wird erst dann geloggt, wenn der Ausgang bBusy wieder FALSE ist. Der Füllstand des internen Buffers wird am Ausgang **fBufferUsage** angezeigt.
Betrieb mit externem Buffer:	Der Benutzer muss einen Buffer des Typs *ARRAY OF BYTES* anlegen, der größer als 1500 Byte ist. Die einzelnen Meldungen werden temporär in dem externen Buffer abgelegt und dieser Buffer wird so schnell wie möglich in die Datei geschrieben. Der Füllstand des Buffers wird am Ausgang **fBufferUsage** angezeigt. Wenn es zu einem Bufferüberlauf kommt, wird der Baustein gestoppt und ein Fehler ausgegeben.

VAR_INPUT

```
VAR_INPUT
    fLogData    : T_CTRL_LOGGER_DATA;
    eMode   : E_CTRL_MODE;
END_VAR
```

```
VAR_GLOBAL CONSTANT
    nCTRL_LOGGER_DATA_ARRAY_SIZE     :UINT := 10;
END_VAR
```

```
TYPE
    T_CTRL_LOGGER_DATA     :ARRAY [1..nCTRL_LOGGER_DATA_ARRAY_SIZE]
OF FLOAT;
END_TYPE
```

fLogData : Array mit den Werten, die in das Log-File geschrieben werden.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eState      : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT     (* Buffer fill level in percent [0 ... 100] *)
    nLoggedColumns  : DINT;
    bBusy       : BOOL
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

eState : State des Funktionsbausteins.

bFileOpen : Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geöffnet wurde.

bFileClosed : Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geschlossen wurde.

fBufferUsage : Aktueller Füllstand des externen Buffers in Prozent.

nLoggedColumns : Anzahl der in die Datei geschriebenen Spalten.

bBusy : Ein TRUE an diesem Ausgang signalisiert, dass das Loggen einer Zeile aktiv ist.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_MAT_FILE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Log-Bausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_LOG_MAT_FILE_PARAMS:
STRUCT
    tLogCycleTime    : TIME := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime       : TIME := T#0ms; (* task cycle time
[TIME] *)
    sFileName        : STRING;
    nNumberOfRows    : INT(1..10);
    sMatrixName      : STRING;
    arRowDescription     : ARRAY [1..10] OF STRING(25);
    bWriteTimeStamps     : BOOL := TRUE;
    bWriteRowDescription : BOOL := TRUE;
    pLogBuffer_ADR       : POINTER TO
ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    nLogBuffer_SIZEOF    : UDINT;
END_STRUCT
END_TYPE
```

tLogCycleTime : Zykluszeit, mit der Einträge in das Log-File geschrieben werden. Diese muss größer oder gleich der TaskCycleTime sein.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

sFileName : Name und Pfad des Log-Files, z.B.: d:\Logfile.mat.

nNumberOfRows : Anzahl der Spalten, die in die Datei geschrieben werden (maximal 10).

sMatrixName : Name des Daten-Arrays.

arRowDescription : Array aus Strings, die die Zeilenüberschriften enthalten.

bWriteTimeStamps : Wenn dieser Parameter auf TRUE gesetzt ist, wird in die 1. Zeile des Daten-Arrays der Zeitstempel der Messung geschrieben.

bWriteRowDescription : Wenn dieser Parameter auf TRUE gesetzt ist, wird ein cell-Array angelegt, in das die Beschreibungen der Zeilen geschrieben werden.

pLogBuffer_ADR : Adresse des externen LogBuffers. Um einen Buffer zu erkennen muss die Adresse ungleich 0 sein.

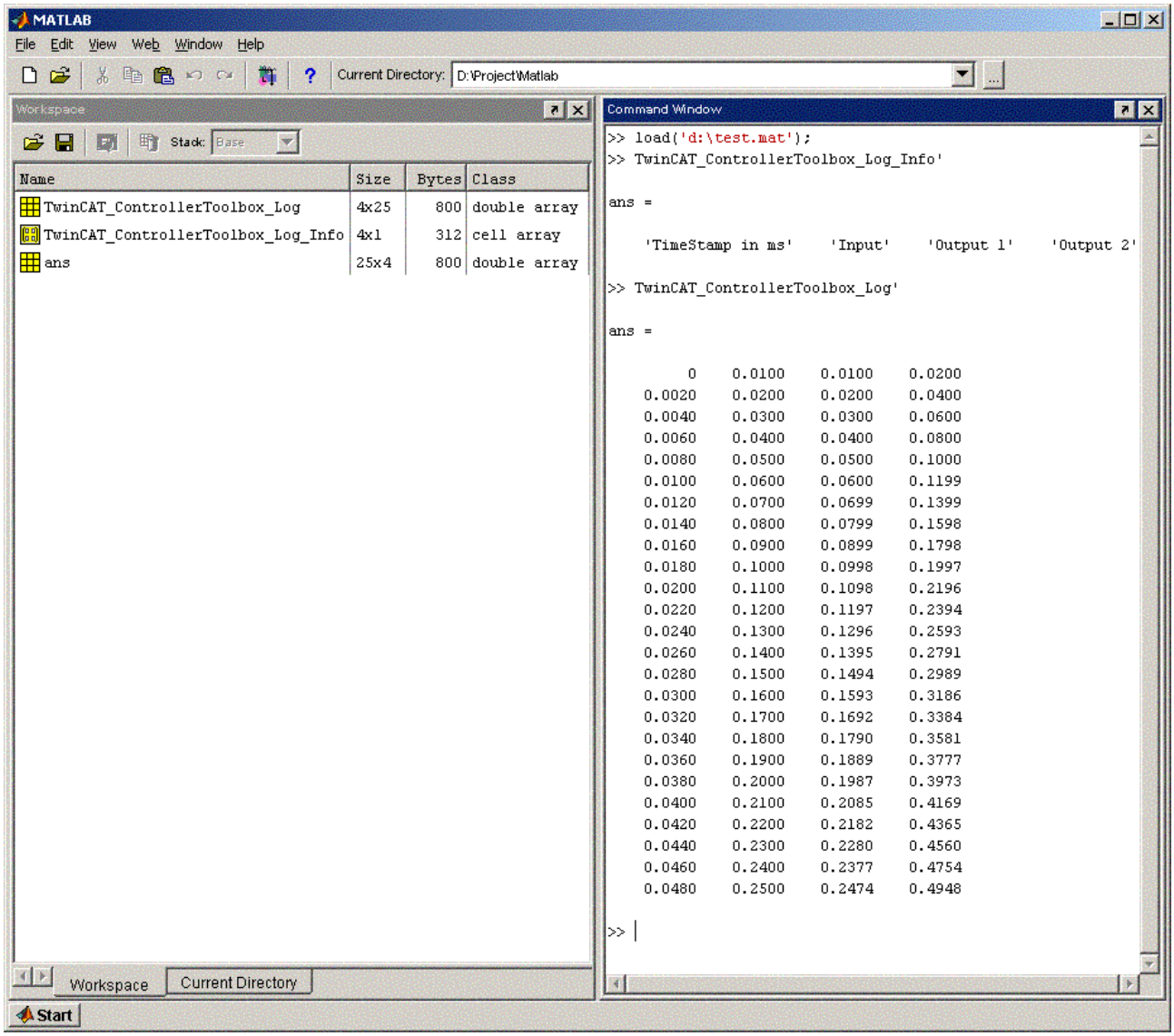
nLogBuffer_SIZEOF: Größe des LogBuffers. Der Buffer muss ein ARRAY OF BYTE mit mindestens 1501 Elementen sein. Die Größe des Buffers kann mit Hilfe des Ausgangs fBufferUsage optimiert werden.

HINWEIS

Der Parametersatz darf nur dann geändert werden, wenn die Datei geschlossen ist (bFileClosed = TRUE)! Anderenfalls kann es zu Fehlern bei dem Dateihandling kommen!

Beispiel

```
PROGRAM PRG_LOG_MAT_FILE_TEST_BUFFERED
VAR
    eMode       : E_CTRL_MODE;
    stParams        : ST_CTRL_LOG_MAT_FILE_PARAMS;
    LoggerData      : T_CTRL_LOGGER_DATA;

    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;

    fbCtrlLogMatFile    : FB_CTRL_LOG_MAT_FILE;
    LogBuffer       : ARRAY[0..2000] OF BYTE;

    bInit       : BOOL := TRUE;

    fIn         : LREAL;
    fOut        : LREAL;

    fMaxBufferUsage     : LREAL;
END_VAR

IF bInit
THEN
    stCtrl_GLOBAL_CycleTimeInterpretation.bInterpretCycleTimeAsTicks := FALSE;
    stCtrl_GLOBAL_CycleTimeInterpretation.fBaseTime := 0;

    stParams.tLogCycleTime      := T#2ms;
    stParams.tTaskCycleTime         := T#2ms;
    stParams.nNumberOfRows      := 3;
    stParams.sFileName          := 'D:\test.mat';
    stParams.sMatrixName        := 'TwinCAT_ControllerToolbox_Log';

    stParams.arRowDescription[1]    := 'Input';
    stParams.arRowDescription[2]    := 'Output 1';
    stParams.arRowDescription[3]    := 'Output 2';

    stParams.bWriteRowDescription   := TRUE;
    stParams.bWriteTimeStamps           := TRUE;

    eMode   := eCTRL_MODE_ACTIVE;
    bInit   := FALSE;
END_IF

stParams.nLogBuffer_SIZEOF      := SIZEOF( LogBuffer );
stParams.pLogBuffer_ADR := ADR( LogBuffer );

(* creat test data *)
  fIn   := fIn + 0.01;
  fOut  := SIN(fIn);

(* copy test data to the Logger-Input *)
  LoggerData[ 1 ] := fIn;
  LoggerData[ 2 ] := fOut;
  LoggerData[ 3 ] := fOut * 2;

(* log only 1000 columns *)
  IF fbCtrlLogMatFile.nLoggedColumns >= 25
  THEN
    eMode := eCTRL_MODE_Passive;
  END_IF

(* call logger *)
  fbCtrlLogMatFile( fLogData := LoggerData,
          eMode := eMode,
          stParams :=stParams,
          eErrorId => eErrorId,
          bError => bError);

(* get max buffer usage *)
  fMaxBufferUsage := MAX(fbCtrlLogMatFile.fBufferUsage, fMaxBufferUsage);
```

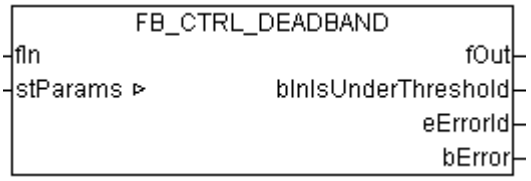


Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

5.8 Output To Controlling Equipment

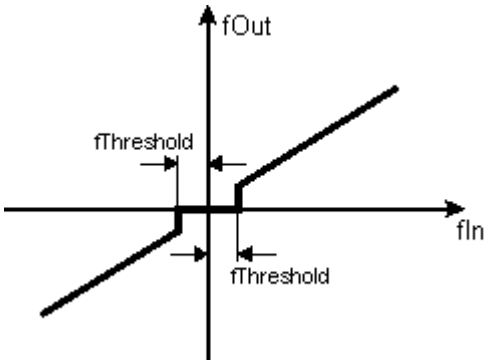
5.8.1 FB_CTRL_DEADBAND



Dieser Baustein stellt eine Totzone für das Eingangssignal dar. Wenn sich das Eingangssignal in der Totzone befindet, wird dieses durch den Ausgang **bInIsUnderThreshold** signalisiert.

Beschreibung des Ausgangsverhaltens:

$$f_{out} = \begin{cases} 0.0 & : \quad |f_{in}| \leq fThreshold \\ f_{in} & : \quad else \end{cases}$$



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

fIn : Eingangsgröße.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut            : FLOAT;
    bInIsUnderThreshold     : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

fOut : Ausgang des Funktionsbausteins.

bInIsUnderThreshold : Ein TRUE an diesem Ausgang signalisiert, dass sich der Eingangswert in der Totzone befindet.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_DEADBAND_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_DEADBAND_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME      := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME      := T#0ms; (* task
cycle time [TIME] *)
    fThreshold          : FLOAT;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

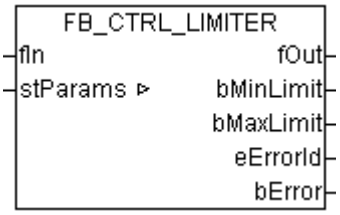
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fThreshold : Totzone des Funktionsbausteins, siehe Bild.

Voraussetzungen

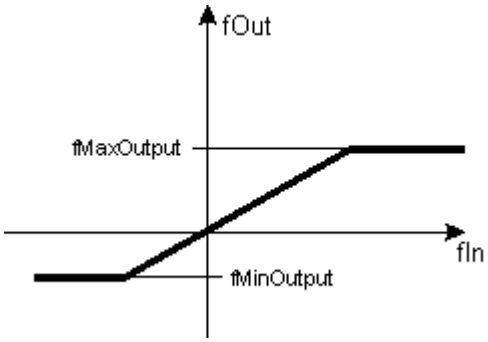
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.2 FB_CTRL_LIMITER



Dieser Baustein begrenzt ein Eingangssignal auf ein parametrierbares Intervall.

Beschreibung des Ausgangsverhaltens:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

fIn : Eingangsgröße des Funktionsbausteins.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    bMinLimit       : BOOL;
    bMaxLimit       : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des Funktionsbausteins.

bMinLimit : Ein TRUE an diesem Ausgang signalisiert, dass der Ausgang die untere Grenze erreicht hat.

bMaxLimit : Ein TRUE an diesem Ausgang signalisiert, dass der Ausgang die obere Grenze erreicht hat.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIMITER_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_LIMITER_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME      := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME      := T#0ms; (* task
cycle time [TIME] *)
    fMinOutput          : FLOAT;
    fMaxOutput          : FLOAT;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

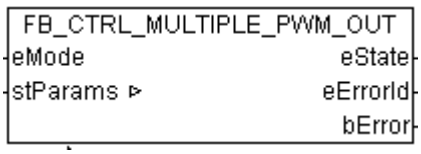
fMinOutput : Unteres Limit, auf das der Ausgang begrenzt wird.

fMaxOutput : Oberes Limit, auf das der Ausgang begrenzt wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

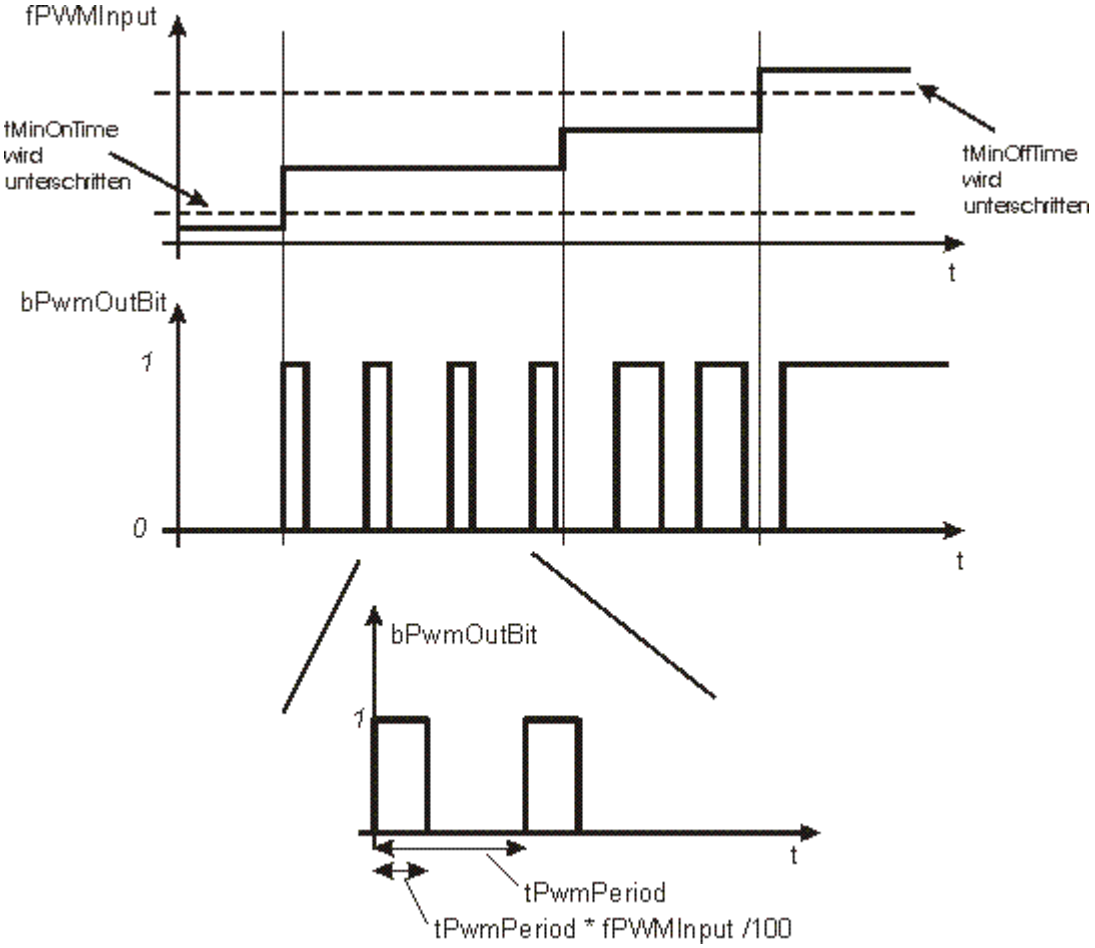
5.8.3 FB_CTRL_MULTIPLE_PWM_OUT



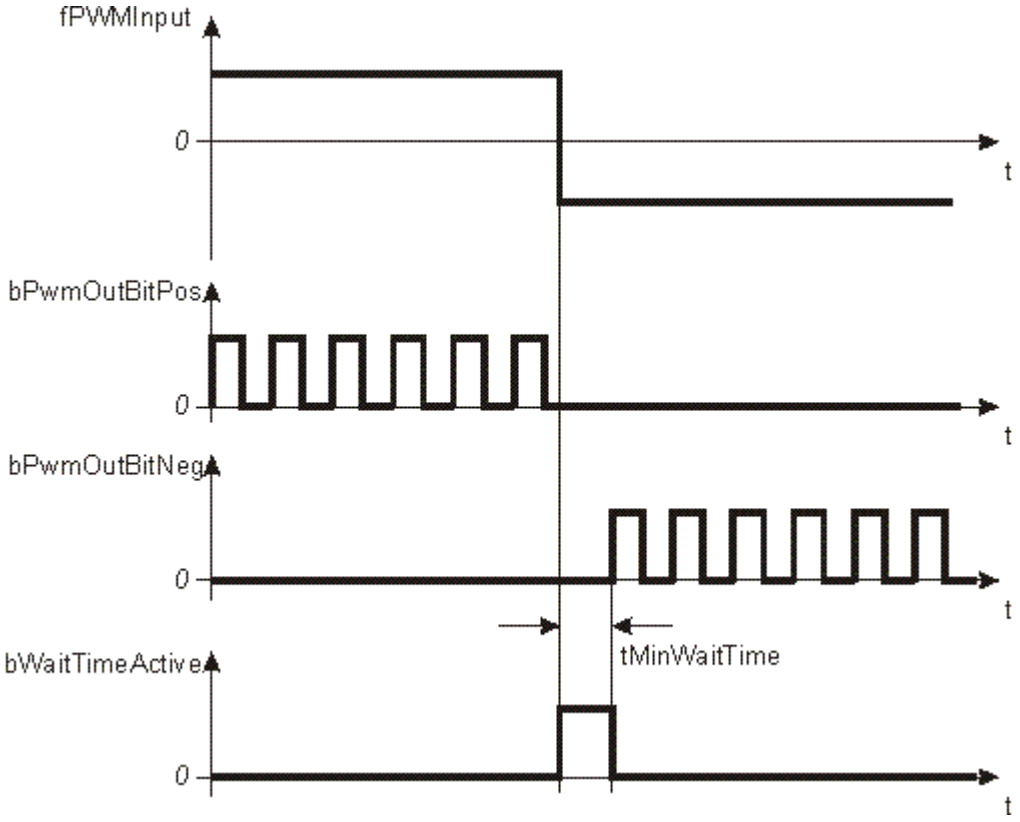
Dieser Baustein erzeugt aus mehreren Eingangssignalen PWM modulierte Ausgangssignale, wobei die Ausgangssignale zeitlich zueinander so angeordnet werden, dass möglichst wenige Ausgänge zeitgleich eingeschaltet sind. Durch dieses zeitliche Anordnung wird das für die Stellglieder benötigte Leistungsmaximum reduziert.

Bei diesem erweiterten Baustein kann neben dem Puls-Pausen-Verhältnis auch die minimale Einschaltdauer und die minimale Ausschaltdauer parametriert werden.

Beschreibung des Ausgangsverhaltens (1):



Beschreibung des Ausgangsverhaltens (2):



Um diesen Funktionsbaustein nutzen zu können, müssen vom Programmierer in der SPS die folgenden Arrays angelegt werden:

