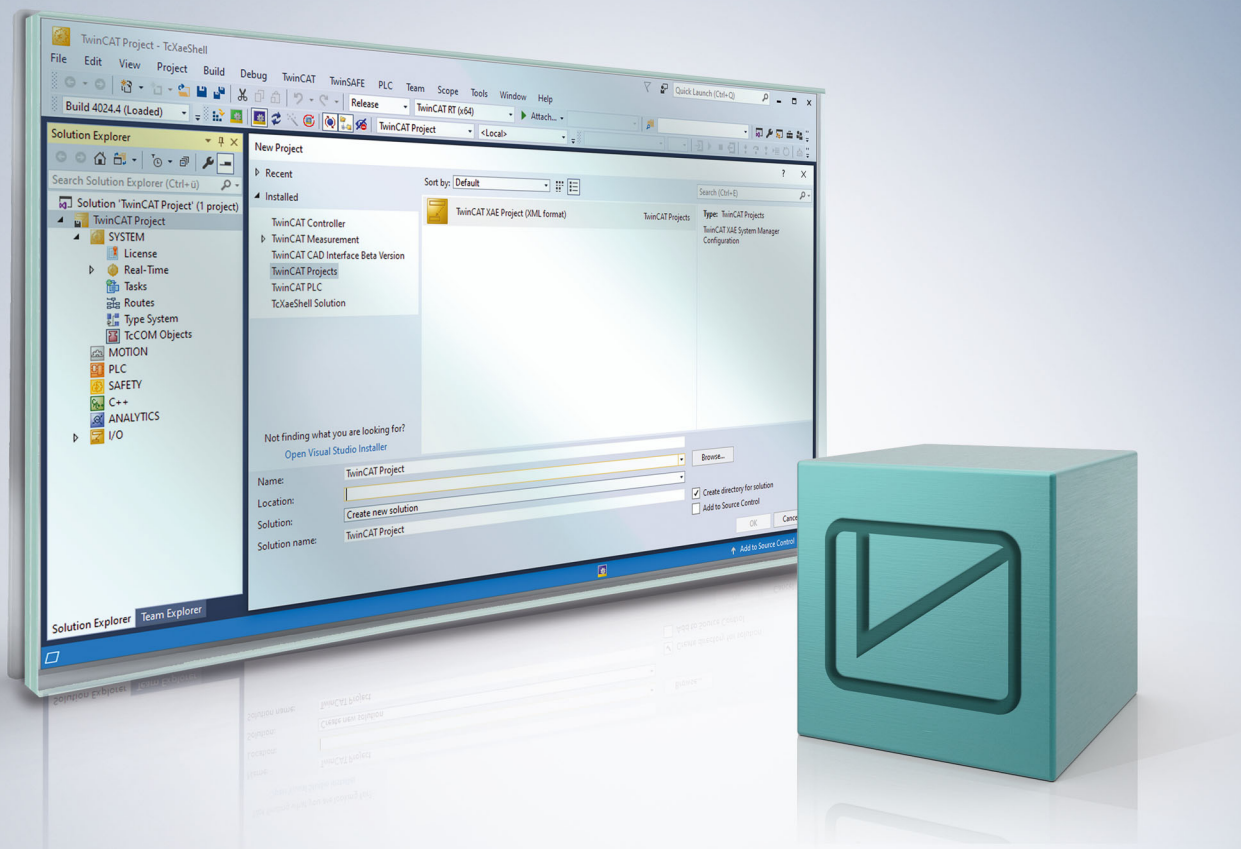
BECKHOFF New Automation Technology

Handbuch | DE

TF4100

TwinCAT 3 | Controller Toolbox



12.07.2023 | Version: 2.2.3

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 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

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.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

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.

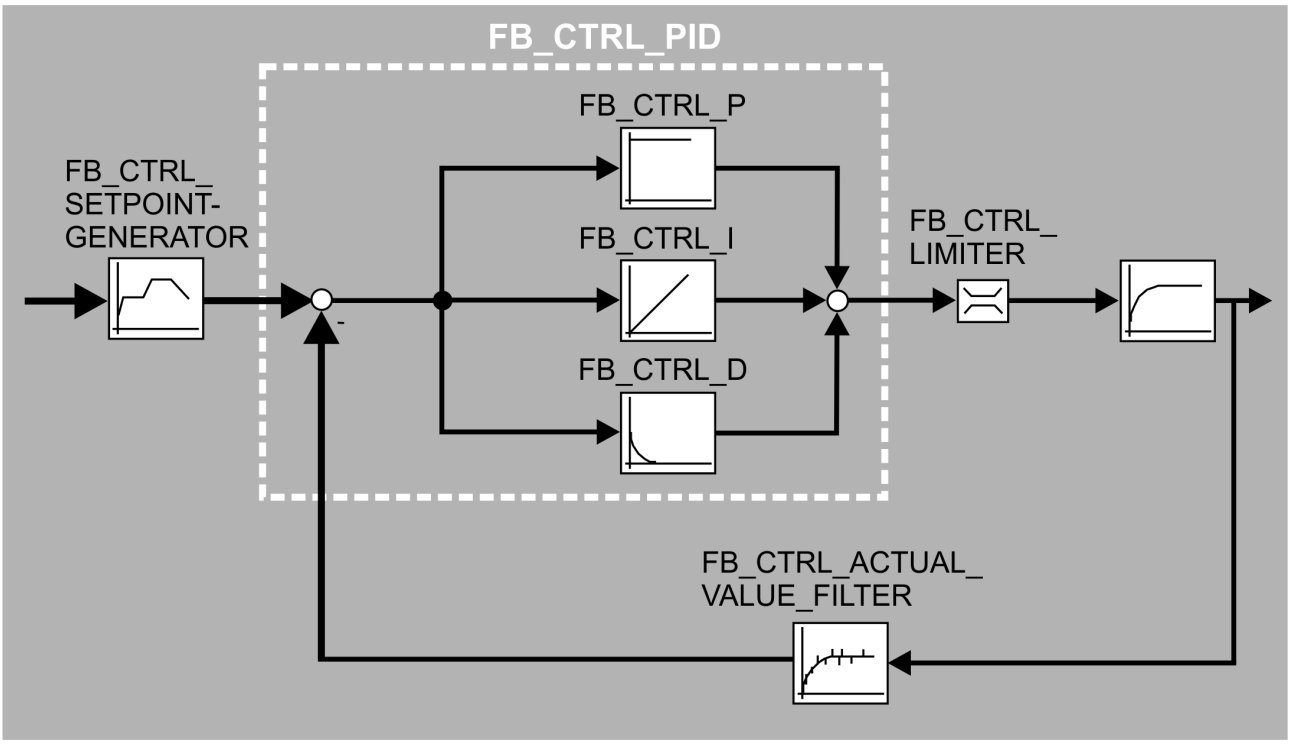
Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

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 Anwendungen einsetzbare komplexe Regler enthalten, aber auch Basisbausteine, mit denen für spezielle Anwendungen eigene Reglerstrukturen implementiert werden können.



Funktionsbausteine

Name	Beschreibung
FB_CTRL_2POINT [▶ 41]	2-Punkt-Regler
FB_CTRL_2POINT_PWM_ADAPTIVE [▶ 43]	Adaptiver 2-Punkt-Regler mit PWM-Ausgang
FB_CTRL_3PHASE_SETPOINT_GENERATOR [▶ 153]	3 Phasen Sollwertgenerator
FB_CTRL_3POINT [▶ 45]	3-Punkt-Regler
FB_CTRL_3POINT_EXT [▶ 47]	Erweiterter 3-Punkt-Regler
FB_CTRL_ACTUAL_VALUE_FILTER [▶ 85]	Istwertfilter
FB_CTRL_ARITHMETIC_MEAN [▶ 87]	Arithmetischer Mittelwertfilter
FB_CTRL_CHECK_IF_IN_BAND [▶ 123]	Bereichsüberwachung
FB_CTRL_D [▶ 33]	D-Glied
FB_CTRL_DEADBAND [▶ 132]	Totzone
FB_CTRL_DIGITAL_FILTER [▶ 89]	Digitaler Filter
FB_CTRL_FLOW_TEMP_SETPOINT_GEN [▶ 162]	Vorgabe der Vorlauftemperatur in Abhängigkeit der Außentemperatur
FB_CTRL_GET_SYSTEM_TIME [▶ 20]	Ausgabe der Windows-Systemzeit
FB_CTRL_GET_TASK_CYCLETIME [▶ 22]	Ermittlung der Task-Zykluszeit
FB_CTRL_HYSTERESIS [▶ 26]	Hysterese-Glied
FB_CTRL_I [▶ 29]	I-Glied
FB_CTRL_I_WITH_DRIFTCOMPENSATION [▶ 31]	I-Glied mit Driftkompensation
FB_CTRL_LEAD_LAG [▶ 94]	Lead-Lag-Glied
FB_CTRL_LIMITER [▶ 133]	Stellgrößenbegrenzung
FB_CTRL_LIN_INTERPOLATION [▶ 118]	Lineares Interpolationsglied
FB_CTRL_LOG_DATA [▶ 124]	Datenlogger im *.csv ASCII-Format
FB_CTRL_LOG_MAT_FILE [▶ 127]	Datenlogger im Matlab 5 - Format
FB_CTRL_LOOP_SCHEDULER [▶ 23]	Verteilung der Rechenleistung bei mehreren Regelkreisen
FB_CTRL_MOVING_AVERAGE [▶ 93]	Gleitender Mittelwertfilter
FB_CTRL_MULTIPLE_PWM_OUT [▶ 135]	PWM-Glied mit mehreren Ausgängen
FB_CTRL_NORMALIZE [▶ 120]	Kennlinien - Linearisierung
FB_CTRL_NOISE_GENERATOR [▶ 97]	Rauschgenerator
FB_CTRL_NOTCH_FILTER [▶ 98]	Notch-Filter
FB_CTRL_nPOINT [▶ 50]	n-Punkt-Regler
FB_CTRL_P [▶ 28]	P-Glied
FB_CTRL_PARAMETER_SWITCH [▶ 52]	Parameter-Umschaltalgorithmus für einen Splitrange-Regler
FB_CTRL_PI [▶ 55]	PI-Regler
FB_CTRL_PI_PID [▶ 57]	Kaskadierter PI-PID-Regler
FB_CTRL_PID [▶ 61]	PID-Regler
FB_CTRL_PID_EXT [▶ 72]	Erweiterter PID-Regler
FB_CTRL_PID_EXT_SPLITRANGE [▶ 65]	Erweiterter PID-Regler mit Parametersatzumschaltung
FB_CTRL_PID_SPLITRANGE [▶ 77]	PID-Regler mit Parametersatzumschaltung
FB_CTRL_PT1 [▶ 100]	PT_1-Glied
FB_CTRL_PT2 [▶ 102]	PT_2-Glied
FB_CTRL_PT2oscillation [▶ 104]	Schwingungsfähiges PT_2-Glied
FB_CTRL_PT3 [▶ 106]	PT_3-Glied

Name	Beschreibung
FB_CTRL_PTn [▶ 108]	PT_n-Glied
FB_CTRL_PTt [▶ 110]	PT_t-Glied
FB_CTRL_PWM_OUT [▶ 140]	PWM-Glied
FB_CTRL_PWM_OUT_EXT [▶ 142]	Erweitertes PWM-Glied
FB_CTRL_RAMP_GENERATOR [▶ 164]	Rampengenerator
FB_CTRL_RAMP_GENERATOR_EXT [▶ 166]	Erweiterter Rampengenerator
FB_CTRL_SCALE [▶ 144]	Bereichsanpassung
FB_CTRL_SERVO_MOTOR_OUT [▶ 146]	Ansteuerung eines Stellantriebs
FB_CTRL_SERVO_MOTOR_SIMULATION [▶ 112]	Simulation eines Stellantriebs
FB_CTRL_SETPOINT_GENERATOR [▶ 168]	Sollwertgenerator
FB_CTRL_SIGNAL_GENERATOR [▶ 171]	Signalgenerator
FB_CTRL_SPLITRANGE [▶ 148]	Signalzerlegung in den positiven und negativen Anteil.
FB_CTRL_STEPPING_MOTOR_OUT [▶ 150]	Ansteuerung eines Schrittmotors
FB_CTRL_TRANSFERFUNCTION_1 [▶ 35]	Übertragungsfunktion nach der 1. Standardform
FB_CTRL_TRANSFERFUNCTION_2 [▶ 38]	Übertragungsfunktion nach der 2. Standardform
FB_CTRL_TuTg [▶ 114]	TuTg-Glied
FB_CTRL_ZERO_ZONE_DAMPING [▶ 116]	Nullpunkt-Dämpfung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4016	PC oder CX	Tc2_ControllerToolbox

3 Installation

3.1 Systemvoraussetzungen

Die folgenden Systemvoraussetzungen müssen für eine ordnungsgemäße Funktion der TC3 Function Controller Toolbox erfüllt sein.

Entwicklungsumgebung

Eine Entwicklungsumgebung beschreibt einen Computer, auf dem SPS-Programme entwickelt, aber nicht ausgeführt werden. Auf einem Entwicklungscomputer muss folgendes installiert sein:

- TwinCAT 3 XAE (Engineering) Build 4012 oder höher
- TwinCAT 3 Function TF4100 Controller Toolbox Version 3.4.0.0 oder höher
- Bitte beachten Sie: Für die Entwicklungsumgebung kann eine 7-Tage Testlizenz genutzt werden, siehe Lizenzierung [▶ 15].

Laufzeitumgebung

Eine Laufzeitumgebung beschreibt einen Computer, auf dem SPS-Programme ausgeführt werden. Auf einem Laufzeitcomputer muss folgendes installiert sein:

- TwinCAT3 XAR build 4012 oder höher
- Lizenzen Für TC1200 PLC und TF4100 Controller Toolbox
- Bitte beachten Sie: Für Testzwecke kann eine 7-Tage Testlizenz genutzt werden, siehe Lizenzierung [▶ 15].

Entwickler- und Laufzeit auf einem Computer

Sollen auf einem Computer Laufzeit- und Entwicklungsumgebung laufen (z.B. um ein SPS-Programm zu testen, bevor es auf den Ziel-Computer geladen wird), müssen folgende Anforderungen erfüllt sein:

- TwinCAT3 XAE (engineering installation) build 4012 oder höher
- TwinCAT 3 Function TF4100 Controller Toolbox Version 3.4.0.0 oder höher
- Bitte beachten Sie: Für Testzwecke kann eine 7-Tage Testlizenz genutzt werden, siehe Lizenzierung [▶ 15].

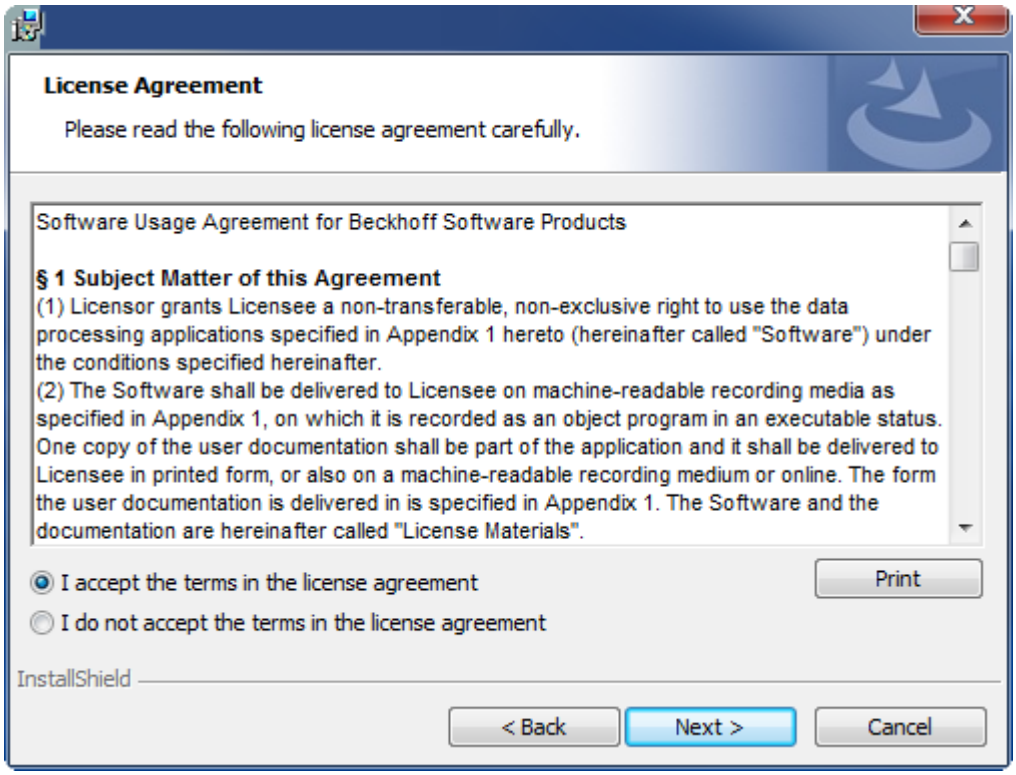
3.2 Installation

Nachfolgend wird beschrieben, wie die TwinCAT 3 Function für Windows-basierte Betriebssysteme installiert wird.

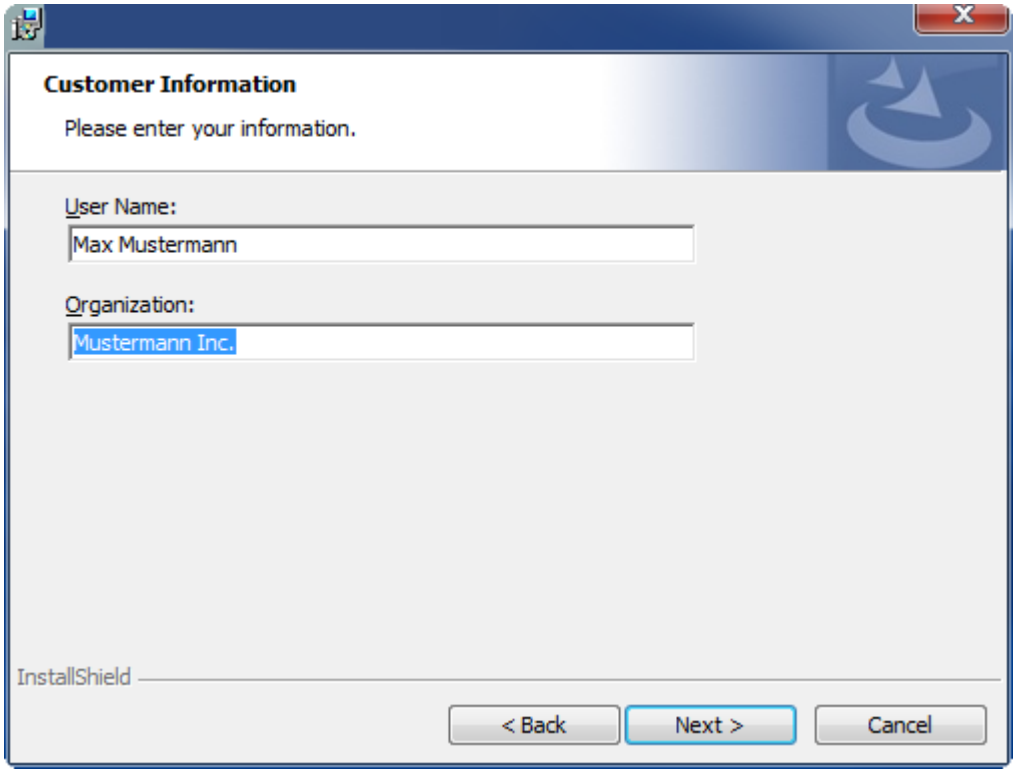
✓ Die Setup-Datei der TwinCAT 3 Function wurde von der Beckhoff-Homepage heruntergeladen.

1. Führen Sie die Setup-Datei als Administrator aus. Wählen Sie dazu im Kontextmenü der Datei den Befehl **Als Administrator ausführen**.
   ⇨ Der Installationsdialog öffnet sich.

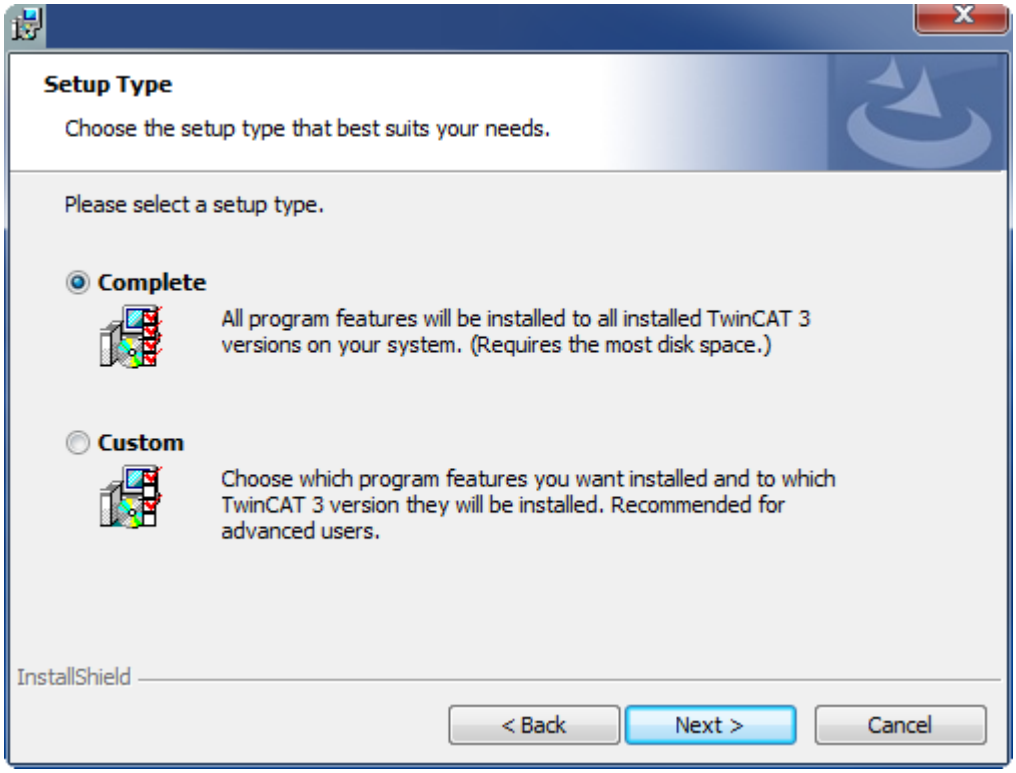
2. Akzeptieren Sie die Endbenutzerbedingungen und klicken Sie auf **Next**.



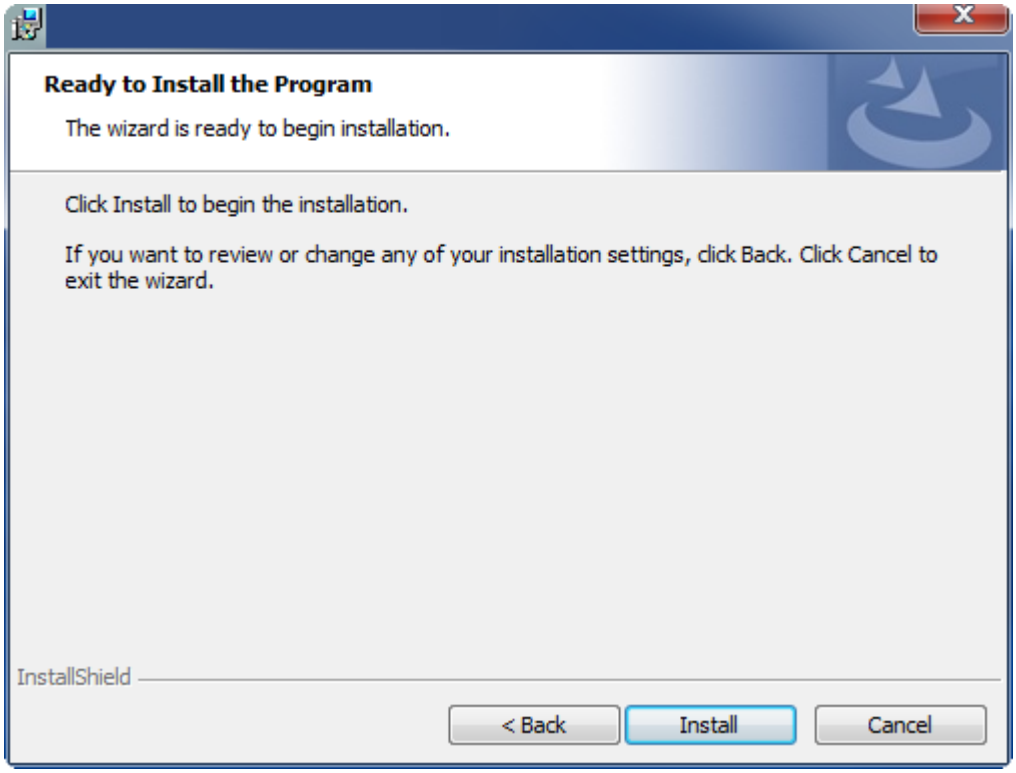
3. Geben Sie Ihre Benutzerdaten ein.



4. Wenn Sie die TwinCAT 3 Function vollständig installieren möchten, wählen Sie **Complete** als Installationstyp. Wenn Sie die Komponenten der TwinCAT 3 Function separat installieren möchten, wählen Sie **Custom**.

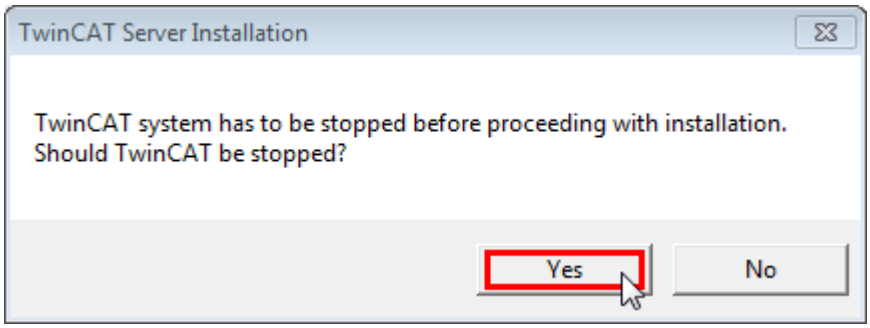


5. Wählen Sie **Next** und anschließend **Install**, um die Installation zu beginnen.

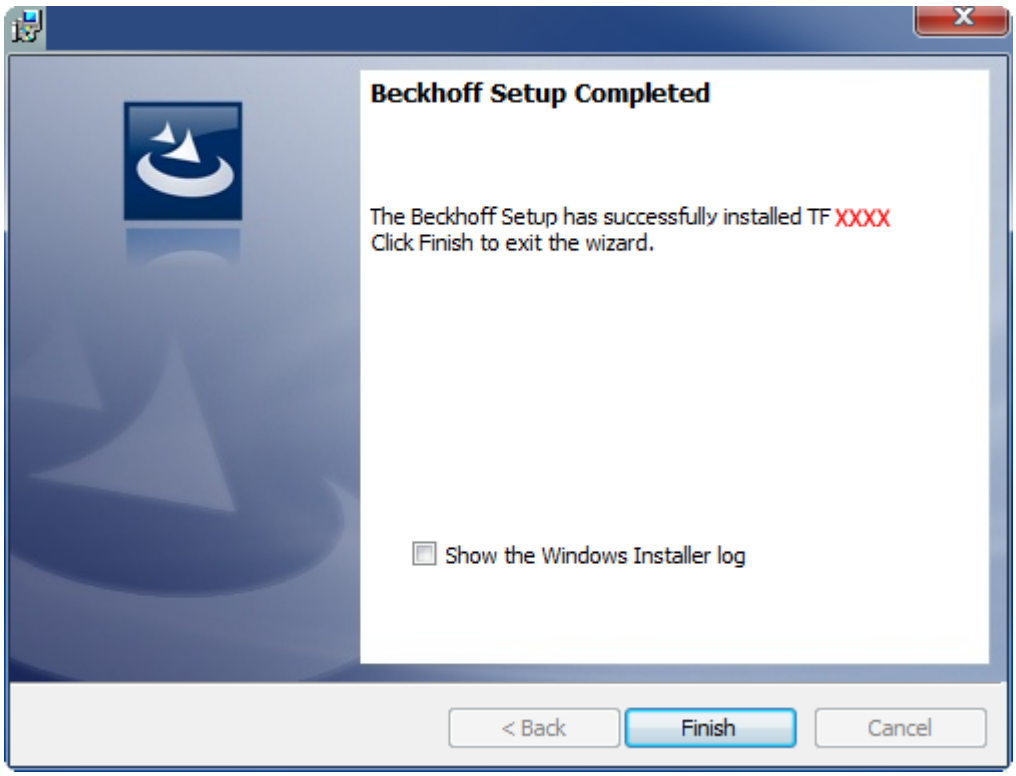


⇨ Ein Dialog weist Sie darauf hin, dass das TwinCAT-System für die weitere Installation gestoppt werden muss.

6. Bestätigen Sie den Dialog mit **Yes.**



7. Wählen Sie **Finish**, um das Setup zu beenden.



⇨ Die TwinCAT 3 Function wurde erfolgreich installiert und kann lizenziert werden (siehe Lizenzierung [▶ 15]).

3.3 Lizenzierung

Die TwinCAT 3 Function ist als Vollversion oder als 7-Tage-Testversion freischaltbar. Beide Lizenztypen sind über die TwinCAT-3-Entwicklungsumgebung (XAE) aktivierbar.

Lizenzierung der Vollversion einer TwinCAT 3 Function

Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT 3 Lizenzierung“.

Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

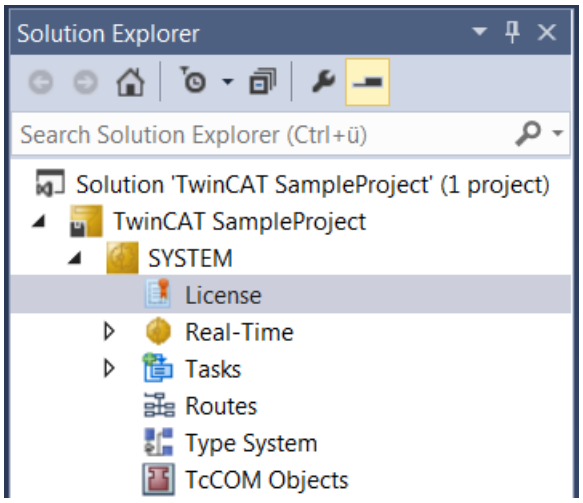
Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.

3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **Choose Target System** das Zielsystem aus.

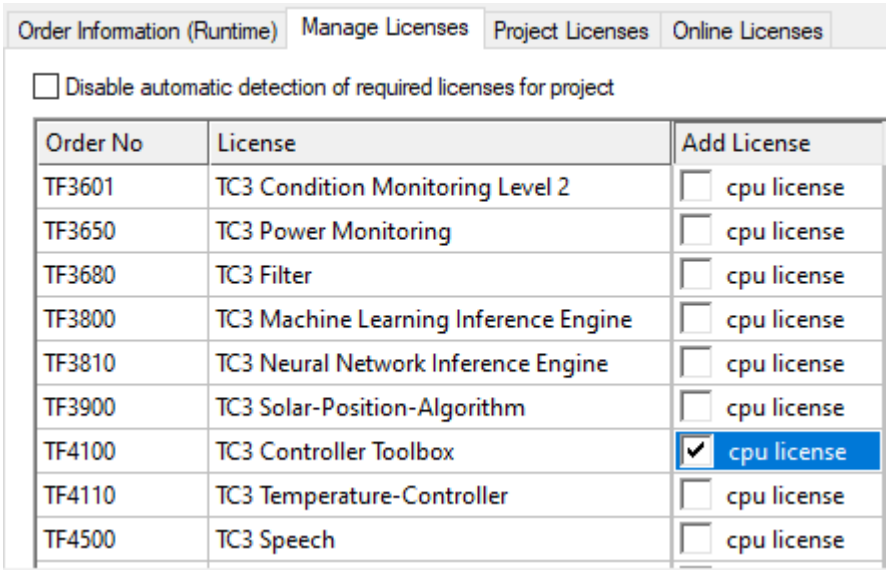
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.

4. Klicken Sie im **Solution Explorer** im Teilbaum **SYSTEM** doppelt auf **License.**



   ⇨ Der TwinCAT-3-Lizenzmanager öffnet sich.

5. Öffnen Sie die Registerkarte **Manage Licenses.** Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (z. B. „TF4100 TC3 Controller Toolbox“).

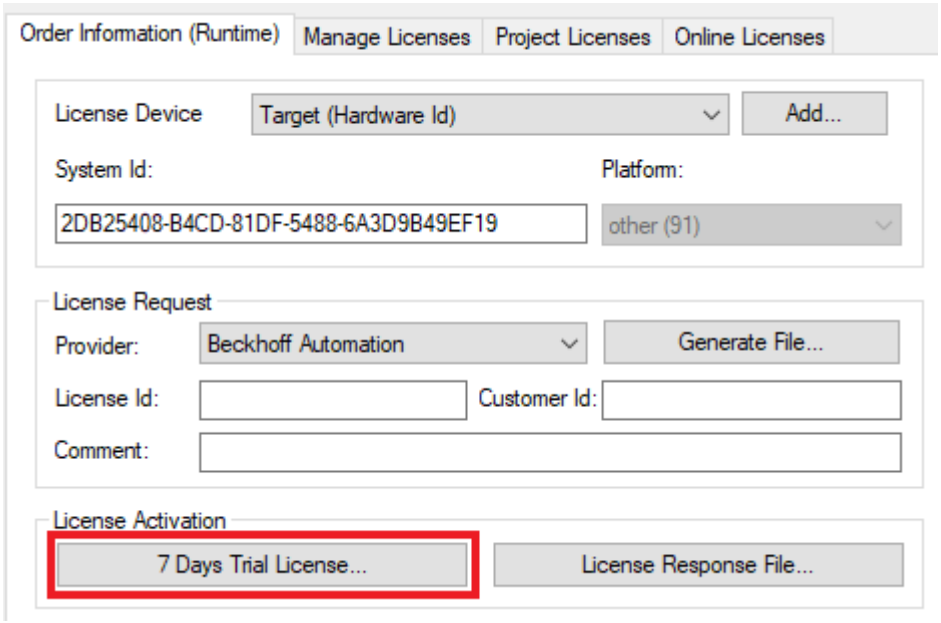


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

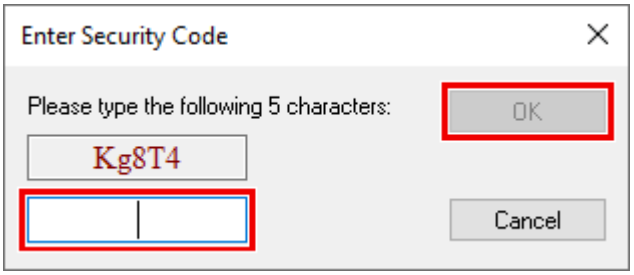
6. Öffnen Sie die Registerkarte **Order Information (Runtime).**

   ⇨ In der tabellarischen Übersicht der Lizenzen wird die zuvor ausgewählte Lizenz mit dem Status „missing“ angezeigt.