```
    ar_fPwmInput          :
ARRAY[1..nNumberOfPwmOutputs] OF FLOAT;
    ar_stWaitTimesConfig      : ARRAY[1..nNumberOfPwmOutputs] OF
ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
    ar_stPwmOutputs       : ARRAY[1..nNumberOfPwmOutputs] OF
ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
    ar_stPwmDataVars      : ARRAY[1..nNumberOfPwmOutputs] OF
ST_CTRL_MULTIPLE_PWM_OUT_DATA;
```

In das Array ar_fPwmInput werden die Eingangsgrößen für die einzelnen Kanäle des PWM-Bausteins geschrieben. In dem Array ar_stWaitTimesConfig kann der Programmierer die parametrierbaren Zeiten für die einzelnen Kanäle ablegen. Die Ausgangsbits werden von dem Funktionsbaustein in das Array ar_stPwmOutputs geschrieben. Die internen Daten, die der Baustein benötigt, werden in dem Array ar_stPwmDataVars abgelegt. Die in dem letzt genannten Array enthaltenen Strukturen dürfen innerhalb des SPS-Programms keinesfalls geändert werden. Diese Vorgehensweise wird auch in dem unten aufgeführten Beispielprogramm dargestellt.

VAR_INPUT

```
VAR_INPUT
    eMode       : E_CTRL_MODE;
END_VAR
```

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eState      : E_CTRL_STATE;
    bError      : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PWM-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_MULTIPLE_PWM_OUT_PARAMS :
STRUCT
    tTaskCycleTime       : TIME; (* PLC/PWM cycle time in
seconds *)
    tPWMPeriod           : TIME; (* controller cycle time
in seconds *)
    nNumberOfPwmOutputs      : USINT;
    pPwmInputArray_ADR       : POINTER TO FLOAT := 0;
    nPwmInputArray_SIZEOF    : UINT;
    pPwmTimesConfig_ADR      : POINTER TO
ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
    nPwmTimesConfig_SIZEOF   : UINT;
    pPwmOutputArray_ADR      : POINTER TO
ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
    nPwmOutputArray_SIZEOF   : UINT;
    pPwmData_ADR         : POINTER TO
ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    nPwmData_SIZEOF      : UINT;
END_STRUCT
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tPWMPeriod : Periodendauer des PWM-Signals.

nNumberOfPwmOutputs : Anzahl der PWM-Ausgänge. [1...n]

pPwmInputArray_ADR : Adresse des PWM-Input-Arrays.

nPwmInputArray_SIZEOF : Größe des PWM-Input-Arrays in Bytes.

pPwmTimesConfig_ADR : Adresse des PWM-Times-Arrays.

nPwmTimesConfig_SIZEOF : Größe des PWM-Times-Arrays in Bytes.

pPwmData_ADR : Adresse des internen PWM-Data-Arrays.

pPwmData_SIZEOF : Größe des internen PWM-Data-Arrays in Bytes.

```
TYPE
ST_CTRL_MULTIPLE_PWM_OUT_TIMES :
STRUCT
    tMinOnTime      : TIME; (* min. switch on time *)
    tMinOffTime     : TIME; (* min. switch off time *)
    tMinWaitTime    : TIME; (* min. waiting time when switching from
pos to neg or vv*)
END_STRUCT
END_TYPE
```

tMinOnTime : Minimale Einschaltdauer des PWM-Ausgangs.

tMinOffTime : Minimale Ausschaltdauer des PWM-Ausgangs..

tMinWaitTime : Wartezeit zwischen den Umschaltvorgängen zwischen einem positiven und negativen Ausgangsignal.

```
TYPE
ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS :
STRUCT
    bPwmOutBitPos      : BOOL;
    bPwmOutBitNeg      : BOOL;
    bWaitTimeActive    : BOOL;
END_STRUCT
END_TYPE
```

bPwmOutBitPos : PWM-Signal, wenn fPwmInput > 0.0.

bPwmOutBitNeg : PWM-Signal, wenn fPwmInput < 0.0.

bWaitTimeActive : Ein TRUE an diesem Ausgang signalisiert, dass die Wartezeit zwischen dem Umschalten der Ausgangssignale aktiv ist. Dieser Ausgang kann dazu verwendet werden, einen eventuell vorhandenen I-Anteil in dem vorgeschalteten Regler für diese Zeit konstant zu halten.

```
TYPE
ST_CTRL_MULTIPLE_PWM_OUT_DATA :
STRUCT
    Interne Struktur. Auf diese darf nicht schreibend
zugegriffen werden.
END_STRUCT
END_TYPE
```

Beispiel:

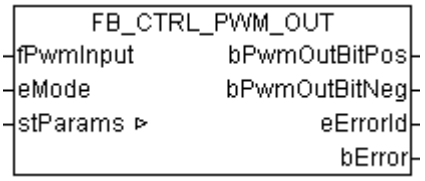
```
PROGRAM PRG_MULTIPLE_PWM_OUT_TEST
VAR CONSTANT
    nNumberOfPwmOutputs : USINT := 3;
END_VAR
VAR
    ar_fPwmInput          : ARRAY[1..nNumberOfPwmOutputs] OF
FLOAT;
    ar_stWaitTimesConfig      : ARRAY[1..nNumberOfPwmOutputs] OF
ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
    ar_stPwmOutputs       : ARRAY[1..nNumberOfPwmOutputs] OF
ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
    ar_stPwmDataVars      : ARRAY[1..nNumberOfPwmOutputs] OF
ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    eMode             : E_CTRL_MODE;
    stParams          :
ST_CTRL_MULTIPLE_PWM_OUT_PARAMS;
    eErrorId          : E_CTRL_ERRORCODES;
```

```
    bError           : BOOL;
    fbPwm_Out        : FB_CTRL_MULTIPLE_PWM_OUT;
    bInit            : BOOL := TRUE;
    fIn1             : FLOAT;
    fIn2             : FLOAT;
    fIn3             : FLOAT;
    bOut1_pos        : BOOL;
    bOut1_neg        : BOOL;
    bOut2_pos        : BOOL;
    bOut2_neg        : BOOL;
    bOut3_pos        : BOOL;
    bOut3_neg        : BOOL;
END_VAR
IF bInit
THEN
    (* set values in the local arrays *)
    ar_stWaitTimesConfig[1].tMinOnTime   := T#500ms;
    ar_stWaitTimesConfig[1].tMinOffTime  := T#300ms;
    ar_stWaitTimesConfig[1].tMinWaitTime := T#3.5s;
    ar_stWaitTimesConfig[2].tMinOnTime   := T#400ms;
    ar_stWaitTimesConfig[2].tMinOffTime  := T#250ms;
    ar_stWaitTimesConfig[2].tMinWaitTime := T#4.5s;
    ar_stWaitTimesConfig[3].tMinOnTime   := T#400ms;
    ar_stWaitTimesConfig[3].tMinOffTime  := T#200ms;
    ar_stWaitTimesConfig[3].tMinWaitTime := T#5.5s;
    (* set values in the parameter struct *)
    stParams.tTaskCycleTime     := T#2ms;
    stParams.tPWMPeriod         := T#2s;
    stParams.nNumberOfPwmOutputs    := nNumberOfPwmOutputs;
    (* set the ctrl-mode *)
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
(* set the addresses *)
stParams.pPwmTimesConfig_ADR    := ADR(
ar_stWaitTimesConfig);
stParams.nPwmTimesConfig_SIZEOF := SIZEOF(
ar_stWaitTimesConfig);
stParams.pPwmInputArray_ADR     := ADR( ar_fPwmInput );
stParams.nPwmInputArray_SIZEOF  := SIZEOF( ar_fPwmInput );
stParams.pPwmOutputArray_ADR    := ADR( ar_stPwmOutputs );
stParams.nPwmOutputArray_SIZEOF := SIZEOF( ar_stPwmOutputs );
stParams.pPwmData_ADR       := ADR( ar_stPwmDataVars );
stParams.nPwmData_SIZEOF    := SIZEOF( ar_stPwmDataVars
);
ar_fPwmInput[1] := fIn1;
ar_fPwmInput[2] := fIn2;
ar_fPwmInput[3] := fIn3;
fbPwm_Out( eMode     := eMode,
       stParams  := stParams,
       bError    => bError,
       eErrorId  => eErrorId);
bOut1_pos := ar_stPwmOutputs[1].bPwmOutBitPos;
bOut1_neg := ar_stPwmOutputs[1].bPwmOutBitNeg;
bOut2_pos := ar_stPwmOutputs[2].bPwmOutBitPos;
bOut2_neg := ar_stPwmOutputs[2].bPwmOutBitNeg;
bOut3_pos := ar_stPwmOutputs[3].bPwmOutBitPos;
bOut3_neg := ar_stPwmOutputs[3].bPwmOutBitNeg;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.4 FB_CTRL_PWM_OUT

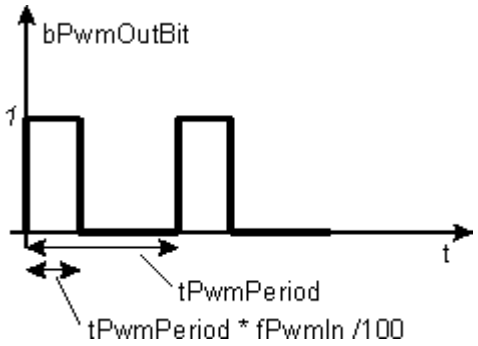


Dieser Baustein erzeugt aus dem Eingangssignal ein PWM-moduliertes Signal.

Beschreibung des Ausgangsverhaltens:

Dieser Baustein erzeugt an den Ausgängen ein PWM Signal mit einem Puls-Pausen-Verhältnis, welches dem Eingang **fPwmInput** entspricht. Das Puls-Pausen-Verhältnis wird am Eingang in % angegeben, wobei ein Wertebereich von -100% bis 100% zur Verfügung steht. Wenn ein positiver Wert angegeben wird, wird das PM-modulierte Signal an dem Ausgang **bPwmOutBitPos** ausgegeben. Bei einem negativ vorgegebenen Puls-Pausenverhältnis wird es an dem Ausgang **bPwmOutBitNeg** ausgegeben. Mit diesen zwei Signalen besteht somit die Möglichkeit, vorzeichenabhängig zwei Stellglieder anzusteuern.

Mit dem Parameter **bInstantPWMUpdate** = TRUE kann eine instantane Übernahme einer neuen Eingangsgröße aktiviert werden. D.h., der neue Eingangswert wird sofort im aktuellen PWM-Zyklus wirksam. Wenn dieser Parameter FALSE ist, erfolgt eine Übernahme der Eingangsgröße erst zu Beginn eines neuen PWM-Zyklusses.



VAR_INPUT

```
VAR_INPUT
    fPwmInput       : FLOAT;        (* controller output = PMW input [-100.0 ... 100.0] *)
    eMode       : E_CTRL_MODE;
END_VAR
```

fPwmInput : Eingangsgröße.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    bPwmOutBitPos   : BOOL;         (* PWM output bit *)
    bPwmOutBitNeg   : BOOL;         (* PWM output bit *)
    eState      : E_CTRL_STATE;
    bError      : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

bPwmOutBitPos : PWM-Signal, wenn fPwmInput > 0.0 ist.

bPwmOutBitNeg : PWM-Signal, wenn fPwmInput < 0.0 ist.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PWM_OUT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PWM-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PWM_OUT_PARAMS:
STRUCT
     tTaskCycleTime         : TIME     (* task cycle time
[TIME] *)
     tPWMPeriod         : TIME;    (* PWM period
duration [TIME] *)
     bInstantPWMUpdate      : BOOL;
END_STRUCT
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

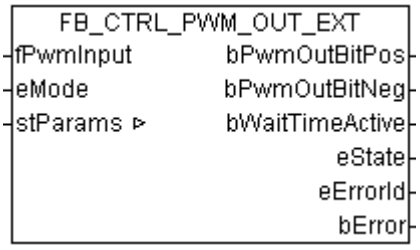
tPWMPeriod : Periodendauer des PWM-Signals.

bInstantPWMUpdate : Wenn dieses Flag TRUE ist, wird eine neue Eingangsgröße sofort im aktuellen PWM-Zyklus übernommen.

Voraussetzungen

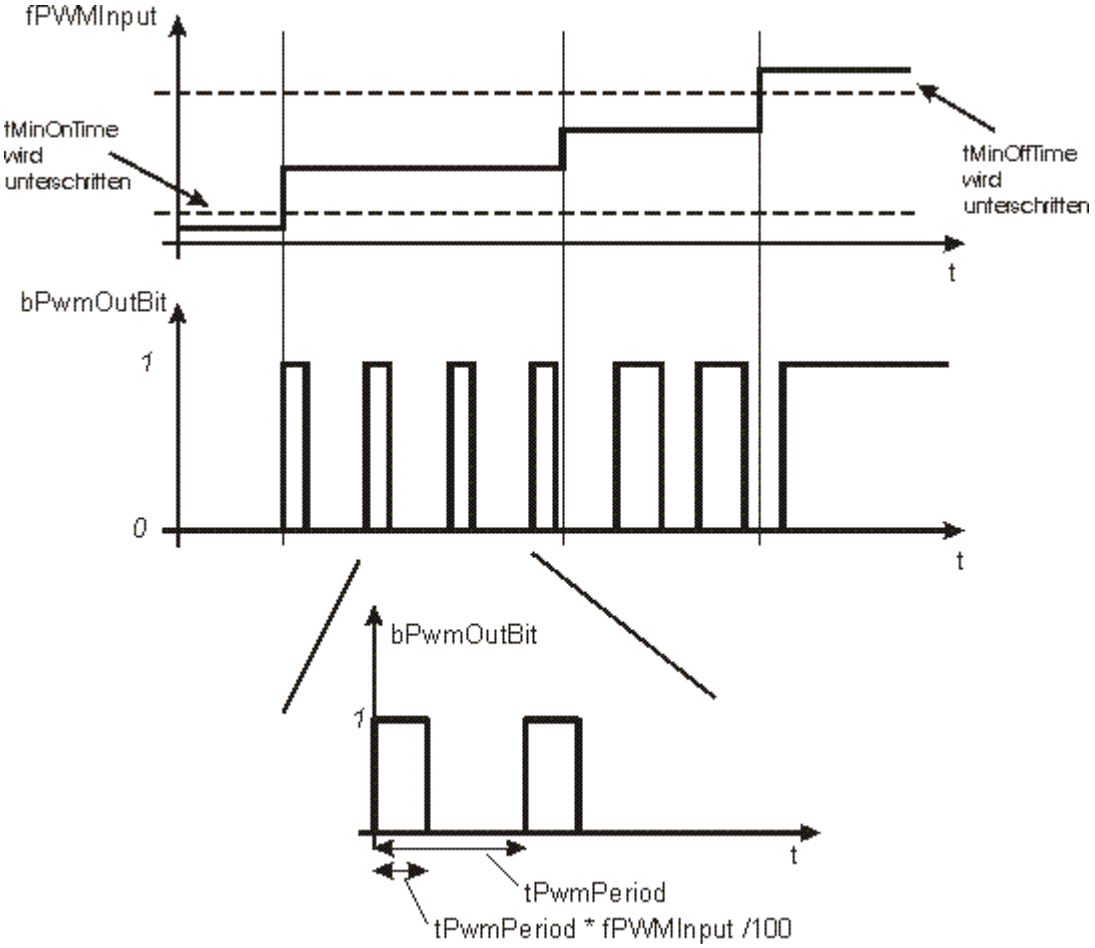
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.5 FB_CTRL_PWM_OUT_EXT

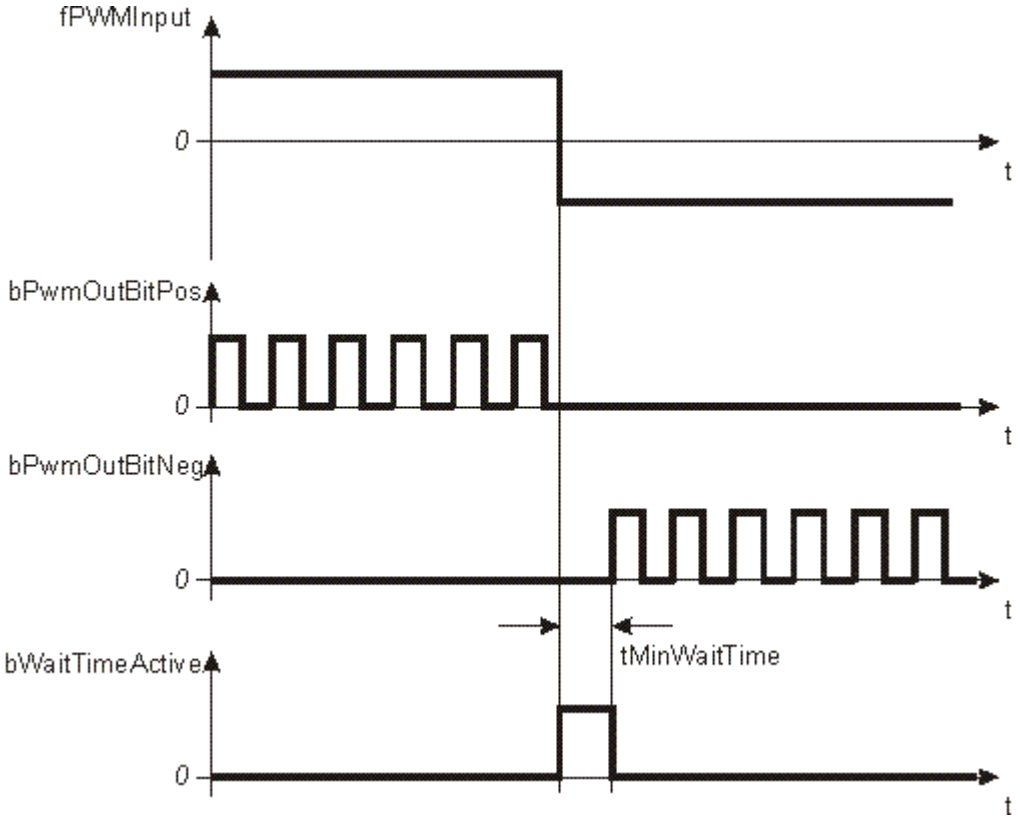


Dieser Baustein erzeugt aus dem Eingangssignal ein PWM-moduliertes Signal. Bei diesem erweiterten Baustein kann neben dem Puls-Pausen-Verhältnis auch die minimale Einschaltdauer und die minimale Ausschaltdauer parametriert werden.

Beschreibung des Ausgangsverhaltens (1):



Beschreibung des Ausgangsverhaltens (2):



VAR_INPUT

```
VAR_INPUT
    fPwmInput       : FLOAT;        (* NEW: controller output = PMW input [-100.0 ... 100.0] *)
    eMode       : E_CTRL_MODE;
END_VAR
```

fPwmInput : Eingangsgröße des Funktionsbausteins.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    bPwmOutBitPos   : BOOL;         (* PWM output bit *)
    bPwmOutBitNeg   : BOOL;         (* PWM output bit *)
    eState      : E_CTRL_STATE;
    bError      : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

bPwmOutBitPos : PWM-Signal, wenn fPwmInput > 0.0.

bPwmOutBitNeg : PWM-Signal, wenn fPwmInput < 0.0.

bWaitTimeActive : Ein TRUE an diesem Ausgang signalisiert, dass die Wartezeit zwischen dem Umschalten der Ausgangssignale aktiv ist. Dieser Ausgang kann dazu verwendet werden, einen eventuell vorhandenen I-Anteil in dem vorgeschalteten Regler für diese Zeit konstant zu halten.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des PWM-Glieds. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PWM_OUT_EXT_PARAMS :
STRUCT
    tTaskCycleTime   : TIME; (* PLC/PWM cycle time in seconds
*)
    tPWMPeriod       : TIME; (* controller cycle time in
seconds *)
    tMinOnTime       : TIME; (* min. switch on time *)
    tMinOffTime      : TIME; (* min. switch off time *)
    tMinWaitTime     : TIME; (* min. waiting time when
switching from pos to neg or vv*)
END_STRUCT
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tPWMPeriod : Periodendauer des PWM-Signals.

tMinOnTime : Minimale Einschaltdauer.

tMinOffTime : Minimale Ausschaltdauer.

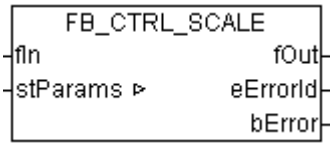
tMinWaitTime : Wartezeit zwischen den Umschaltvorgängen zwischen einem positiven und negativen Ausgangsignal.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.6 FB_CTRL_SCALE



Mit diesem Funktionsbaustein ist es möglich, eine lineare Signalanpassung zwischen zwei Wertebereichen vorzunehmen.

VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

fIn : Eingangsgröße des Funktionsbausteins.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Skalierte Ausgangsgröße.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_SCALE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_SCALE_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME     := T#0ms; (* controller cycle
time [TIME] *)
    tTaskCycleTime     : TIME     := T#0ms; (* task cycle time
[TIME] *)
    fInMin          : FLOAT;
    fInMax          : FLOAT;
    fOutMin         : FLOAT;
    fOutMax         : FLOAT;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

fInMin : Minimum der Eingangsgröße.

fInMax : Maximum der Eingangsgröße.

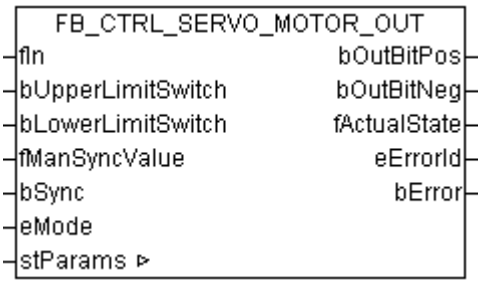
fOutMin : Minimum der Ausgangsgröße.

fOutMax : Maximum der Ausgangsgröße.

Voraussetzungen

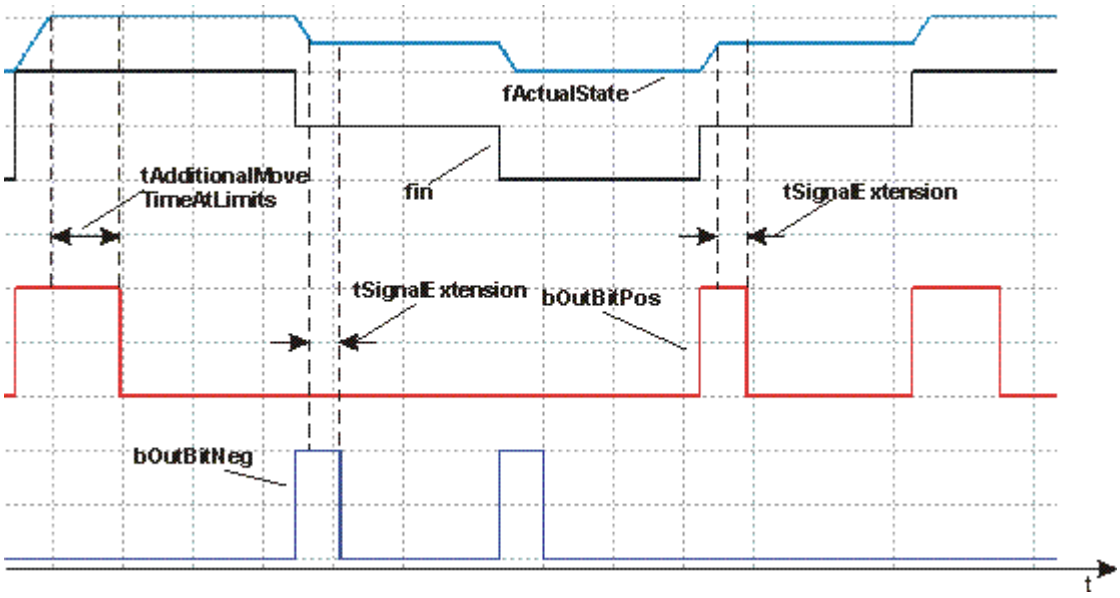
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.7 FB_CTRL_SERVO_MOTOR_OUT



Dieser Funktionsbaustein erzeugt Impulse, mit denen ein Stellmotor auf eine definierte Position gefahren werden kann.

Verhalten des Ausgangs:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT; (* controller output = SERVO_MOTOR_OUT input [ fCtrlOutMin ... fCtrlOutMax
] *)
    bUpperLimitSwitch       : BOOL;
    bLowerLimitSwitch       : BOOL;
    fManSyncValue    : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Stellgröße des Reglers im Intervall [fCtrlOutMin ... fCtrlOutMax] (Reglerausgang).

bUpperLimitSwitch : Endschalter: TRUE wenn der obere Anschlag erreicht ist.

bLowerLimitSwitch : Endschalter: TRUE wenn der untere Anschlag erreicht ist.

fManSyncValue : Eingang, mit dem der interne Zustand der aktuellen Motorstellung angepasst werden kann, oder auf dessen Wert im Manual-Mode gefahren wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird die interne Motorposition, die der tatsächlichen entsprechen muss, auf den Wert fManSyncValue gesetzt.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

Dieser Baustein besitzt zur Synchronisation mit der aktuellen Ventilstellung den Mode "eCTRL_MODE_SYNC_MOVEMENT". In diesem Mode wird der Ausgang zum Schließen des Ventils so lange gesetzt, bis das Ventil sicher geschlossen ist. Daran anschließend wird der Baustein auf diese Ventilstellung synchronisiert.

Wenn der Synchronisierungsvorgang abgeschlossen ist, wird automatisch in den State "eCTRL_STATE_ACTIVE" geschaltet!