7. Klicken Sie auf **7 Days Trial License...**, um die 7-Tage-Testlizenz zu aktivieren.



⇨ Es öffnet sich ein Dialog, der Sie auffordert, den im Dialog angezeigten Sicherheitscode einzugeben.



8. Geben Sie den Code genauso ein, wie er angezeigt wird, und bestätigen Sie ihn.
9. Bestätigen Sie den nachfolgenden Dialog, der Sie auf die erfolgreiche Aktivierung hinweist.
   ⇨ In der tabellarischen Übersicht der Lizenzen gibt der Lizenzstatus nun das Ablaufdatum der Lizenz an.
10. Starten Sie das TwinCAT-System neu.

⇨ Die 7-Tage-Testversion ist freigeschaltet.

4 SPS-Bibliotheken

4.1 Generelle Funktionsweise

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

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 zweiten 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 fünften 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 angegebenen Zykluszeiten bezogen werden. Die Verwendung der Base-Time wird in den folgenden Beispielen erläutert.

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 ist 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 im 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_...	Eventuelle 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 [▶ 174] ausgegeben, wenn der Ausgang `bError` TRUE ist.

Verwendung der globalen Base-Time

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. ...
```

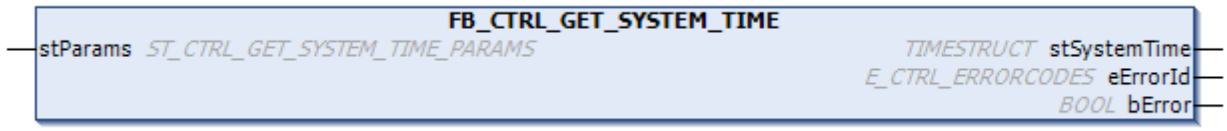
Die an den Bausteinen angegebene **TaskCycleTime** beträgt „2•200E-6s = 400µs“ und entspricht somit der eingestellten SPS-Zykluszeit. Die **CtrlCycleTime** wird auf „800µs = 4•200E-6s“ gesetzt, sodass der Regelkreis mit einer Abtastzeit von 800µs arbeitet, also in jedem zweiten SPS-Zyklus bearbeitet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

4.2 Referenz

4.2.1 Funktionsbausteine

4.2.1.1 Auxiliary

4.2.1.1.1 FB_CTRL_GET_SYSTEM_TIME



Der Funktionsbaustein liest die aktuelle Windows-Systemzeit und stellt sie im **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 10ms 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
```

Name	Typ	Beschreibung
stSystemTime	TIME STRUCT	Struktur, in der die Systemzeit ausgegeben wird.
eErrorId	E_CTRL_ ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.
wYear	WORD	Das Jahr: 1970 ~ 2106;
wMonth	WORD	Der Monat: 1 ~ 12 (Januar = 1, Februar = 2 usw.);
wDayOfWeek	WORD	Der Wochentag: 0 ~ 6 (Sonntag = 0, Montag = 1 usw.);
wDay	WORD	Tag des Monats: 1 ~ 31;
wHour	WORD	Stunde: 0 ~ 23;
wMinute	WORD	Minute: 0 ~ 59;
wSecond	WORD	Sekunde: 0 ~ 59;
WMilliseconds	WORD	Millisekunde: 0 ~ 999;

VAR_IN_OUT

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

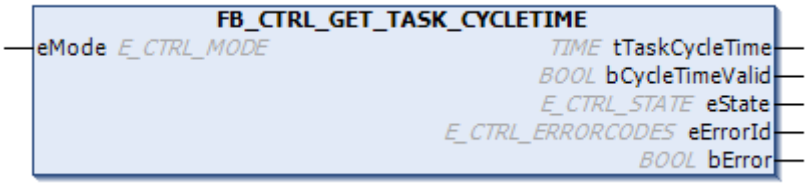
Name	Typ	Beschreibung
stParams	ST_CTRL_ GET_SYSTEM_TI ME	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_GET_SYSTEM_TIME:
STRUCT
    tTaskCycleTime : TIME;
    tCtrlCycleTime : TIME;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Task-Zyklus aufgerufen wird.

4.2.1.1.2 FB_CTRL_GET_TASK_CYCLETIME



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

Korrekte Verwendung des Bausteins

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 Sie Zykluszeiten verwenden, die kleiner 1 ms sind oder die kein Vielfaches von 1 ms sind, diesen Baustein nicht verwenden

VAR_INPUT

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

Name	Typ	Beschreibung
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    tTaskCycleTime    : TIME;
    bCycleTimeValid   : BOOL;
    eState            : E_CTRL_STATE;
    eErrorId          : E_CTRL_ERRORCODES;
    bError            : BOOL;
END_VAR
```

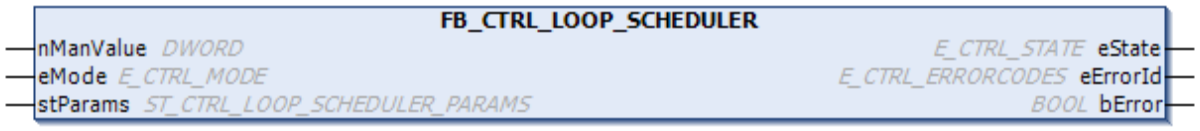
Name	Typ	Beschreibung
tTaskCycleTime	TIME	Dieser Ausgang gibt die aktuelle Task-Zykluszeit mit einer Auflösung von 1ms an.
bCycleTimeValid	BOOL	Wenn dieser Ausgang TRUE ist, ist die am Ausgang `tTaskCycleTime` angegebene Zeit gültig.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	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;
    bInit               : BOOL := TRUE;
    fSetpointValue      : FLOAT := 45.0;
```

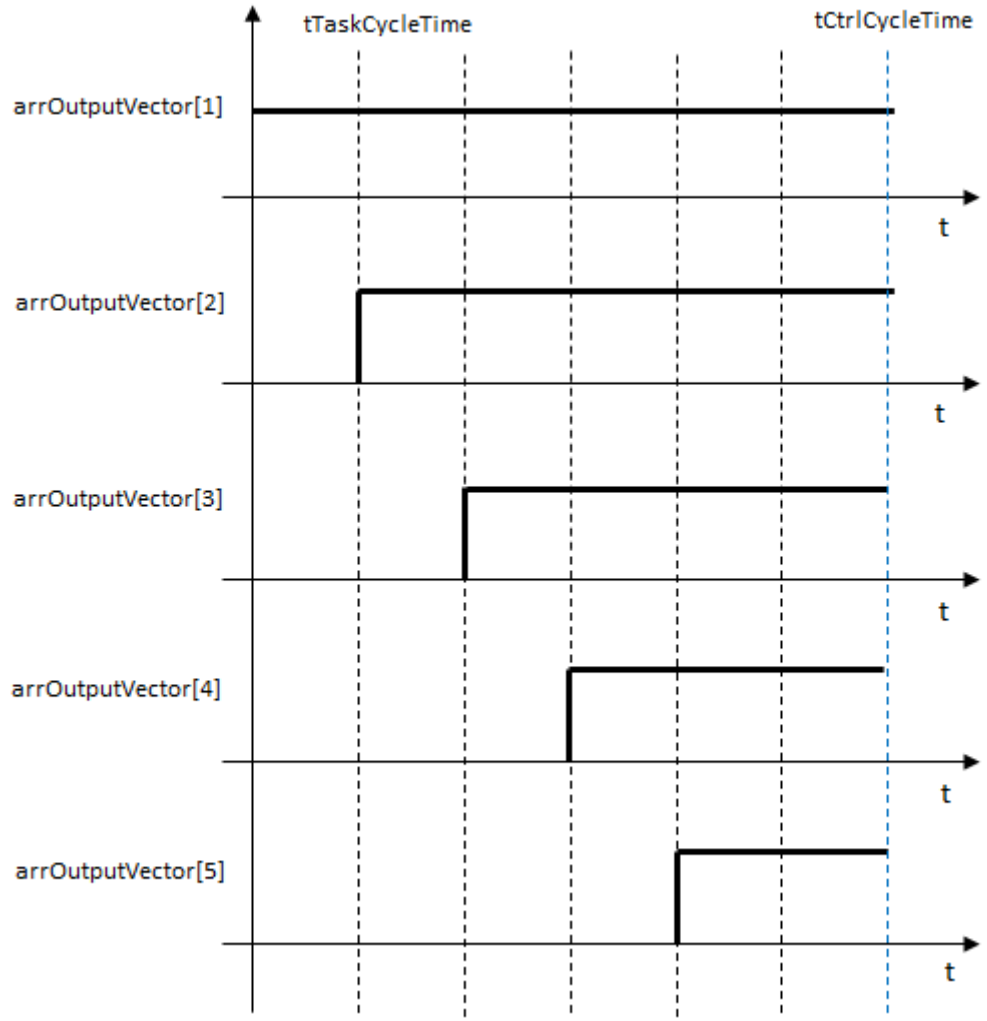
```
    fActualValue        : FLOAT;
    fbCTRL_PI           : FB_CTRL_PI;
    stCTRL_PI_Params    : ST_CTRL_PI_PARAMS;
    fbCTRL_PT1          : FB_CTRL_PT1;
    stCTRL_PT1_Params   : ST_CTRL_PT1_PARAMS;
END_VAR
fbCTRL_GET_TASK_CYCLETIME( eMode    := eCTRL_MODE_ACTIVE,
                           tTaskCycleTime  => tTaskCycleTime,
                           bCycleTimeValid => bCycleTimeValid,
                           eState          => eCTRL_MODE_ACTIVE,
                           eErrorId        => eErrorId,
                           bError          => bError );
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
fbCTRL_PI( fActualValue := fbCTRL_PT1.fOut,
         fSetpointValue := fSetpointValue,
         eMode := eCTRL_MODE_ACTIVE,
         stParams := stCTRL_PI_Params );
fbCTRL_PT1( fIn := fbCTRL_PI.fOut,
            eMode := eCTRL_MODE_ACTIVE,
            stParams := stCTRL_PT1_Params,
            fOut => fActualValue );
END_IF
```

4.2.1.1.3 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 5 Regelkreise verwaltet. Bei diesen gilt „tCtrlCycleTime = 6 · 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
```

Name	Typ	Beschreibung
nManValue	DWORD	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	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_ LOOP_ SCHEDULER_ PARAMS	Parameterstruktur des Loop-Schedulers.

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_LOOP_SCHEDULER_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    nNumberOfControlLoops   : UINT;
    pOutputVector_ADR       : POINTER TO BOOL := 0;
    nOutputVector_SIZEOF    : UINT := 0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der die Regelkreise bearbeitet werden, die mit dem Loop-Scheduler verwaltet werden. Diese muss größer oder gleich der **TaskCycleTime** sein.
tTaskCycle Time	TIME	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.
nNumberOfContr olLoops	UINT	Anzahl der Regelkreise, die verwaltet werden.
pOutputVector_A DR	POINTER TO BOOL	Adresse des Ausgangsvektors
nOutputVector_S IZEOF	UINT	Größe des Ausgangsvektors in Byte

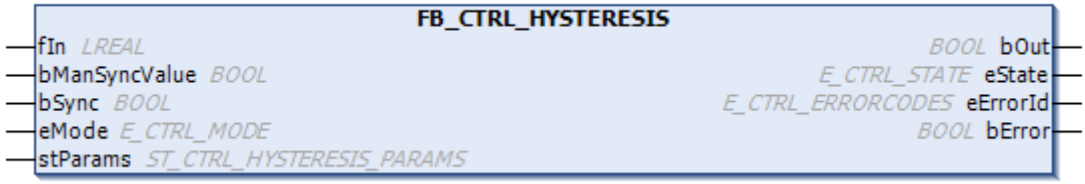
Beispiel:

```
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;
    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
stParams.nOutputVector_SIZEOF  := SIZEOF(arrOutputVector);
stParams.pOutputVector_ADR     := ADR(arrOutputVector);
fbCTRL_LoopScheduler( eMode     := eMode,
                      stParams  := stParams,
                      eErrorId  => eErrorId,
                      bError    => bError);
IF arrOutputVector[1] THEN
    eMode_CtrlLoop_1 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[2] THEN
    eMode_CtrlLoop_2 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[3] THEN
    eMode_CtrlLoop_3 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[4] THEN
    eMode_CtrlLoop_4 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[5]
THEN
    eMode_CtrlLoop_5 := eCTRL_MODE_ACTIVE;
END_IF
```

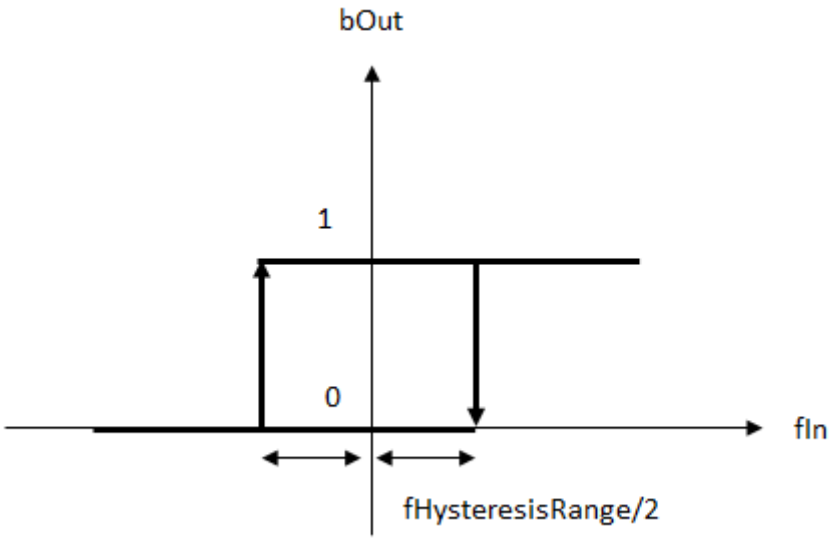
4.2.1.2 Base

4.2.1.2.1 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang des Hysterese-Glieds
bManSync Value	BOOL	Eingang, mit dem das Hysterese-Glied auf einen der beiden Zweige gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das Hysterese-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
bOut	BOOL	Ausgang des Hysterese-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

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

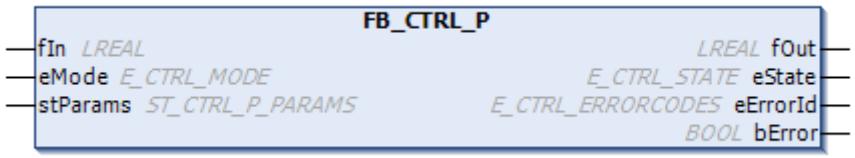
Name	Typ	Beschreibung
stParams	ST_CTRL_HYSTERESIS_PARAMS	Parameterstruktur des Hysterese-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_HYSTERESIS_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fHysteresisRange    : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fHysteresis Range	FLOAT	Hysterese-Bereich, siehe Bild oben.

4.2.1.2.2 FB_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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang des P-Glieds
eMode	E_CTRL_MODE	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des P-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

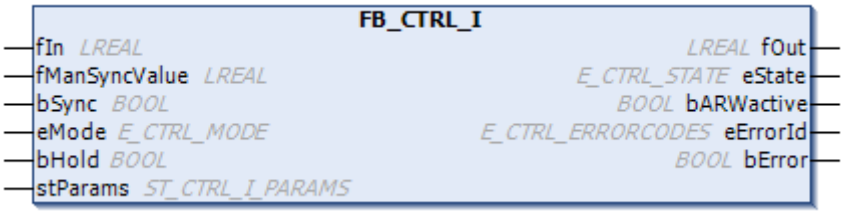
Name	Typ	Beschreibung
stParams	ST_CTRL_P_ PARAMS	Parameterstruktur des P-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_P_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fKp               : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Proportionalverstärkung des P-Glieds

4.2.1.2.3 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang des I-Glieds
fManSyncValue	FLOAT	Eingang, auf den das I-Glied synchronisiert werden kann, oder dessen Wert im Manual-Mode am Ausgang anliegt.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird der Integrator auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des I-Gliedes
bARWactive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich der Integrator in der Begrenzung befindet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer.
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

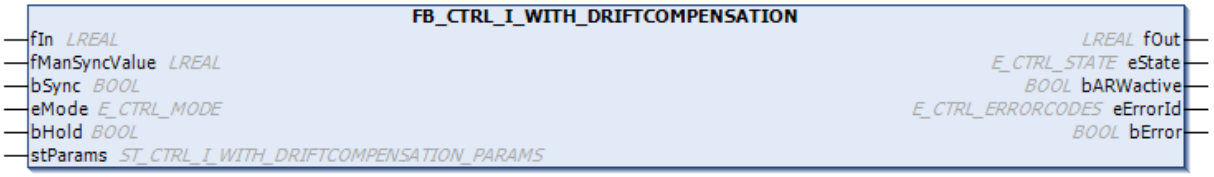
Name	Typ	Beschreibung
stParams	ST_CTRL_I_ PARAMS	Parameterstruktur des I-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_I_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    tTi               : TIME := T#0ms;
    fOutMaxLimit      : FLOAT := 1E38;
    fOutMinLimit      : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

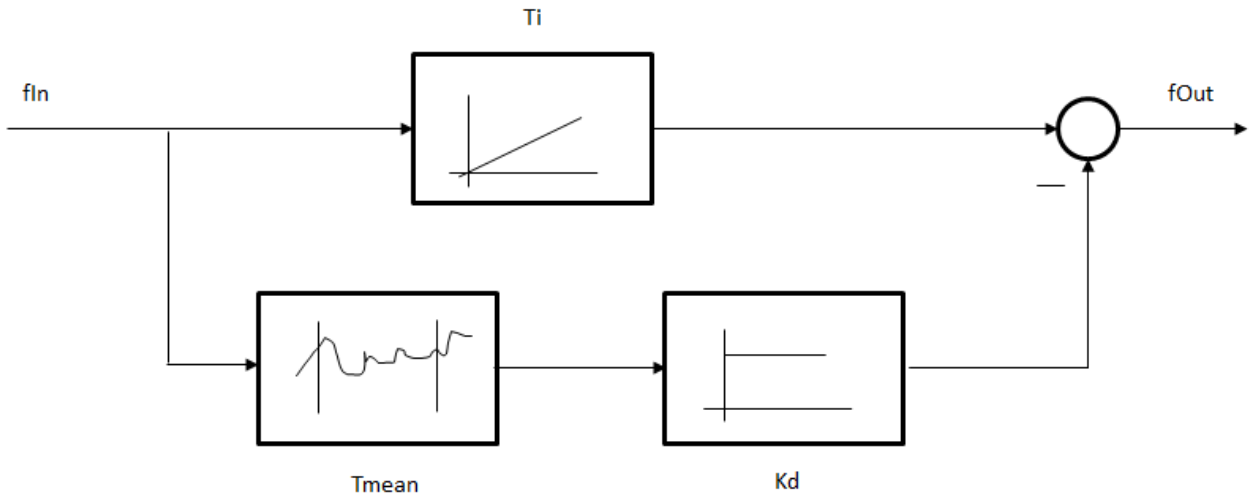
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der `TaskCycleTime` sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tTi	TIME	Integrationszeit des I-Glieds
fOutMaxLimit	FLOAT	Oberes Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang `bARWActive` signalisiert.
fOutMinLimit	FLOAT	Unteres Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang `bARWActive` signalisiert.

4.2.1.2.4 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang des I-Glieds
fManSyncValue	FLOAT	Eingang, auf den das I-Glied synchronisiert werden kann, oder dessen Wert im Manual-Mode am Ausgang anliegt.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird der Integrator auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des I-Gliedes
bARWactive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich der Integrator in der Begrenzung befindet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_I_ WITH_DRIFTCO MPENSATION_ PARAMS	Parameterstruktur des I-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    tTi                     : TIME := T#0ms;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    fDampingCoefficient     : FLOAT := 0.0;
    tAveragingTime          : TIME := T#0ms;
    pWorkArray_ADR          : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF       : UINT := 0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tTi	TIME	Integrationszeit des I-Glieds
fOutMaxLimit	FLOAT	Oberes Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang `bARWActive` signalisiert.
fOutMinLimit	FLOAT	Unteres Limit, an dem die Integration angehalten wird (ARW-Maßnahme). Das Erreichen dieses Limits wird durch ein TRUE an dem Ausgang `bARWActive` signalisiert.
fDampingCoefficient	FLOAT	Faktor k_d im Wirkungsplan
tAveragingTime	TIME	Zeit, über die das gleitende Mittelwertfilter den Mittelwert bildet.
pWorkArray_ADR	POINTER TO FLOAT	Adresse eines Array of FLOAT der Größe `tAveragingTime / tCtrlCycleTime` (Siehe Beschreibung des Bausteins: FB_CTRL_MOVING_AVERAGE.)
nWorkArray_SIZEOF	UINT	Größe eines Array of FLOAT der Größe `tAveragingTime / tCtrlCycleTime` (Siehe Beschreibung des Bausteins: FB_CTRL_MOVING_AVERAGE.)

4.2.1.2.5 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang des D-Gliedes
fManSyncValue	FLOAT	Eingang, auf den das D-Glied synchronisiert werden kann oder dessen Wert im Manual-Mode am Ausgang anliegt.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das D-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des D-Gliedes
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_D_ PARAMS	Parameterstruktur des D-Gliedes

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_D_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME  := T#0ms;
    tTaskCycleTime  : TIME  := T#0ms;
    tTv             : TIME  := T#0ms;
    tTd             : TIME  := T#0ms;
    fOutMaxLimit    : FLOAT := 1E38;
    fOutMinLimit    : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tTv	TIME	Differenzierzeitkonstante
tTd	TIME	Dämpfungszeitkonstante
fOutMaxLimit	FLOAT	Oberes Limit, an dem der Ausgang des D-Gliedes begrenzt wird.
fOutMinLimit	FLOAT	Unteres Limit, an dem der Ausgang des D-Gliedes begrenzt wird.

4.2.1.2.6 FB_CTRL_TRANSFERFUNCTION_1

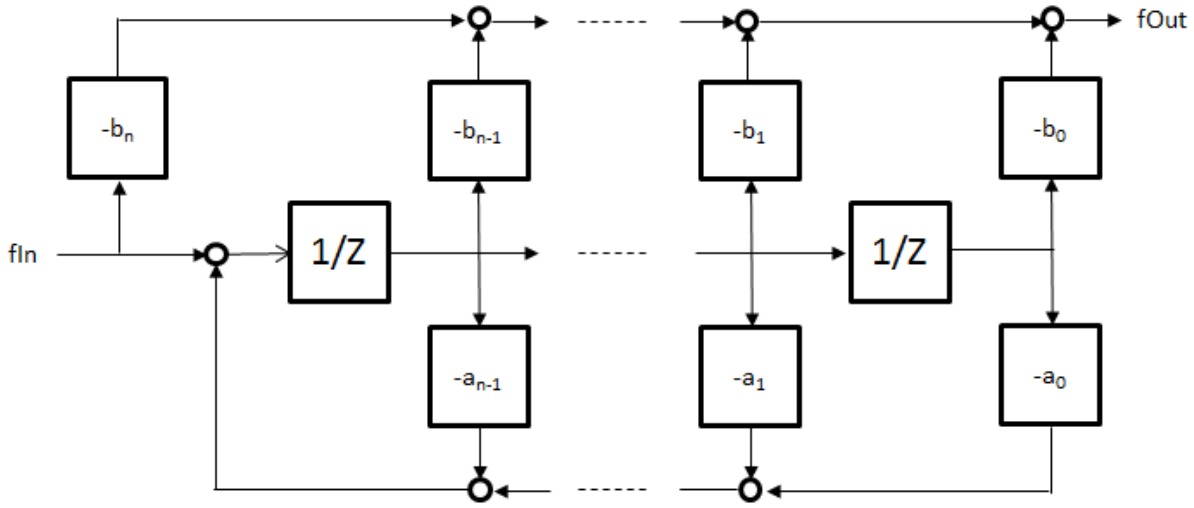
Dieser Baustein berechnet eine diskrete Übertragungsfunktion mit der unten dargestellten ersten Standardform. Die Übertragungsfunktion kann hierbei von beliebiger Ordnung „n“ sein.

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

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

Beschreibung des Übertragungsverhaltens

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



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

```
aNumArray        : ARRAY[0..nNumOrder] OF FLOAT;
aDenArray        : ARRAY[0..nDenOrder] OF FLOAT;
aStructTfData    : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_1_DATA;
```

In dem Array `aNumArray` werden die Koeffizienten „b_0“ „bis b_n“ abgelegt. Diese müssen folgendermaßen angeordnet werden:

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

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

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

Name	Typ	Beschreibung
fIn	FLOAT	Eingang der Übertragungsfunktion
fManValue	FLOAT	Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.
eMode	E_CTRL_MODE	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang der Übertragungsfunktion
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
eErrorId	E_CTRL_ERRORCODES	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_TRANSFERFUNCTION_1_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_TRANSFERFUNCTION_1_PARAMS:
STRUCT
    tTaskCycleTime                : TIME;
    tCtrlCycleTime                : TIME := T#0ms;
    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_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tCtrlCycleTime	TIME	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	USINT	Ordnung der Übertragungsfunktion [0...]
pNumeratorArray_ADR	POINTER TO FLOAT	Adresse des Arrays mit den Zählerkoeffizienten
nNumeratorArray_SIZEOF	UINT	Größe des Arrays mit den Zählerkoeffizienten in Byte
pDenominatorArray_ADR	POINTER TO FLOAT	Adresse des Arrays mit den Nennerkoeffizienten
nDenomiantorArray_SIZEOF	UINT	Größe des Arrays mit den Nennerkoeffizienten in Byte
pTransferfunction1Data_ADR	POINTER TO ST_CTRL_TRANSFER FUNCTION_1_DATA	Adresse des Daten-Arrays
nTransferfunction1Data_SIZEOF	UINT	Größe des Daten-Arrays in Byte

Beispiel:

```
PROGRAM PRG_TRANSFERFUNCTION_1_TEST
VAR CONSTANT
    nTfOrder    : USINT := 2;
END_VAR
VAR
    aNumArray          : ARRAY[0..nNumOrder] OF FLOAT;
    aDenArray          : ARRAY[0..nDenOrder] OF FLOAT;
    aStTfData          : ARRAY[0..nTfOrder] 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
    aNumArray[0] := 1.24906304658218E-007;
    aNumArray[1] := 2.49812609316437E-007;
    aNumArray[2] := 1.24906304658218E-007;
    aDenArray[0] := 0.998501124344101;
    aDenArray[1] := -1.99850062471888;
    aDenArray[2] := 1.0;
    stParams.tTaskCycleTime        := T#2ms;
    stParams.tCtrlCycleTime        := T#2ms;
    stParams.nOrderOfTheTransferfunction := nTfOrder;
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
stParams.pNumeratorArray_ADR           := ADR(aNumArray);
stParams.nNumeratorArray_SIZEOF        := SIZEOF(aNumArray);
stParams.pDenominatorArray_ADR         := ADR(aDenArray);
stParams.nDenominatorArray_SIZEOF      := SIZEOF(aDenArray);
stParams.pTransferfunction1Data_ADR    := ADR(aStructTfData);
stParams.nTransferfunction1Data_SIZEOF := SIZEOF(aStructTfData);
fbTansferfunction (fIn := fIn,
                   eMode := eMode,
                   stParams := stParams,
```

```
                fOut => fOut,
                eErrorId => eErrorId,
                bError => bError);
```

4.2.1.2.7 FB_CTRL_TRANSFERFUNCTION_2

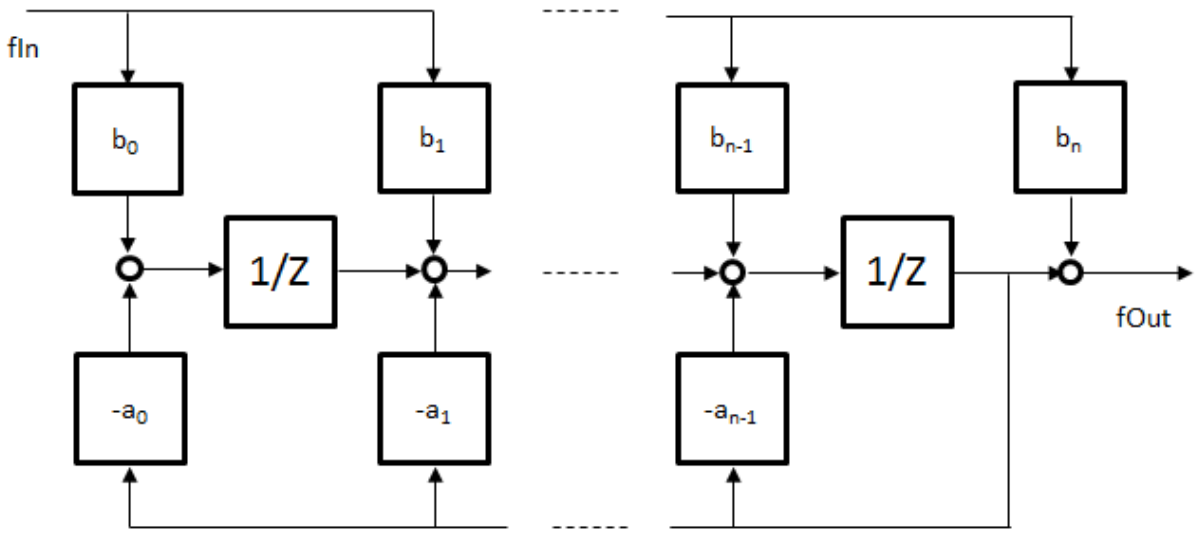
Dieser Baustein berechnet eine diskrete Übertragungsfunktion mit der unten dargestellten zweiten Standardform. Die Übertragungsfunktion kann hierbei von beliebiger Ordnung „n“ sein.

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

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

Beschreibung des Übertragungsverhaltens

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



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

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

Die Koeffizienten des Zählerpolynoms berechnen sich dabei nach der folgenden Vorschrift:

$$\tilde{b}_i = b_i - b_n a_i \quad \forall 0 \le i < n$$

$$\tilde{b}_n = b_n$$

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

```
aNumArray    : ARRAY[0..nTfOrder] OF FLOAT;
aDenArray    : ARRAY[0..nTfOrder] OF FLOAT;
aStTfData    : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_2_DATA;
```

In dem Array `aNumArray` werden die Koeffizienten „b_0“ bis „b_n“ abgelegt. Diese müssen folgendermaßen angeordnet werden:

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

In dem Array `aDenArray` werden die Koeffizienten „a_0“ bis a_n“ abgelegt. Diese müssen folgendermaßen angeordnet werden:

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

Die internen Daten, die der Baustein benötigt, werden in dem Array `aStTfData` 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang der Übertragungsfunktion
fManValue	FLOAT	Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.
eMode	E_CTRL_MODE	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang der Übertragungsfunktion
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
eErrorId	E_CTRL_ERRORCODES	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_TRANSFERFUNCTION_2_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_TRANSFERFUNCTION_2_PARAMS:
STRUCT
    tTaskCycleTime                : TIME;
    tCtrlCycleTime                : TIME := T#0ms;
    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_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tCtrlCycleTime	TIME	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	USINT	Ordnung der Übertragungsfunktion [0...]
pNumeratorArray_ADR	POINTER TO FLOAT	Adresse des Arrays mit den Zählerkoeffizienten
nNumeratorArray_SIZEOF	UINT	Größe des Arrays mit den Zählerkoeffizienten in Byte
pDenominatorArray_ADR	POINTER TO FLOAT	Adresse des Arrays mit den Nennerkoeffizienten
nDenomiantorArray_SIZEOF	UINT	Größe des Arrays mit den Nennerkoeffizienten in Byte
pTransferfunction2Data_ADR	POINTER TO ST_CTRL_TRANSFERFUNCTION_2_DATA	Adresse des Daten-Arrays
nTransferfunction2Data_SIZEOF	UINT	Größe des Daten-Arrays in Byte

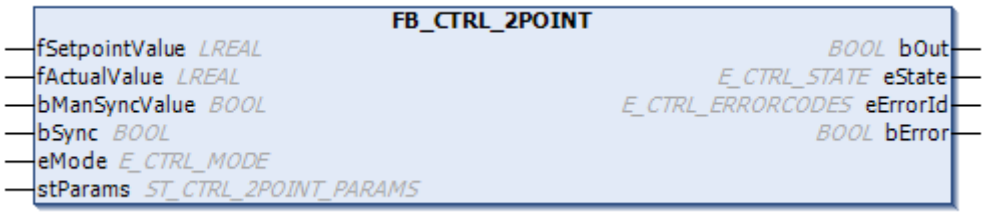
Beispiel:

```
PROGRAM PRG_TRANSFERFUNCTION_2_TEST
VAR CONSTANT
    nTfOrder : USINT := 2;
END_VAR
VAR
    aNumArray           : ARRAY[0..nTfOrder] OF FLOAT;
    aDenArray           : ARRAY[0..nTfOrder] OF FLOAT;
    aStTfData           : ARRAY[0..nTfOrder] 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
    aNumArray[0] := 1.24906304658218E-007;
    aNumArray[1] := 2.49812609316437E-007;
    aNumArray[2] := 1.24906304658218E-007;
    aDenArray[0] := 0.998501124344101;
    aDenArray[1] := -1.99850062471888;
    aDenArray[2] := 1.0;
    stParams.tTaskCycleTime    := T#2ms;
    stParams.tCtrlCycleTime    := T#2ms;
    stParams.nOrderOfTheTransferfunction := nTfOrder;
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
stParams.pNumeratorArray_ADR := ADR(aNumArray);
stParams.nNumeratorArray_SIZEOF := SIZEOF(aNumArray);
stParams.pDenominatorArray_ADR := ADR(aDenArray );
stParams.nDenominatorArray_SIZEOF := SIZEOF(aDenArray );
stParams.pTransferfunction2Data_ADR := ADR(aStTfData );
stParams.nTransferfunction2Data_SIZEOF := SIZEOF(aStTfData);
fbTansferfunction (fIn := fIn,
                   eMode := eMode,
                   stParams := stParams,
                   fOut => fOut,
                   eErrorId => eErrorId,
                   bError => bError);
```

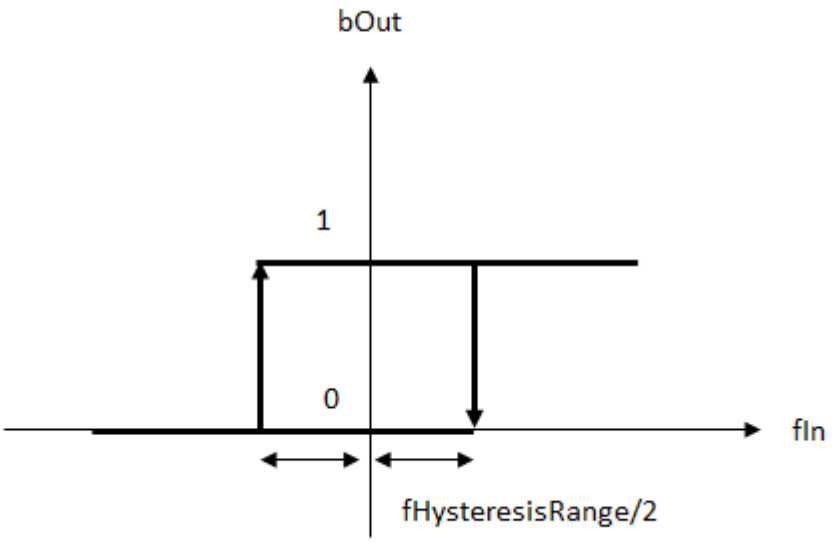
4.2.1.3 Controller

4.2.1.3.1 FB_CTRL_2POINT



Der Funktionsbaustein stellt ein 2-Punkt-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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
bManSyncValue	BOOL	Eingang, mit dem das 2-Punkt-Glied auf einen der beiden Zweige gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das 2-Punkt-Glied auf den Wert „bManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
bOut	BOOL	Ausgang des 2-Punkt-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

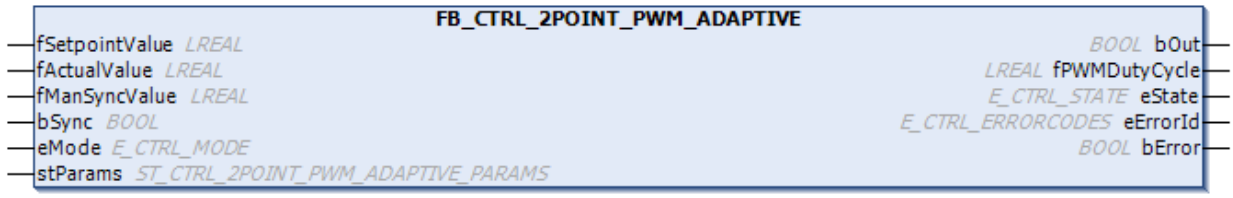
Name	Typ	Beschreibung
stParams	ST_CTRL_2POINT_PARAMS	Parameterstruktur des 2-Punkt-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fHysteresisRange    : FLOAT;
END_STRUCT
END_TYPE
```

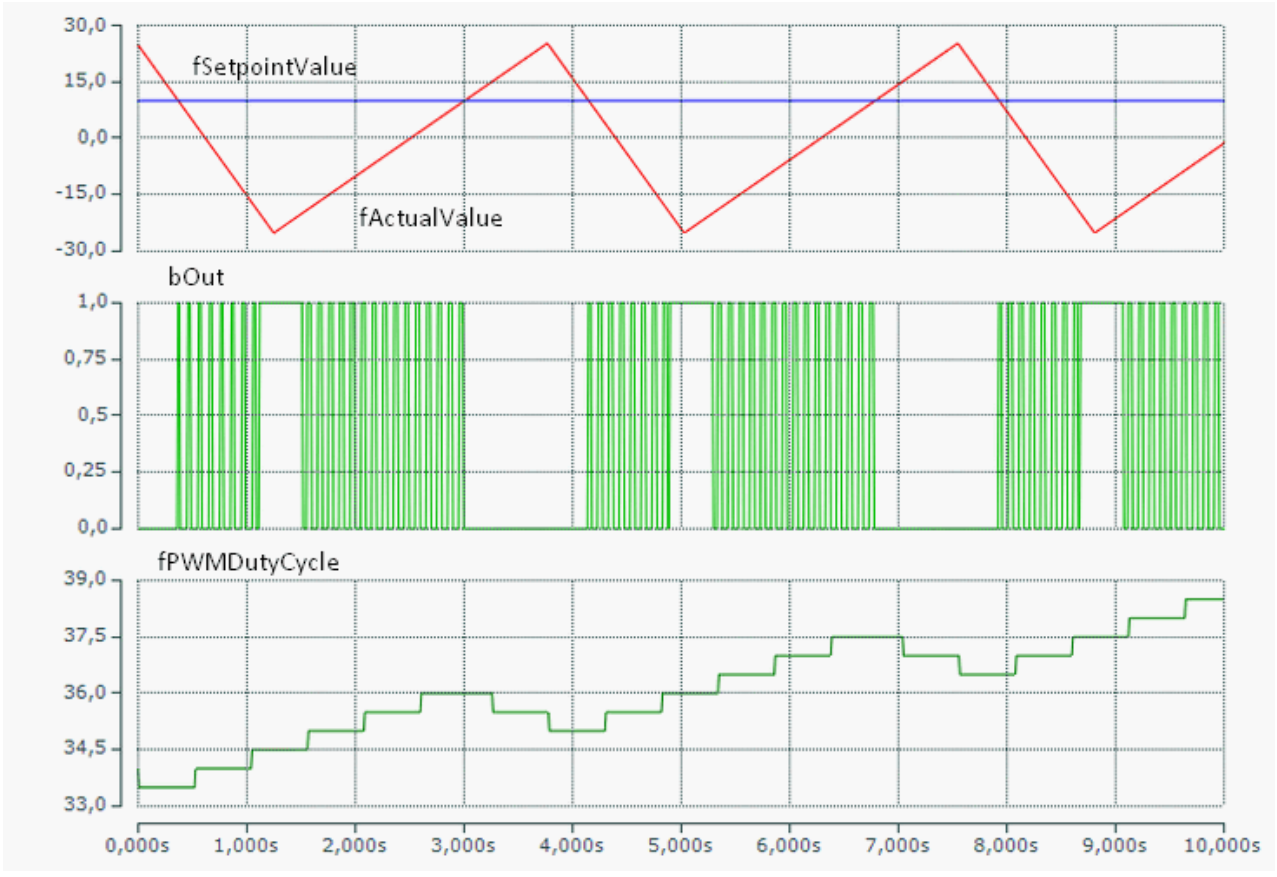
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycle Time	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fHysteresis Range	FLOAT	Hysterese-Bereich, siehe Bild oben.

4.2.1.3.2 FB_CTRL_2POINT_PWM_ADAPTIVE



Der Funktionsbaustein stellt einen adaptiven 2-Punkt-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 nutzt intern einen PWM-Baustein, der zur Ansteuerung des thermischen Stellgliedes 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 „0“ ist und abgeschaltet, wenn die Regelabweichung kleiner „0“ 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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
fManSync Value	FLOAT	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	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das Puls-Pausen-Verhältnis des internen PWM-Bausteins auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
bOut	BOOL	Ausgang des Reglers
fPWMDutyCycle	FLOAT	Aktuelles Puls-Pausen-Verhältnis des internen PWM-Bausteins
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS	Parameterstruktur des 2-Punkt-Gliedes

`stParams` besteht aus den folgenden Elementen:

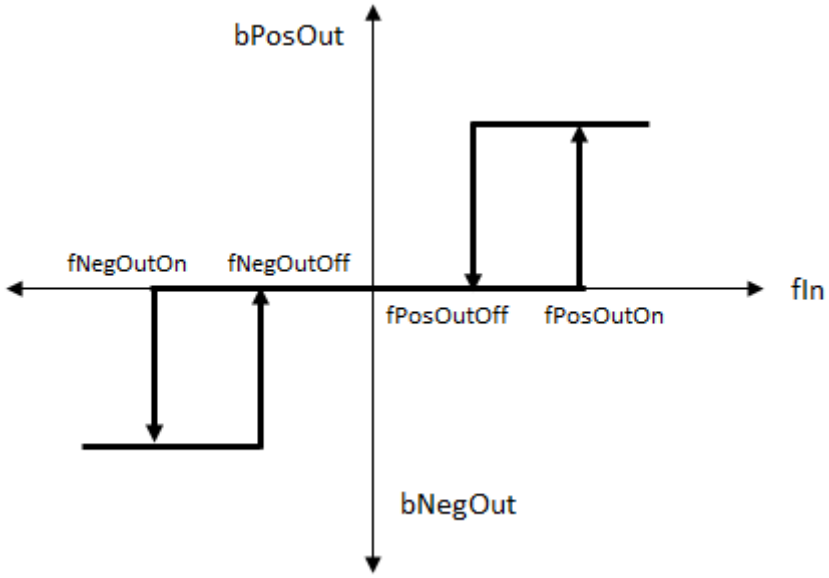
```
TYPE
ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    tPWMPeriod        : TIME;
    fOkRange          : FLOAT;
    fForceRange       : FLOAT;
    fStepSize         : FLOAT;
    fMinLimit         : FLOAT;
    fMaxLimit         : FLOAT;
    tWaitTime         : TIME;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tPWMPeriod	TIME	Periodendauer des PWM-Signals
fOkRange	FLOAT	Bereich von „fE“, in dem das Puls-Pausen-Verhältnis nicht verändert wird.
fForceRange	FLOAT	Wenn „fE“ diesen Bereich überschreitet, wird das Ausgang dauerhaft auf TRUE gesetzt.
fStepSize	FLOAT	Wert, um den das Puls-Pausen-Verhältnis bei jeder Adaption variiert wird. [0% ... 100%]
fMinLimit	FLOAT	Maximales Puls-Pausen-Verhältnis in Prozent [0% ... 100%]
fMaxLimit	FLOAT	Minimales Puls-Pausen-Verhältnis in Prozent [0% ... 100%]
tWaitTime	TIME	Wartezeit zwischen den einzelnen Variationen des Puls-Pausen-Verhältnisses

4.2.1.3.3 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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße

nManSyncValue	INT	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	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das 3-Punkt-Glied auf den Wert „bManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
bPosOut	BOOL	Dieser Ausgang des 3-Punkt-Glieds ist TRUE, wenn der obere Zweig der Kennlinie aktiv ist.
bNegOut	BOOL	Dieser Ausgang des 3-Punkt-Glieds ist TRUE, wenn der untere Zweig der Kennlinie aktiv ist.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

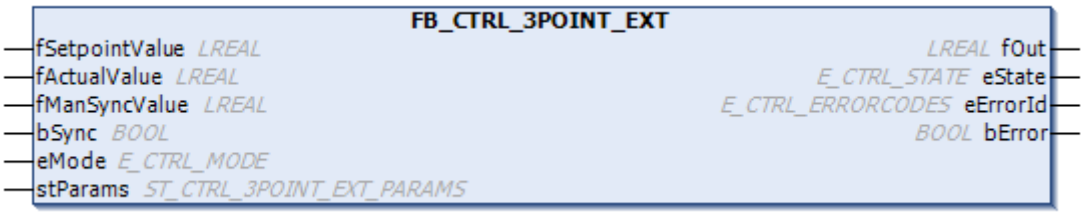
Name	Typ	Beschreibung
stParams	ST_CTRL_3POIN T_PARAMS	Parameterstruktur des 3-Punkt-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_3POINT_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fPosOutOn         : FLOAT;
    fPosOutOff        : FLOAT;
    fNegOutOn         : FLOAT;
    fNegOutOff        : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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.
tTaskCycle Time	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fPosOutOn	FLOAT	Regelabweichung, bei der von „bPosOut = FALSE zu bPosOut = TRUE“ geschaltet wird (bNegOut = FALSE).
fPosOutOff	FLOAT	Regelabweichung, bei der von „bPosOut = TRUE zu bPosOut = FALSE“ geschaltet wird (bNegOut = FALSE).
fNegOutOn	FLOAT	Regelabweichung, bei der von „bNegOut = FALSE zu bNegOut = TRUE“ geschaltet wird (bPosOut = FALSE).
fNegOutOff	FLOAT	Regelabweichung, bei der von „bNegOut = TRUE zu bNegOut = FALSE“ geschaltet wird (bPosOut = FALSE).

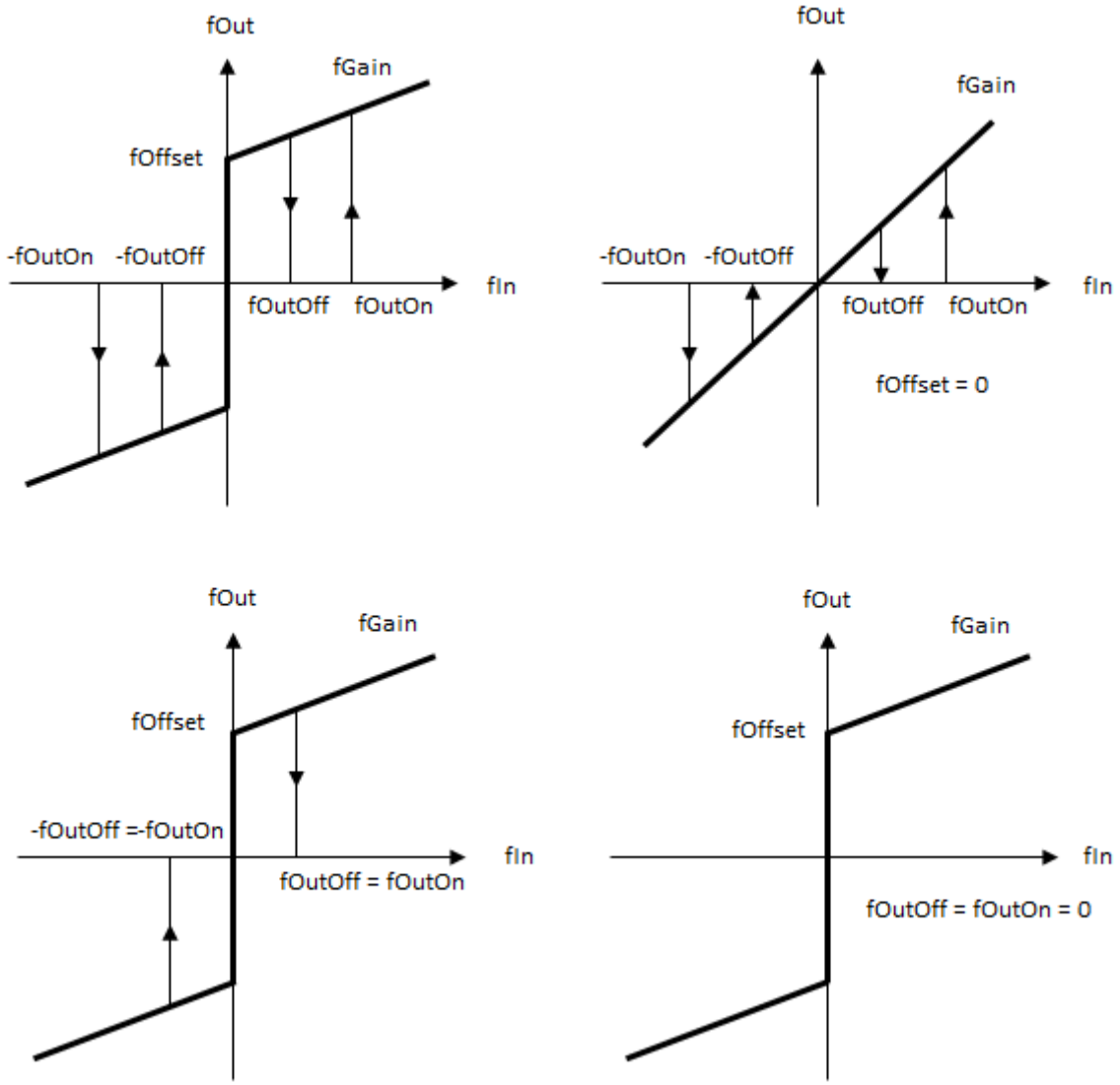
4.2.1.3.4 FB_CTRL_3POINT_EXT



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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße

fManSyncValue	FLOAT	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	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das 3-Punkt-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des erweiterten 3-Punkt-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

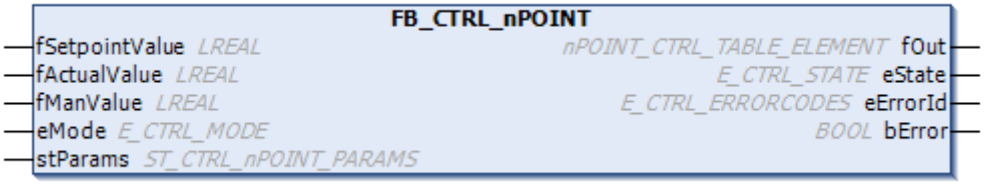
Name	Typ	Beschreibung
stParams	ST_CTRL_ 3POINT_EXT_ PARAMS	Parameterstruktur des erweiterten 3-Punkt-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_3POINT_EXT_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fOutOff           : FLOAT;
    fOutOn            : FLOAT;
    fGain             : FLOAT;
    fOffset           : FLOAT;
END_STRUCT
END_TYPE
```

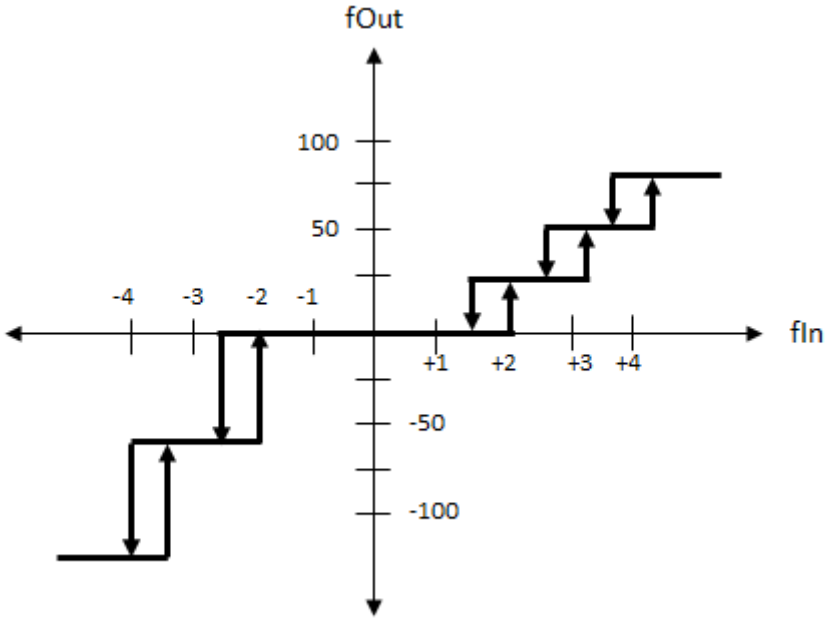
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycle Time	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fOutOff	FLOAT	Wenn die Regelabweichung diesen Wert unterschreitet, wird der Ausgang abgeschaltet (auf „0“ gesetzt).
fOutOn	FLOAT	Wenn die Regelabweichung diesen Wert überschreitet, wird der Ausgang eingeschaltet.
fGain	FLOAT	Verstärkungsfaktor
fOffset	FLOAT	Offset

4.2.1.3.5 FB_CTRL_nPOINT



Der Funktionsbaustein stellt ein n-Punkt-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 ersten 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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
fManValue	BOOL	Eingang, der im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	nPOINT_CTRL_TABLE_ELEMENT	Ausgang des n-Punkt-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

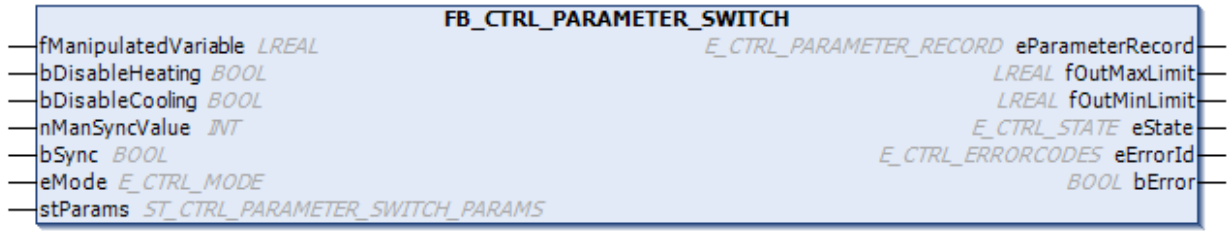
Name	Typ	Beschreibung
stParams	ST_CTRL_nPOINT_PARAMS	Parameterstruktur des n-Punkt-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_nPOINT_PARAMS :
STRUCT
    tCtrlCycleTime           : TIME := T#0ms;
    tTaskCycleTime           : TIME := T#0ms;
    pDataTable_ADR           : POINTER TO nPOINT_CTRL_TABLE_ELEMENT
     := 0;
    nDataTable_SIZEOF        : UINT := 0;
    nDataTable_NumberOfRows  : UINT := 0;
    fHysteresisRange         : FLOAT;
END_STRUCT
END_TYPE
```

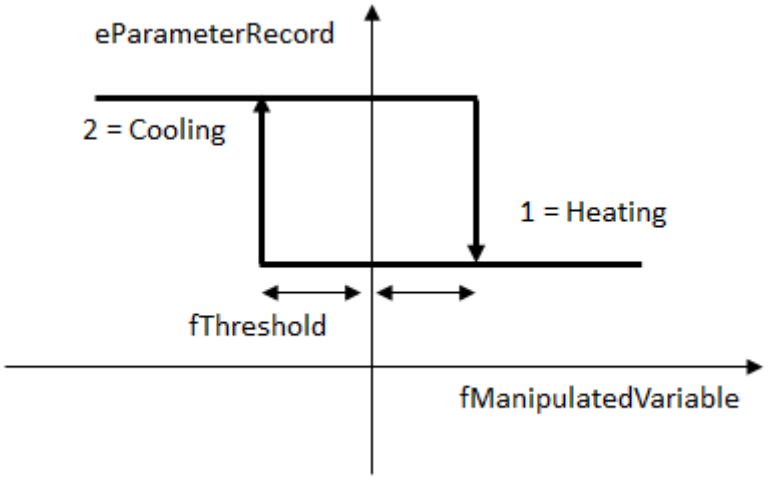
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycle Time	TIME	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	POINTER TO nPOINT_CTRL_ TABLE_ ELEMENT	Adresse der Daten-Tabelle
nDataTable_ SIZEOF	UINT	Größe der Daten-Tabelle in Bytes
nDataTable_ NumberOf Rows	UINT	Zeilenanzahl der Daten-Tabelle
fHysteresis Range	FLOAT	Hysterese-Bereich, siehe Bild oben. Der Hysterese-Bereich wirkt wie bei dem FB_CTRL_2POINT [▶ 41] beschrieben.

4.2.1.3.6 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 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;
    nManSyncValue         : eCTRL_PARAMETER_RECORD_HEATING;
    bSync                 : BOOL;
    eMode                 : E_CTRL_MODE;
END_VAR
```

Name	Typ	Beschreibung
fManipulated Variable	FLOAT	Eingangsgröße des `FB_Parameter_Switch`, diese sollte gleich der Ausgangsgröße des Reglers sein.
nManSync Value	eCTRL_PARAMETER_RECORD_HEATING	Eingang, mit dem der Funktionsbaustein auf einen der beiden Parameterbereiche gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird der Funktionsbaustein auf den Wert „nManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] 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
```

Name	Typ	Beschreibung
eParameterRecord	E_CTRL_PARAMETER_RECORD	Ausgang des Funktionsbausteins, der den Parameterbereich angibt.
fOutMaxLimit	FLOAT	Maximale Ausgangsgröße, mit der der Regler begrenzt wird. (Sollte in die Parameter-Struktur des Reglers kopiert werden.)
fOutMinLimit	FLOAT	Minimale Ausgangsgröße, mit der der Regler begrenzt wird. (Sollte in die Parameter-Struktur des Reglers kopiert werden.)
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_PARAMETER_SWITCH_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tTaskCycleTime    : TIME;
    tCtrlCycleTime    : TIME;
    fThreshold        : FLOAT;
    fOutMaxLimit      : FLOAT;
    fOutMinLimit      : FLOAT;
```

```
    tMinWaitTime      : TIME;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tCtrlCycleTime	TIME	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	FLOAT	Schaltschwelle, siehe Bild oben.
fOutMaxLimit	FLOAT	Max-Limit, welches an den Regler weitergegeben wird.
fOutMinLimit	FLOAT	Min-Limit, welches an den Regler weitergegeben wird.
tMinWaitTime	TIME	Wartezeit, siehe Beschreibung oben.

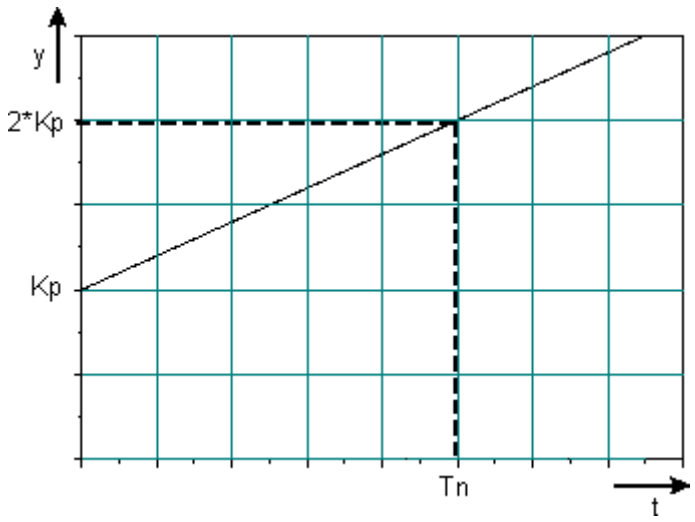
4.2.1.3.7 FB_CTRL_PI

Der Funktionsbaustein stellt ein PI-Übertragungsglied im Wirkungsplan dar.

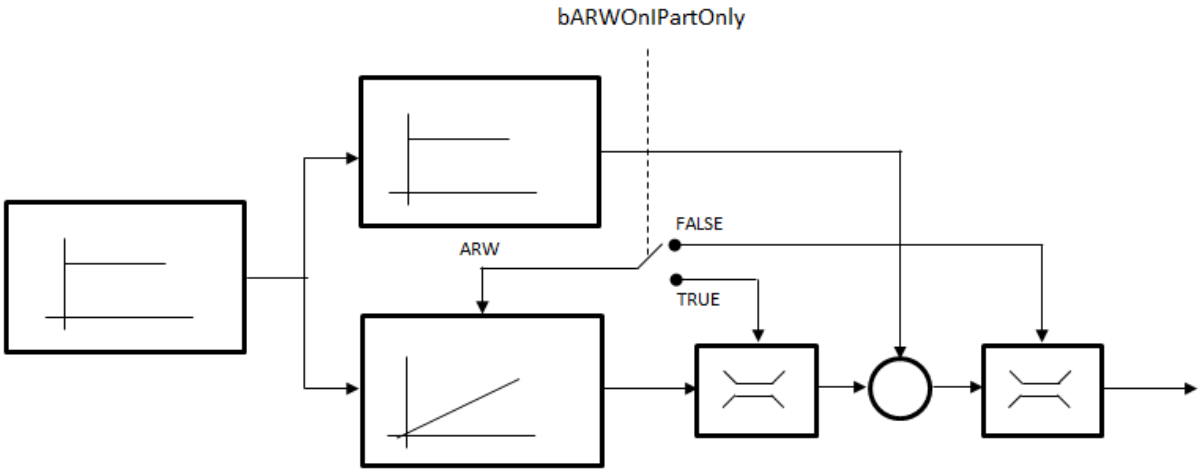
Verhalten des Ausgangs

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

Sprungantwort



ARW



VAR_INPUT

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

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
fManSyncValue	FLOAT	Eingang, mit dem das PI-Glied gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PI-Glieds
bARWactive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PI-Glied in der Begrenzung befindet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

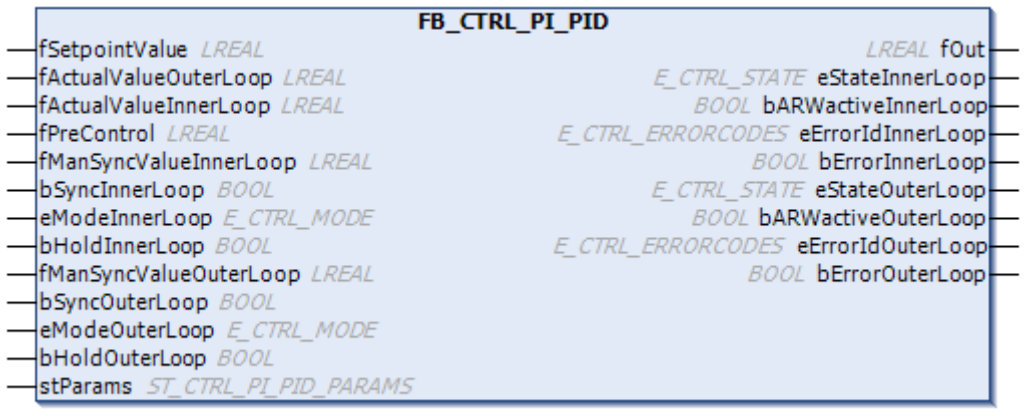
Name	Typ	Beschreibung
stParams	ST_CTRL_PI_PARAMS	Parameterstruktur des PI-Gliedes

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_PI_PARAMS:
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    tTn                 : TIME := T#0ms;
    fKp                 : FLOAT := 0;
    fOutMaxLimit        : FLOAT := 1E38;
    fOutMinLimit        : FLOAT := -1E38;
    bARWOnIPartOnly     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tTn	TIME	Nachstellzeit
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
fOutMaxLimit	FLOAT	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	FLOAT	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	BOOL	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. (Vergl. Wirkungsplan.)

4.2.1.3.8 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`.

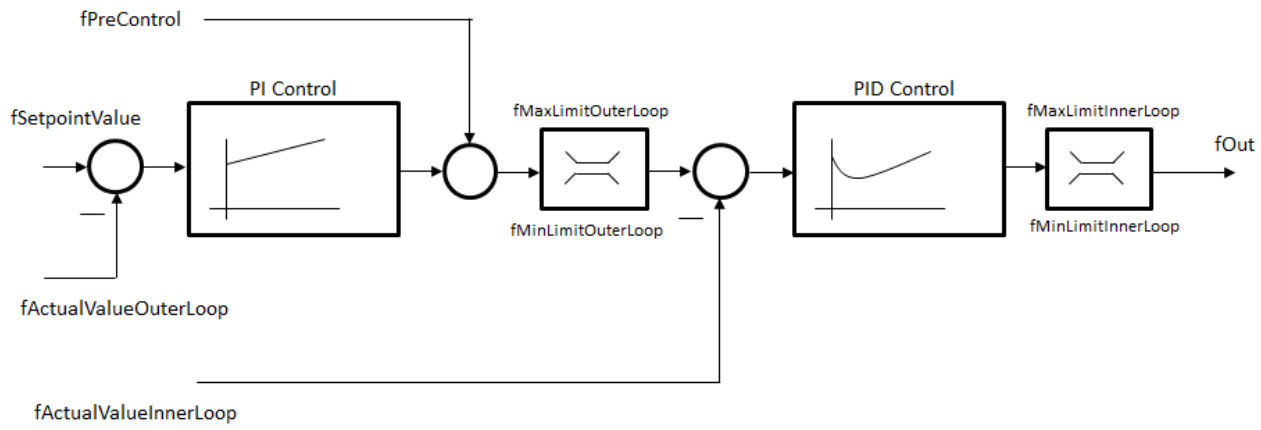
Übertragungsfunktion des PI-Glieds

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

Übertragungsfunktion des PID-Glieds

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

Wirkungsplan des kaskadierten Übertragungsglieds



VAR_INPUT

```
VAR_INPUT
    fSetpointValue          : FLOAT;
    fActualValueOuterLoop   : FLOAT;
    fActualValueInnerLoop   : FLOAT;
    fPreControl             : FLOAT;
    fManSyncValueInnerLoop  : FLOAT;
    bSyncInnerLoop          : BOOL;
    eModeInnerLoop          : E_CTRL_MODE;
    bHoldInnerLoop          : BOOL;
    fManSyncValueOuterLoop  : FLOAT;
    bSyncOuterLoop          : BOOL;
    eModeOuterLoop          : E_CTRL_MODE;
    bHoldOuterLoop          : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValueOuterLoop	FLOAT	Istwert der Regelgröße, die auf den PI-Regler des äußeren Regelkreises zurückgeführt wird.
fActualValueInnerLoop	FLOAT	Istwert der Regelgröße, die auf den PID-Regler des inneren Regelkreises zurückgeführt wird.
fPreControl	FLOAT	Vorsteuerung, die hinter dem PI-Regler aufgeschaltet wird.
fManSyncValueInnerLoop	FLOAT	Eingang, auf dessen Wert der interne Zustand des PID-Glieds (innerer Regelkreis) gesetzt werden kann.
bSyncInnerLoop	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PID-Glied (innerer Regelkreis) auf den Wert „fManSyncValueInnerLoop“ gesetzt.
eModeInnerLoop	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des PID-Glieds (innerer Regelkreis) festlegt.
bHoldInnerLoop	BOOL	Ein TRUE an diesem Eingang hält den internen Zustand des PID-Glieds (innerer Regelkreis) konstant auf dem aktuellen Wert.
fManSyncValueOuterLoop	FLOAT	Eingang, auf dessen Wert der interne Zustand des PI-Glieds (äußerer Regelkreis) gesetzt werden kann.
bSyncOuterLoop	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied (äußerer Regelkreis) auf den Wert „fManSyncValueOuterLoop“ gesetzt.
eModeOuterLoop	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des PI-Glieds (äußerer Regelkreis) festlegt.
bHoldOuterLoop	BOOL	Ein TRUE an diesem Eingang hält den internen Zustand des PI-Glieds (äußerer Regelkreis) konstant auf dem aktuellen Wert.

VAR_OUTPUT

```
VAR_OUTPUT
    fOut                : FLOAT;
    eStateInnerLoop     : E_CTRL_STATE;
    bARWactiveInnerLoop : BOOL;
    eErrorIdInnerLoop   : E_CTRL_ERRORCODES;
    bErrorInnerLoop     : BOOL;
    eStateOuterLoop     : E_CTRL_STATE;
    bARWactiveOuterLoop : BOOL;
    eErrorIdOuterLoop   : E_CTRL_ERRORCODES;
    bErrorOuterLoop     : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PI-PID-Glieds
eStateInnerLoop	E_CTRL_STATE	State des internen PID-Glieds (innerer Regelkreis)
bARWactiveInnerLoop	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied (innerer Regelkreis) in der Begrenzung befindet.
eErrorIdInnerLoop	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174] des PID-Glieds (innerer Regelkreis).
bErrorInnerLoop	BOOL	Wird TRUE, sobald ein Fehler in dem PID-Glied (innerer Regelkreis) eintritt.
eStateOuterLoop	E_CTRL_STATE	State des internen PI-Glieds (äußerer Regelkreis)
bARWactiveOuterLoop	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PI-Glied (äußerer Regelkreis) in der Begrenzung befindet.
eErrorIdOuterLoop	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174] des PI-Glieds (äußerer Regelkreis).
bErrorOuterLoop	BOOL	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
```

Name	Typ	Beschreibung
stParams	ST_CTRL_PI_PID_PARAMS	Parameterstruktur des PI-PID-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PI_PID_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp_OuterLoop           : FLOAT := 0;
    tTn_OuterLoop           : TIME := T#0s;
    fMaxLimit_OuterLoop     : FLOAT := 1E38;
    fMinLimit_OuterLoop     : FLOAT := -1E38;
    fKp_InnerLoop           : FLOAT := 0;
    tTn_InnerLoop           : TIME := T#0ms;
    tTv_InnerLoop           : TIME := T#0ms;
    tTd_InnerLoop           : TIME := T#0ms;
    fMaxLimit_InnerLoop     : FLOAT := 1E38;
    fMinLimit_InnerLoop     : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	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	FLOAT	Reglerverstärkung / Reglerbeiwert des PI-Glieds (äußerer Regelkreis)
tTn_OuterLoop	TIME	Nachstellzeit des PI-Glieds (äußerer Regelkreis). Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
fMaxLimit_OuterLoop	FLOAT	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	FLOAT	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	FLOAT	Reglerverstärkung / Reglerbeiwert des PID-Glieds (innerer Regelkreis)
tTn_InnerLoop	TIME	Nachstellzeit des PID-Glieds (innerer Regelkreis). Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv_InnerLoop	TIME	Vorhaltzeit des PID-Glieds (innerer Regelkreis). Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd_InnerLoop	TIME	Dämpfungszeit des PID-Glieds (innerer Regelkreis)
fMaxLimit_InnerLoop	FLOAT	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	FLOAT	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.