VAR_OUTPUT

```
VAR_OUTPUT
    bOutBitPos      : BOOL; (* PWM output bit *)
    bOutBitNeg      : BOOL; (* PWM output bit *)
    fActualState    : FLOAT; (* Actual state of the motor [ 0% ... 100%] *)
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bOutBitPos : Ausgang, um den Motor in positiver Richtung zu verfahren.

bOutBitNeg : Ausgang, um den Motor in negativer Richtung zu verfahren.

fActualState : Aktuelle Motorstellung im Intervall [fCtrlOutMin ... fCtrlOutMax], in dem der sich der Motor aktuell befindet.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_SERVO_MOTOR_OUT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_SERVO_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime            : TIME     := T#0ms;
(* controller cycle time [TIME] *)
    tTaskCycleTime            : TIME     := T#0ms;
(* task cycle time [TIME] *)
    tMovingTime               : TIME;
(* time to move from closed (e.g. 0%) to open (e.g. 100%)*)
    tSignalExtension          : TIME;
(* output signal extension time *)
    tAdditionalMoveTimeAtLimits       : TIME;
(* add these moving time to ensure that the limits are
reached*)
    tMinWaitTimeBetweenDirectionChange: TIME;
(* wait time between pos and neg output pulses and vice versa
*)
    tMinimumPulseTime         : TIME;
(* minimum output pulse length *)
    bMoveOnLimitSwitch        : BOOL;
```

```
    bStopAdditionalMoveTimeIfInputValueIsChanged : BOOL;
(* if an additional move on the limits is active, this will be
stopped if the input value is changed *)
    fCtrlOutMax               : FLOAT := 100.0;
(* controller output to move to the max limit *)
    fCtrlOutMin               : FLOAT := 0.0;
(* controller output to move to the min limit *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen in aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tMovingTime : Die Zeit, die benötigt wird, um das Stellglied von einem Anschlag zum anderen zu fahren.

tSignalExtension : Signalverlängerung, um die jeder Ausgangsimpuls verlängert wird, um eine Totzeit zu kompensieren.

tAdditionalMovingTimeAtLimits : Signalverlängerung, die zum sicheren Erreichen der Grenzen zusätzlich ausgegeben wird, wenn das Stellglied auf +/-100% gefahren werden soll. Wirkt nur, wenn bMoveOnLimitSwitch FALSE ist.

tMinWaitTimeBetweenDirectionChange : Minimale Wartezeit zwischen positiven und negativen Ausgangsimpulsen.

tMinimumPulseTime : Minimale Länge eines Ausgangsimpulses.

bMoveOnLimitSwitch : Wenn TRUE, wird bei einer Stellgröße von fCtrlOutMin oder fCtrlOutMax solange ein Signal ausgegeben, bis der entsprechende Endschalter erreicht wird.

bStopAdditionalMoveTimeIfInputValueIsChanged : Wenn dieses Flag TRUE ist, wird das Fahren des Ventils auf die Endlage, welches durch die Zeit "Additional Moving Time At Limits" verursacht wird, gestoppt, wenn eine Eingangsgröße ungleich der Endlage vorgegeben wird. Wenn dieses Flag FALSE ist, wird bei einer Eingangsgröße, die einer Endlage entspricht, immer erst sicher auf die Endlage gefahren, bevor wieder eine andere Ventilstellung angefahren werden kann.

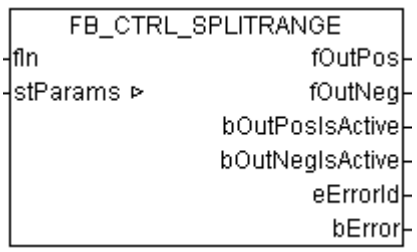
fCtrlOutMax : Stellgröße, bei der das Ventil auf 100% gefahren wird.

fCtrlOutMin : Stellgröße, bei der das Ventil auf 0% gefahren wird.

Voraussetzungen

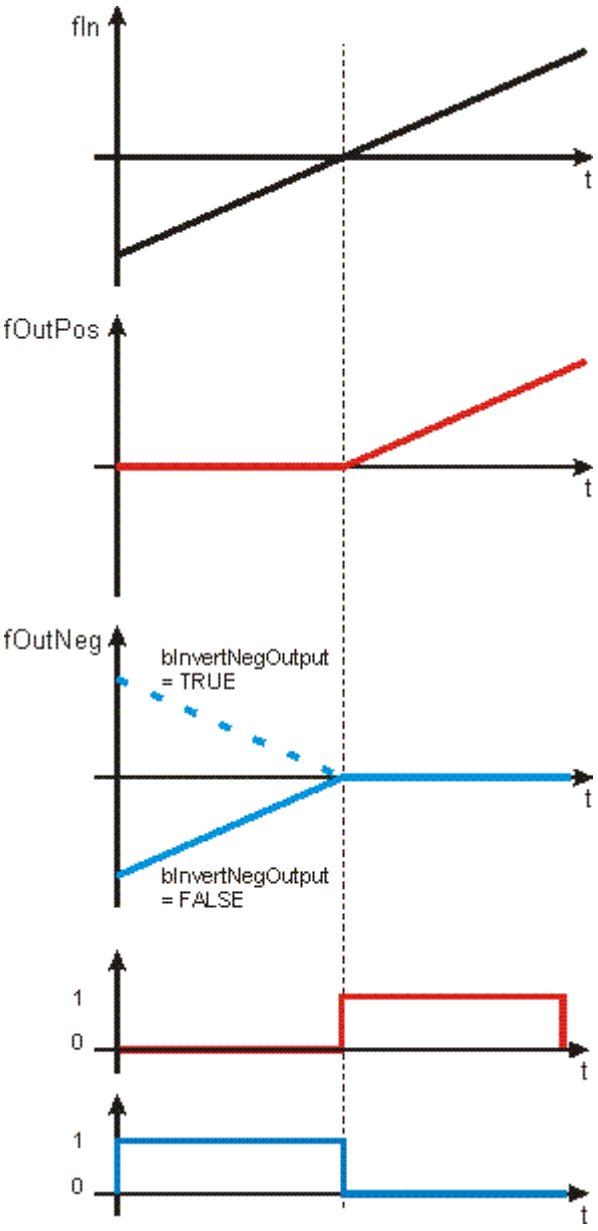
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.8 FB_CTRL_SPLITRANGE



Dieser Baustein zerlegt ein Eingangssignal in einen positiven und einen negativen Anteil. Mit den Parametern bDisablePosOut und bDisableNegOut kann der positive oder negative Ausgang deaktiviert werden → Heizbetrieb nur im Winter, Kühlbetrieb nur im Sommer. Der Parameter bInvertNegOutput ermöglicht das Invertieren des negativen Ausgangs.

Beschreibung des Ausgangsverhaltens:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

fIn : Eingangsgröße des Funktionsbausteins.

VAR_OUTPUT

```
VAR_OUTPUT
    fOutPos         : FLOAT;
    fOutNeg         : FLOAT;
    bOutPosIsActive     : BOOL;
    bOutNegIsActive     : BOOL;

    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOutPos : Positiver Teil von fIn.

fOutNeg : Negativer Teil von fIn.

bOutPosIsActive : Ein TRUE signalisiert, das fIn > 0.0 ist,

bOutNegIsActive : Ein TRUE signalisiert, das fIn < 0.0 ist,

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_SPLITRANGE_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_SPLITRANGE_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME      := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME      := T#0ms; (* task
cycle time [TIME] *)
    bInvertNegOutput        : BOOL;
    bDisablePosOut          : BOOL;
    bDisableNegOut          : BOOL;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

bInvertNegOutput : Wenn dieser Parameter TRUE ist, wird fOutNeg invertiert.

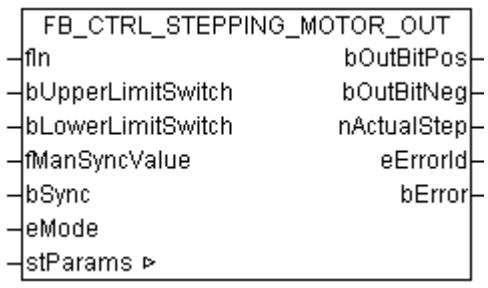
bDisablePosOut : Der Ausgang fOutPos wird deaktiviert und ist immer 0.0.

bDisableNegOut : Der Ausgang fOutNeg wird deaktiviert und ist immer 0.0.

Voraussetzungen

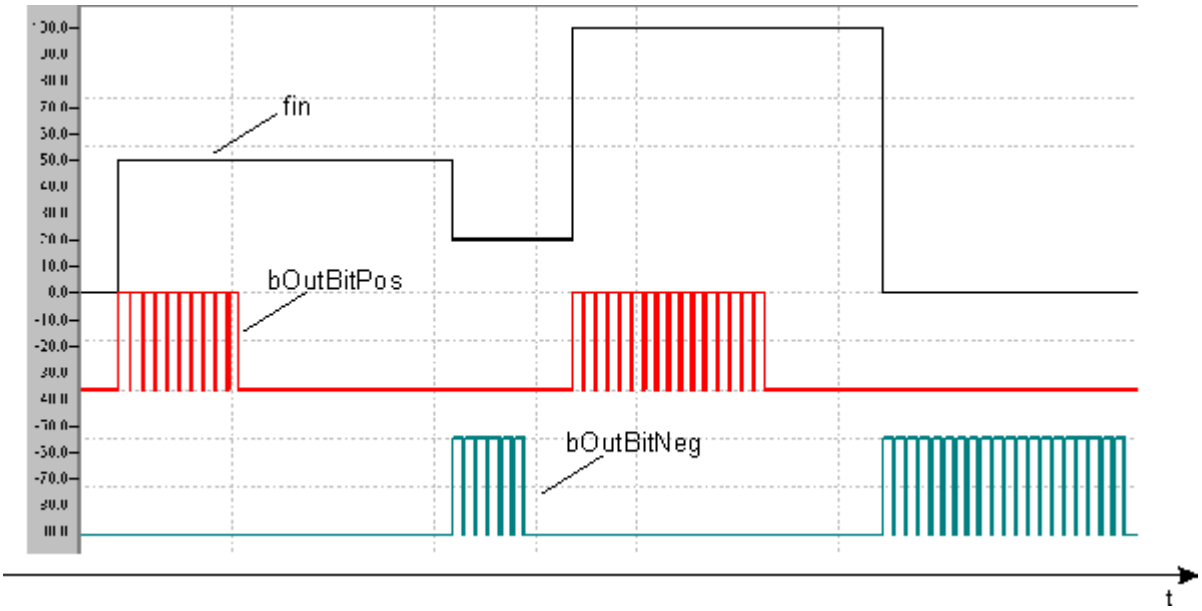
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.8.9 FB_CTRL_STEPPING_MOTOR_OUT



Dieser Funktionsbaustein erzeugt eine Stellgröße für einen Schrittmotor.

Verhalten des Ausgangs:



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT; (* controller output = STEPPING_MOTOR_OUT input [ fCtrlOutMin ... fCtrlOutM
ax ] *)
    bUpperLimitSwitch       : BOOL;
    bLowerLimitSwitch       : BOOL;
    fManSyncValue   : FLOAT;
    bSync       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fIn : Stellgröße des Reglers (Reglerausgang).

bUpperLimitSwitch : Endschalter: TRUE wenn der obere Anschlag erreicht ist.

bLowerLimitSwitch : Endschalter: TRUE wenn der untere Anschlag erreicht ist.

fManSyncValue : Eingang, mit dem der interne Zustand der Motorstellung angepasst werden kann, oder auf dessen Wert im Manual-Mode gefahren wird.

bSync : Mit einer steigenden Flanke an diesem Eingang wird der interne Schrittzähler auf den Schritt gesetzt, der dem Wert fManSyncValue entspricht.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

Dieser Baustein besitzt zur Synchronisation mit der aktuellen Ventilstellung den Mode "eCTRL_MODE_SYNC_MOVEMENT". In diesem Mode werden so lange Impulse zum Schließen des Ventils ausgegeben, bis das Ventil sicher geschlossen ist. Daran anschließend wird der Baustein auf diese Ventilstellung synchronisiert.

Wenn der Synchronisierungsvorgang abgeschlossen ist, wird automatisch in den State "eCTRL_STATE_ACTIVE" geschaltet!

VAR_OUTPUT

```
VAR_OUTPUT
    bOutBitPos      : BOOL; (* output signal to move to the maximum *)
    bOutBitNeg      : BOOL; (* output signal to move to the minimum*)
    nActualStep     : DINT; (* Actual state of the motor [ 0 ... nMaxMovingPulses ] *)
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

bOutBitPos : Ausgang, um den Motor in positiver Richtung zu verfahren.

bOutBitNeg : Ausgang, um den Motor in negativer Richtung zu verfahren.

nActualStep : Aktueller Schritt, in dem der Motor steht..

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_STEPPING_MOTOR_OUT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_STEPPING_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime         : TIME     := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime         : TIME     := T#0ms; (* task
cycle time [TIME] *)
    tOnTime            : TIME;          (*
impulse length *)
    tOffTime               : TIME;          (*
delay between two impulses *)
    nMaxMovingPulses           : DINT;          (*
necessary pulses to move from closed to open or v. v. *)
    nMinMovingPulses           : UINT := 0;     (* min
pulses to start movement *)
    bHoldWithOutputOn          : BOOL;          (* if
TRUE then a ouput is on when no pulses are generated *)
    nAdditionalPulsesAtLimits      : DINT;          (* add
these pulses to ensure that the limits are reached*)
    bMoveOnLimitSwitch         : BOOL;
    fCtrlOutMax            : FLOAT := 100.0;    (*
controller output to move to the max limit *)
    fCtrlOutMin            : FLOAT := 0.0;      (*
controller output to move to the min limit *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen in aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

tOnTime :Impulslänge.

tOffTime : Pausenlänge.

nMaxMovingPulses : Impulse, die benötigt werden, von einem Limit zum anderen zu fahren..

bHoldWithOutputOn : Wenn dieser Parameter auf TRUE gesetzt ist, bleibt im Stillstand des Antriebs ein Ausgang gesetzt. Dieser wird somit gebremst.

nAdditionalPulsesAtLimits : Impulse, die zum sicheren Erreichen der Grenzen zusätzlich ausgegeben werden.

bMoveOnLimitSwitch : Wenn TRUE, werden bei 0% oder 100% Stellgröße solange Impulse ausgegeben, bis ein Endschalter erreicht wird.

fCtrlOutMax : Stellgröße, bei der das Ventil auf 100% gefahren wird..

fCtrlOutMin : Stellgröße, bei der das Ventil auf 0% gefahren wird..

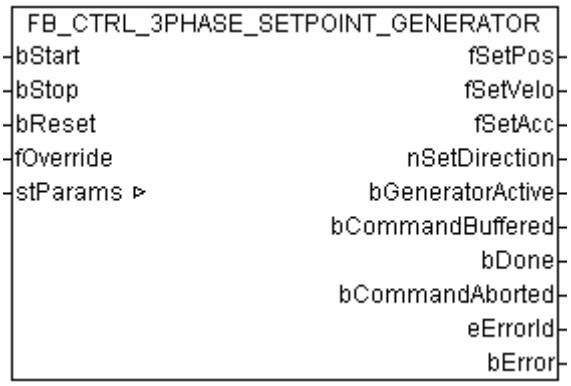
fMinCtrtlDeltaToStartMovement : Minimale Stellgrößendifferenz, die überschritten werden muss, damit der Motor verstellt wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.9 Setpointgeneration

5.9.1 FB_CTRL_3PHASE_SETPOINT_GENERATOR (nur auf einem PC-System)



Der Funktionsbaustein stellt einen 3-Phasen Sollwertgenerator dar.

Beschreibung:

Dieser Funktionsbaustein erzeugt ein 3-Phasen Sollwertprofil mit einem rechteckigen Beschleunigungsverlauf. Während der Generator aktiv ist, ist es möglich einen neuen Parametersatz vorzugeben. Je nach angegebenem Typ des Parametersatzes wird dieser sofort zur Wirkung gebracht (eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_Instant), oder alternativ wird erst die aktuelle Bewegung zu Ende geführt und dann eine neue Bewegung mit dem neuen Parametersatz gestartet. (eNewPosType := eCTRL_NEW_PARAMETER_TYPE_NotInstant).

HINWEIS

1. Bei Vorgabe eines neuen Parametersatzes ist es möglich, dass die alte Endposition überfahren wird. Siehe Beispiel.
2. Es wird im Allgemeinen immer empfohlen, dem Sollwertgenerator eine Endlagenüberwachung nachzuschalten.

VAR_INPUT

```
VAR_INPUT
    bStart      : BOOL;
    bStop       : BOOL;
    bReset      : BOOL;
    fOverride   : LREAL;
END_VAR
```

bStart : Mit einer positiven Flanke am Eingang *bStart* wird die Sollwertgenerierung gestartet, wenn der Generator nicht aktiv ist und die Eingänge *bStop* und *bReset* FALSE sind.

bStop : Mit einer positiven Flanke am Eingang *bStop* wird die Sollwertgenerierung gestoppt. Es wird mit der Verzögerung des aktuellen Parametersatzes gebremst und der eventuell gespeicherte Folgeauftrag gelöscht.

bReset : Reset des Sollwertgenerators. Eine eventuell aktive Positionierung wird sofort abgebrochen, die Ausgänge fSetVelo und fSetAcc werden zu 0.0, die Sollposition wird auf die Startposition gesetzt und die internen Zustände werden gelöscht.

fOverride : Mit dem Override im Intervall [0 .. 100.0 %] kann die Sollgeschwindigkeit skaliert werden. Bei einem Override von 100% wird ein Profil mit der im Parametersatz angegebenen Sollgeschwindigkeit generiert. Der hier implementierte Override berechnet bei einer Overrideänderung keine Skalierung der aktuellen Laufzeittabellen, sondern es wird ein interner Nachstartauftrag mit geänderter Sollgeschwindigkeit generiert. Die Auflösung des Overrides beträgt 0.1%.

VAR_OUTPUT