4.2.1.3.9 FB_CTRL_PID

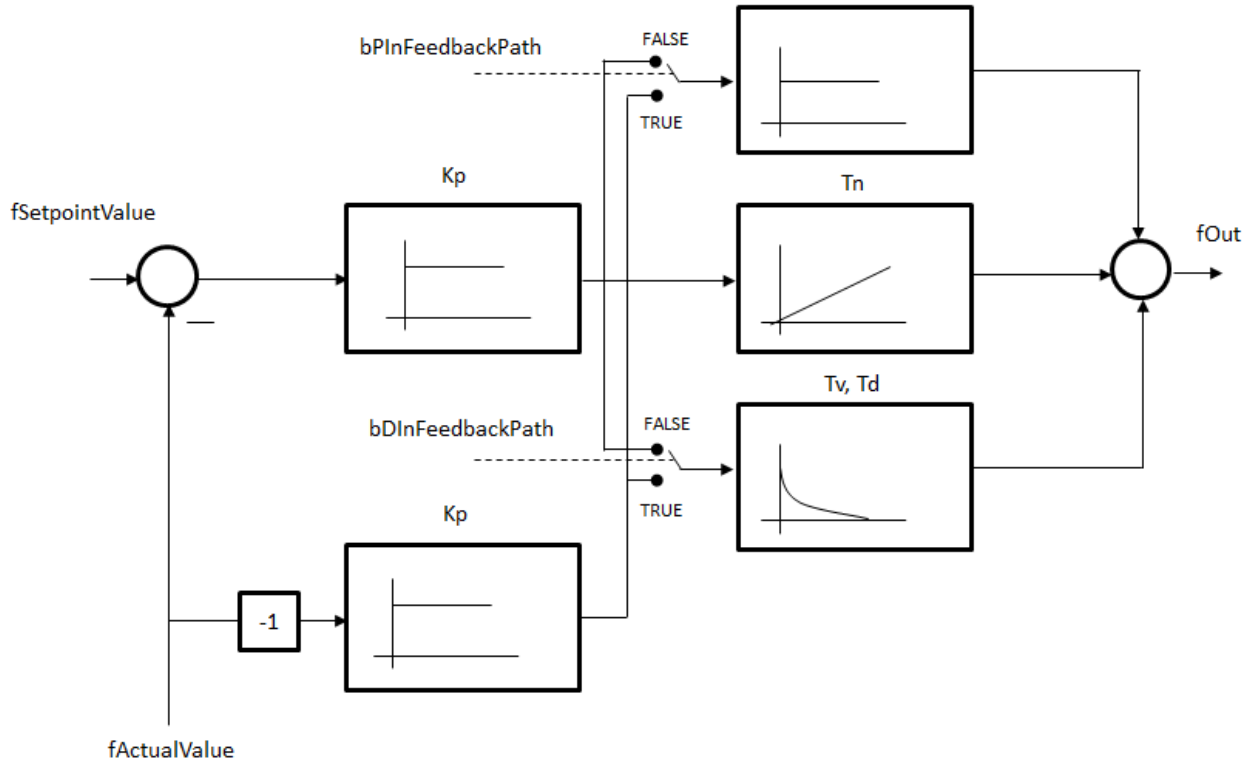
Der Funktionsbaustein stellt ein PID-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge `bPInTheFeedbackPath` und `bDInTheFeedbackPath` FALSE sind, anderenfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blocks.

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

Wirkungsplan



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

Der regelungstechnische Hintergrund ist der, dass bei Sollwertsprüngen durch den Differential-Anteil des Regelalgorithmus große Stellgrößen entstehen, die die Stellelemente belasten und das Regelsystem zum Schwingen anregen können. Ein Regelalgorithmus, dessen Differentialanteil 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	$G(s) = \frac{G_I(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$

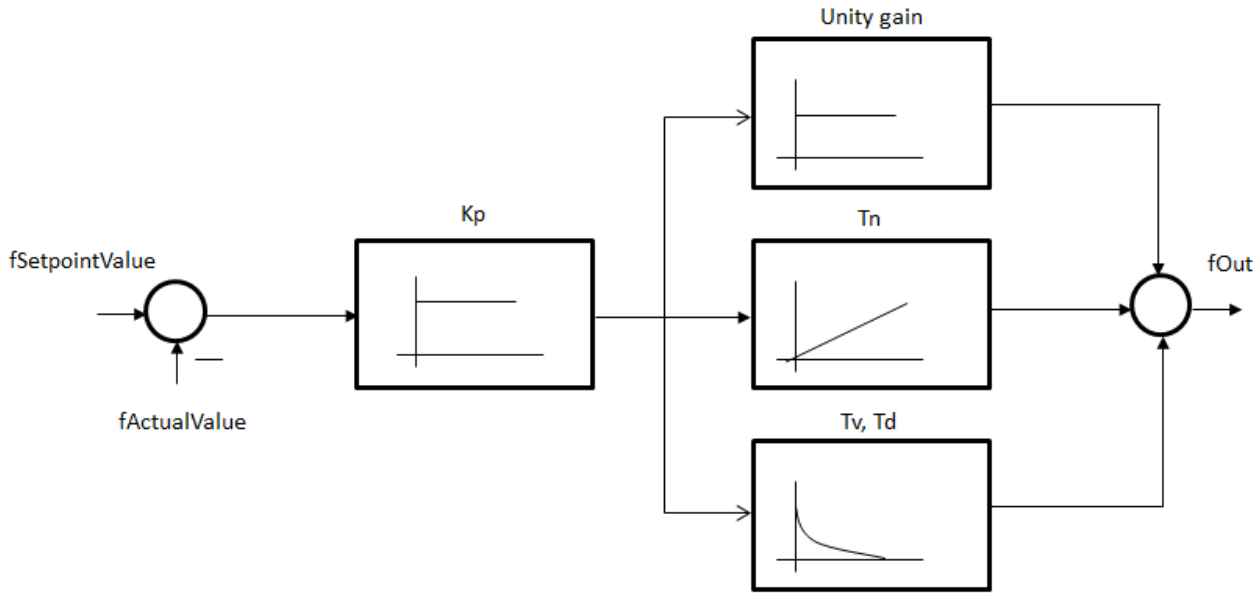
mit:

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

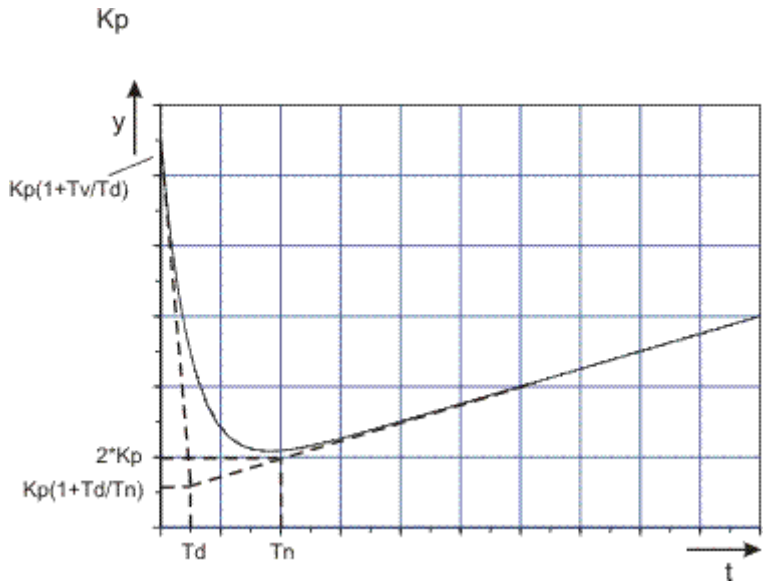
$$G_{PI}(s) = K_p \left(1 + \frac{1}{T_n s} \right)$$

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

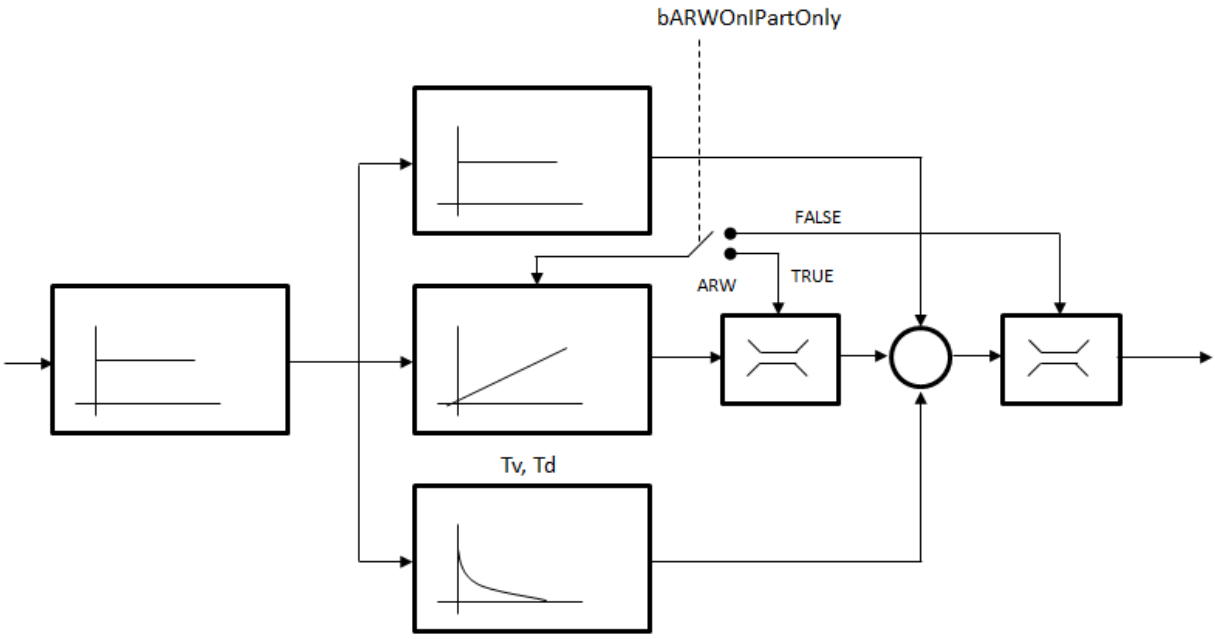
Die Standardeinstellung für die beiden Eingänge `bPInTheFeedbackPath` und `bDInTheFeedbackPath` ist FALSE. Der PID-Regler entspricht dann einem standardmäßigen PID-Regler in additiver Form.



Sprungantwort



ARW



VAR_INPUT

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

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
fManSyncValue	FLOAT	Eingang, auf dessen Wert der interne Zustand des PID-Glieds gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PID-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PID-Glieds
bARWactive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

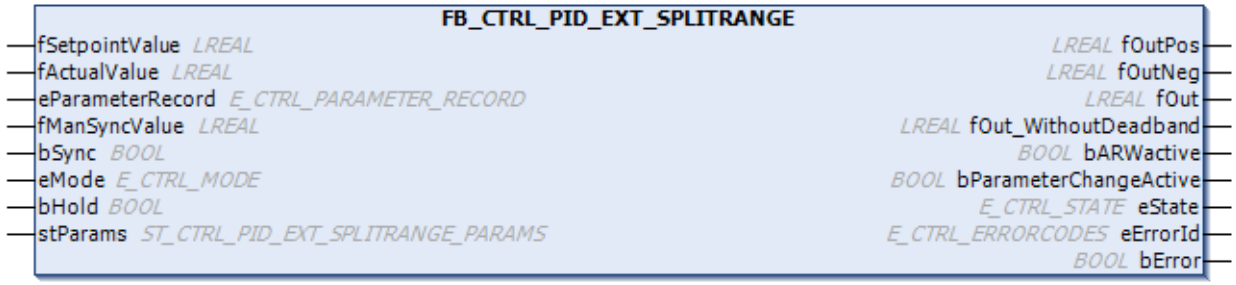
Name	Typ	Beschreibung
stParams	ST_CTRL_PID_PARAMS	Parameterstruktur des PID-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp                     : FLOAT := 0;
    tTn                     : TIME := T#0ms;
    tTv                     : TIME := T#0ms;
    tTd                     : TIME := T#0ms;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Reglerbeiwert
tTn	TIME	Nachstellzeit, wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv	TIME	Vorhaltzeit, wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd	TIME	Dämpfungszeit
fOutMaxLimit	FLOAT	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	FLOAT	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	BOOL	Eingang, mit dem die Eingangsgröße des internen P-Glieds ausgewählt werden kann (siehe Wirkungsplan). Standardeinstellung: FALSE
bDInTheFeedbackPath	BOOL	Eingang, mit dem die Eingangsgröße des internen D-Glieds ausgewählt werden kann (siehe Wirkungsplan). Standardeinstellung: FALSE
bARWOnIPartOnly	BOOL	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. (Vergl. Wirkungsplan.)

4.2.1.3.10 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-Windows sowie des Input- und Output-Deadbands 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 Parameter-Satzumschaltung erfolgt dabei so, dass sich auch während der Umschaltung eine stetige Stellgröße ergibt.

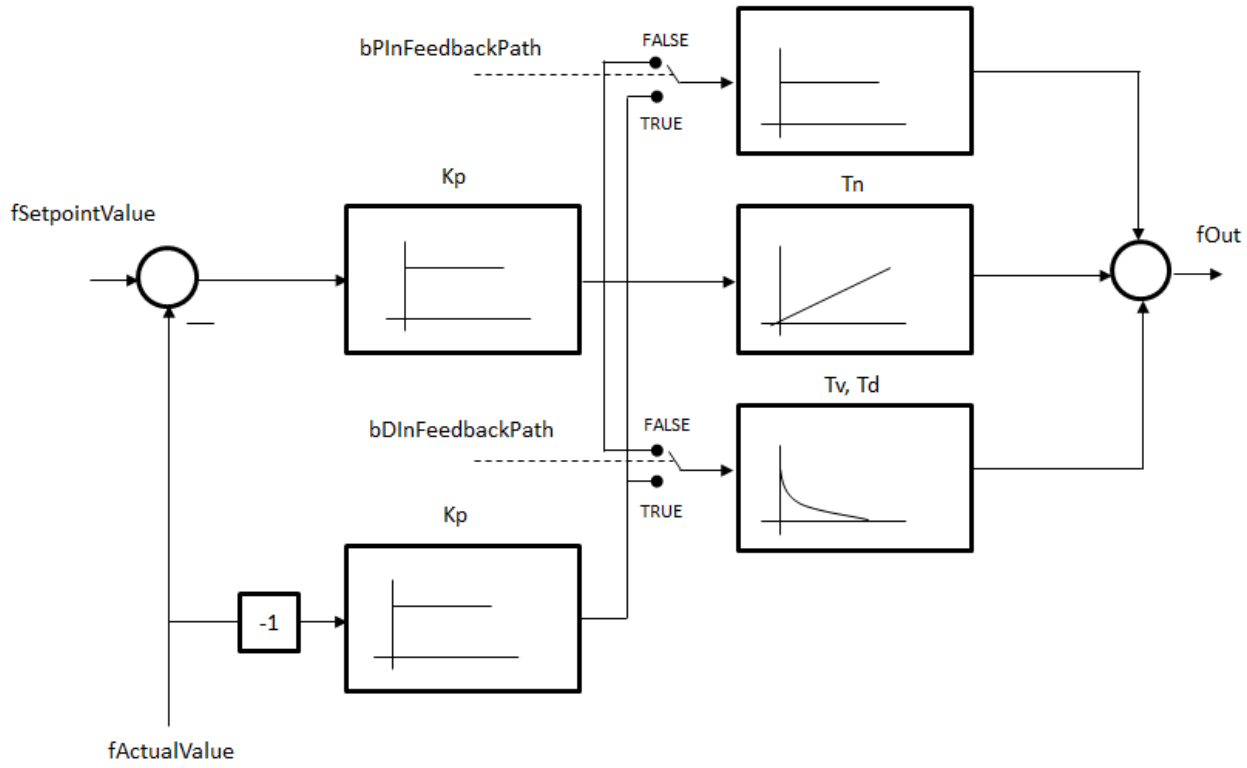
Der Umschaltalgorithmus berechnet eine lineare zeitabhängige Transition zwischen den beiden Parametersätzen. Mit dem Parameter `nParameterChangeCycleTicks` kann die Anzahl der Task-Zyklen vorgegeben werden, in denen stetig zwischen den Parametersätzen umgeschaltet wird.

Übertragungsfunktion

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge `bPInTheFeedbackPath` und `bDInTheFeedbackPath` FALSE sind, anderenfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blocks:

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

Wirkungsplan



Der Standardwirkungsplan 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	$G(s) = \frac{G_{PID}(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$
true	true	$G(s) = \frac{G_I \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$

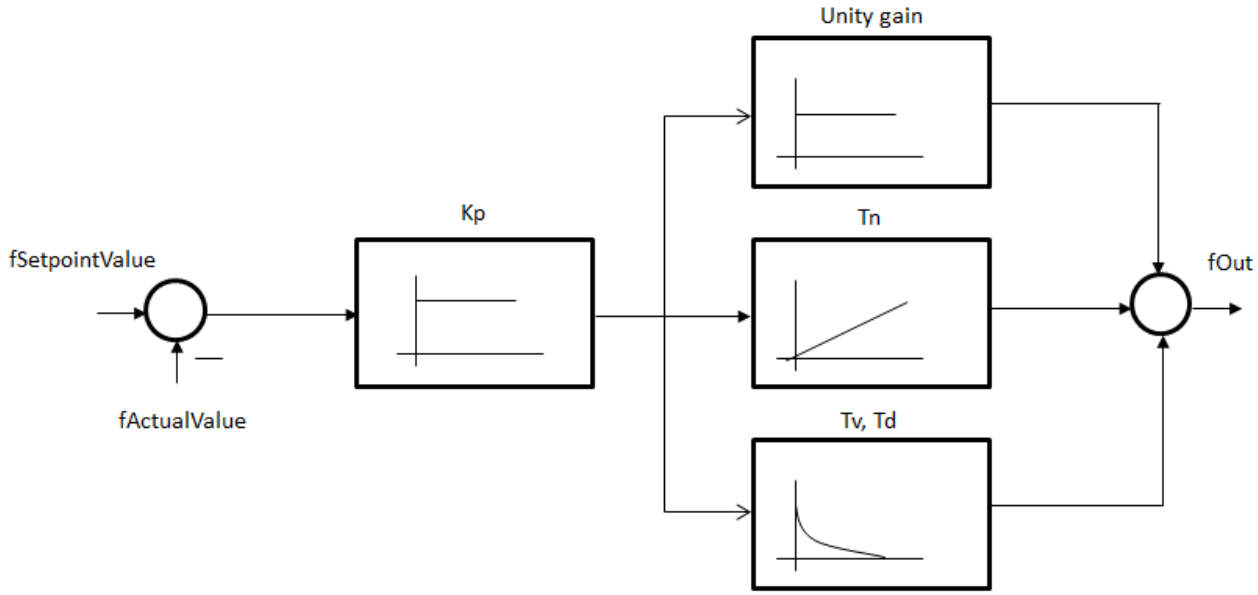
mit:

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

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

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

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.

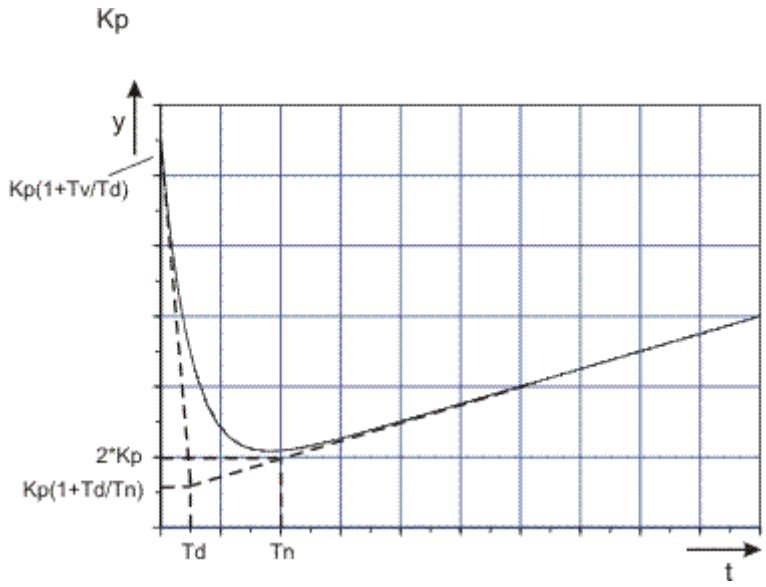
Totzone des Ausgangs

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

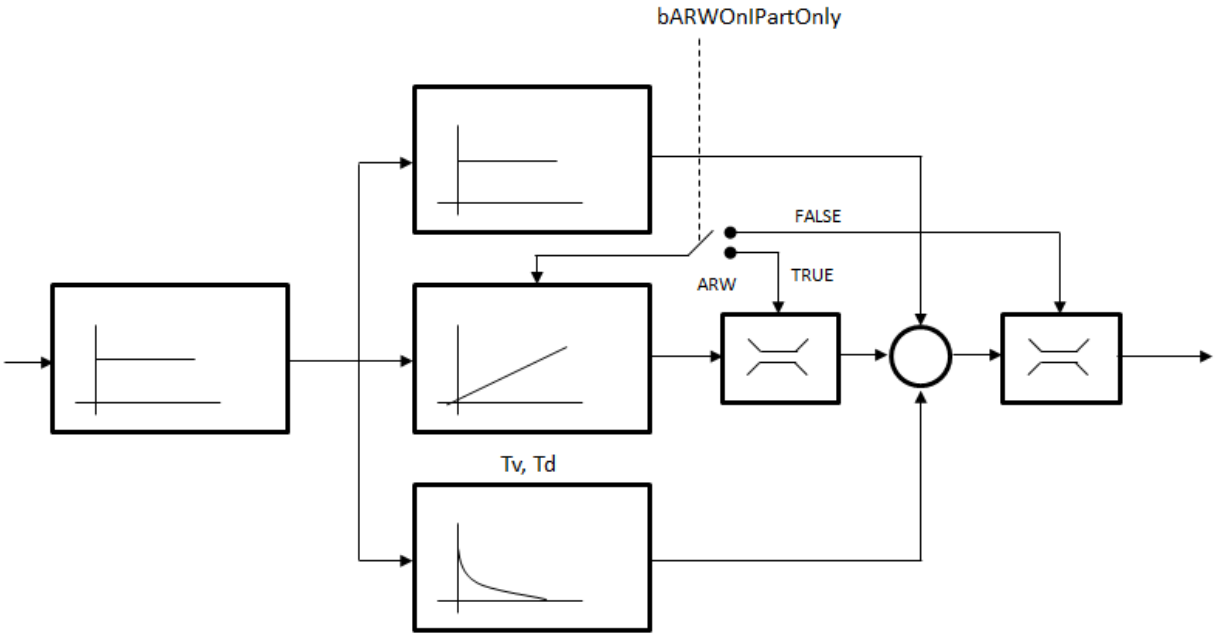
Totzone des Eingangs

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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
eParameterRecord	E_CTRL_PARAMETER_RECORD	Index des aktiven Parametersatzes
fManSyncValue	FLOAT	Eingang, mit dem das PI-Glied gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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;
    bParameterChangeActive  : BOOL;
    bError                  : BOOL;
    eErrorId                : E_CTRL_ERRORCODES;
END_VAR
```

Name	Typ	Beschreibung
fOutPos	FLOAT	Ausgang des PID-Glieds, wenn die Stellgröße positiv ist. Anderenfalls wird eine Null ausgegeben.
fOutNeg	FLOAT	Ausgang des PID-Glieds, wenn die Stellgröße negativ ist. Anderenfalls wird eine Null ausgegeben.
fOut	FLOAT	Ausgang des PID-Glieds
bARWActive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.
bParameter ChangeActive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass ein Wechsel von einem zum anderen Parametersatz erfolgt.
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
fCtrlDerivation		Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_PID_EXT_SPLITRANGE_PARAMS;
END_VAR
```

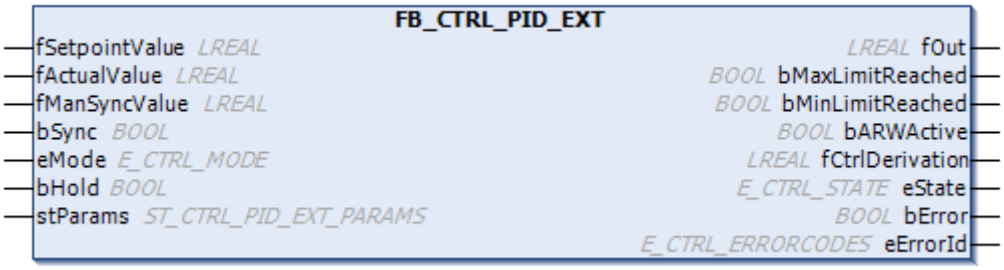
Name	Typ	Beschreibung
stParams	ST_CTRL_ PID_EXT_SPLIT RANGE_PARAMS	Parameterstruktur des PID-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_EXT_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime              : TIME := T#0ms;
    tTaskCycleTime              : TIME := T#0ms;
    fKp_heating                 : FLOAT := 0;
    tTn_heating                 : TIME := T#0ms;
    tTv_heating                 : TIME := T#0ms;
    tTd_heating                 : TIME := T#0ms;
    fKp_cooling                 : FLOAT := 0;
    tTn_cooling                 : TIME := T#0ms;
    tTv_cooling                 : TIME := T#0ms;
    tTd_cooling                 : TIME := T#0ms;
    nParameterChangeCycleTicks  : INT;
    fDeadBandInput              : REAL := 0.0;
    fDeadBandOutput             : REAL := 0.0;
    fInnerWindow                : REAL := 0.0;
    fOuterWindow                : REAL := 0.0;
    fOutMaxLimit                : FLOAT := 1E38;
    fOutMinLimit                : FLOAT := -1E38;
    bPInTheFeedbackPath         : BOOL;
    bDInTheFeedbackPath         : BOOL;
    bARWOnIPartOnly             : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	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	FLOAT	Reglerverstärkung / Reglerbeiwert
tTn_heating	TIME	Nachstellzeit: Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv_heating	TIME	Vorhaltzeit: Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd_heating	TIME	Dämpfungszeit
Bereich **eCTRL_PARAMETER_RECORD_COOLING:**		
fKp_cooling	FLOAT	Reglerverstärkung / Reglerbeiwert
tTn_cooling	TIME	Nachstellzeit: Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv_cooling	TIME	Vorhaltzeit: Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd_cooling	TIME	Dämpfungszeit
nParameter ChangeCycle Ticks	INT	Anzahl der Task-Zyklen, in denen von einem Parametersatz zum anderen gewechselt wird.
fDeadBandInput	REAL	Siehe obige Beschreibung: Totzone des Eingangs
fDeadBandOutput	REAL	Siehe obige Beschreibung: Totzone des Ausgangs
fInnerWindow	REAL	Siehe obige Beschreibung: Lineare Reduzierung des I-Anteils in dem InnerWindow
fOuterWindow	REAL	Siehe obige Beschreibung: Abschaltung des I-Anteils in dem OuterWindow
fOutMaxLimit	FLOAT	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	FLOAT	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	BOOL	Eingang, mit dem die Eingangsgröße des P-Gliedes ausgewählt werden kann (siehe Wirkungsplan).
bDInTheFeedbackPath	BOOL	Eingang, mit dem die Eingangsgröße des D-Gliedes ausgewählt werden kann (siehe Wirkungsplan).
bARWOnIPartOnly	BOOL	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. (Vergl. Wirkungsplan.)

4.2.1.3.11 FB_CTRL_PID_EXT

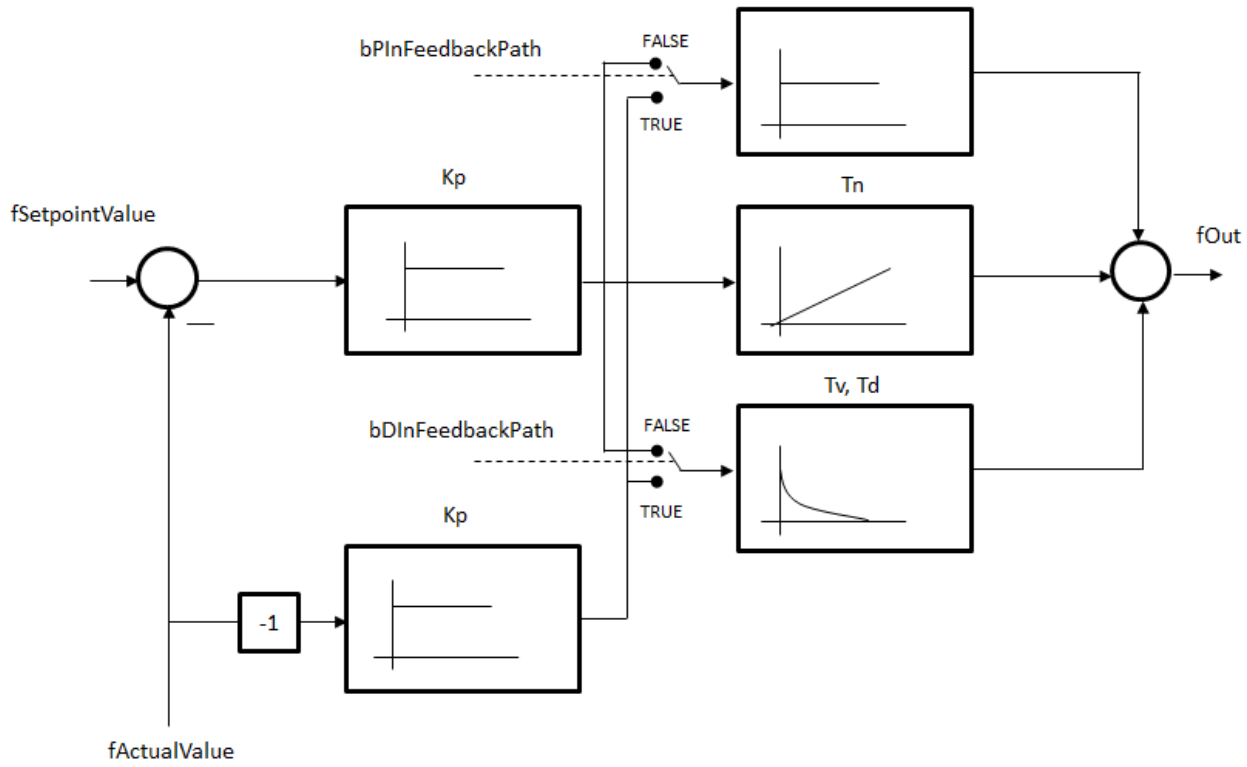


Übertragungsfunktion

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge `bPInTheFeedbackPath` und `bDInTheFeedbackPath` FALSE sind, anderenfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blocks:

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

Wirkungsplan



Der Standardwirkungsplan 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 Stellgrößen 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	$G(s) = \frac{G_{PID}(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$
true	true	$G(s) = \frac{G_I(s) \bullet G_S(s)}{1 + G_{PID}(s) \bullet G_S(s)}$

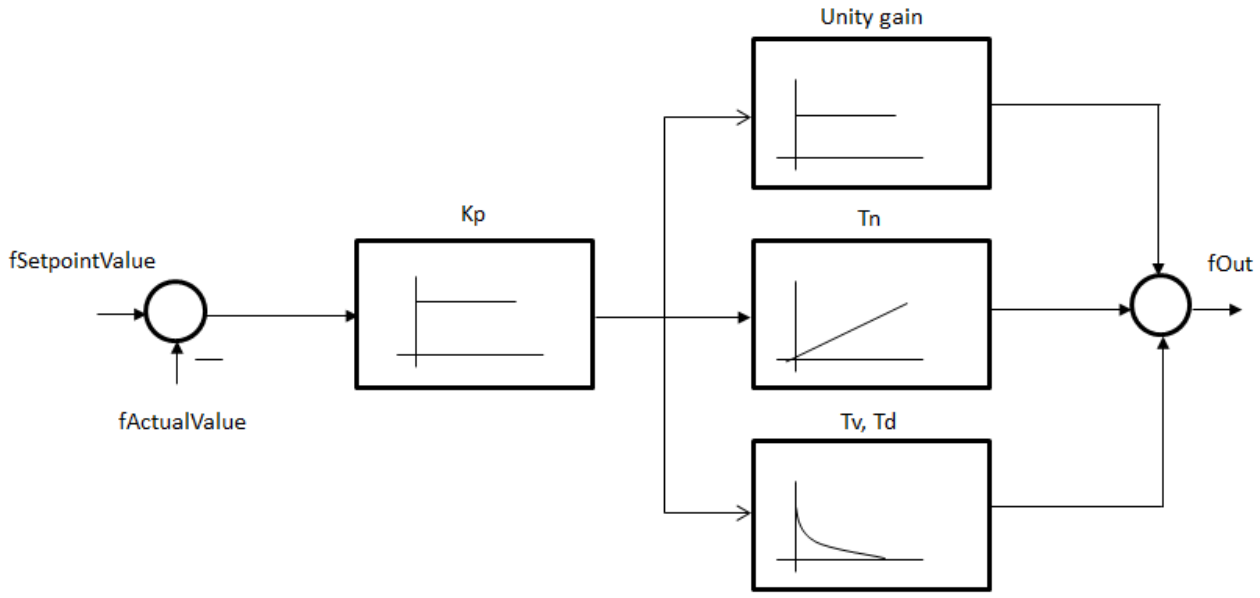
mit:

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

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

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

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



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.

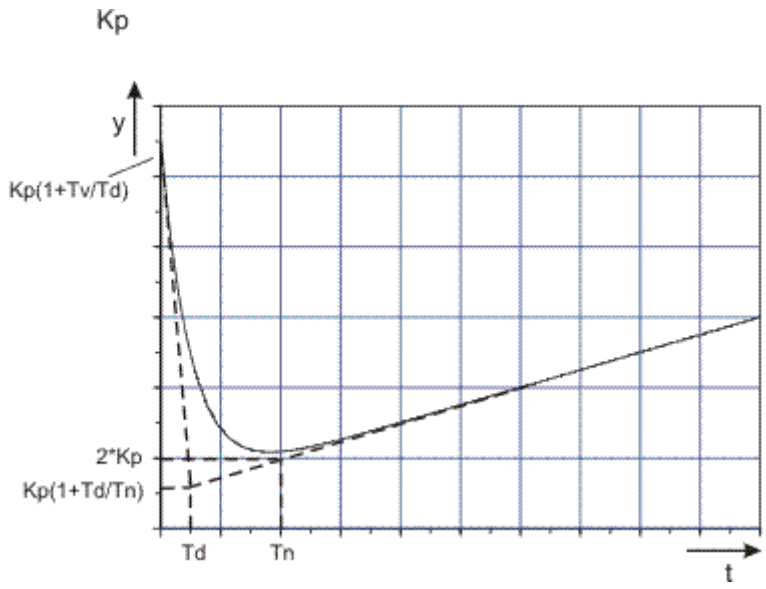
Totzone des Ausgangs

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

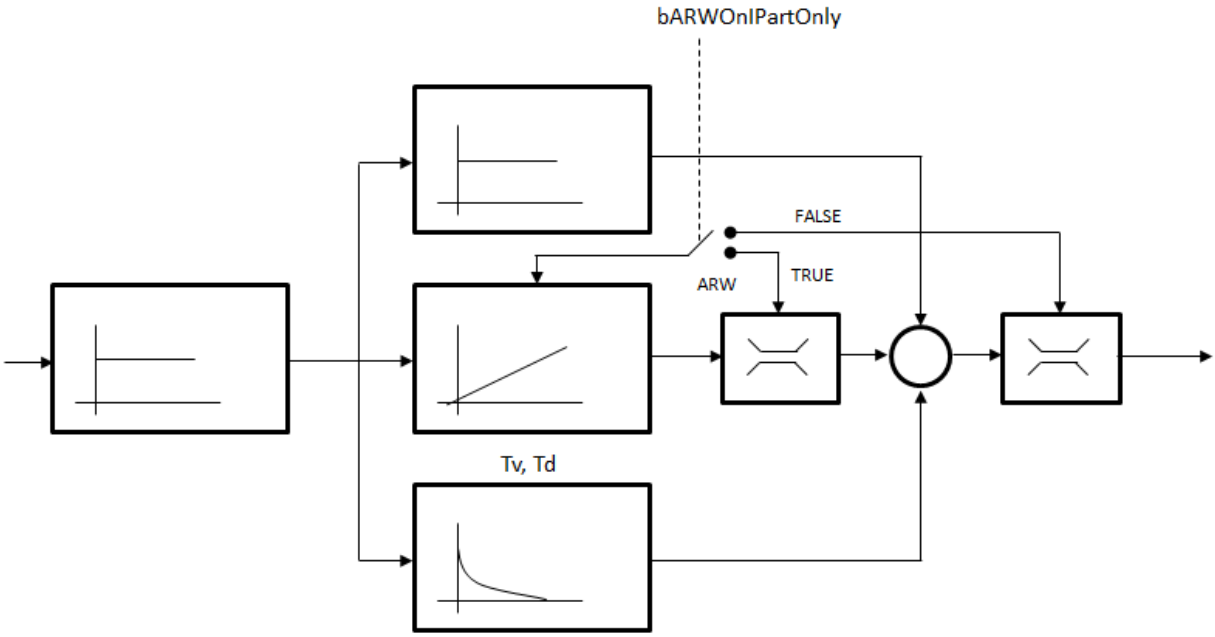
Totzone des Eingangs

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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
fManSyncValue	FLOAT	Eingang, mit dem das PID-Glied gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PID-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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;
    bMinLimitReached    : BOOL := FALSE;
    bARWActive          : BOOL := FALSE;
    fCtrlDerivation     : FLOAT;
    eState              : E_CTRL_STATE;
    bError              : BOOL;
    eErrorId            : E_CTRL_ERRORCODES;
END_VAR
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PID-Glieds
bMaxLimit Reached	BOOL	Der Ausgang ist TRUE, wenn sich der Baustein in der oberen Begrenzung befindet.
bMinLimit Reached	BOOL	Der Ausgang ist TRUE, wenn sich der Baustein in der unteren Begrenzung befindet.
bARWActive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.
fCtrlDerivation	FLOAT	Der aktuelle Wert der Regelabweichung
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
eErrorId	E_CTRL_ ERRORCODES	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

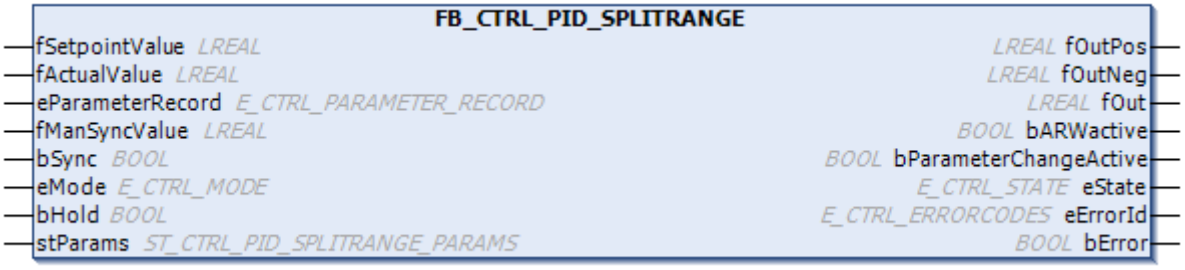
Name	Typ	Beschreibung
stParams	ST_CTRL_PID_ EXT_PARAMS	Parameterstruktur des PID-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_EXT_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp                     : FLOAT := 0;
    tTn                     : TIME := T#0ms;
    tTv                     : TIME := T#0ms;
    tTd                     : TIME := T#0ms;
    fDeadBandInput          : REAL := 0.0;
    fDeadBandOutput         : REAL := 0.0;
    fInnerWindow            : REAL := 0.0;
    fOuterWindow            : REAL := 0.0;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Reglerbeiwert
tTn	TIME	Nachstellzeit, wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv	TIME	Vorhaltzeit, wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd	TIME	Dämpfungszeit
fDeadBandInput	REAL	Siehe obere Beschreibung.
fDeadBand Output	REAL	Siehe obere Beschreibung.
fInnerWindow	REAL	Siehe obere Beschreibung.
fOuterWindow	REAL	Siehe obere Beschreibung.
fOutMaxLimit	FLOAT	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	FLOAT	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.
bPInTheFeed backPath	BOOL	Eingang, mit dem die Eingangsgröße des P-Glieds ausgewählt werden kann (siehe Wirkungsplan).
bDInTheFeed backPath	BOOL	Eingang, mit dem die Eingangsgröße des D-Glieds ausgewählt werden kann (siehe Wirkungsplan).
bARWOnIPart Only	BOOL	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. (Vergl. Wirkungsplan.)

4.2.1.3.12 FB_CTRL_PID_SPLITRANGE



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

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.

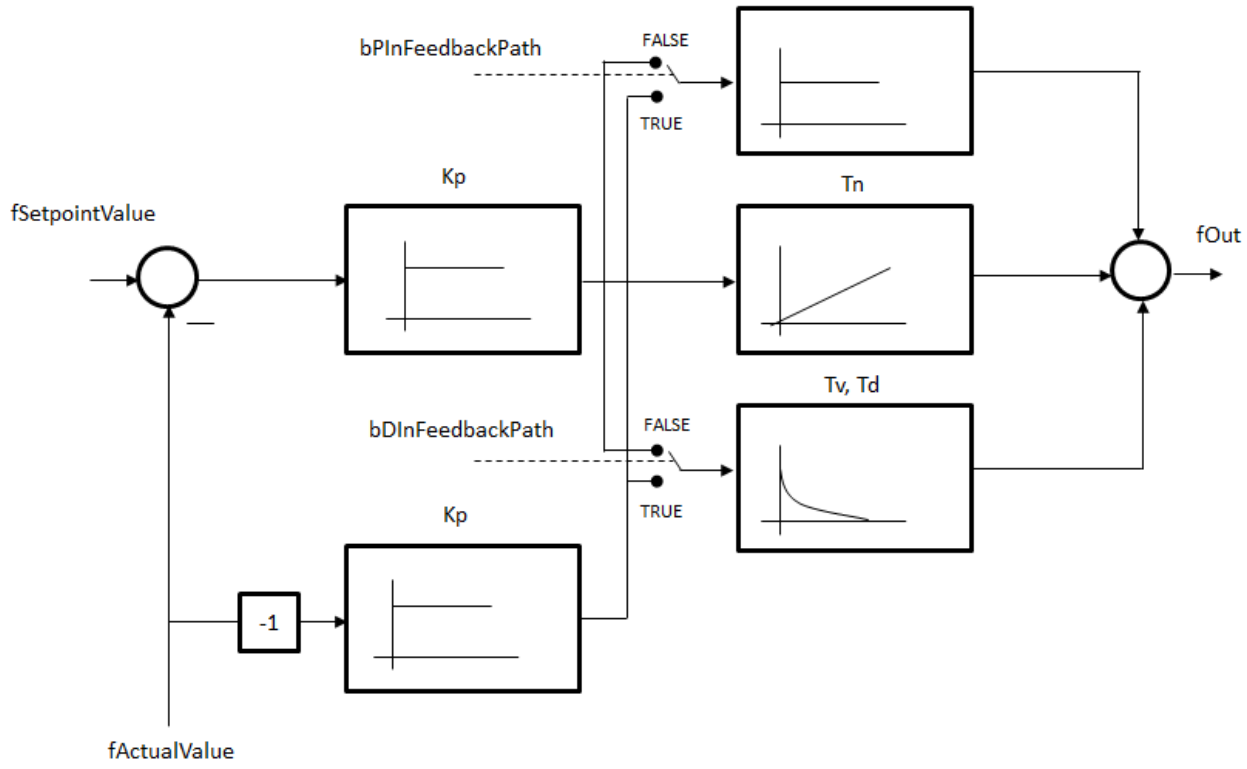
Der Umschaltalgorithmus berechnet eine lineare zeitabhängige Transition zwischen den beiden Parametersätzen. Mit dem Parameter `nParameterChangeCycleTicks` kann die Anzahl der Task-Zyklen vorgegeben werden, in denen stetig zwischen den Parametersätzen umgeschaltet wird.

Übertragungsfunktion

Die folgende Übertragungsfunktion lässt sich angeben, wenn die boolschen Eingänge `bPInTheFeedbackPath` und `bDInTheFeedbackPath` FALSE sind, anderenfalls beschreibt die Übertragungsfunktion nur einen Teil des Übertragungsverhaltens des Blocks:

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

Wirkungsplan



Der Standardwirkungsplan 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	$G(s) = \frac{G_I(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$

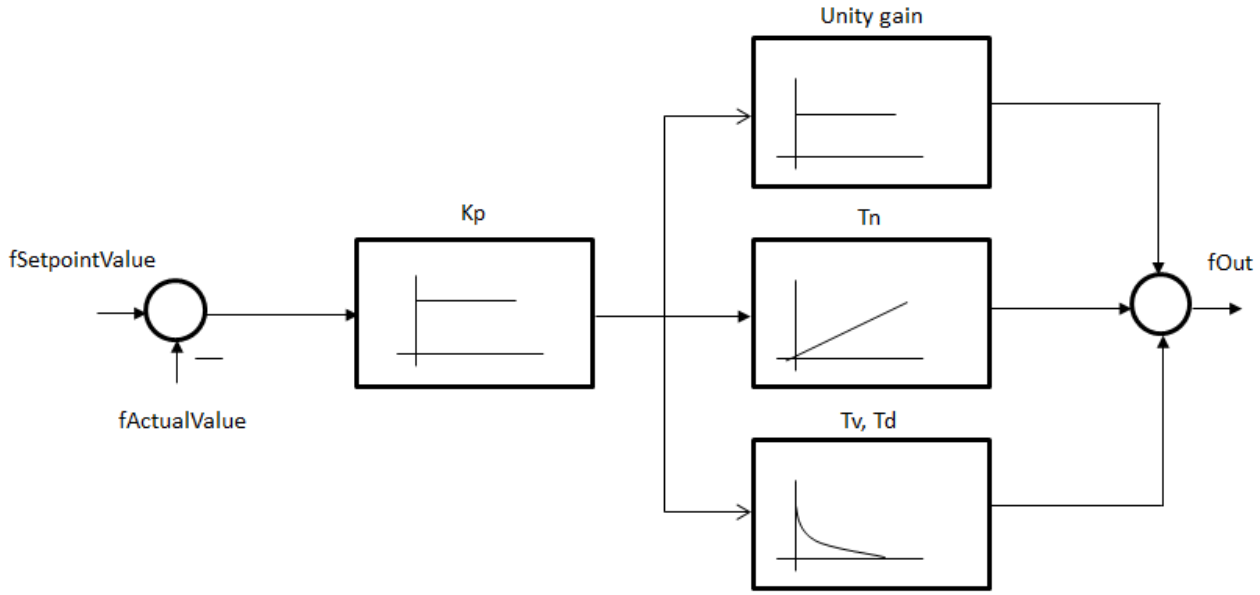
mit:

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

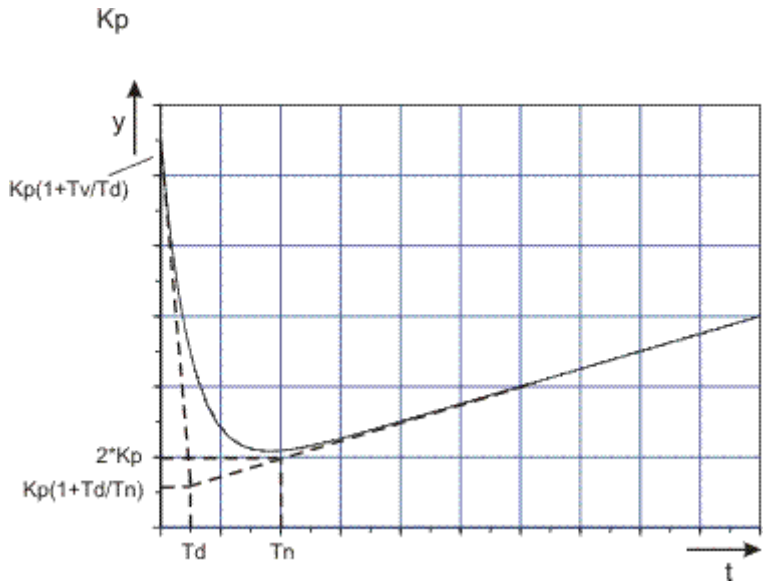
$$G_{PI}(s) = K_p \left(1 + \frac{1}{T_n s} \right)$$

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

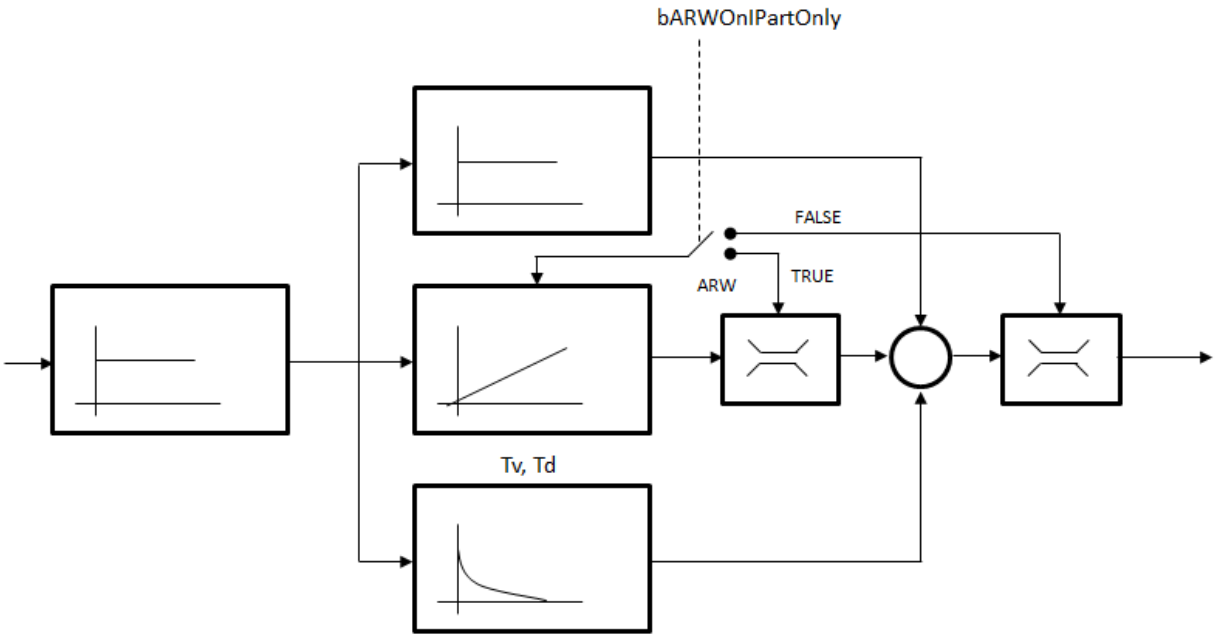
Die Standardeinstellung für die beiden Eingänge `bPInTheFeedbackPath` und `bDInTheFeedbackPath` ist **FALSE**. Der PID-Regler entspricht dann einem Standard-PID-Regler in additiver Form.



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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
eParameter Record	E_CTRL_ PARAMETER_ RECORD	Index des aktiven Parametersatzes
fManSyncValue	FLOAT	Eingang, mit dem das PI-Glied gesetzt werden kann.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PI-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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;
    bParameterChangeActive  : BOOL;
    eState                  : E_CTRL_STATE;
    bError                  : BOOL;
    eErrorId                : E_CTRL_ERRORCODES;
END_VAR
```

Name	Typ	Beschreibung
fOutPos	FLOAT	Ausgang des PID-Glieds, wenn die Stellgröße positiv ist. Anderenfalls wird eine Null ausgegeben.
fOutNeg	FLOAT	Ausgang des PID-Glieds, wenn die Stellgröße negativ ist. Anderenfalls wird eine Null ausgegeben.
fOut	FLOAT	Ausgang des PID-Glieds
bARWActive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich das PID-Glied in der Begrenzung befindet.
bParameter ChangeActive	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass ein Wechsel von einem zum anderen Parametersatz erfolgt.
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_PID_ SPLITRANGE_ PARAMS	Parameterstruktur des PID-Glieds

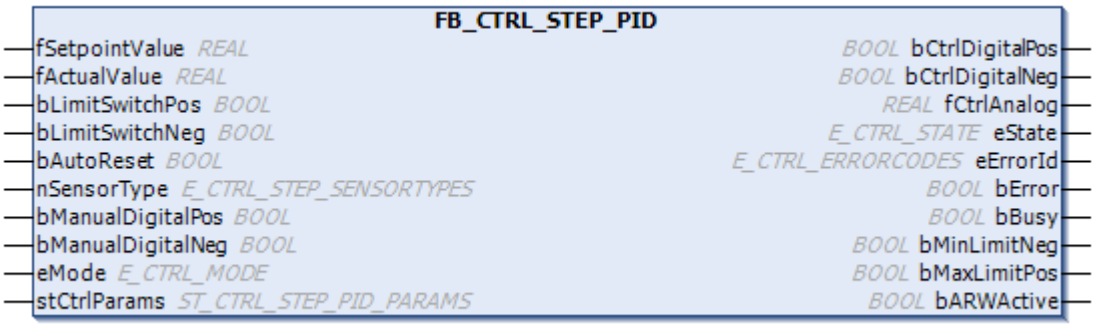
`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PID_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime              : TIME := T#0ms;
    tTaskCycleTime              : TIME := T#0ms;
    fKp_heating                 : FLOAT := 0;
    tTn_heating                 : TIME := T#0ms;
    tTv_heating                 : TIME := T#0ms;
```

```
    tTd_heating                     : TIME := T#0ms;
    fKp_cooling                     : FLOAT := 0;
    tTn_cooling                     : TIME := T#0ms;
    tTv_cooling                     : TIME := T#0ms;
    tTd_cooling                     : TIME := T#0ms;
    nParameterChangeCycleTicks      : INT;
    fOutMaxLimit                    : FLOAT := 1E38;
    fOutMinLimit                    : FLOAT := -1E38;
    bPInTheFeedbackPath             : BOOL;
    bDInTheFeedbackPath             : BOOL;
    bARWOnIPartOnly                 : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	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	FLOAT	Reglerverstärkung / Reglerbeiwert
tTn_heating	TIME	Nachstellzeit: Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv_heating	TIME	Vorhaltzeit: Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd_heating	TIME	Dämpfungszeit
Bereich **eCTRL_PARAMETER_RECORD_COOLING**:		
fKp_cooling	FLOAT	Reglerverstärkung / Reglerbeiwert
tTn_cooling	TIME	Nachstellzeit: Wenn diese zu T#0s parametriert ist, wird der I-Anteil deaktiviert.
tTv_cooling	TIME	Vorhaltzeit: Wenn diese zu T#0s parametriert ist, wird der D-Anteil deaktiviert.
tTd_cooling	TIME	Dämpfungszeit
nParameter ChangeCycle Ticks	INT	Anzahl der Task-Zyklen, in denen von einem zum anderen Parametersatz gewechselt wird.
fOutMaxLimit	FLOAT	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	FLOAT	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.
bPInTheFeed backPath	BOOL	Eingang, mit dem die Eingangsgröße des P-Gliedes ausgewählt werden kann (siehe Wirkungsplan).
bDInTheFeed backPath	BOOL	Eingang, mit dem die Eingangsgröße des D-Gliedes ausgewählt werden kann (siehe Wirkungsplan).
bARWOnIPartOn ly	BOOL	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. (Vergl. Wirkungsplan.)

4.2.1.3.13 FB_CTRL_STEP_PID



Der Funktionsbaustein regelt eine digital manipulierte Größe zur Integration von Aktoren. Er basiert auf einem Standard-PID-Regler mit einem zusätzlichen binären Ausgangssignal zusammen mit dem analogen Ausgang und funktioniert ohne ein präzises Positionsrückmeldesignal.

Das zur Berechnung des Reglerausgangs erforderliche Positionsrückmeldesignal wird mithilfe der begrenzenden Werte und der zum Erreichen dieser Grenzen erforderlichen Zeit geschätzt. Der Funktionsbaustein generiert dann Impulse, die zum Antrieb des Aktors als Servomotor erforderlich sind.

VAR_INPUT

```
VAR_INPUT
fSetpointValue         : LREAL;
fActualValue           : LREAL;
bLimitSwitchPos        : BOOL;
bLimitSwitchNeg        : BOOL;
bAutoReset             : BOOL;
nSensorType            : E_CTRL_STEP_SENSORTYPES;
fDisturbanceValue      : REAL;
bManualDigitalPos      : BOOL;
bManualDigitalNeg      : BOOL;
eMode                  : E_CTRL_MODE;
END_VAR
```

Name	Typ	Beschreibung
fSetpointValue	LREAL	Sollwert der Regelgröße
fActualValue	LREAL	Ist-Wert der Regelgröße
bLimitSwitchPos	BOOL	Endschalter, wird TRUE, wenn der obere Anschlag erreicht wurde.
bLimitSwitchNeg	BOOL	Endschalter, wird TRUE, wenn der untere Anschlag erreicht wurde.
bAutoReset	BOOL	Wenn die Variable TRUE ist, stellt sie den Regler auf Anfangsbedingungen zurück.
nSensorType	E_CTRL_STEP_SENSORTYPES	Ermöglicht die Auswahl des korrekten Thermoelementtyps.
fDisturbance Value	REAL	Ist-Wert der Störgröße
bManualDigital Pos	BOOL	Wenn die Variable TRUE ist, stellt den Motor manuell in die positive Richtung
bManualDigital Neg	BOOL	wenn TRUE, stellt den Motor manuell in die negative Richtung
eMode	E_CTRL_MODE	Eingabe, die die Betriebsart des Reglers spezifiziert

VAR_OUTPUT

```
VAR_OUTPUT
bCtrlDigitalPos    : BOOL;
bCtrlDigitalNeg    : BOOL;
fCtrlAnalog        : LREAL;
eState             : E_CTRL_STATE;
eErrorId           : E_CTRL_ERRORCODES;
```

```
bError            : BOOL;
bBusy             : BOOL;
bMinLimitNeg      : BOOL;
bMinLimitPos      : BOOL;
bARWActive        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bCtrlDigitalPos	BOOL	Ausgang erforderlich, um den Motor in eine positive Richtung laufen zu lassen.
bCtrlDigitalNeg	BOOL	Ausgang erforderlich, um den Motor in eine negative Richtung laufen zu lassen.
fCtrlAnalog	LREAL	Analogausgang des Reglers
eState	E_CTRL_STATE	Ist-Zustand des Reglers
eErrorId	E_CTRL_ERROR CODES	Liefert die Fehlernummer, wenn `bError` TRUE ist.
bError	BOOL	TRUE, wenn ein Fehler im Funktionsbaustein auftritt.
bBusy	BOOL	TRUE, wenn der Funktionsbaustein aktiv ist.
bMinLimitNeg	BOOL	Ein TRUE an diesem Ausgang zeigt an, dass der Regler seinen Mindestgrenzwert erreicht hat.
bMinLimitPos	BOOL	Ein TRUE an diesem Ausgang zeigt an, dass der Regler seinen Höchstgrenzwert erreicht hat.
bARWActive	BOOL	Ein TRUE an diesem Ausgang zeigt an, dass der Regler derzeit eingeschränkt ist.

VAR_IN_OUT

```
VAR_IN_OUT
stCtrlParams : ST_CTRL_STEP_PID_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stCtrlParams	ST_CTRL_STEP_ PID_PARAMS	Parameterstruktur des Reglers.

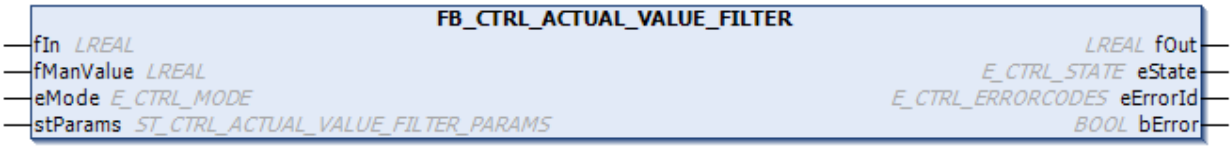
`stCtrlParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_STEP_PID_PARAMS:
STRUCT
tCtrlCycleTime     : TIME;
tTaskCycleTime     : TIME;
fKp                : REAL;
fTn                : REAL;
fTv                : REAL;
fTd                : REAL;
fTM                : REAL;
fDeadBandWidth     : REAL;
tMinimumPulseTime  : TIME;
tFilterTime        : TIME;
fWmax              : REAL;
fWmin              : REAL;
fYMax              : REAL;
fYMin              : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis verarbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein verwendet diese Eingabegröße zur internen Berechnung, ob der Zustand und die Ausgangsgrößen im aktuellen Zyklus aktualisiert wurden.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Wenn der Baustein in jedem Zyklus aufgerufen wird, entspricht diese der Taskzykluszeit der aufrufenden Task.
fKp	REAL	Reglerverstärkung oder Proportionalverstärkung
fTn	REAL	Nachstellzeit: Der I-Anteil ist deaktiviert, wenn dieser Parameter null ist.
fTv	REAL	Vorhaltzeit, der D-Anteil ist deaktiviert, wenn dieser Parameter null ist.
fTd	REAL	Dämpfungszeit
fTM	REAL	Digitale Ventilbetätigungszeit in Sekunden
fDeadBandWidth	REAL	Zulässige Breite der Reglerabweichung
tMinimumPulse Time	TIME	Mindestimpulszeit für Impulsgeber
tFilterTime	TIME	Filterzeit für Ist-Wert-Filter
fWmax	REAL	Maximalwert des Sollwerts in °C
fWmin	REAL	Minimalwert des Sollwerts in °C
fYMax	REAL	Maximalwert des Reglerausgangs
fYMin	REAL	Minimalwert des Reglerausgangs

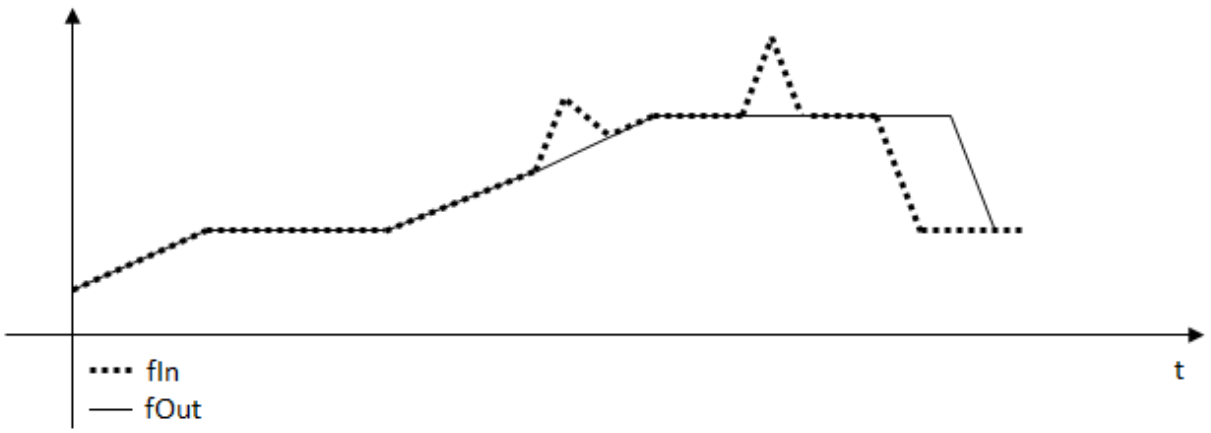
4.2.1.4 Filter / ControlledSystemSimulation

4.2.1.4.1 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Filters
fManValue	FLOAT	Eingangsgröße, dessen Wert im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Funktionsbausteins
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

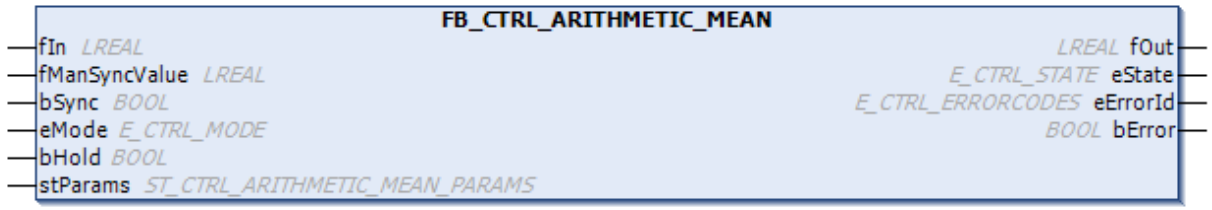
Name	Typ	Beschreibung
stParams	ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS	Parameterstruktur des Actual-Value-Filter-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fDeltaMax         : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fDeltaMax	FLOAT	Maximale Differenz zweier aufeinander folgender Eingangswerte. Siehe Beschreibung oben.

4.2.1.4.2 FB_CTRL_ARITHMETIC_MEAN



$$\overline{\chi} = \frac{1}{n}\sum_{n}\chi_n$$

VAR_INPUT

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

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Mittelwert-Filters
fManSync Value	FLOAT	Eingangsgröße, auf die der Mittelwert-Filter gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird der Mittelwert-Filter auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Mittelwert-Filters
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_ ARITHMETIC_ MEAN_PARAMS	Parameterstruktur des Mittelwertfilters

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_ARITHMETIC_MEAN_PARAMS:
STRUCT
    tCtrlCycleTime : TIME := T#0ms;
    tTaskCycleTime : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

4.2.1.4.3 FB_CTRL_MEDIAN_FILTER

Der Funktionsbaustein gibt den Median an. Der Median ist der mittlere Wert einer nach Größe geordneten Auflistung von Werten. Das bedeutet, dass die eine Hälfte der gesammelte Datenwerte kleiner als der Medianwert ist, die andere Hälfte größer.

Von dem Programmierer muss ein Array „ARRAY [1..2(n+2)] OF LREAL“ angelegt werden, in dem der Funktionsbaustein die internen benötigten Daten ablegen kann.

VAR_INPUT

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

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Median-Filters
fManSyncValue	FLOAT	Eingangsgröße, auf die der Median-Filter gesetzt werden kann oder die im Manual-Mode am Ausgang ausgegeben wird.
eMode	E_CTRL_MODE	Mit einer steigenden Flanke an diesem Eingang wird der Median-Filter auf den Wert „fManSyncValue“ gesetzt.
bSync	FLOAT	Eingang, der die Betriebsart des Bausteins festlegt.
bHold	BOOL	Ein TRUE an diesem Eingang hält den internen Zustand und somit auch den Ausgang.

VAR_OUTPUT

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

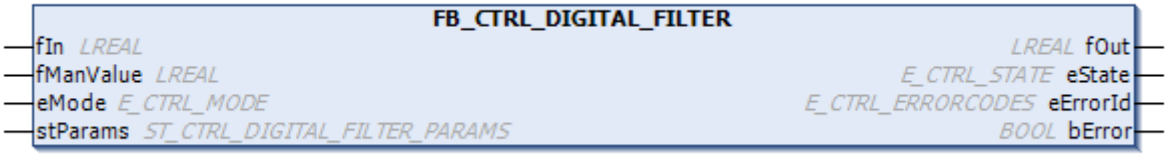
Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Median-Filters
eState	E_CTRL_STATE	Status des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer.
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    tCtrlCycleTime        : TIME := T#0ms;
    tTaskCycleTime        : TIME := T#0ms;
    nSamplesToFilter      : UINT;
    pWorkArray_ADR        : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF     : UINT := 0;
END_VAR
```