```
VAR_OUTPUT
    fSetPos         : LREAL; (* generated setpoint position *)
    fSetVelo        : LREAL; (* generated setpoint velocity *)
    fSetAcc         : LREAL; (* generated setpoint acceleration *)
    nSetDirection   : INT;   (* generated direction *)
     bCommandBuffered       : BOOL;
    bDone       : BOOL;
    bCommandAborted     : BOOL;

    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fSetPos : Sollposition

fSetVelo : Sollgeschwindigkeit

fSetAcc : Sollbeschleunigung

nSetDirection : Bewegungsrichtung [-1, 0, 1],
1 --> Bewegungsrichtung positiv
0 --> Generator inaktiv
-1 --> Bewegungsrichtung negativ

bGeneratorActive : Signalisiert, ob der Generator aktiv ist.

bCommandBuffered : Dieser Ausgang signalisiert mit einem TRUE, dass ein Fahrauftrag gespeichert ist, der nach dem aktuellen Auftrag gestartet wird.
Ein gespeicherter Auftrag wird gelöscht, wenn der folgende Sonderfall als Parametersatz angegeben wird:

```
          fAcceleration      := 0.0;
fDeceleration      := 0.0;
fStartPos      := 0.0;
fStartVelo     := 0.0;
fTargetPos     := 0.0;
fTargetVelo    := 0.0;
fVelocity      := 0.0;
tCtrlCycleTime     := T#0s;
tTaskCycleTime     := T#0s;
eNewParameterType  := eCTRL_NEW_PARAMETER_TYPE_NotInstant;
```

bDone : Dieser Ausgang wird TRUE, wenn die Bewegung abgeschlossen ist und die Zielpositon erreicht wurde.

bCommandAborted : Dieser Ausgang wird TRUE, wenn die aktuelle Bewegung abgebrochen wurde. Dieses kann beispielsweise durch eine steigende Flanke am Eingang *bStop* verursacht werden.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Sollwertgenerators. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime     : TIME;  (* task cycle time [TIME]
*)
    tCtrlCycleTime     : TIME;  (* controller cycle time [TIME]
*)
    fStartPos      : LREAL;
    fStartVelo     : LREAL;
    fVelocity      : LREAL; (* >= 0.0 *)
    fTargetPos     : LREAL;
    fTargetVelo    : LREAL;
    fAcceleration      : LREAL; (* > 0.0 *)
    fDeceleration      : LREAL; (* > 0.0 *)
    eNewParameterType  : E_CTRL_NEW_PARAMETER_TYPE;
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Task-Zyklus aufgerufen wird.

fStartPos : Startposition des Bewegungsprofils.

fStartVelo : Startgeschwindigkeit des Bewegungsprofils.

fVelocity : Geschwindigkeit in Einheiten / Sekunde.

fTargetPos : Zielposition des Bewegungsprofils.

fTargetVelo : Zielgeschwindigkeit des Bewegungsprofils.

Hinweis **Die Zielgeschwindigkeit bleibt nach dem Erreichen der Zielposition (setzen des bDone Flags) anstehen, die Position wird aber ab diesem Zeitpunkt nicht weiter berechnet (konstante Position bei einer Geschwindigkeit ≠ 0.0).**

fAcceleration : Beschleunigung in Einheiten / Sekunde^2.

fDeceleration : Verzögerung in Einheiten / Sekunde^2.

eNewParameterType :

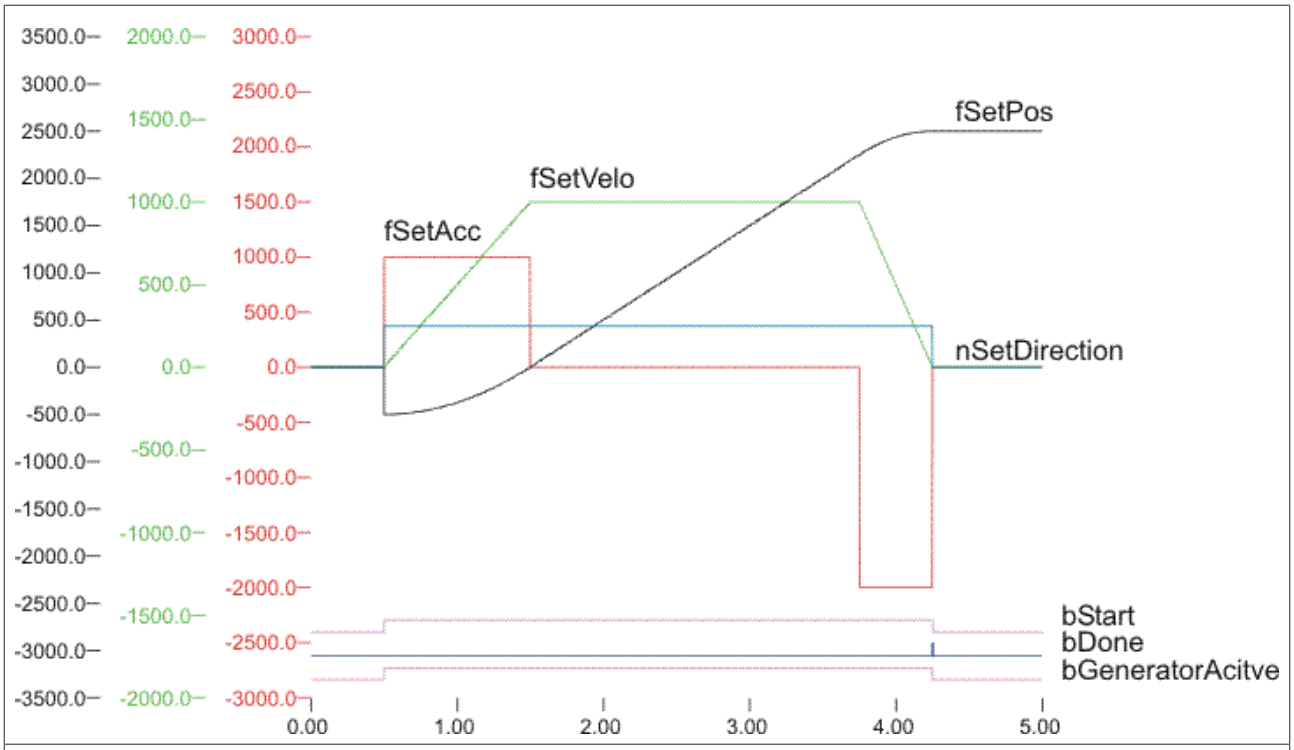
```
           TYPE E_CTRL_NEW_PARAMETER_TYPE :
(
    eCTRL_NEW_PARAMETER_TYPE_NotInstant := 0,
    eCTRL_NEW_PARAMETER_TYPE_Instant    := 1
);
END_TYPE
```

eCTRL_NEW_PARAMETER_TYPE_Instant: Wenn ein Nachstartauftrag mit einem neuen Parametersatz erfolgt, wird dieser sofort übernommen. D. h., aus dem aktuellen Bewegungszustand wird eine Transition auf die Daten des neuen Parametersatzes berechnet, wobei die alten Parameter verworfen werden.

eCTRL_NEW_PARAMETER_TYPE_NotInstant: Wenn ein Nachstartauftrag mit einem neuen Parametersatz erfolgt, wird dieser nicht sofort übernommen. D. h., die aktuelle Bewegung wird erst zu Ende ausgeführt und danach wird mit den neuen Parametern auf das neue Ziel positioniert. Ein TRUE an dem Ausgang bCommandBuffered zeigt an, dass ein nicht instantaner Nachstartauftrag gespeichert ist. Ein bereits gespeicherter Auftrag kann durch einen anderen neuen nicht instantanen Parametersatz überschrieben oder gelöscht werden.

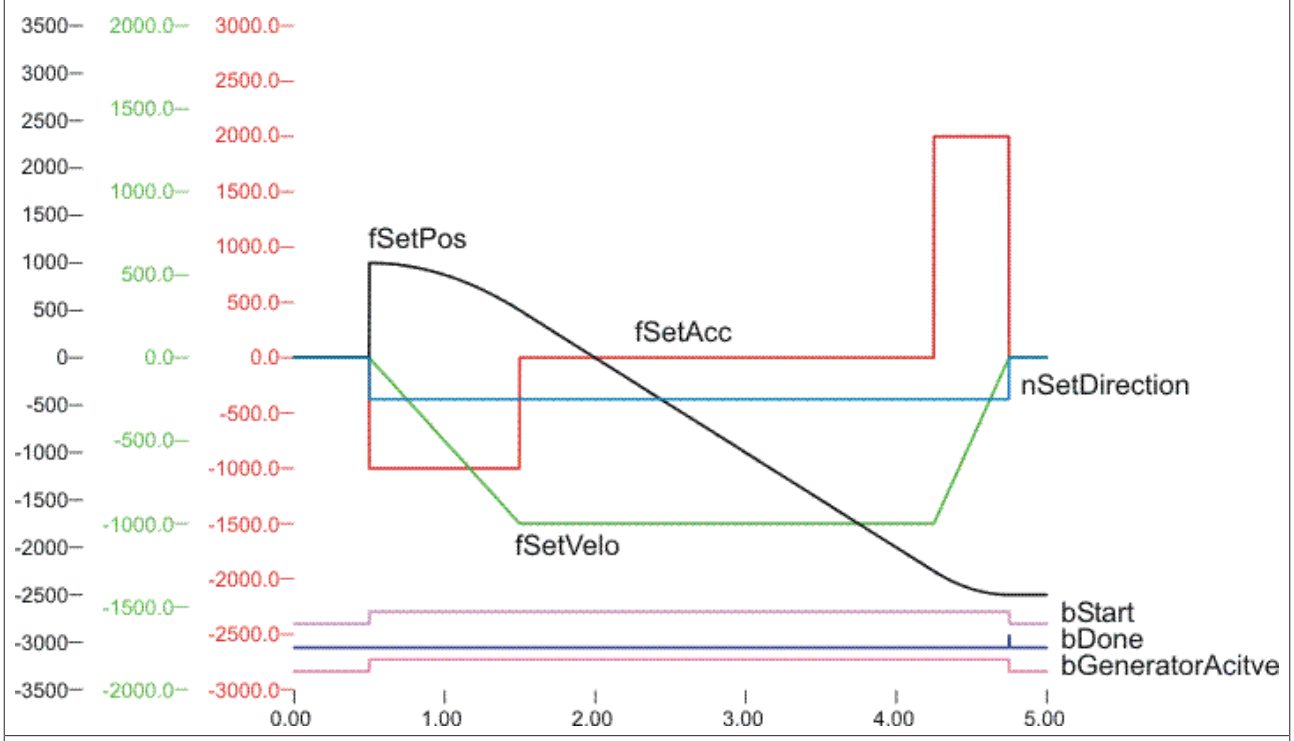
Beispielpositionierungen

Beispielpositionierung 1:	`stParams.fStartPos := -500.0;`
	`stParams.fTargetPos := 2500.0;`
	`stParams.fStartVelo := 0.0;`
	`stParams.fVelocity := 1000.0;`
	`stParams.fTargetVelo := 0.0;`
	`stParams.fAcceleration := 1000.0;`
	`stParams.fDeceleration := 2000.0;`
	`fOverride := 100.0;`



Beispielpositionierung 2:

```
stParams.fStartPos := 1000.0;
stParams.fTargetPos := -2500.0;
stParams.fStartVelo := 0.0;
stParams.fVelocity := 1000.0;
stParams.fTargetVelo := 0.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 2000.0;
fOverride := 100.0;
```



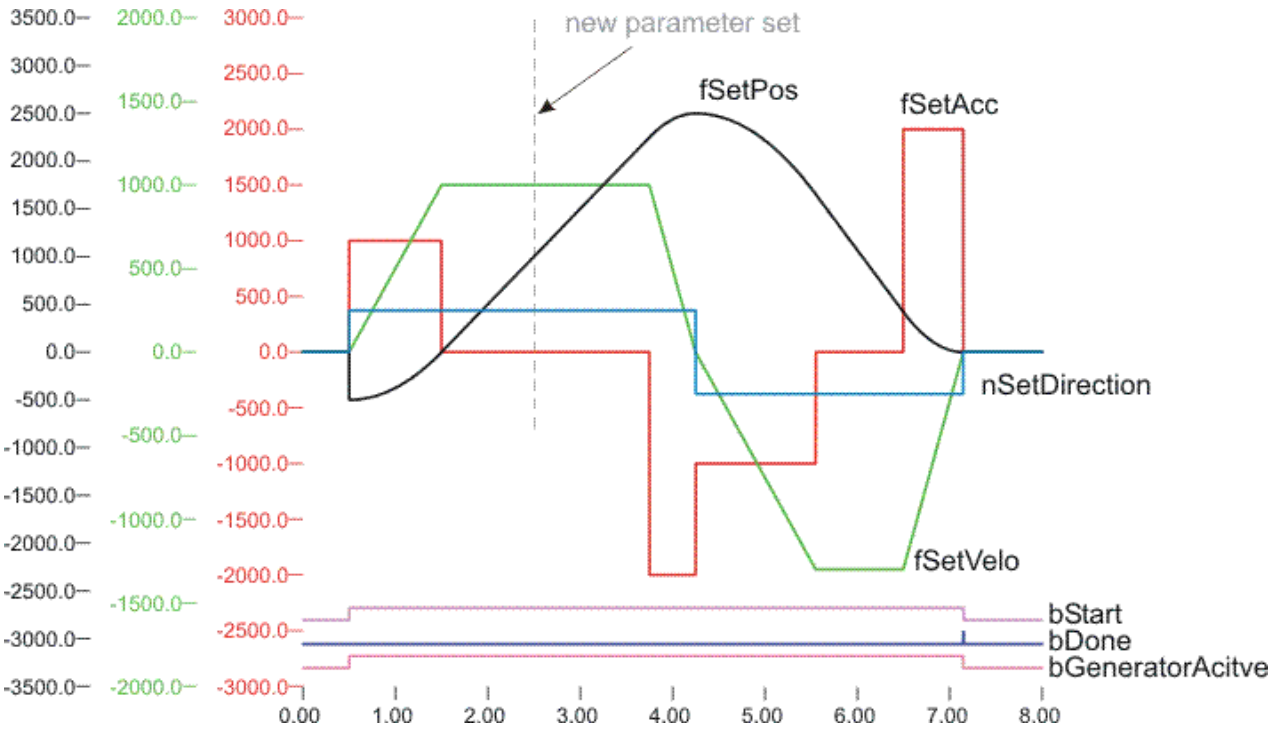
Beispielpositionierung 3:

```
stParams.fStartPos := -500.0;
stParams.fTargetPos := 2500.0;
stParams.fStartVelo := 0.0;
stParams.fVelocity := 1000.0;
stParams.fTargetVelo := 0.0;
stParams.fAcceleration := 1000.0;
```

```
stParams.fDeceleration := 2000.0; stParams.eNewParameterType :=
eCTRL_NEW_PARAMETER_TYPE_NotInstant;
fOverride := 100.0;
```

Parameteränderung wenn fSetPos > 1000.0, eNewPosType := eCTRL_NEW_POS_TYPE_NotInstant

```
stParams.fTargetPos := 0.0;
stParams.fStartVelo := 0.0;
stParams.fVelocity := 1300.0;
stParams.fTargetVelo := 0.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 2000.0;
fOverride := 100.0;
```

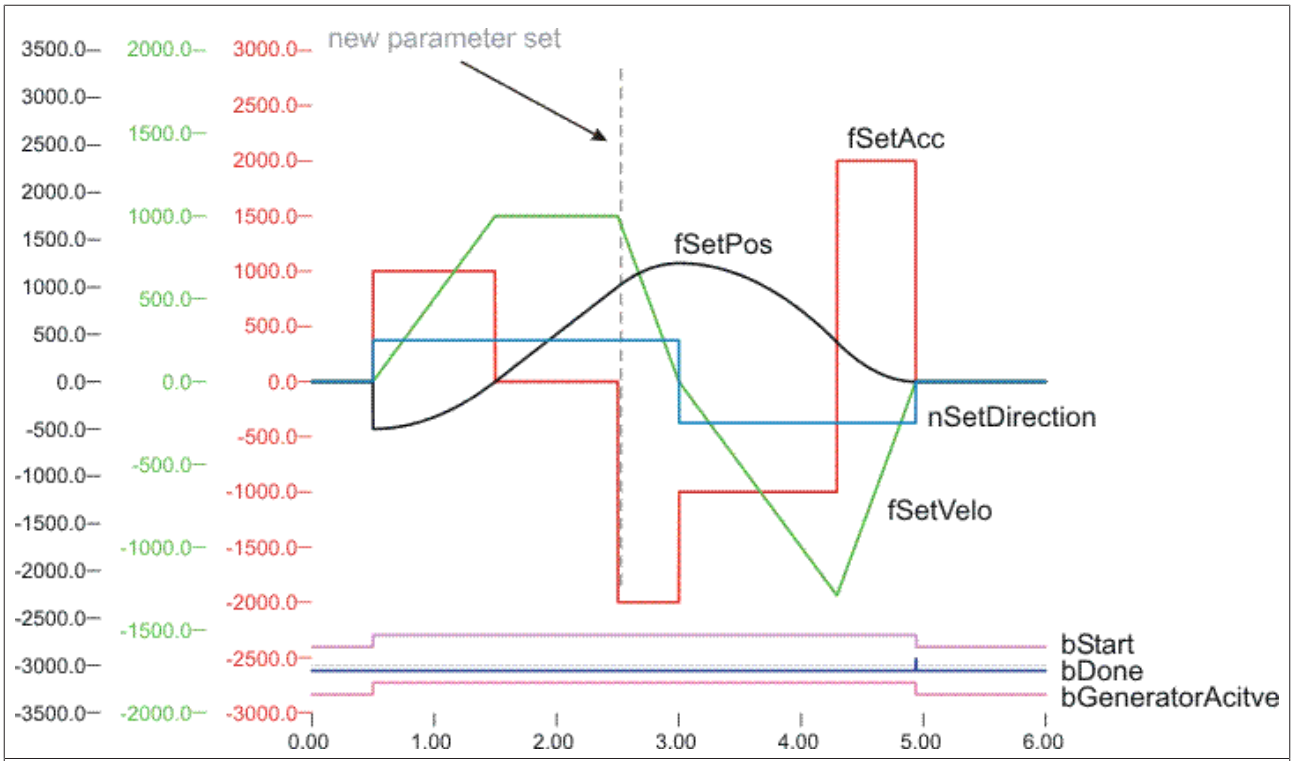


Beispielpositionierung 4:

```
stParams.fStartPos := -500.0;
stParams.fTargetPos := 2500.0;
stParams.fStartVelo := 0.0;
stParams.fVelocity := 1000.0;
stParams.fTargetVelo := 0.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 2000.0;
stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;
fOverride := 100.0;
```

Parameteränderung wenn fSetPos > 1000.0, eNewPosType := eCTRL_NEW_POS_TYPE_Instant

```
stParams.fTargetPos := 0.0;
stParams.fStartVelo := 0.0;
stParams.fVelocity := 1300.0;
stParams.fTargetVelo := 0.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 2000.0;
fOverride := 100.0;
```



Beispielpositionierung 5:

Start auf Punkt 1:

```
stParams.fStartPos := -100.0;
stParams.fTargetPos := 200.0;
stParams.fStartVelo := 0.0;
stParams.fVelocity := 250.0;
stParams.fTargetVelo := 150.0;
stParams.fAcceleration := 500.0;
stParams.fDeceleration := 400.0;
stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_Instant
```

Nachstarten auf Punkt 2:

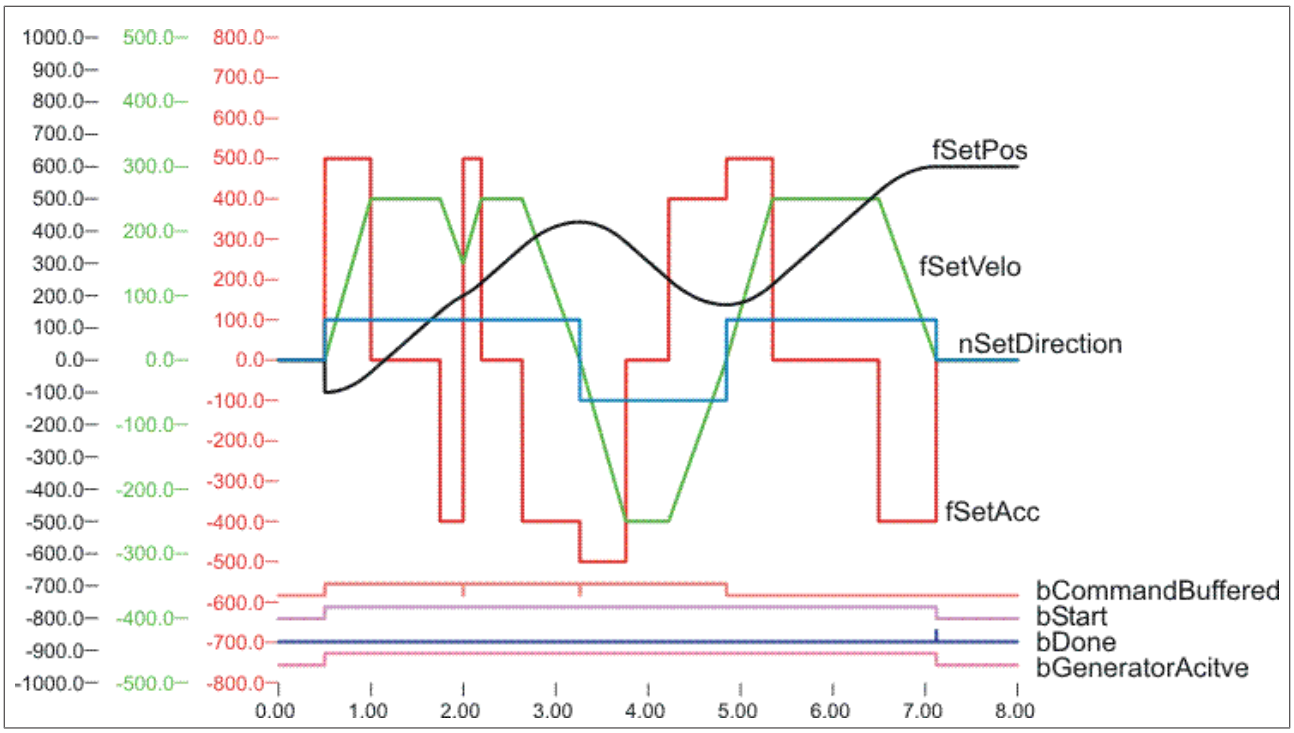
```
stParams.fTargetPos := 400.0;
stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;
```

Nachstarten auf Punkt 3:

```
stParams.fTargetPos := 200.0;
stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;
```

Nachstarten auf Punkt 4:

```
stParams.fTargetPos := 600.0;
stParams.fTargetVelo := 0.0;
stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;
```



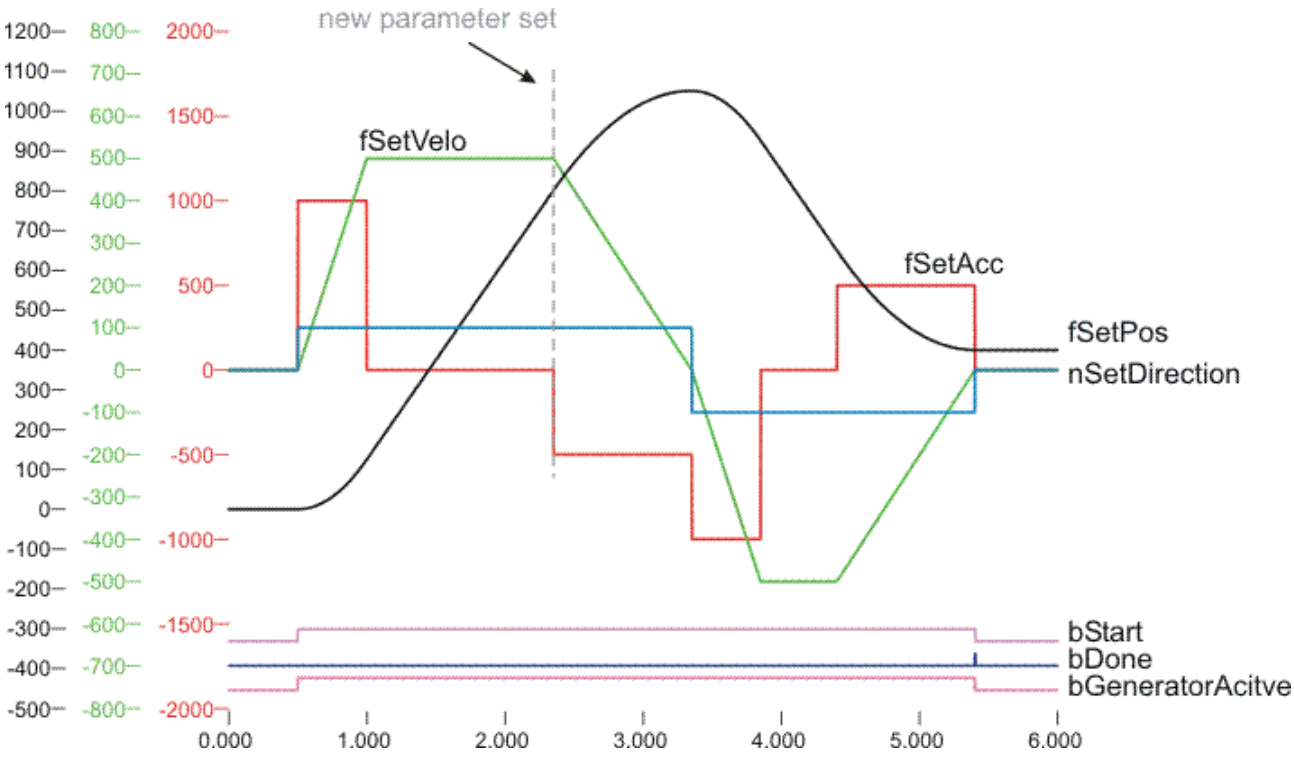
HINWEIS

Wenn einer neuer Parametersatz mit dem Type "`eCTRL_NEW_POS_TYPE_Instant`" an den Baustein übergeben wird, in dem die Verzögerung verringert wird, ist es möglich, dass die alte Zielposition überfahren wird.

Beispiel:

```
stParams.fTargetPos      := 1000.0;
stParams.fStartPos       := 0.0;
stParams.fVelocity       := 500.0;
stParams.fAcceleration   := 1000.0;
stParams.fDeceleration   := 1000.0;
IF fSetPos > 800.0
THEN
    stParams.fTargetPos    := 400.0;
    stParams.fVelocity     := 500.0;
    stParams.fAcceleration := 1_000.0;
    stParams.fDeceleration := 500.0;
    stParams.eNewPosType   := eCTRL_NEW_POS_TYPE_Instant;
END_IF
```

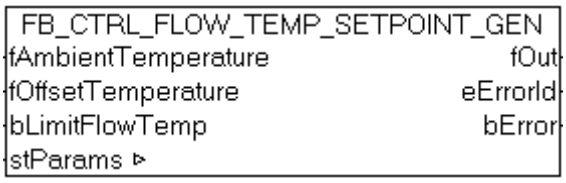
In der nachfolgenden Scope-Aufnahme ist deutlich zu sehen, dass die ursprüngliche Zielposition von 1000 mm überfahren wird, was darauf zurück zu führen ist, dass die Verzögerung in dem neuen Parametersatz verringert wird.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib

5.9.2 FB_CTRL_FLOW_TEMP_SETPOINT_GEN

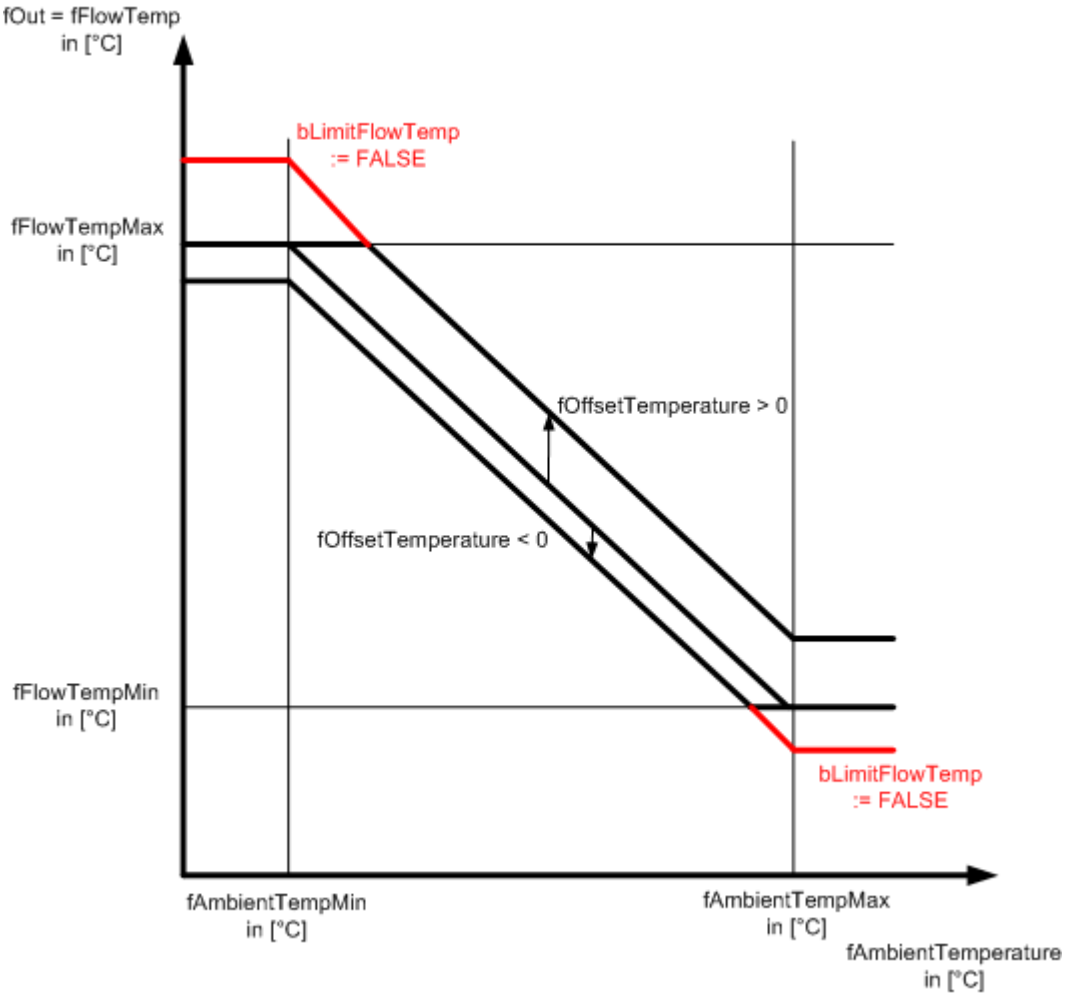


Der Funktionsbaustein ermöglicht die Vorgabe einer Vorlauftemperatur in Abhängigkeit der Außentemperatur.

Beschreibung:

Aus der Umgebungstemperatur (**fAmbientTemperature**) wird der Sollwert der Vorlauftemperatur (**fOut**) bestimmt. Dies geschieht über eine Gerade die über einen Offset (**fOffsetTemperature**) verschiebbar ist. Die Steigung der Geraden ergibt sich aus den vorgegebenen Eckpunkten der Umgebungs- und der Vorlauftemperatur. Über ein Flag (**bLimitFlowTemp**) kann bestimmt werden, ob die Vorlauftemperatur auf ihre Grenzwerte begrenzt wird oder nicht. Mit Hilfe der Offset-Temperatur kann einfach eine Nachtabsenkung oder eine Vorsteuerung durchgeführt werden.