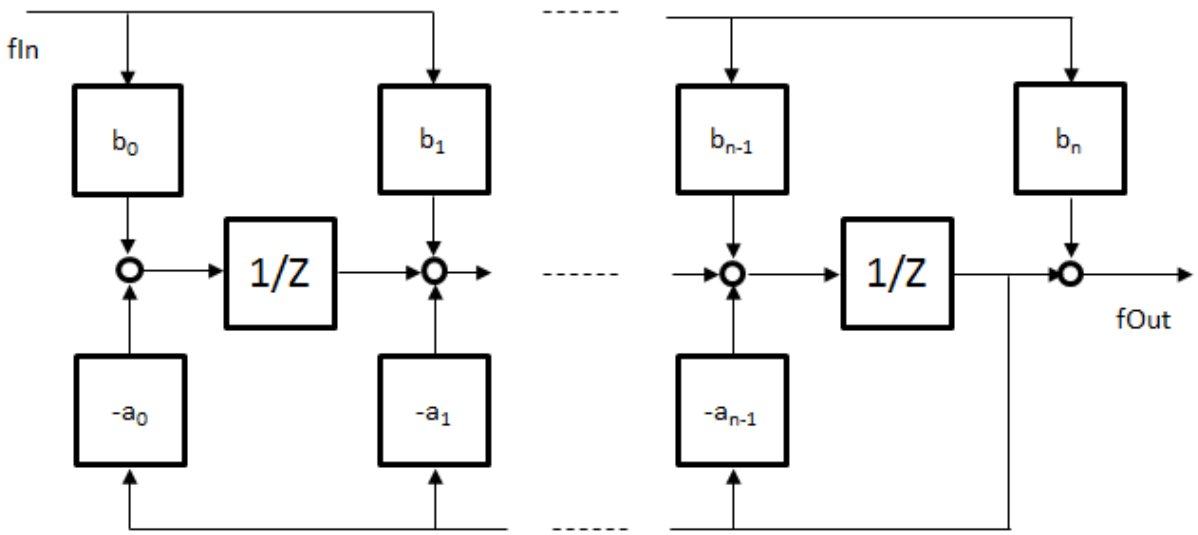
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
nSamplesTo Filter	UINT	Anzahl der Werte „n“, über die der Medianwert gebildet wird.
pWorkArray_ ADR	POINTER TO FLOAT	Adresse des Arrays, in dem die Eingangswerte zwischengespeichert werden.
nWorkArray_ SIZEOF	UINT	Größe des `WorkArrays`

4.2.1.4.4 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:



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

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

```
aCoefficientsArray_a        : ARRAY[1..nFilterOrder+1] OF FLOAT;
aCoefficientsArray_b        : ARRAY[1..nFilterOrder+1] OF FLOAT;
aStDigitalFilterData        : ARRAY[1..nFilterOrder] OF
ST_CTRL_DIGITAL_FILTER_DATA;
```

In dem Array `aCoefficientsArray_b` werden die Koeffizienten „b_1“ bis „b_n“ abgelegt. Diese müssen folgendermaßen angeordnet werden:

```
aCoefficientsArray_b[1] := b1;
aCoefficientsArray_b[2] := b2;
…
aCoefficientsArray_b[n-1] := bn-1;
aCoefficientsArray_b[n]   := bn;
```

In dem Array `aCoefficientsArray_a` werden die Koeffizienten „a1“ bis „an“ abgelegt.
Diese müssen folgendermaßen angeordnet werden:

```
aCoefficientsArray_a[1] := xxx; (* not being evaluated *)
aCoefficientsArray_a[2] := a2;
...
aCoefficientsArray_a[n-1] := an-1;
aCoefficientsArray_a[n]   := an;
```

Die internen Daten, die der Baustein benötigt, werden in dem Array `aStDigitalFilterData` 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingang des Digitalfilters
fManValue	FLOAT	Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.
eMode	E_CTRL_MODE	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Digitalfilters
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_DIGITAL_FILTER_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_DIGITAL_FILTER_PARAMS :
STRUCT
    tTaskCycleTime                : TIME;
    tCtrlCycleTime                : TIME := T#0ms;
    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
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tCtrlCycleTime	TIME	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	USINT	Ordnung des Digitalfilters [0...]
pCoefficientsArray_a_ADR	POINTER TO FLOAT	Adresse des Arrays mit den Koeffizienten a
nCoefficientsArray_a_SIZEOF	UINT	Größe des Arrays mit den Koeffizienten a in Byte
pCoefficientsArray_b_ADR	POINTER TO FLOAT	Adresse des Arrays mit den Koeffizienten b
nCoefficientsArray_b_SIZEOF	UINT	Größe des Arrays mit den Koeffizienten b in Byte
pDigitalFilterData_ADR	POINTER TO ST_CTRL_DIGITAL_FILTER_DATA	Adresse des Daten-Arrays
nDigitalFilterData_SIZEOF	UINT	Größe des Daten-Arrays in Byte

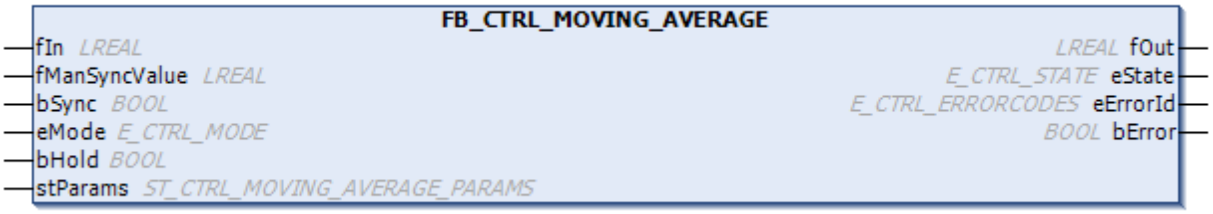
```
TYPE
ST_CTRL_DIGITAL_FILTER_DATA
STRUCT
 Internal structure. This must not be written to.
END_STRUCT
END_TYPE
```

Beispiel:

```
PROGRAM PRG_DIGITAL_FILTER_TEST
VAR
    fbDigitalFilter        : FB_CTRL_DIGITAL_FILTER;
    aCoefficientsArray_a    : ARRAY[1..3] OF FLOAT;
    aCoefficientsArray_b    : ARRAY[1..3] OF FLOAT;
    aStDigitalFilterData    : 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
    aCoefficientsArray_a[1] := 0.0; (* not used *)
    aCoefficientsArray_a[2] := 0.2;
    aCoefficientsArray_a[3] := 0.1;
    aCoefficientsArray_b[1] := 0.6;
    aCoefficientsArray_b[2] := 0.4;
    aCoefficientsArray_b[3] := 0.2;

    stParams.tTaskCycleTime := T#2ms;
    stParams.tCtrlCycleTime := T#2ms;
    stParams.nFilterOrder := 2;

    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF

stParams.pCoefficientsArray_a_ADR := ADR(aCoefficientsArray_a);
stParams.nCoefficientsArray_a_SIZEOF := SIZEOF(aCoefficientsArray_a);
stParams.pCoefficientsArray_b_ADR := ADR(aCoefficientsArray_b);
stParams.nCoefficientsArray_b_SIZEOF := SIZEOF(aCoefficientsArray_b);
stParams.pDigitalFilterData_ADR := ADR(astDigitalFilterData);
stParams.nDigitalFilterData_SIZEOF := SIZEOF(aStDigitalFilterData);
```

```
fbDigitalFilter ( fIn := fIn,
                  eMode := eMode,
                  stParams := stParams,
                  fOut => fOut,
                  eErrorId => eErrorId,
                  bError => bError);
```

4.2.1.4.5 FB_CTRL_MOVING_AVERAGE



Der Funktionsbaustein stellt einen gleitenden Mittelwertfilter im Wirkungsplan dar.

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

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

Von dem Programmierer muss in der SPS 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Moving-Average-Filters
fManSyncValue	FLOAT	Eingangsgröße, auf die das Moving-Average-Filter gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das Moving-Average-Filter auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Moving-Average-Filters
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

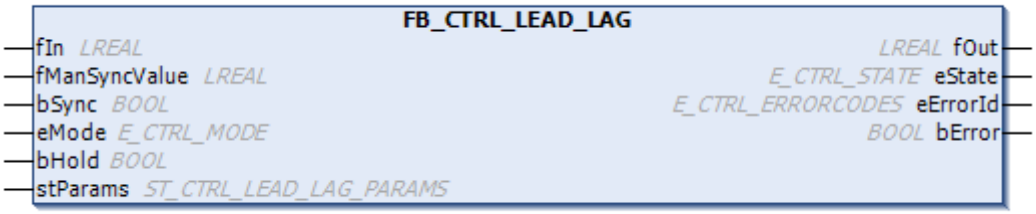
Name	Typ	Beschreibung
stParams	ST_CTRL_MOVIN G_AVERAGE_PA RAMS	Parameterstruktur des Moving-Average-Filters

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_MOVING_AVERAGE_PARAMS:
STRUCT
    tCtrlCycleTime        : TIME := T#0ms;
    tTaskCycleTime        : TIME := T#0ms;
    nSamplesToFilter      : UINT;
    pWorkArray_ADR        : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF     : UINT     := 0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
nSamplesTo Filter	UINT	Anzahl der Werte „n“, über die der arithmetische Mittelwert gebildet wird.
pWorkArray_ ADR	POINTER TO FLOAT	Adresse des Arrays, indem die Eingangswerte zwischengespeichert werden.
nWorkArray_ SIZEOF	UINT	Größe des `WorkArrays`

4.2.1.4.6 FB_CTRL_LEAD_LAG

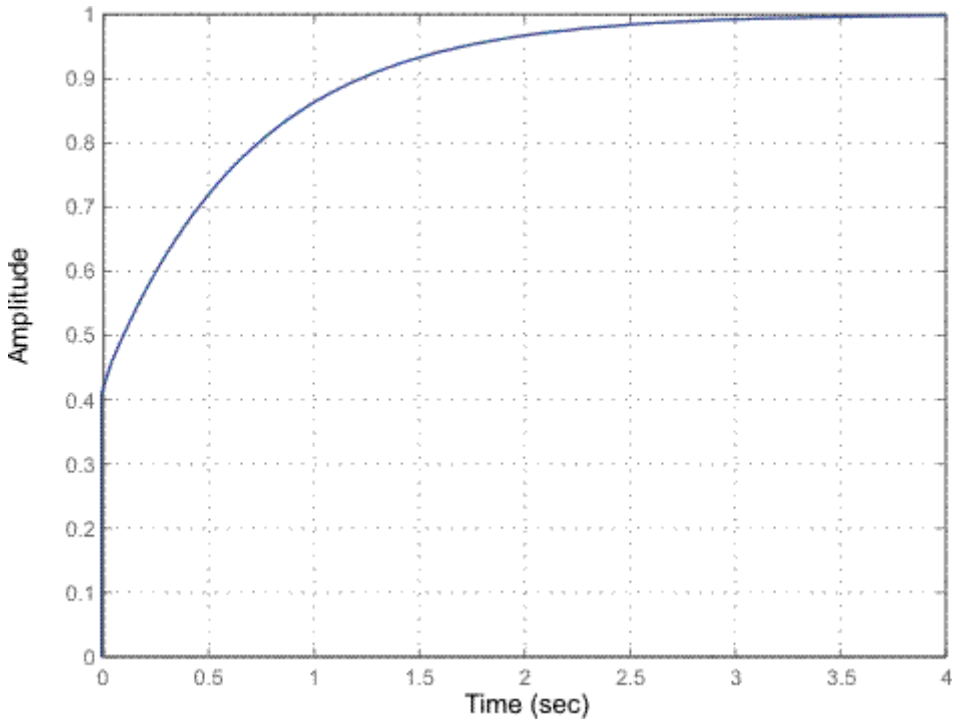


Der Funktionsbaustein stellt ein digitales Lead-Lag-Filter dar.

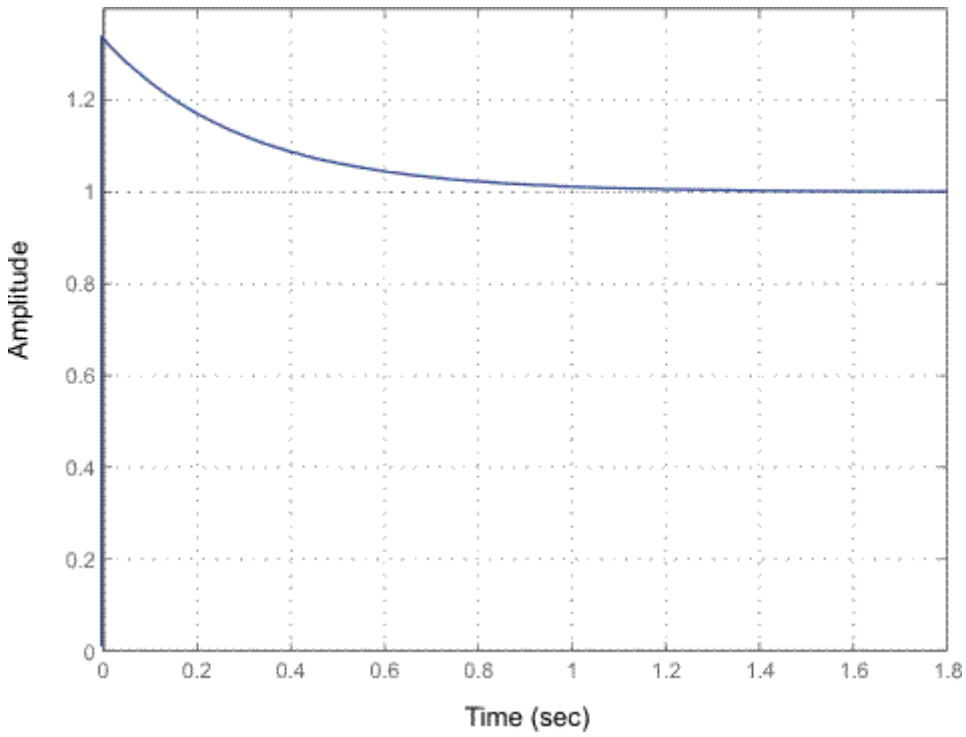
Übertragungsfunktion

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

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Notch-Filters
fManSyncValue	FLOAT	Eingangsgröße, die im Manual-Mode am Ausgang ausgegeben wird.
bSync	FLOAT	Reserve
fManValue	FLOAT	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
eMode	E_CTRL_MODE	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Lead-Lag-Filters
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174]. (p>)
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

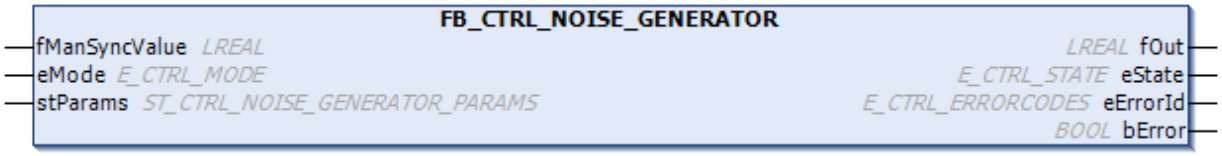
Name	Typ	Beschreibung
stParams	ST_CTRL_LEAD_ LAG_FILTER_ PARAMS	Parameterstruktur des Lead-Lag-Filters

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_LEAD_LAG_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    tT1               : TIME := T#0ms; (* T1 *)
    tT2               : TIME := T#0ms; (* T2 *)
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tT1	TIME	T1, siehe G(s).
tT2	TIME	T2, siehe G(s).

4.2.1.4.7 FB_CTRL_NOISE_GENERATOR



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

Ausgangssignal

Ausgangssignal bei einer Amplitude von 5.0.

VAR_INPUT

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

Name	Typ	Beschreibung
fManSyncValue	FLOAT	Eingangsgröße, die im Manual-Mode am Ausgang ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Rauschgenerators
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

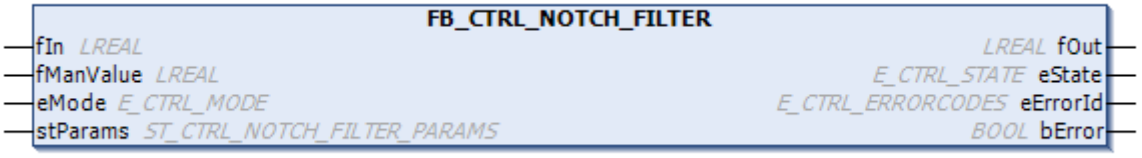
Name	Typ	Beschreibung
stParams	ST_CTRL_NOISE_GENERATOR_PARAMS	Parameterstruktur des Rauschgenerators

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_NOISE_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime : TIME := T#0ms;
    tTaskCycleTime : TIME := T#0ms;
    fAmplitude     : FLOAT := 0;
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fAmplitude	FLOAT	Amplitude des Ausgangssignals: Es wird am Ausgang des Funktionsbausteins ein Rauschsignal im Intervall [-fAmplitude/2.0 ... fAmplitude/2.0] erzeugt.

4.2.1.4.8 FB_CTRL_NOTCH_FILTER

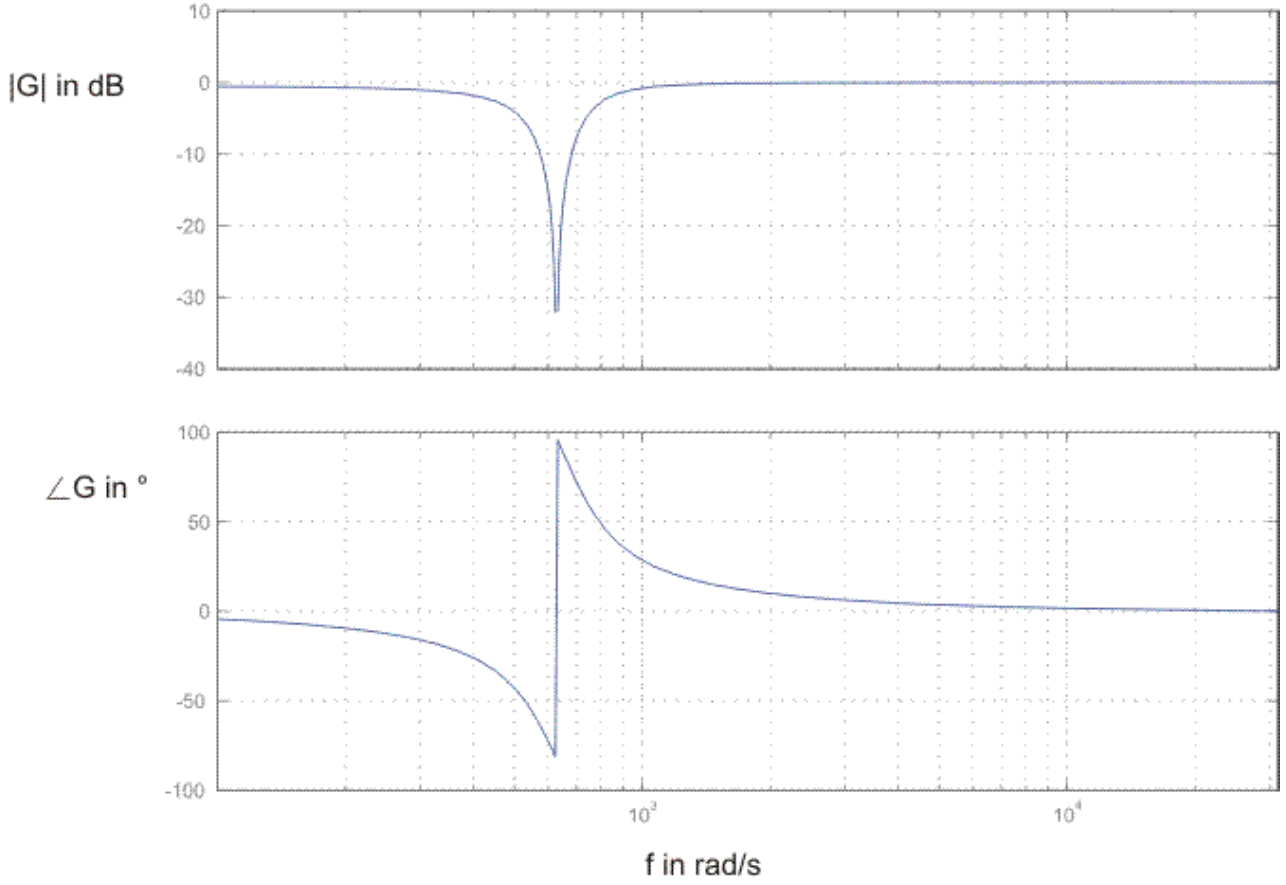


Der Funktionsbaustein stellt einen digitalen Notch-Filter dar.

Transferfunktion

$$G(s) = \frac{s^2 + w_0^2}{s^2 + ew_0 s + w_0^2}$$

Bodediagramm



mit:

$f_0 = 100Hz$

$e = 0.25$

VAR_INPUT

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

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Notch-Filters
fManValue	FLOAT	Eingangsgröße, die im Manual-Mode am Ausgang ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Notch-Filters
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

Name	Typ	Beschreibung
stParams	ST_CTRL_ NOTCH_FILTER_ PARAMS	Parameterstruktur des Notch-Filters

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_NOTCH_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fNotchFreq      : FLOAT := 0;
    fBandwidth      : FLOAT := 0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fNotchFreq	FLOAT	Notch-Frequenz in Hz
fBandwidth	FLOAT	Bandbreite bezogen auf die Notchfrequenz: Bandbreite in Hz = fNotchFreq * fBandwidth

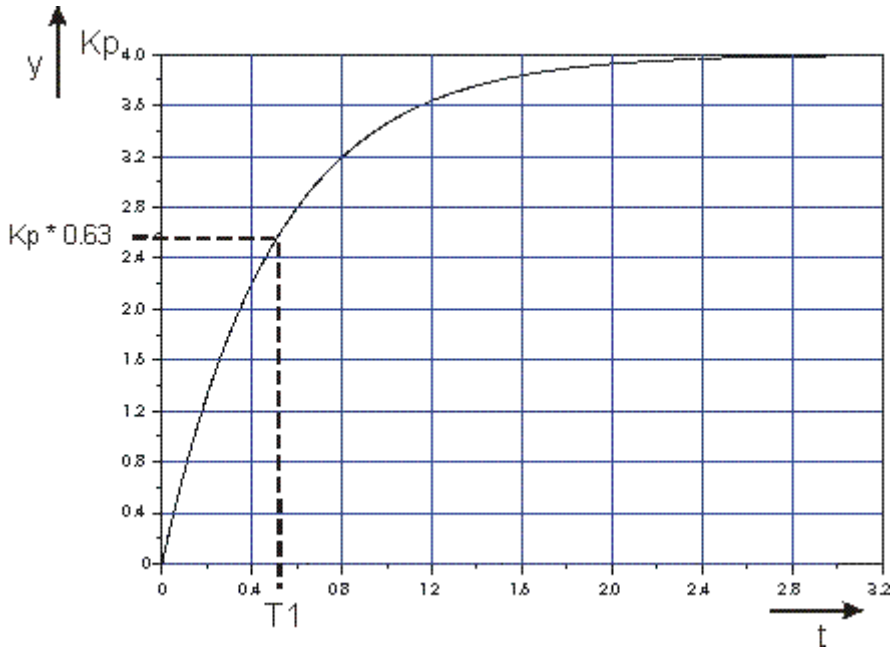
4.2.1.4.9 FB_CTRL_PT1

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des PT1-Glieds
fManSyncValue	FLOAT	Eingangsgröße, auf die das PT1-Glied gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PT1-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PT1-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

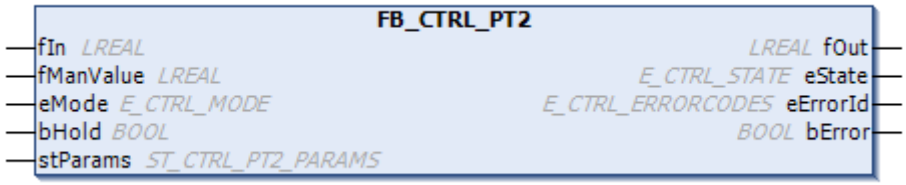
Name	Typ	Beschreibung
stParams	ST_CTRL_PT1_ PARAMS	Parameterstruktur des PT1-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT1_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tT1             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
tT1	TIME	Zeitkonstante

4.2.1.4.10 FB_CTRL_PT2

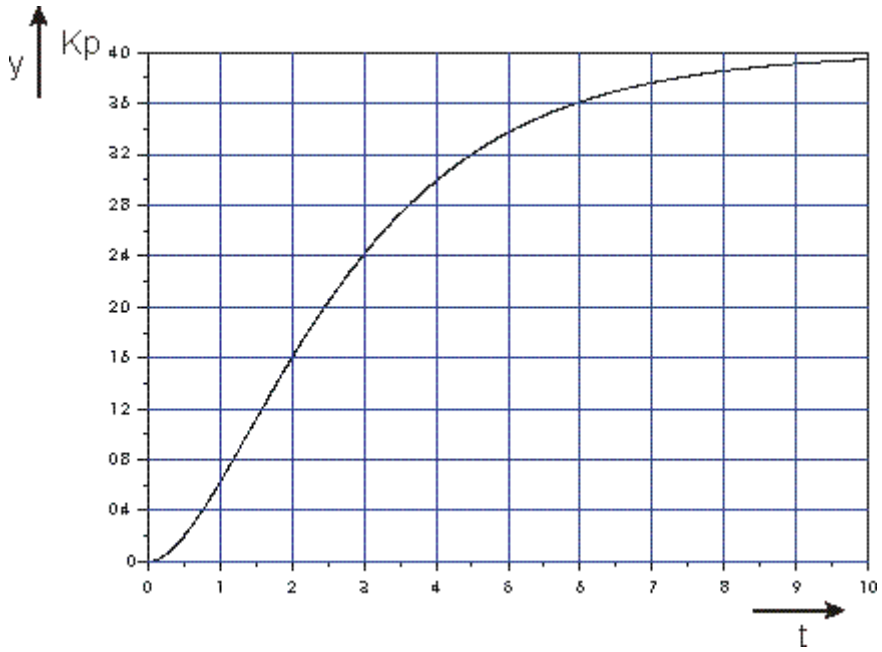


Der Funktionsbaustein stellt ein nicht schwingungsfähiges PT2-Übertragungsglied im Wirkungsplan dar.

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des PT2-Glieds
fManValue	FLOAT	Eingangsgröße, die im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PT2-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

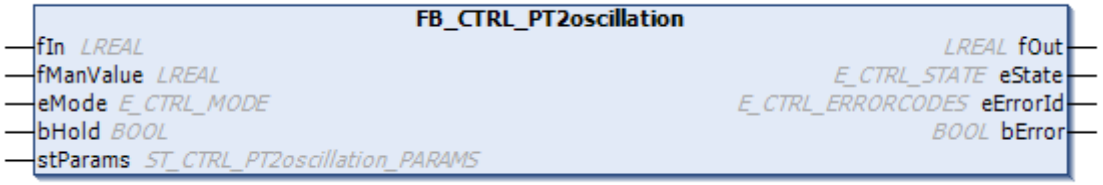
Name	Typ	Beschreibung
stParams	ST_CTRL_PT2_PARAMS	Parameterstruktur des PT2-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT2_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tT1             : TIME := T#0ms;
    tT2             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
tT1	TIME	Zeitkonstante T1
tT2	TIME	Zeitkonstante T2

4.2.1.4.11 FB_CTRL_PT2oscillation

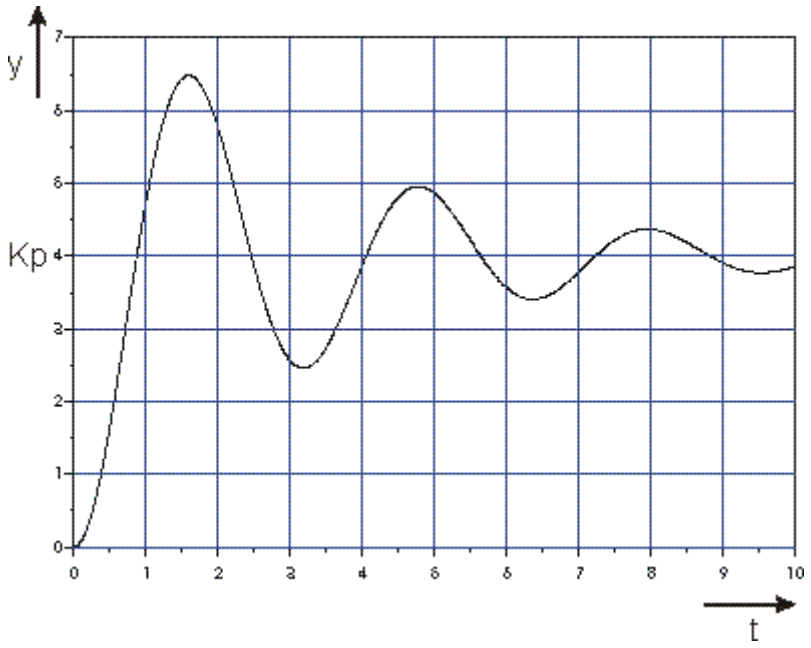


Der Funktionsbaustein stellt ein schwingungsfähiges PT2-Übertragungsglied im Wirkungsplan dar.

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des schwingungsfähigen PT2-Glieds
fManValue	FLOAT	Eingangsgröße, die im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PT2-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

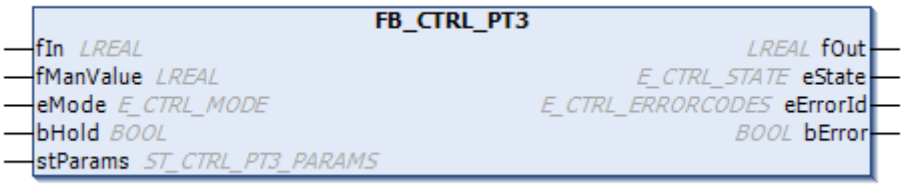
Name	Typ	Beschreibung
stParams	ST_CTRL_PT2 oscillation_ PARAMS	Parameterstruktur des schwingungsfähigen PT2-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT2oscillation_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    fTheta          : FLOAT := 0;
    tT0             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Proportionalbeiwert
fTheta	FLOAT	Dämpfungsgrad
tT0	TIME	Kennzeit

4.2.1.4.12 FB_CTRL_PT3

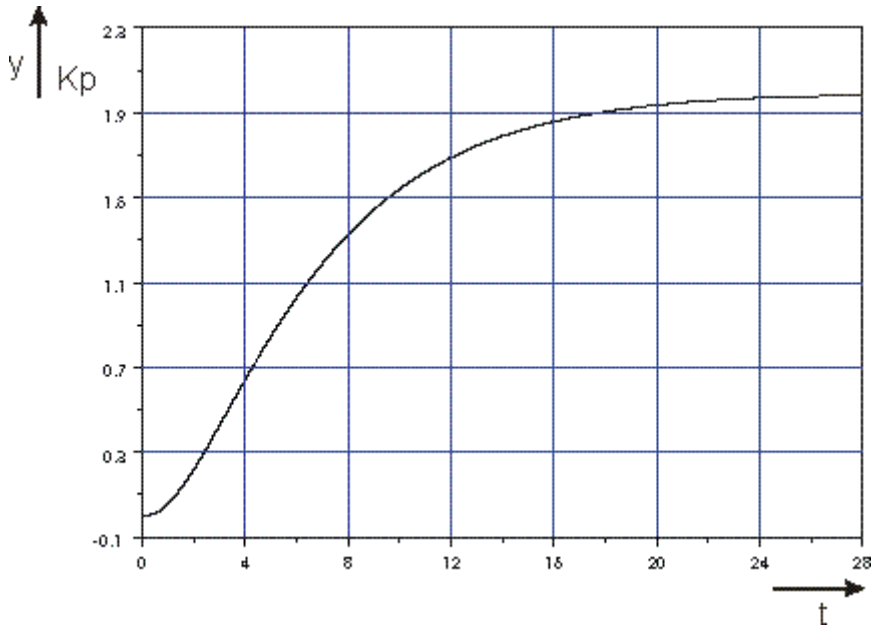


Der Funktionsbaustein stellt ein nicht schwingungsfähiges PT3-Übertragungsglied im Wirkungsplan dar.

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des PT3-Glieds
fManValue	FLOAT	Eingangsgröße, die im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PT3-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

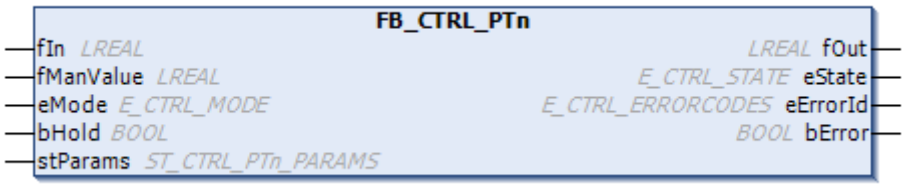
Name	Typ	Beschreibung
stParams	ST_CTRL_PT3_PARAMS	Parameterstruktur des PT3-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PT3_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tT1             : TIME := T#0ms;
    tT2             : TIME := T#0ms;
    tT3             : TIME := T#0ms;
END_STRUCT
END_TYPE>
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
tT1	TIME	Zeitkonstante T1
tT2	TIME	Zeitkonstante T2p
tT3	TIME	Zeitkonstante T3

4.2.1.4.13 FB_CTRL_PTn

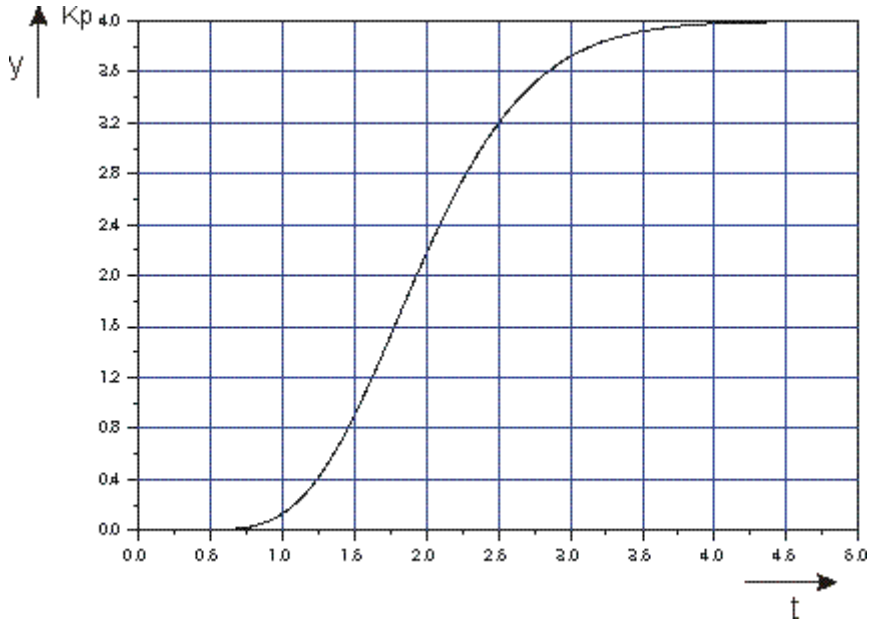


Der Funktionsbaustein stellt ein nicht schwingungsfähiges PTn-Übertragungsglied mit „n<= 10“ und gleichen Zeitkonstanten im Wirkungsplan dar.

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des PTn-Glieds
fManValue	FLOAT	Eingangsgröße, die im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	BOOL	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PTn-Glieds
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

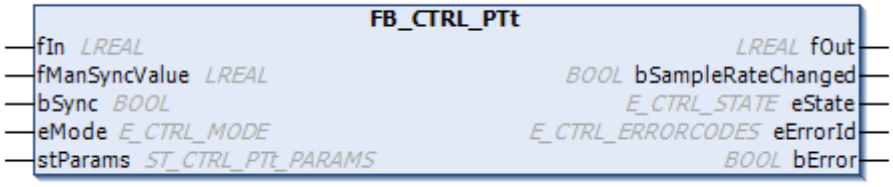
Name	Typ	Beschreibung
stParams	ST_CTRL_PTn_PARAMS	Parameterstruktur des PTn-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PTn_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fKp               : FLOAT := 0;
    tT1               : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
tT1	TIME	Zeitkonstante T1

4.2.1.4.14 FB_CTRL_PTt



Der Funktionsbaustein stellt ein PTt-Übertragungsglied im Wirkungsplan dar.

Übertragungsfunktion

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

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des PTt-Glieds
fManSyncValue	FLOAT	Eingangsgröße, auf die das PTt-Glied gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das PTt-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des PTt-Glieds
bSampleRate Changed	BOOL	Ausgang, der anzeigt, ob der Baustein intern die Abtastrate reduziert hat, da das Array zur Verzögerung des Eingangssignals sonst nicht genügend Platz bietet.
eErrorId	E_CTRL_ ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

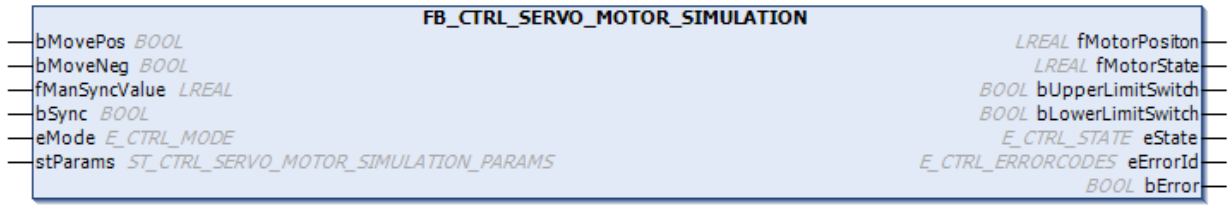
Name	Typ	Beschreibung
stParams	ST_CTRL_PTt_ PARAMS	Parameterstruktur des PTt-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_PTt_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tTt             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
tTt	TIME	Totzeit

4.2.1.4.15 FB_CTRL_SERVO_MOTOR_SIMULATION



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
```

Name	Typ	Beschreibung
bMovePos	BOOL	Eingang, der den simulierten Stellantrieb in positive Richtung fährt.
bMoveNeg	BOOL	Eingang, der den simulierten Stellantrieb in negative Richtung fährt.
fManSyncValue	FLOAT	Eingang, mit dem die simulierte Motorstellung gesetzt werden kann, oder auf dessen Wert im Manual-Mode gefahren wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird die simulierte Motorposition auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    fMotorPositon      : FLOAT;
    fMotorState        : FLOAT;
    bUpperLimitSwitch  : BOOL;
    bLowerLimitSwitch  : BOOL;
    eState             : E_CTRL_STATE;
    eErrorId           : E_CTRL_ERRORCODES;
    bError             : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fMotorPositon	FLOAT	Simulierte Motorstellung im Intervall [fMovingRangeMin ... fMovingRangeMax]
fMotorState	FLOAT	Simulierte Motorstellung im Intervall [0 ... 100.0]
bUpperLimit Switch	BOOL	Simulierter Endschalter am positiven Anschlag des Stellantriebs
bLowerLimit Switch	BOOL	Simulierter Endschalter am negativen Anschlag des Stellantriebs.
eState	E_CTRL_ STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

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

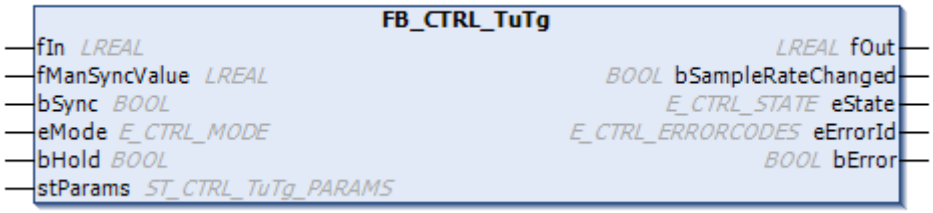
Name	Typ	Beschreibung
stParams	ST_CTRL_ SERVO_MOTOR_ SIMULATION_ PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fMovingRangeMin   : FLOAT := 0;
    fMovingRangeMax   : FLOAT := 0;
    tMovingTime       : TIME := T#0ms;
    tDeadTime         : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fMovingRange Min	FLOAT	Minimale Position des simulierten Stellantriebs
fMovingRange Max	FLOAT	Maximale Position des simulierten Stellantriebs
tMovingTime	TIME	Die Zeit, die benötigt wird, um den simulierten Stellantrieb von einem Anschlag zum anderen zu fahren.
tDeadTime	TIME	Totzeit des simulierten Stellantriebs

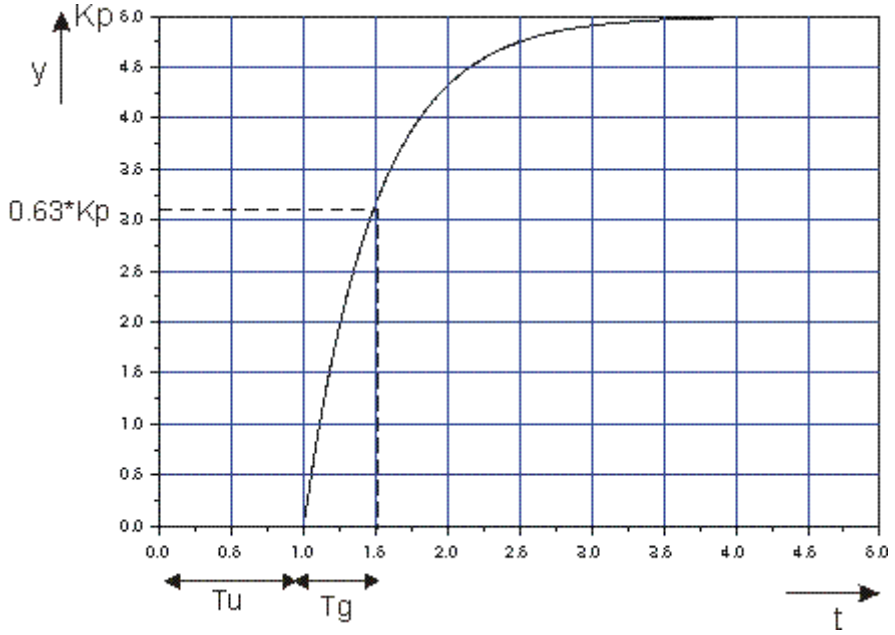
4.2.1.4.16 FB_CTRL_TuTg



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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des TuTg-Glieds
fManSyncValue	FLOAT	Eingangsgröße, auf die das TuTg-Glied gesetzt werden kann, oder die im Manual-Mode am Ausgang ausgegeben wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird das TuTg-Glied auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.
bHold	FLOAT	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des TuTg-Glieds
bSampleRate Changed	BOOL	Ausgang, der anzeigt, ob der Baustein intern die Abtastrate reduziert hat, da das Array zur Verzögerung des Eingangssignals sonst nicht genügend Platz bietet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams      : ST_CTRL_TuTg_PARAMS;
END_VAR
```

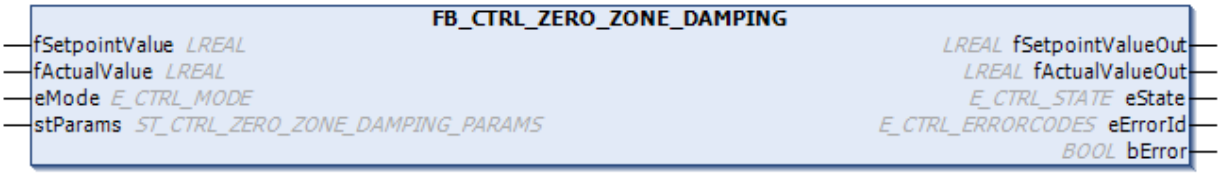
Name	Typ	Beschreibung
stParams	ST_CTRL_TuTg_ PARAMS	Parameterstruktur des TuTg-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_TuTg_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME  := T#0ms;
    tTaskCycleTime    : TIME  := T#0ms;
    fKp               : FLOAT := 0;
    tTu               : TIME  := T#0ms;
    tTg               : TIME  := T#0ms;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fKp	FLOAT	Reglerverstärkung / Übertragungsbeiwert
tTu	TIME	Totzeit
tTg	TIME	Zeitkonstante

4.2.1.4.17 FB_CTRL_ZERO_ZONE_DAMPING

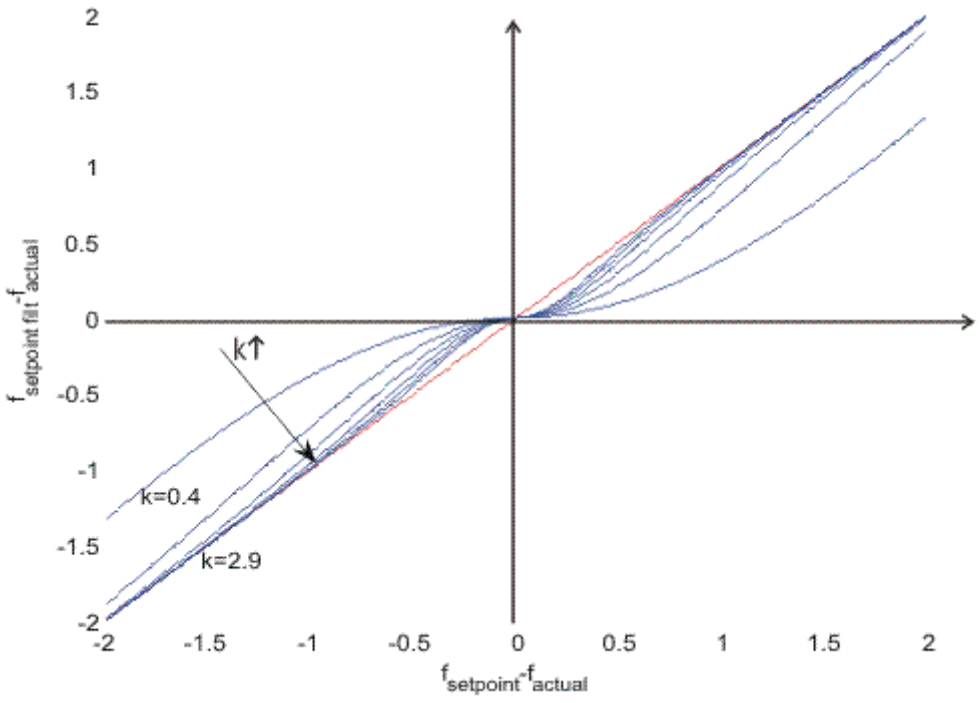


Mit diesem Funktionsbaustein kann eine Nullpunktdämpfung realisiert werden, um Regeleingriffe im Bereich | Istwert.-Sollwert | < ε zu minimieren.

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
```

Name	Typ	Beschreibung
fSetpointValue	FLOAT	Sollwert der Regelgröße
fActualValue	FLOAT	Istwert der Regelgröße
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fSetpointValue Out	FLOAT	Gefilterter Sollwert zum Regler
fActualValueOut	FLOAT	Istwert zum Regler
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_ZERO_ZONE_DAMPING_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_ZERO_ ZONE_ DAMPING_ PARAMS	Parameterstruktur des Übertragungselementes

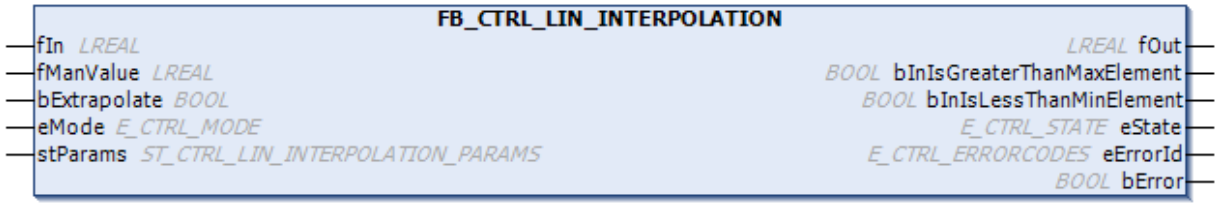
`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_PI_PST_CTRL_ZERO_ZONE_DAMPING_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME  := T#0ms;
    tTaskCycleTime          : TIME  := T#0ms;
    fDampingCoefficient     : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fDampingCoefficient	FLOAT	Der Parameter entspricht $k_{damping}$ in der Übertragungsfunktion.

4.2.1.5 Interpolation

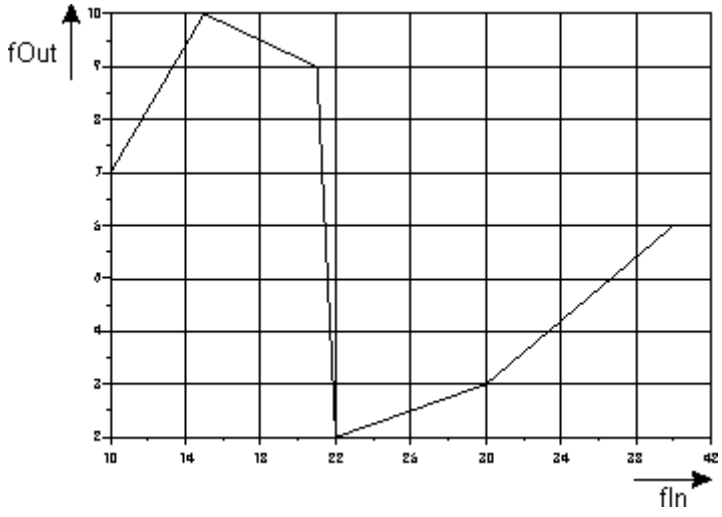
4.2.1.5.1 FB_CTRL_LIN_INTERPOLATION



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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Interpolationsbausteins
fManValue	FLOAT	Eingangsgröße, dessen Wert im Manual-Mode ausgegeben wird.
bExtrapolate	BOOL	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	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] 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
```

Name	Typ	Beschreibung
fOut	FLOAT	Linear interpolierter Tabellenwert
bInIsGreaterThanMax Element	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße größer ist als die größte Stützestelle.
bInIsLessThanMin Element	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße kleiner ist als die kleinste Stützestelle.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIN_INTERPOLATION_PARAMS;
END_VAR
```

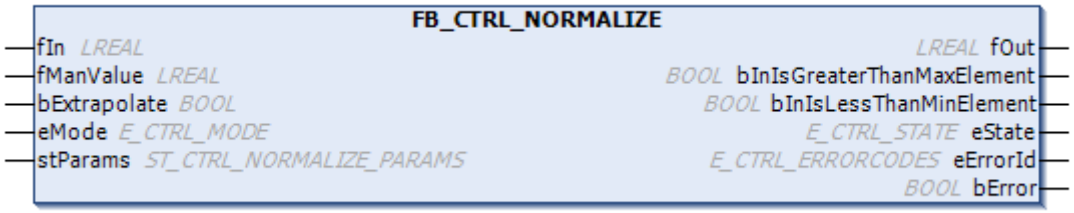
Name	Typ	Beschreibung
stParams	ST_CTRL_LIN_INTERPOLATION_PARAMS	Parameterstruktur des Interpolations-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime           : TIME := T#0ms;
    tTaskCycleTime           : TIME := T#0ms;
    pDataTable_ADR           : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF        : UINT     := 0;
    nDataTable_NumberOfRows  : UINT     := 0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	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	POINTER TO FLOAT	Adresse des n x 2 - Arrays, welches linear interpoliert wird.
nDataTable_ SIZEOF	UINT	Größe des n x 2 - Arrays
nDataTable_ NumberOfRows	UINT	Anzahl der Zeilen des Arrays

4.2.1.5.2 FB_CTRL_NORMALIZE

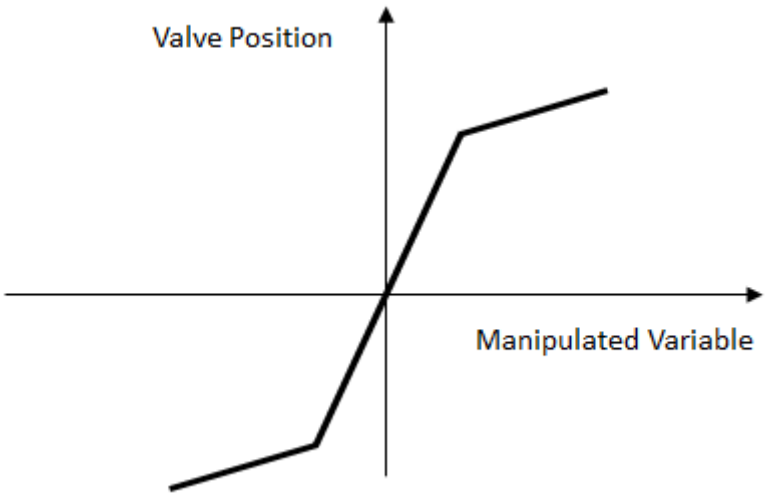


Mit diesem Funktionsbaustein kann ein nichtlineares Übertragungsglied mit Hilfe einer inversen Kennlinie linearisiert werden.

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.

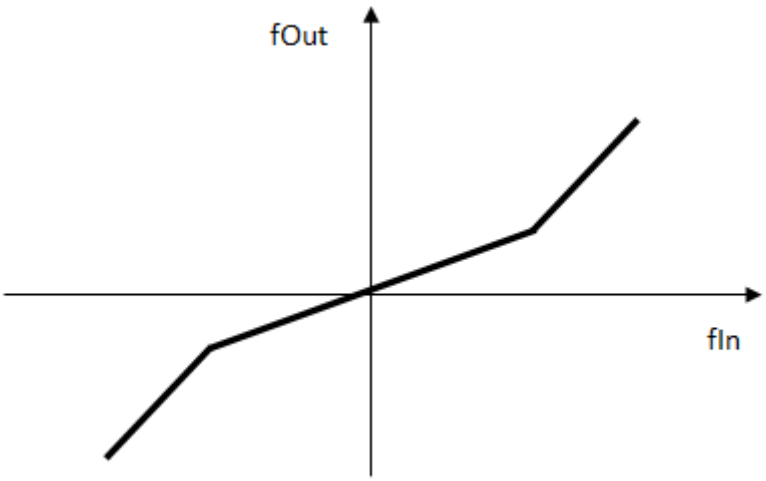
Beispiel

Die folgende Ventilkennlinie wird mit 4 Stützstellen in der Tabelle abgelegt.

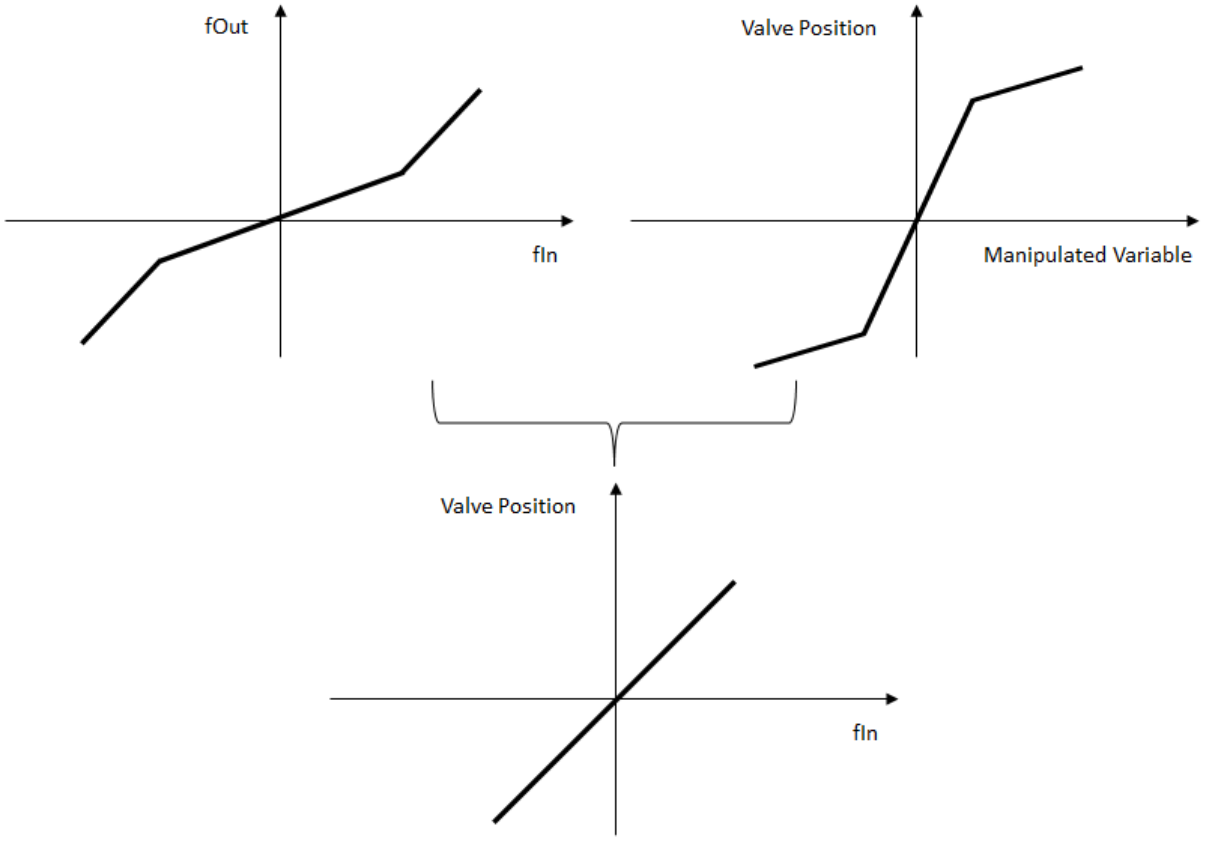


Stellgröße	Ventilstellung
arrTable[1,1] := -6;	arrTable[1,2] := -100;
arrTable[2,1] := -1;	arrTable[2,2] := -70;
arrTable[3,1] := 1;	arrTable[3,2] := 70;
arrTable[4,1] := 6;	arrTable[4,2] := 100;

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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße
fManValue	FLOAT	Eingangsgröße, dessen Wert im Manual-Mode ausgegeben wird.
bExtrapolate	BOOL	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	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] 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
```

Name	Typ	Beschreibung
fOut	FLOAT	Linear interpolierter Tabellenwert
bInIsGreaterThanMaxElement	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße größer ist als die größte Stützestelle.
bInIsLessThanMinElement	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Eingangsgröße kleiner ist als die kleinste Stützestelle.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_NORMALIZE_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_NORMALIZE_PARAMS	Parameterstruktur des Funktionsbausteins

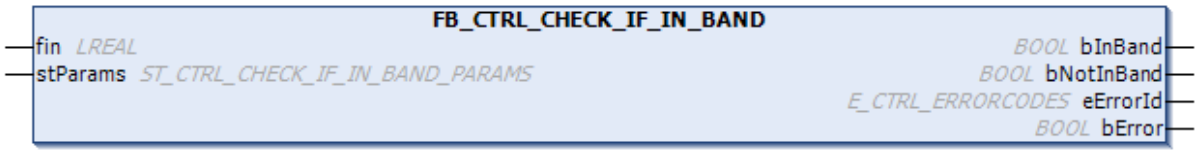
`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_NORMALIZE_PARAMS:
STRUCT
    tCtrlCycleTime           : TIME := T#0ms;
    tTaskCycleTime           : TIME := T#0ms;
    pDataTable_ADR           : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF        : UINT     := 0;
    nDataTable_NumberOfRows  : UINT     := 0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	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	POINTER TO FLOAT	Adresse des n x 2 - Arrays, welches linear interpoliert wird.
nDataTable_ SIZEOF	UINT	Größe des n x 2 - Arrays
nDataTable_ NumberOfRows	UINT	Anzahl der Zeilen des Arrays

4.2.1.6 Monitoring / Alarming

4.2.1.6.1 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße, die überwacht wird.

VAR_OUTPUT

```
VAR_OUTPUT
    bInBand     : BOOL;
    bNotInBand  : BOOL;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bInBand	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich die Eingangsgröße in dem angegebenen Intervall befindet.
bNotInBand	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich die Eingangsgröße **nicht** in dem angegebenen Intervall befindet.
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_CHECK_IF_IN_BAND_PARAMS;
END_VAR
```

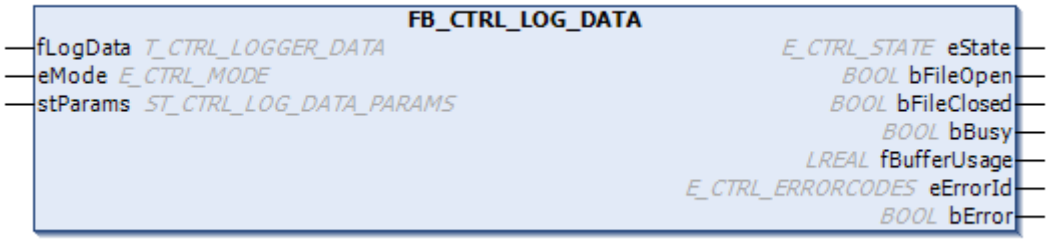
Name	Typ	Beschreibung
stParams	ST_CTRL_ CHECK_IF_IN_ BAND_PARAMS	Parameterstruktur des Funktionsbausteins

stParams besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_CHECK_IF_IN_BAND_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fMin            : FLOAT;
    fMax            : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fMin	FLOAT	Untere Grenze des Intervalls
fMax	FLOAT	Obere Grenze des Intervalls

4.2.1.6.2 FB_CTRL_LOG_DATA



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 Einträge 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.

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.

Wenn eine Pufferadresse und eine Puffergröße ungleich Null parametriert wird, wird der externe Puffer genutzt. Ohne einen externen Puffer wird ein interner Puffer mit einer Größe von 255 Byte verwendet.

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
```

Name	Typ	Beschreibung
fLogData	T_CTRL_LOGGER_DATA	Array mit den Werten, die in das Log-File geschrieben werden.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eState        : E_CTRL_STATE;
    bFileOpen     : BOOL
    bFileClosed   : BOOL
    fBufferUsage  : FLOAT
    bBusy         : BOOL
    eErrorId      : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
eState	E_CTRL_STATE	State des Funktionsbausteins
bFileOpen	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geöffnet wurde.
bFileClosed	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geschlossen wurde.
fBufferUsage	FLOAT	Aktueller Füllstand des externen Puffers in Prozent
bBusy	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass das Loggen einer Zeile aktiv ist.
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_DATA_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_LOG_ DATA_PARAMS	Parameterstruktur des Log-Bausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_LOG_DATA_PARAMS:
STRUCT
    tLogCycleTime             : TIME := T#0ms;
    tTaskCycleTime            : TIME := T#0ms;
    sFileName                 : STRING;
    sNetId                    : T_AmsNetId := '';
    tFileOperationTimeou      : TIME := T#3s;
    nNumberOfColumn           : INT(1..10);
    arColumnHeadings          : ARRAY [1..10] OF STRING;
    bAppendData               : BOOL := FALSE;
    bWriteTimeStamps          : BOOL := TRUE;
    bWriteColumnHeadings      : BOOL := TRUE;
    bWriteAbsoluteTimeStamps  : BOOL := FALSE
    pLogBuffer_ADR            : POINTER TO BYTE;
    nLogBuffer_SIZEOF         : UDINT;
END_STRUCT
END_TYPE
```

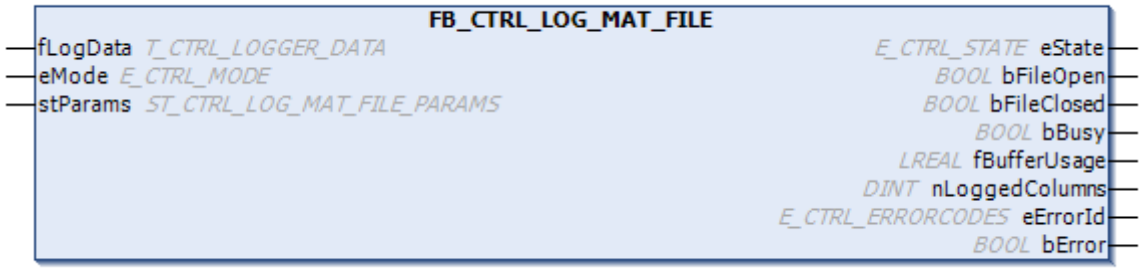
Name	Typ	Beschreibung
tLogCycleTime	TIME	Zykluszeit, mit der Einträge in das Log-File geschrieben werden. Diese muss größer oder gleich der **TaskCycleTime** sein.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
sFileName	STRING	Name und Pfad des Log-Files, z.B.: d:\Logfile.csv
sNetId	T_AmsNetId	Das Log-File wird auf das System mit der hier angegebenen NetId geschrieben.
tFileOperation Timeou	TIME	Timeout für alle Dateioperationen
nNumberOf Column	INT	Anzahl der Spalten, die in die Datei geschrieben werden (maximal 10)
arColumn Headings	ARRAY	Array aus Strings, die die Spaltenüberschriften enthalten.
bAppendData	BOOL	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.
bWriteTime Stamps	BOOL	Wenn dieser Parameter auf TRUE gesetzt ist, wird in die erste Spalte der Datei der Zeitstempel der Messung geschrieben.
bWriteColumn Headings	BOOL	Wenn dieser Parameter auf TRUE gesetzt ist, werden in die 1. Zeile der Datei die Spaltenüberschriften geschrieben.
bWriteAbsolute TimeStamps	BOOL	Wenn dieser Parameter auf TRUE gesetzt wird, wird als Timestamp die lokale NT-Zeit verwendet. In diesem Fall beträgt die minimale `LogCycleTime` 5s!
pLogBuffer_ADR	POINTER TO BYTE	Adresse des externen `LogBuffers` – um einen Puffer zu erkennen, muss die Adresse ungleich 0 sein.
nLogBuffer_ SIZEOF	UDINT	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

Fehler beim Dateihandling

Der Parametersatz darf nur dann geändert werden, wenn die Datei geschlossen ist (bFileClosed = TRUE). Anderenfalls kann es zu Fehlern bei dem Dateihandling kommen.

4.2.1.6.3 FB_CTRL_LOG_MAT_FILE



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 Namen der Variablen 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.

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 Buffer-Überlauf kommt, wird der Baustein gestoppt und ein Fehler ausgegeben.

Wenn eine Bufferadresse und eine Buffergröße ungleich Null parametriert werden, wird der externe Buffer genutzt. Ohne einen externen Buffer wird ein interner Buffer mit einer Größe von 1500 Byte verwendet.