Verhalten der Ausgangsgröße:



VAR_INPUT

```
VAR_INPUT
    fAmbientTemperature     : FLOAT;
    fOffsetTemperature      : FLOAT;
    bLimitFlowTemp      : BOOL;
END_VAR
```

fAmibientTemperature : Start der Rampengenerierung.

fOffestTemperature : Startwert der Rampe.

bLimitFlowTemp : Zielwert der Rampe.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Sollwert der Vorlauftemperatur.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Rampengenerators. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS:
STRUCT
    tTaskCycleTime   : TIME;  (* task cycle time [TIME]
*)
    tCtrlCycleTime   : TIME;  (* controller cycle time [TIME]
*)
    fForeRunTempMax  : FLOAT;
    fForeRunTempMin  : FLOAT;
    fAmbientTempMax  : FLOAT;
    fAmbientTempMin  : FLOAT;
END_STRUCT
END_TYPE
```

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Task-Zyklus aufgerufen wird.

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

fFlowTempMax : Maximale Vorlauftemperatur (siehe Diagramm).

fFlowTempMin : Minimale Vorlauftemperatur (siehe Diagramm).

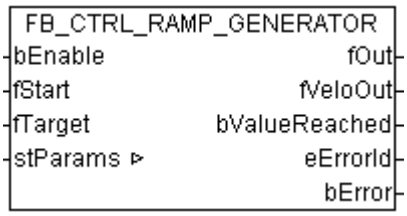
fAmbientTempMax : Außentemperatur, bei der die minimale Vorlauftemperatur vorgegeben wird.

fAmbientTempMin : Außentemperatur, bei der die maximale Vorlauftemperatur vorgegeben wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.9.3 FB_CTRL_RAMP_GENERATOR

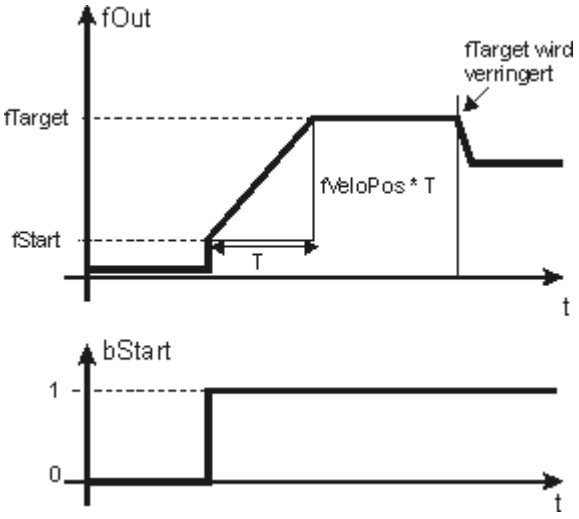


Der Funktionsbaustein stellt einen parametrierbaren Rampengenerator dar.

Beschreibung:

Dieser Funktionsbaustein erzeugt eine Rampe, die den Anfangswert **fStart** mit dem Zielwert **fTarget** verbindet. Die Steigung der Rampe (also die Geschwindigkeit), wird mit den Parametern **fVeloPos** und **fVeloNeg** in Einheiten/s angegeben. Der Startwert wird mit der steigenden Flanke von **bEnable** übernommen und die Berechnung der Rampe wird gestartet. Während das Signal bEnable = TRUE ist, kann der Zielwert variiert werden und der Ausgangswert ändert sich rampenförmig von dem aktuellen Wert auf den jeweils aktuellen Zielwert.

Verhalten der Ausgangsgröße:



VAR_INPUT

```
VAR_INPUT
    bEnable     : BOOL;
    fStart  : FLOAT;
    fTarget     : FLOAT; (* target value *)
END_VAR
```

bEnable : Start der Rampengenerierung.

fStart : Startwert der Rampe.

fTarget : Zielwert der Rampe.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    fVeloOut        : FLOAT;
    bValueReached   : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des Rampengenerators.

fVeloOut : Aktuelle Geschwindigkeit des Rampengenerators.

bValueReached : Der Ausgang signalisiert mit einem TRUE, das der Ausgang fOut den Wert fTarget erreicht hat.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_RAMP_GENERATOR_PARAMS; (* RAMP_Generator parameter struct *)
END_VAR
```

stParams : Parameterstruktur des Rampengenerators. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime    : TIME;   (* task cycle time [TIME]
*)
    tCtrlCycleTime    : TIME;   (* controller cycle time [TIME]
```

```
*)
    fVeloPos     : FLOAT; (* velocity ramp by time range:
> 0.0 *)
    fVeloNeg     : FLOAT; (* velocity ramp by time range:
> 0.0 *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Task-Zyklus aufgerufen wird.

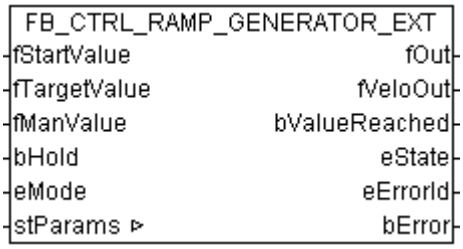
fVeloPos : Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem niedrigeren Wert auf einen höheren überführt wird.

fVeloNeg : Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem höheren Wert auf einen niedrigeren überführt wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.9.4 FB_CTRL_RAMP_GENERATOR_EXT

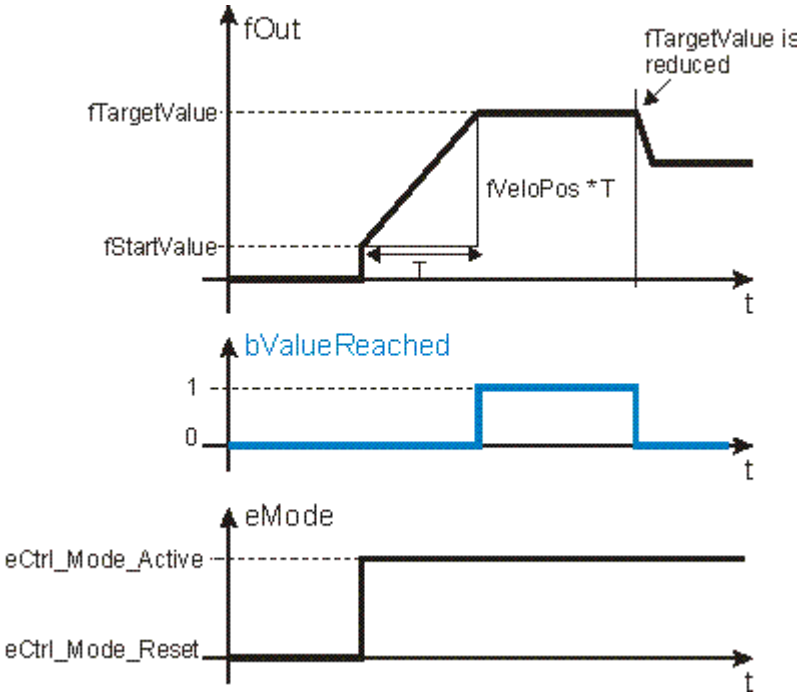


Der Funktionsbaustein stellt einen parametrierbaren Rampengenerator dar, welcher gegenüber dem FB_CTRL_RAMP_GENERATOR die E_CTRL_MODE's unterstützt.

Beschreibung:

Dieser Funktionsbaustein erzeugt eine Rampe, die den Anfangswert **fStartValue** mit dem Zielwert **fTargetValue** verbindet. Die Steigung der Rampe (also die Geschwindigkeit), wird mit den Parametern **fVeloPos** und **fVeloNeg** in Einheiten/s angegeben. Der Startwert wird bei einem Wechsel vom eCTRL_MODE_RESET in den eCTRL_MODE_ACTIVE übernommen und die Berechnung der Rampe wird gestartet. Während sich der Baustein im eCTRL_MODE_ACTIVE befindet, kann der Zielwert variiert werden und der Ausgangswert ändert sich rampenförmig von dem aktuellen Wert auf den jeweils aktuellen Zielwert. An dem Ausgang fVeloOut wird die jeweils aktuelle Geschwindigkeit ausgegeben. Diese kann eventuell zur Vorsteuerung des Regelkreises genutzt werden.

Verhalten der Ausgangsgröße:



VAR_INPUT

```
VAR_INPUT
    fStartValue     : FLOAT;
    fTargetValue    : FLOAT;
    fManValue           : FLOAT;
    bHold       : BOOL;
    eMode       : E_CTRL_MODE;
END_VAR
```

fStartValue : Startwert der Rampe.

fTargetValue : Zielwert der Rampe.

fManValue : Eingangsgröße, auf die der Ausgang im eCTRL_MODE_MANUAL gesetzt wird.

bHold : Die Berechnung der Rampe wird auf dem aktuellen Wert angehalten.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    fVeloOut        : FLOAT;
    bValueReached   : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des Rampengenerators.

fVeloOut : Aktuelle Geschwindigkeit des Rampengenerators.

bValueReached : Der Ausgang signalisiert mit einem TRUE, das der Ausgang fOut den Wert fTargetValue erreicht hat.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_RAMP_GENERATOR_EXT_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Rampengenerators. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_RAMP_GENERATOR_EXT_PARAMS :
STRUCT
    tTaskCycleTime   : TIME;  (* task cycle time [TIME]
*)
    tCtrlCycleTime   : TIME;  (* controller cycle time [TIME]
*)
    fVeloPos     : FLOAT; (* velocity ramp by time range:
> 0.0 *)
    fVeloNeg     : FLOAT; (* velocity ramp by time range:
> 0.0 *)
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Task-Zyklus aufgerufen wird.

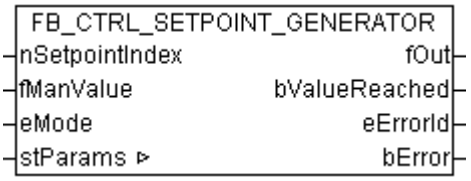
fVeloPos : Geschwindigkeit (>0.0) in Einheiten / s, mit der der Ausgang von einem niedrigeren Wert auf einen höheren überführt wird.

fVeloNeg : Geschwindigkeit (>0.0) in Einheiten / s, mit der der Ausgang von einem höheren Wert auf einen niedrigeren überführt wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.9.5 FB_CTRL_SETPOINT_GENERATOR



Der Funktionsbaustein stellt einen Sollwertgenerator dar, der aus einer Tabelle den angewählten Sollwert ausgibt. Der Wechsel von einem zum anderen Sollwert kann stetig oder unstetig erfolgen.

Verhalten der Ausgangsgröße:

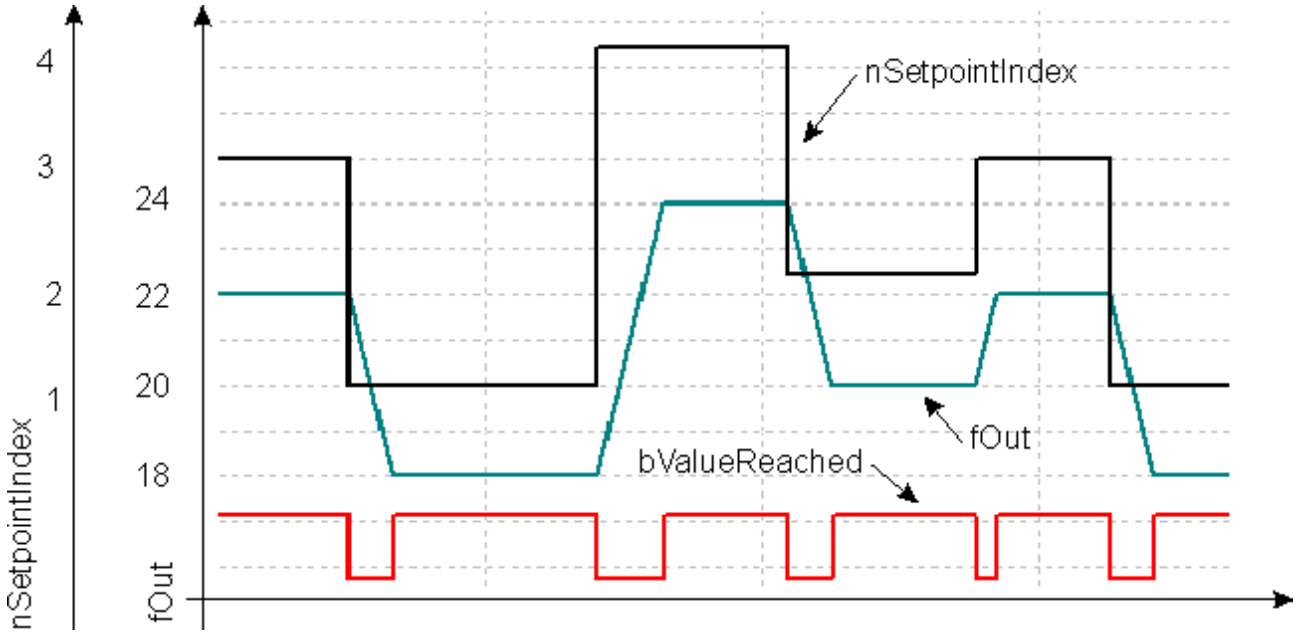


Tabelle des Beispiels:

18
20
22
24

Die erste Zeile der Tabelle entspricht dem Index 1, die zweite dem Index 2 usw.

Beschreibung:

In dem Array, welches mit den entsprechenden Parametern dem Baustein bekannt gemacht wird, werden die einzelnen Sollwerte abgelegt. Über den Eingang **nSetpointIndex** wird ein in der Tabelle hinterlegter Sollwert ausgewählt. Dieser wird dann am Ausgang ausgegeben und kann als Sollwert für die Regelung verwendet werden. Der Wechsel von einem Wert auf einen anderen kann linear oder sprungförmig erfolgen. Die Geschwindigkeit eines stetigen Übergangs wird mit den Parametern **fVeloPos** und **fVeloNeg** festgelegt. Der Ausgang **bValueReached** signalisiert das Erreichen des angewählten Sollwertes.

VAR_INPUT

```
VAR_INPUT
    nSetpointIndex  : INT  (*[1 ... n] *);
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

nSetpointIndex : Index des angewählten Sollwertes.

fManValue : Eingang, der im Manual-Mode ausgegeben wird.

eMode : Eingang, der die Betriebsart [▶ 16] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : SETPOINT_TABLE_ELEMENT;
    bValueReached   : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des Sollwert-Generators.

bValueReached : Der Ausgang ist TRUE, wenn der angewählte Sollwert erreicht worden ist, also wenn das Rampen auf den angewählten Wert beendet ist.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_SETPOINT_GENERATOR_PARAMS; (* parameters *)
END_VAR
```

stParams : Parameterstruktur des Rampengenerators. Diese besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_SETPOINT_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME     := T#0ms; (*
controller cycle time [TIME] *)
    tTaskCycleTime          : TIME     := T#0ms; (* task
cycle time [TIME] *)
    pDataTable_ADR          : POINTER TO
INTERPOLATION_TABLE_ELEMENT := 0;
    nDataTable_SIZEOF       : UINT     := 0;
    nDataTable_NumberOfRows     : UINT     := 0;
    fVeloPos            : FLOAT; (* velocity ramp by
time range > 0.0 *)
    fVeloNeg            : FLOAT; (* velocity ramp by
time range > 0.0 *)
    bDisableRamping         : BOOL     := FALSE;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.indent: -103; margin-left: 108">
tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

pDataTable_ADR : Adresse des Daten-Arrays.

pDataTable_SIZEOF : Größe des Daten-Arrays.

pDataTable_NumberOfRows : Anzahl der Zeilen des Daten-Arrays.

fVeloPos : Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem niedrigeren Wert auf einen höheren überführt wird.

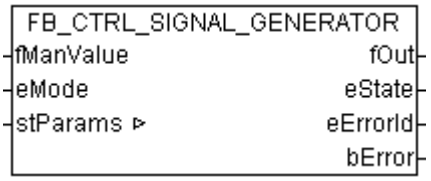
fVeloNeg : Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem höheren Wert auf einen niedrigeren überführt wird.

bDisableRamping: Wenn dieser Parameter TRUE ist, wird keine stetige Ausgangsgröße berechnet. Es wird sprungförmig zwischen den Ausgangswerten umgeschaltet.

Voraussetzungen

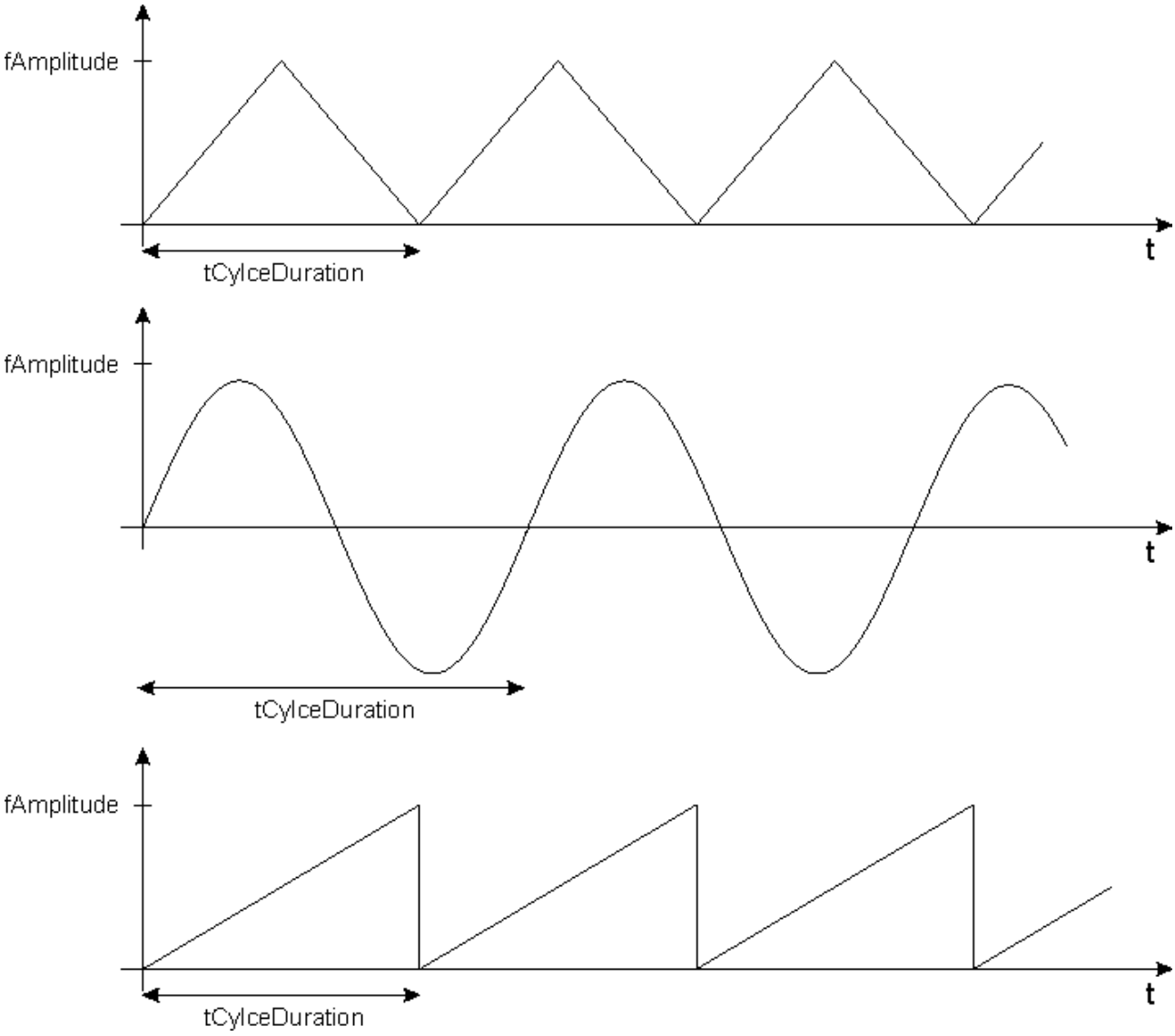
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

5.9.6 FB_CTRL_SIGNAL_GENERATOR



Der Funktionsbaustein stellt einen Signalgenerator mit den Signalformen **Dreieck, Sinus** und **Sägezahn** dar.

Ausgangssignale:



VAR_INPUT

```
VAR_INPUT
    fManValue       : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

fManValue : Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.

eMode : Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

fOut : Ausgang des Signalgenerators.

eState : State des Funktionsbausteins.

eErrorId : Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 16].

bError : Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_SIGNAL_GENERATOR_PARAMS;
END_VAR
```

stParams : Parameterstruktur des Funktionsbausteins. Diese besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_I_PARAMS
:
STRUCT
    tCtrlCycleTime : TIME  :=  T#0ms; (* controller cycle time
*)
    tTaskCycleTime : TIME  :=  T#0ms; (* task cycle time
*)
    eSignalType    : E_CTRL_SIGNAL_TYPE;
    tCylceDuration : TIME;
    fAmplitude     : FLOAT;
    fOffset    : FLOAT := 0.0;
    tStart     : TIME  := T#0s;
END_STRUCT
END_TYPE
```

tCtrlCycleTime : Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der TaskCycleTime sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.

tTaskCycleTime : Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

eSignalType : Anwahl der Signalform. Siehe E_CTRL_SIGNAL_TYPE unten.

```
TYPE
E_CTRL_SIGNAL_TYPE :
    (
    eCTRL_TRIANGLE  := 0,
    eCTRL_SINUS     := 1,
    eCTRL_SAWTOOTH  := 2
    );
END_TYPE
```

tCycleDuration : Periodendauer des erzeugten Signalverlaufs.

fAmplitude : Amplitude erzeugten Signalverlaufs.

fOffset : Offset, der auf den Signalverlauf addiert wird.

tStart : Zeitpunkt innerhalb einer Periode, bei dem der Signalverlauf startet, wenn in den eCTRL_MODE_ACTIVE geschaltet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcControllerToolbox.lib
TwinCAT v2.9 ab Build 947	BC [▶ 10]	TcControllerToolbox.lb6
TwinCAT v2.9 ab Build 956	BX [▶ 10]	TcControllerToolbox.lbx

6 Beispielprojekt

Die Funktionsweise der Bausteine der TcControllerToolbox wird in einem Beispielprojekt gezeigt. In diesem Projekt sind Programme zu den einzelnen Funktionsbausteinen enthalten, mit denen die Basisfunktionalitäten verdeutlicht werden. Es wird an einigen Beispielen aber auch der Aufbau komplexerer Regelstrecken gezeigt. Bei allen Programmen wird die Regelstrecke simuliert, so dass kein Hardwareaufbau für das Projekt benötigt wird.

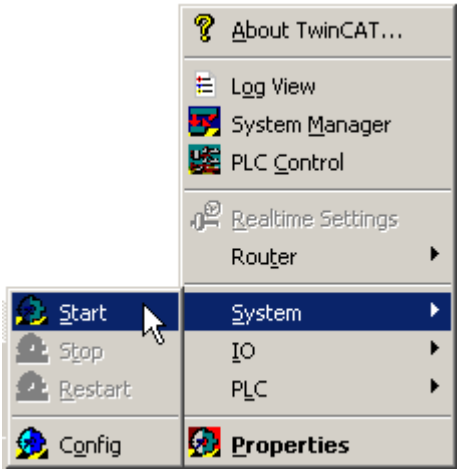
6.1 Beispielprojekt installieren und starten

Die TcControllerToolbox bietet dem Programmierer Übertragungsglieder für die Realisierung verschiedenster Regelungen.

- Beispielprogramm https://infosys.beckhoff.com/content/1031/tcplccontrollertoolbox/Resources/11282752267.zip speichern und entpacken.

Das Beispielprogramm arbeitet mit simulierten Regelstrecken und ist somit ohne zusätzliche Hardware auf jedem Windows NT PC lauffähig.

- Das TwinCAT-System wird mit dem TwinCAT-Icon
  -  in der Task-Leiste gestartet.



Das Beispielprogramm *TcControllerToolbox_Examples.pro* wird in das TwinCAT PLC-Control geladen, übersetzt und gestartet.

- PRO-Datei laden.
- SPS-Projekt über das Menü *Project - Rebuild All* übersetzen.
- SPS-Projekt über das Menü *Online - Login* in das Laufzeitsystem laden.
- Programm über das Menü *Online - Run* starten.

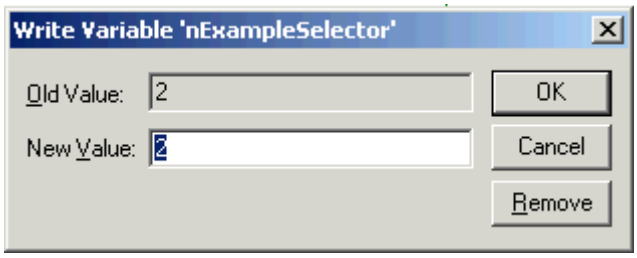
Mit dem TwinCAT ScopeView können die Signalverläufe der einzelnen Beispiele graphisch dargestellt werden. Zu diesem Zweck können die mitgelieferten Einstellungen *TcControllerToolboxExamples_Scope_x.scp* verwendet werden. Die Angabe des bei dem jeweiligen Beispiel zu verwendenden Scope-Files kann den Kommentaren in dem Programm MAIN des Projekts entnommen werden.

- SCP-Datei in das TwinCAT ScopeView laden.
- Die Aufzeichnung über das Menü oder über die Taste F5 starten.

Da das Beispielprojekt mehrere unterschiedliche Beispiele enthält, muss in dem Programm MAIN die Variable nExampleSelector auf die entsprechende Beispielnummer gesetzt werden.

- Doppelklick auf die Variable nExampleSelector.

- Nummer des Beispiels eintragen. OK.



- Taste F7 drücken oder "Force Values" im Menü Online anklicken.

6.2 Zuordnung der Programmnummern

Die Programme werden im MAIN über die Variable *nExampleSelector* ausgewählt.

Base

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_P	1	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_I	2	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_D	3	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_HYSTERESIS	4	TcControllerToolboxExamples_Scope_2.scp
FB_CTRL_TRANSFERFUNCTION_1	5	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_TRANSFERFUNCTION_2	6	TcControllerToolboxExamples_Scope_1.scp

Controller

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_2POINT	10	TcControllerToolboxExamples_Scope_3.scp
FB_CTRL_3POINT	11	TcControllerToolboxExamples_Scope_4.scp
FB_CTRL_3POINT_EXT	12	TcControllerToolboxExamples_Scope_5.scp
FB_CTRL_nPOINT	13	TcControllerToolboxExamples_Scope_5.scp
FB_CTRL_PI	14	TcControllerToolboxExamples_Scope_5.scp
FB_CTRL_PID	15	TcControllerToolboxExamples_Scope_5.scp
FB_CTRL_PID_EXT	16	TcControllerToolboxExamples_Scope_5.scp

Filter / Controlled System Simulation

Beispiel zu dem Funktions-baustein	nExampleSelector	Scope File
FB_CTRL_ACTUAL_VALUE_FILTER	20	TcControllerToolboxExamples_Scope_1.scp

Beispiel zu dem Funktions-baustein	nExampleSelector	Scope File
FB_CTRL_ARITHMETIC_MEAN	21	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_MOVING_AVERAGE	22	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PT1	23	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PT2	24	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PT2oscillation	25	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PT3	26	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PTn	27	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PTt	28	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_TuTg	29	TcControllerToolboxExamples_Scope_1.scp

Interpolation

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_INTERPOLATION	30	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_NORMALIZE	31	TcControllerToolboxExamples_Scope_7.scp

Monitoring / Alarming

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_CHECK_IF_IN_BAND	40	TcControllerToolboxExamples_Scope_8.scp
FB_CTRL_LOG_DATA	41	

Output to Controlling Equipment

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_DEADBAND	50	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_LIMITER	51	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_PWM_OUT	52	TcControllerToolboxExamples_Scope_9.scp
FB_CTRL_PWM_OUT_EXT	53	TcControllerToolboxExamples_Scope_9.scp
FB_CTRL_MULTIPLE_PWM_OUT	54	TcControllerToolboxExamples_Scope_10.scp
FB_CTRL_SCALE	55	TcControllerToolboxExamples_Scope_1.scp

Setpointgeneration

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_RAMP_GENERATOR	60	TcControllerToolboxExamples_Scope_1.scp

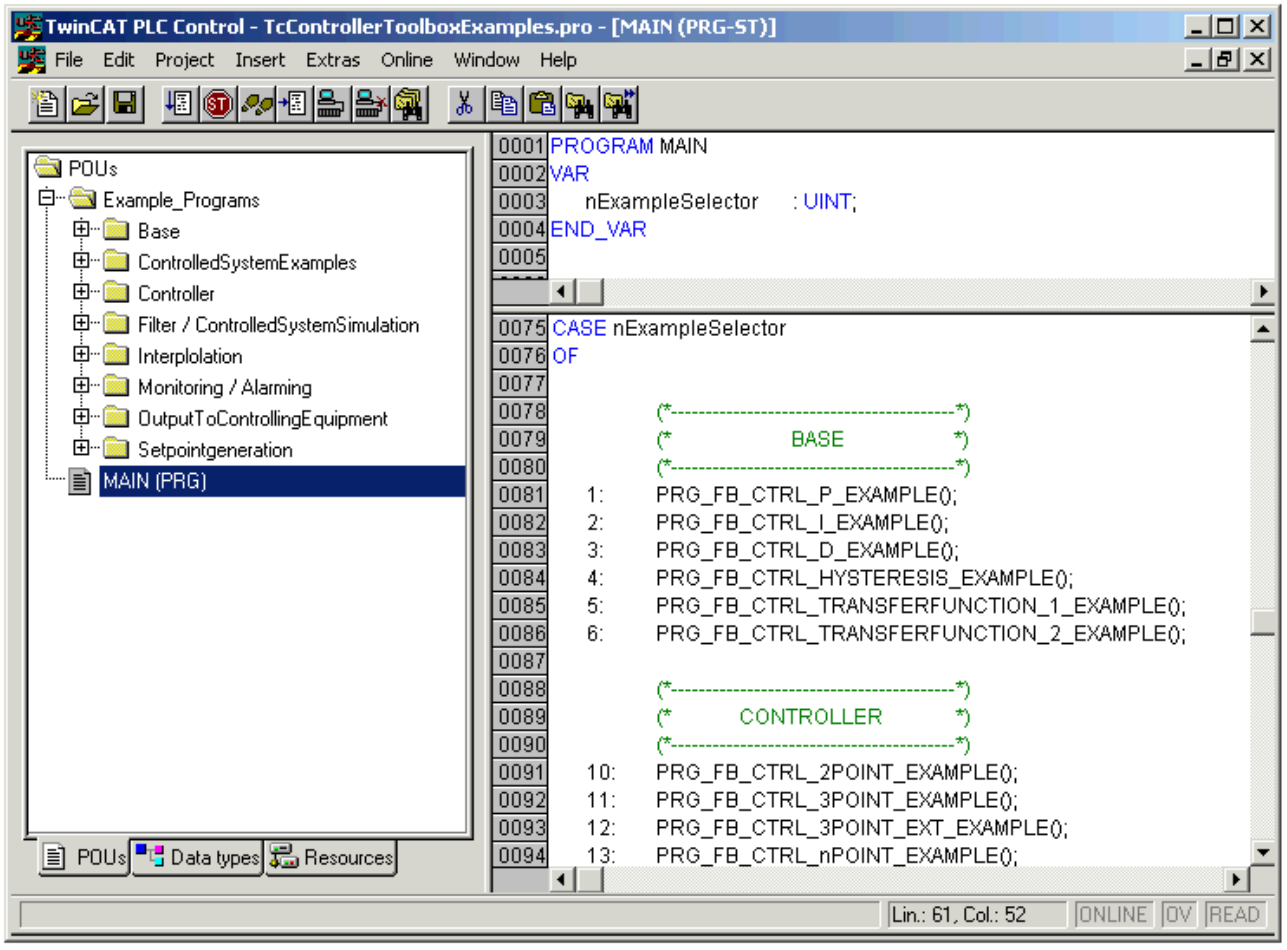
Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_SETPOINT_GENERATOR	61	TcControllerToolboxExamples_Scope_1.scp
FB_CTRL_SIGNAL_GENERATOR	62	TcControllerToolboxExamples_Scope_1.scp

Controlled System Examples

Beispiel zu dem Funktionsbaustein	nExampleSelector	Scope File
FB_CTRL_PID_SPLITRANGE	80	TcControllerToolboxExamples_Scope_6.scp
FB_CTRL_PI_PID	81	TcControllerToolboxExamples_Scope_11.scp

6.3 Programmstruktur

Das Hauptmodul `Main` ruft in Abhängigkeit der Variable `nExampleSelector` das entsprechende Beispielprogramm auf.



Die einzelnen Beispielprogramme sollten durch die Kommentare in den Programmen verständlich und leicht nachvollziehbar sein.

Mehr Informationen:
www.beckhoff.de/ts4100

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.de
www.beckhoff.de