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
```

Name	Typ	Beschreibung
fLogData	T_CTRL_LOGGER_DATA	Array mit den Werten, die in das Log-File geschrieben werden.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

```
VAR_OUTPUT
    eState          : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT
    nLoggedColumns  : DINT;
    bBusy           : BOOL
```

```
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Typ	Beschreibung
eState	E_CTRL_STATE	State des Funktionsbausteins
bFileOpen	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geöffnet wurde.
bFileClosed	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass die Datei erfolgreich geschlossen wurde.
fBufferUsage	FLOAT	Aktueller Füllstand des externen Buffers in Prozent
nLogged Columns	DINT	Anzahl der in die Datei geschriebenen Spalten
bBusy	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass das Loggen einer Zeile aktiv ist.
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_OUTPUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_MAT_FILE_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_LOG_ MAT_FILE_ PARAMS	Parameterstruktur des Log-Bausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_LOG_MAT_FILE_PARAMS:
STRUCT
    tLogCycleTime         : TIME := T#0ms;
    tTaskCycleTime        : TIME := T#0ms;
    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
```

Name	Typ	Beschreibung
tLogCycleTime	TIME	Zykluszeit, mit der Einträge in das Log-File geschrieben werden. Diese muss größer oder gleich der **TaskCycleTime** sein.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
sFileName	STRING	Name und Pfad des Log-Files, z.B.: d:\Logfile.mat
nNumberOfRows	INT	Anzahl der Spalten, die in die Datei geschrieben werden (maximal 10).
sMatrixName	STRING	Name des Daten-Arrays
arRow Description	ARRAY	Array aus Strings, die die Zeilenüberschriften enthalten.
bWriteTime Stamps	BOOL	Wenn dieser Parameter auf TRUE gesetzt ist, wird in die erste Zeile des Daten-Arrays der Zeitstempel der Messung geschrieben.
bWriteRow Description	BOOL	Wenn dieser Parameter auf TRUE gesetzt ist, wird ein cell-Array angelegt, in das die Beschreibungen der Zeilen geschrieben werden.
pLogBuffer_ADR	POINTER TO ST_CTRL_ MULTIPLE_PWM _OUT_DATA	Adresse des externen `LogBuffers` - um einen Buffer zu erkennen, muss die Adresse ungleich 0 sein.
nLogBuffer_ SIZEOF	UDINT	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

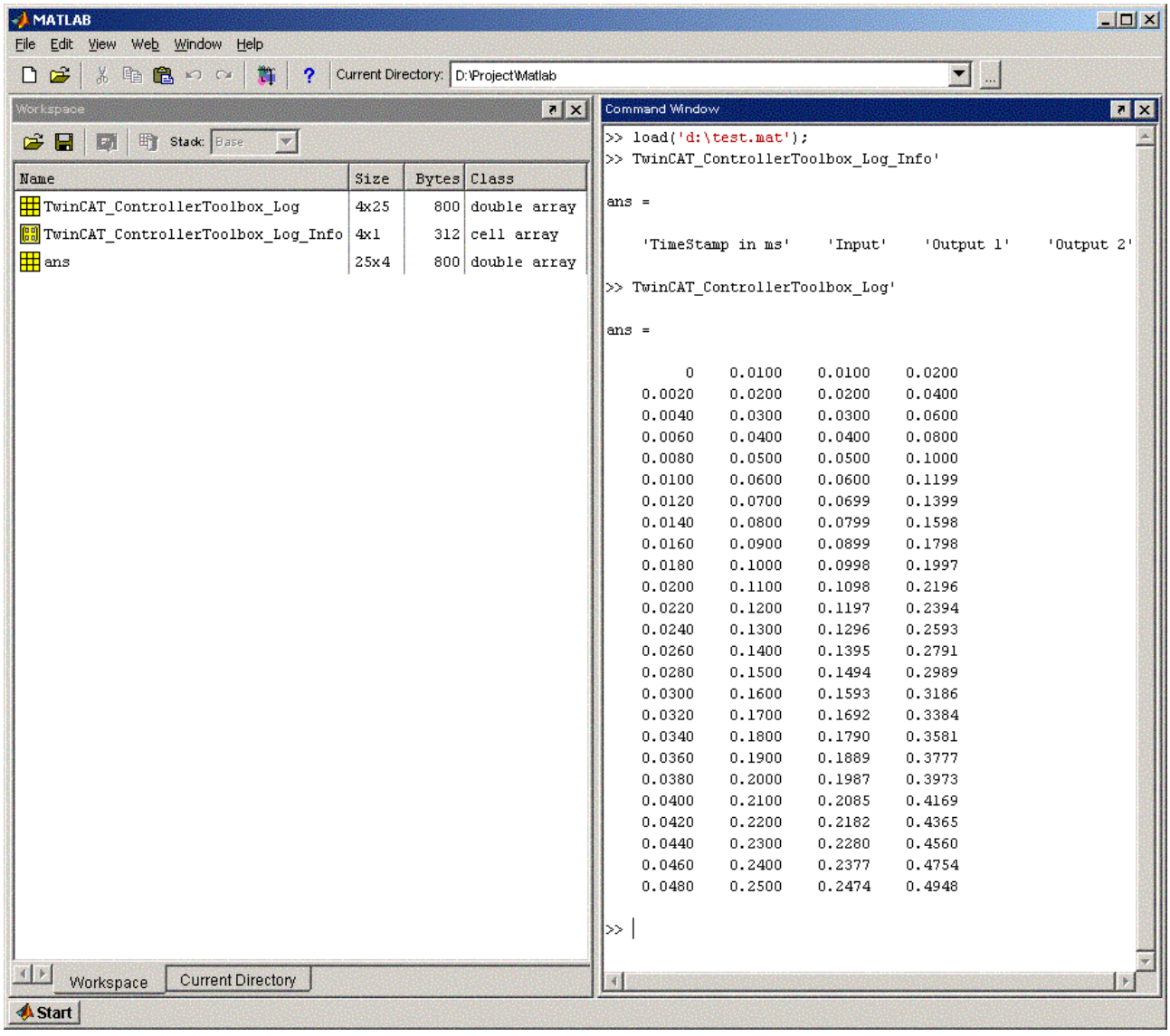
Fehler bei Dateihandling

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 );

fIn := fIn + 0.01;
```

```
fOut := SIN(fIn);

LoggerData[1]:= fIn;
LoggerData[2]:= fOut;
LoggerData[3]:= fOut * 2;

IF fbCtrlLogMatFile.nLoggedColumns >= 25 THEN
    eMode := eCTRL_MODE_Passive;
END_IF

fbCtrlLogMatFile( fLogData := LoggerData,
             eMode := eMode,
             stParams :=stParams,
             eErrorId => eErrorId,
             bError => bError);

fMaxBufferUsage := MAX(fbCtrlLogMatFile.fBufferUsage, fMaxBufferUsage);
```



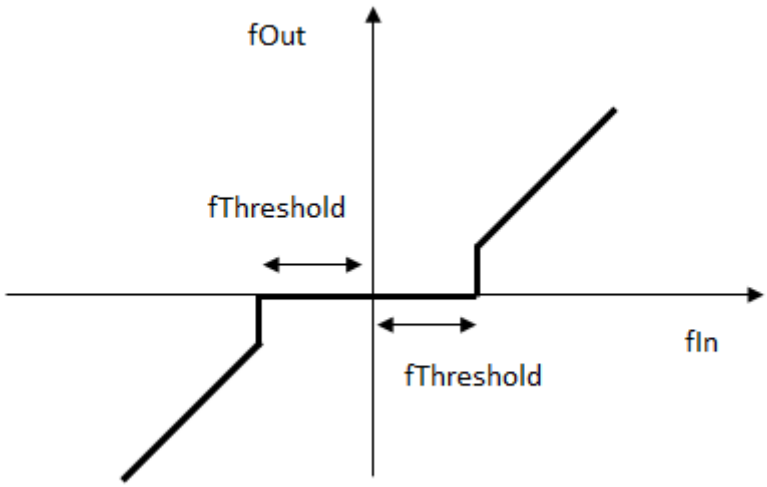
4.2.1.7 Output To Controlling Equipment

4.2.1.7.1 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

$$fout = \begin{cases} 0.0 : | fin | \leq fThreshold \\ fin : else \end{cases}$$



VAR_INPUT

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Funktionsbausteins
bInIsUnder Threshold	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass sich der Eingangswert in der Totzone befindet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_DEADBAND_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_DEAD BAND_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

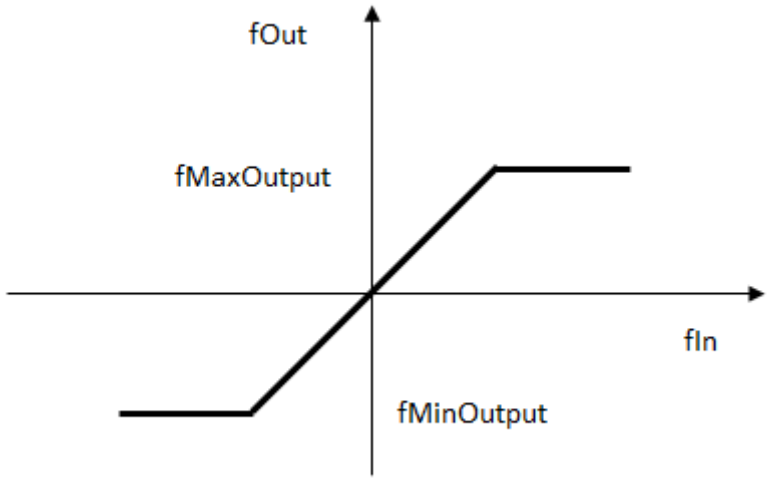
```
TYPE
ST_CTRL_DEADBAND_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fThreshold      : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fThreshold	FLOAT	Totzone des Funktionsbausteins, siehe Bild.

4.2.1.7.2 FB_CTRL_LIMITER

Dieser Baustein begrenzt ein Eingangssignal auf ein parametrierbares Intervall.

Beschreibung des Ausgangsverhaltens



VAR_INPUT

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Funktionsbausteins

VAR_OUTPUT

```
AR_OUTPUT
    fOut        : FLOAT;
    bMinLimit   : BOOL;
    bMaxLimit   : BOOL;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Funktionsbausteins
bMinLimit	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass der Ausgang die untere Grenze erreicht hat.
bMaxLimit	BOOL	Ein TRUE an diesem Ausgang signalisiert, dass der Ausgang die obere Grenze erreicht hat.
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten bError-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIMITER_PARAMS;
END_VAR
```

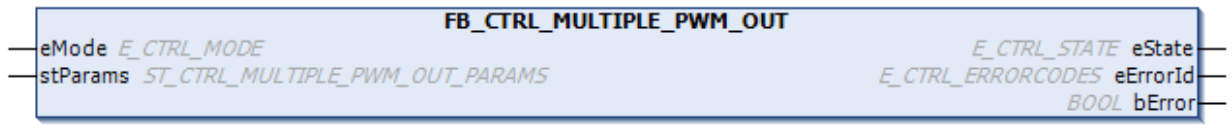
Name	Typ	Beschreibung
stParams	ST_CTRL_ LIMITER_ PARAMS	Parameterstruktur des Funktionsbausteins

stParams besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_LIMITER_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fMinOutput      : FLOAT;
    fMaxOutput      : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fMinOutput	FLOAT	Unteres Limit, auf das der Ausgang begrenzt wird.
fMaxOutput	FLOAT	Oberes Limit, auf das der Ausgang begrenzt wird.

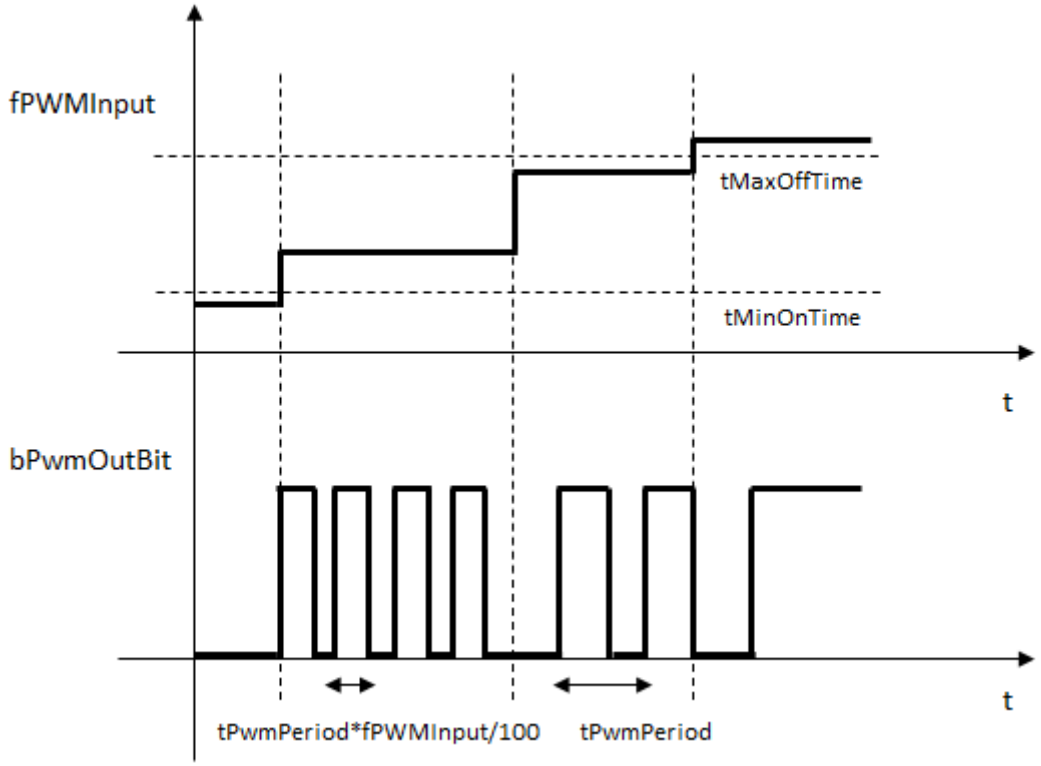
4.2.1.7.3 FB_CTRL_MULTIPLE_PWM_OUT



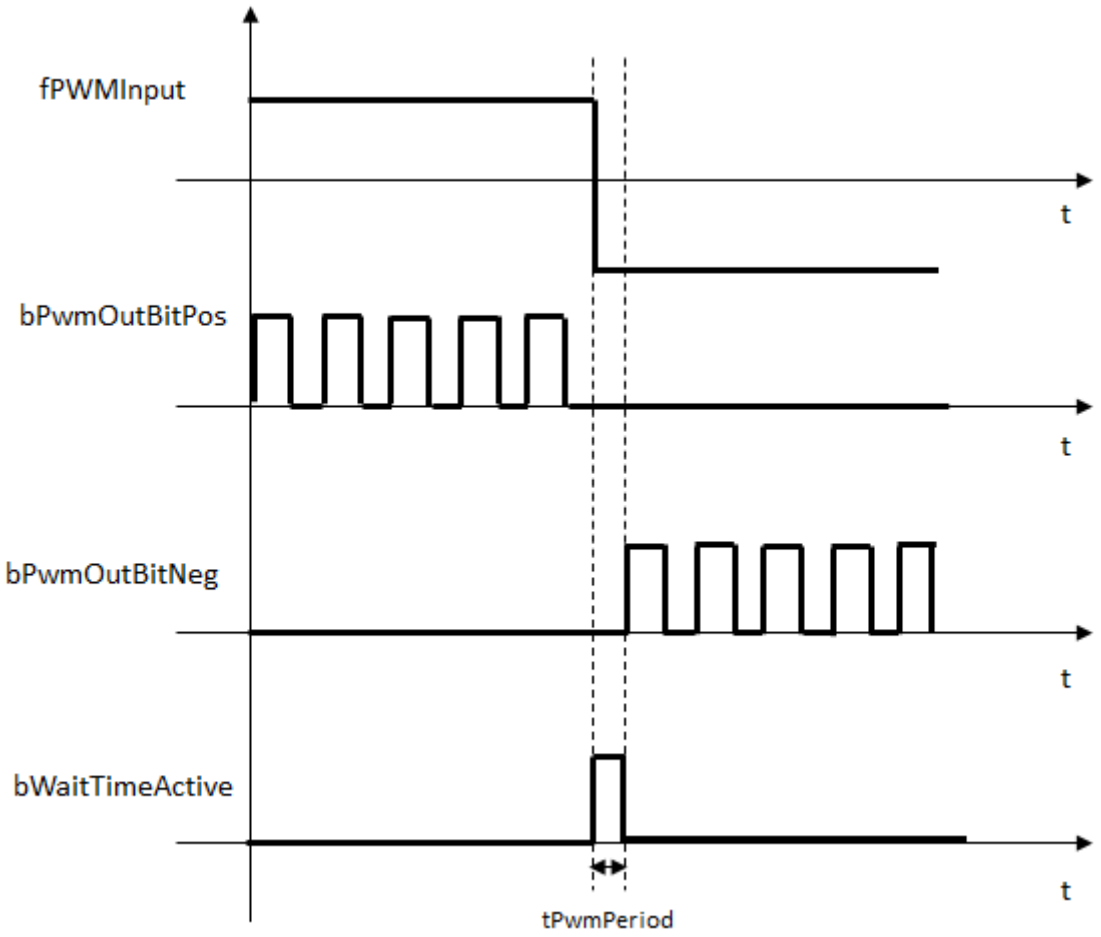
Dieser Baustein erzeugt aus mehreren Eingangssignalen PWM modulierte Ausgangssignale, wobei die Ausgangssignale zeitlich zueinander so angeordnet werden, dass möglichst wenige Ausgänge zeitgleich eingeschaltet sind. Durch diese zeitliche Anordnung wird das für die Stellglieder benötigte Leistungsmaximum reduziert.

Bei diesem erweiterten Baustein können 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:

```
aPwmInput           : ARRAY[1..nNumPwmOut] OF FLOAT;
aStWaitTimesConfig  : ARRAY[1..nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
aStPwmOutputs       : ARRAY[1.. nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
aStPwmDataVars      : ARRAY[1.. nNumPwmOut] 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 Array **ar_stPwmDataVars** 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
```

Name	Typ	Beschreibung
eMode	E_CTRL_MODE	Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
eErrorId	E_CTRL_ERRORCODES	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_PWM_OUT_EXT_PARAMS	Parameterstruktur des PWM-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_PARAMS :
STRUCT
    tTaskCycleTime            : TIME;
    tPWMPeriod                : TIME;
    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
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tPWMPeriod	TIME	Periodendauer des PWM-Signals
nNumberOfPwmOutputs	USINT	Anzahl der PWM-Ausgänge. [1...n]
pPwmInputArray_ADR	POINTER TO FLOAT	Adresse des PWM-Input-Arrays
nPwmInputArray_SIZEOF	UINT	Größe des PWM-Input-Arrays in Bytes
pPwmTimesConfig_ADR	POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_TIMES	Adresse des PWM-Times-Arrays
nPwmTimesConfig_SIZEOF	UINT	Größe des PWM-Times-Arrays in Bytes
pPwmOutputArray_ADR	POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS	Adresse des PWM-Output-Arrays
nPwmOutputArray_SIZEOF	UINT	Größe des PWM-Output-Arrays in Bytes
pPwmData_ADR	POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_DATA	Adresse des internen PWM-Data-Arrays
nPwmData_SIZEOF	UINT	Größe des internen PWM-Data-Arrays in Bytes

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_TIMES :
STRUCT
    tMinOnTime    : TIME;
    tMinOffTime   : TIME;
    tMinWaitTime  : TIME;
END_STRUCT
END_TYPE
```

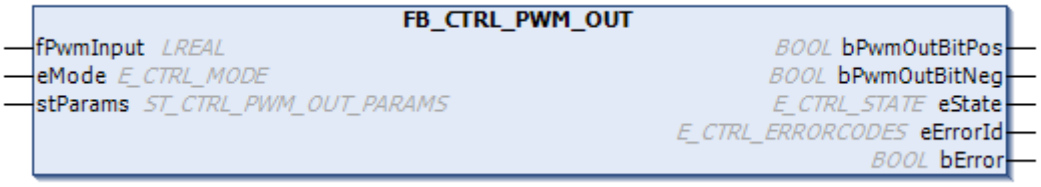
Name	Typ	Beschreibung
tMinOnTime	TIME	Minimale Einschaltdauer des PWM-Ausgangs
tMinOffTime	TIME	Minimale Ausschaltdauer des PWM-Ausgangs
tMinWaitTime	TIME	Wartezeit zwischen den Umschaltvorgängen zwischen einem positiven und negativen Ausgangsignal

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS :
STRUCT
    bPwmOutBitPos    : BOOL;
    bPwmOutBitNeg    : BOOL;
    bWaitTimeActive  : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bPwmOutBit Pos	BOOL	PWM-Signal, wenn fPwmInput > 0.0
bPwmOutBit Neg	BOOL	PWM-Signal, wenn fPwmInput < 0.0
bWaitTime Active	BOOL	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
Internal structure. This must not be written to.
END_STRUCT
END_TYPE
```

4.2.1.7.4 FB_CTRL_PWM_OUT

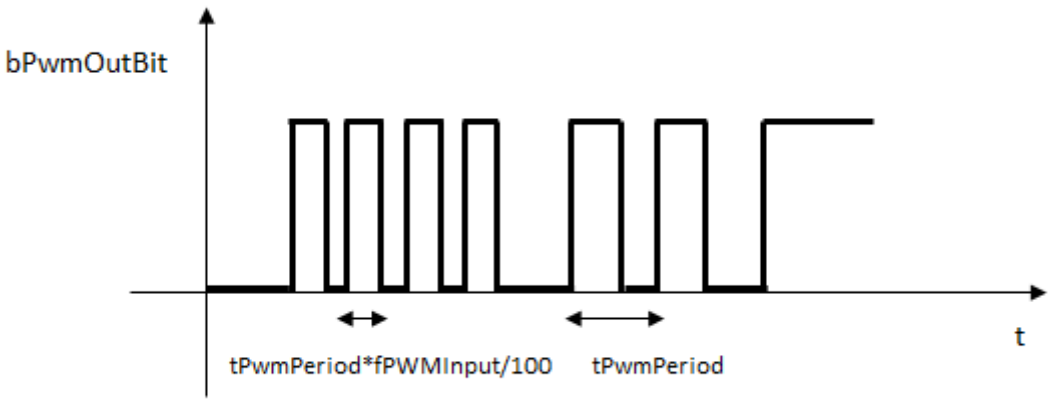


Dieser Baustein erzeugt aus dem Eingangssignal ein PWM-moduliertes Signal.

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-Zyklus.



VAR_INPUT

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

Name	Typ	Beschreibung
fPwmInput	FLOAT	Eingangsgröße
eMode	E_CTRL_MODE	Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
bPwmOutBitPos	BOOL	PWM-Signal, wenn fPwmInput > 0.0 ist.
bPwmOutBitNeg	BOOL	PWM-Signal, wenn fPwmInput < 0.0 ist.
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
eErrorId	E_CTRL_ERRORCODES	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams     : ST_CTRL_PWM_OUT_PARAMS;
END_VAR
```

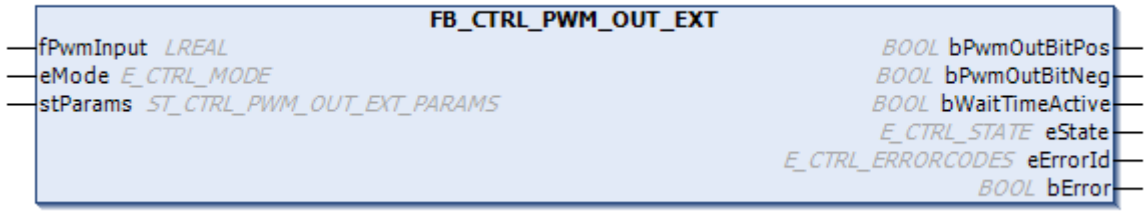
Name	Typ	Beschreibung
stParams	ST_CTRL_PWM_OUT_PARAMS	Parameterstruktur des PWM-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_PWM_OUT_PARAMS:
STRUCT
    tTaskCycleTime      : TIME
    tPWMPeriod          : TIME;
    bInstantPWMUpdate   : BOOL;
END_STRUCT
END_TYPE
```

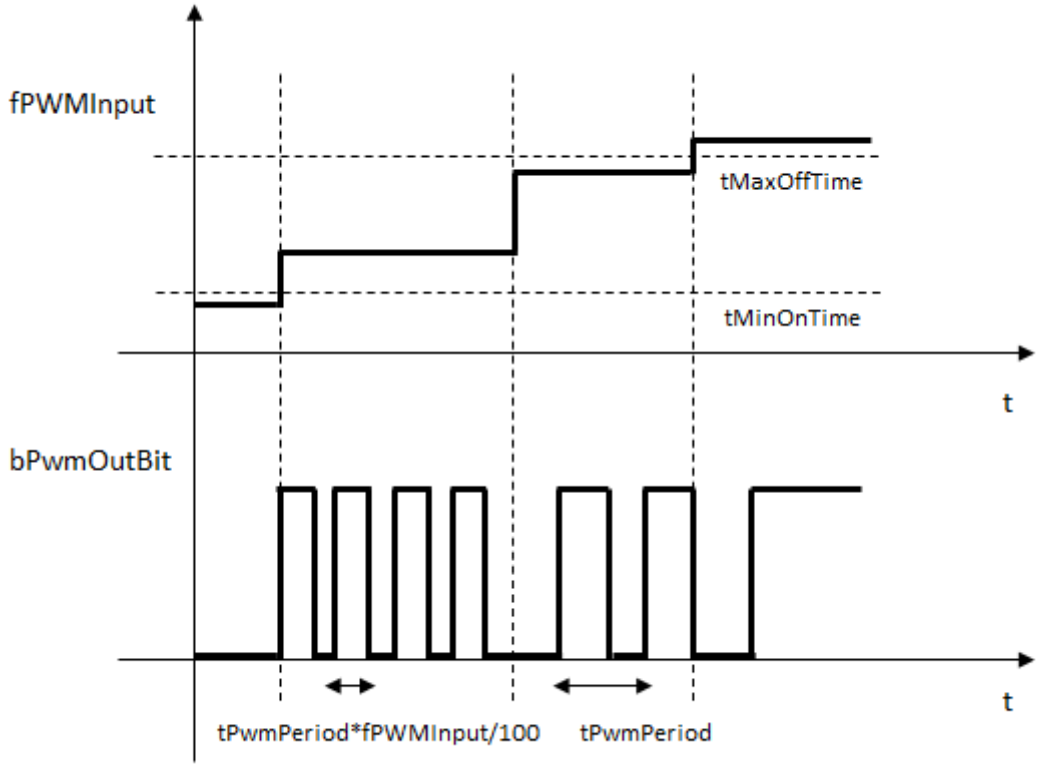
Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tPWMPeriod	TIME	Periodendauer des PWM-Signals
bInstantPWMUpdate	BOOL	Wenn dieses Flag TRUE ist, wird eine neue Eingangsgröße sofort im aktuellen PWM-Zyklus übernommen.

4.2.1.7.5 FB_CTRL_PWM_OUT_EXT

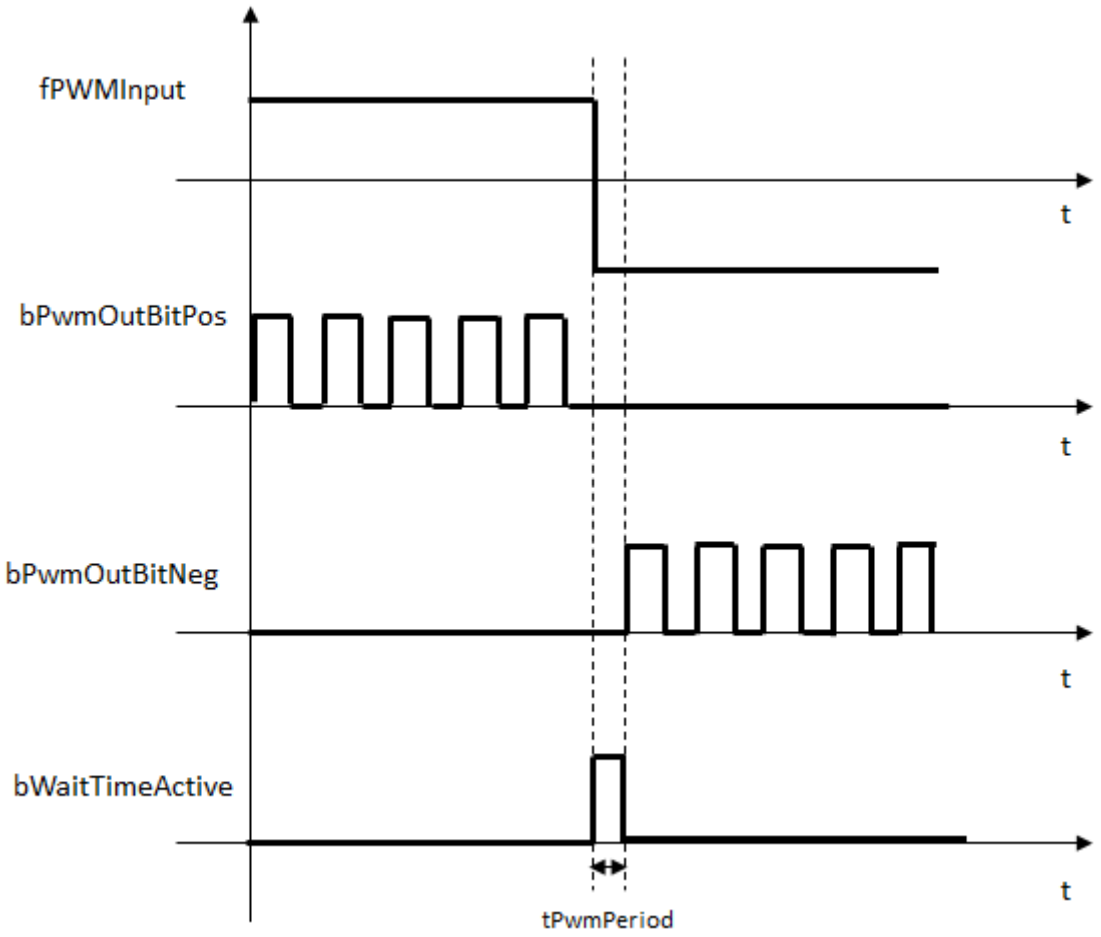


Dieser Baustein erzeugt aus dem Eingangssignal ein PWM-moduliertes Signal. Bei diesem erweiterten Baustein können neben dem Puls-Pausen-Verhältnis auch die minimale Einschaltdauer und die minimale Ausschaltdauer parametriert werden.

Beschreibung des Ausgangsverhaltens 1



Beschreibung des Ausgangsverhaltens 2



VAR_INPUT

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

Name	Typ	Beschreibung
fPwmInput	FLOAT	Eingangsgröße des Funktionsbausteins
eMode	E_CTRL_MODE	Eingang, der die Betriebsart des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
bPwmOutBitPos	BOOL	PWM-Signal, wenn fPwmInput > 0.0
bPwmOutBitNeg	BOOL	PWM-Signal, wenn fPwmInput < 0.0
eState	E_CTRL_STATE	State des Funktionsbausteins
bError	BOOL	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
eErrorId	E_CTRL_ERROR CODES	Wird TRUE, sobald ein Fehler eintritt.

bWaitTimeActive?

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

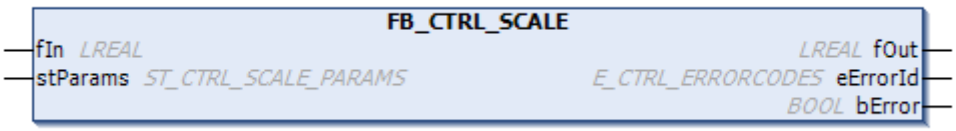
Name	Typ	Beschreibung
stParams	ST_CTRL_PWM_ OUT_EXT_ PARAMS	Parameterstruktur des PWM-Glieds

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_PWM_OUT_EXT_PARAMS :
STRUCT
    tTaskCycleTime  : TIME;
    tPWMPeriod      : TIME;
    tMinOnTime      : TIME;
    tMinOffTime     : TIME;
    tMinWaitTime    : TIME;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tPWMPeriod	TIME	Periodendauer des PWM-Signals
tMinOnTime	TIME	Minimale Einschaltdauer
tMinOffTime	TIME	Minimale Ausschaltdauer
tMinWaitTime	TIME	Wartezeit zwischen den Umschaltvorgängen zwischen einem positiven und negativen Ausgangsignal

4.2.1.7.6 FB_CTRL_SCALE



Mit diesem Funktionsbaustein ist es möglich, eine lineare Signalanpassung zwischen zwei Wertebereichen vorzunehmen.

VAR_INPUT

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Funktionsbausteins

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Skalierte Ausgangsgröße
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_SCALE_PARAMS;
END_VAR
```

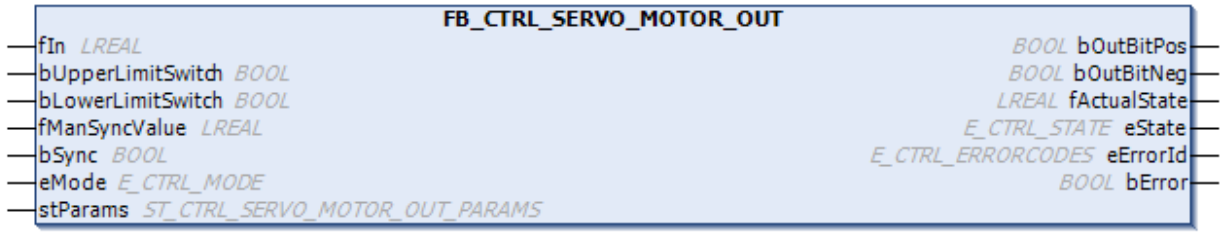
Name	Typ	Beschreibung
stParams	ST_CTRL_SCALE_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_SCALE_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fInMin          : FLOAT;
    fInMax          : FLOAT;
    fOutMin         : FLOAT;
    fOutMax         : FLOAT;
END_STRUCT
END_TYPE
```

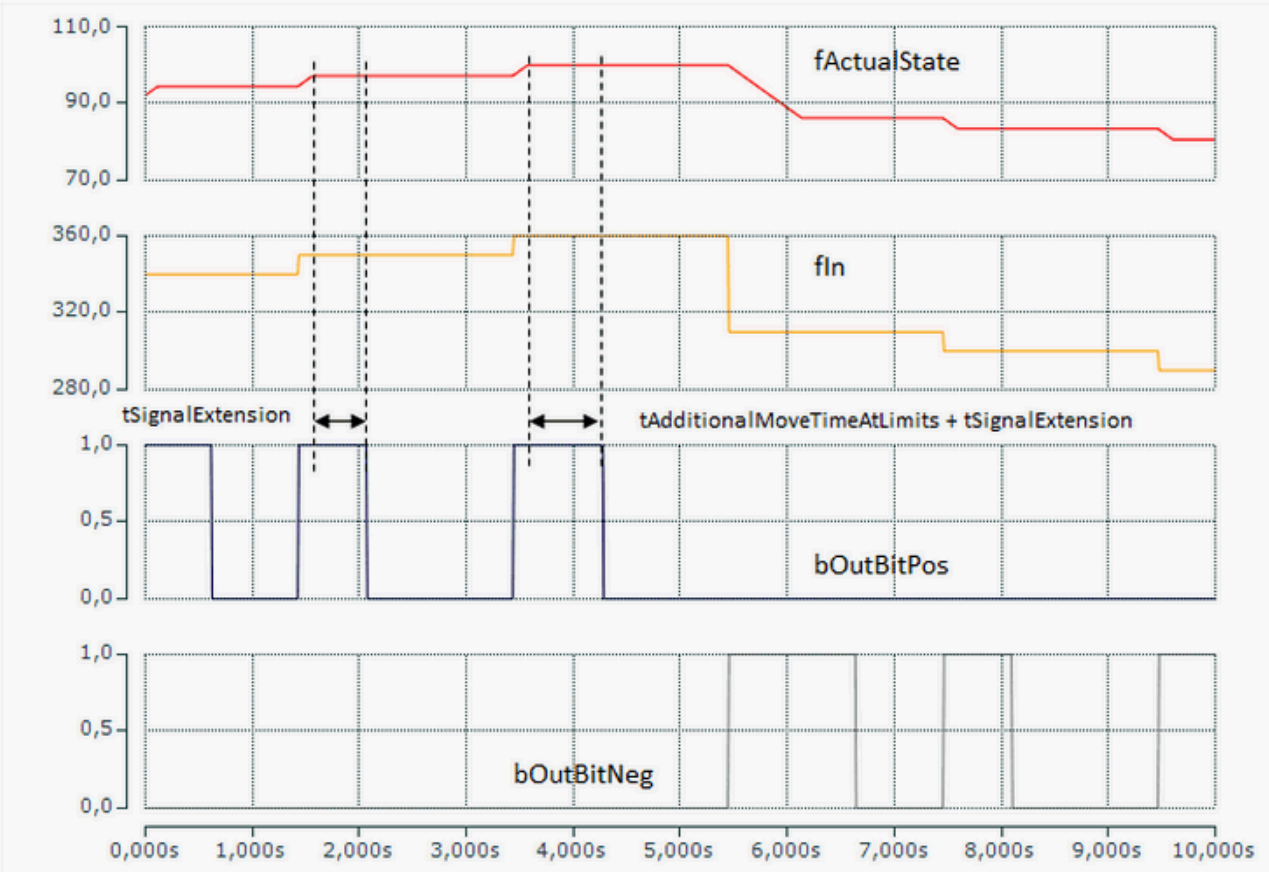
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
fInMin	FLOAT	Minimum der Eingangsgröße
fInMax	FLOAT	Maximum der Eingangsgröße
fOutMin	FLOAT	Minimum der Ausgangsgröße
fOutMax	FLOAT	Maximum der Ausgangsgröße

4.2.1.7.7 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;
    bUpperLimitSwitch   : BOOL;
    bLowerLimitSwitch   : BOOL;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

Name	Typ	Beschreibung
fIn	FLOAT	Stellgröße des Reglers im Intervall [fCtrlOutMin ... fCtrlOutMax] (Reglerausgang).
bUpperLimit Switch	BOOL	Endschalter: TRUE, wenn der obere Anschlag erreicht ist.
bLowerLimit Switch	BOOL	Endschalter: TRUE, wenn der untere Anschlag erreicht ist.
fManSyncValue	FLOAT	Eingang, mit dem der interne Zustand der aktuellen Motorstellung angepasst werden kann, oder auf dessen Wert im Manual-Mode gefahren wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird die interne Motorposition, die der tatsächlichen Motorposition entsprechen muss, auf den Wert „fManSyncValue“ gesetzt.
eMode	E_CTRL_ MODE	Eingang, der die Betriebsart [▶ 174] 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 solange 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;
    bOutBitNeg      : BOOL;
    fActualState    : FLOAT;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bOutBitPos	BOOL	Ausgang, um den Motor in positiver Richtung zu verfahren.
bOutBitNeg	BOOL	Ausgang, um den Motor in negativer Richtung zu verfahren.
fActualState	FLOAT	Aktuelle Motorstellung im Intervall [fCtrlOutMin ... fCtrlOutMax], in dem sich der Motor aktuell befindet.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_SERVO_MOTOR_OUT_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_ SERVO_MOTOR_ OUT_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_SERVO_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime                          : TIME := T#0ms;
    tTaskCycleTime                          : TIME := T#0ms;
    tMovingTime                             : TIME;
```

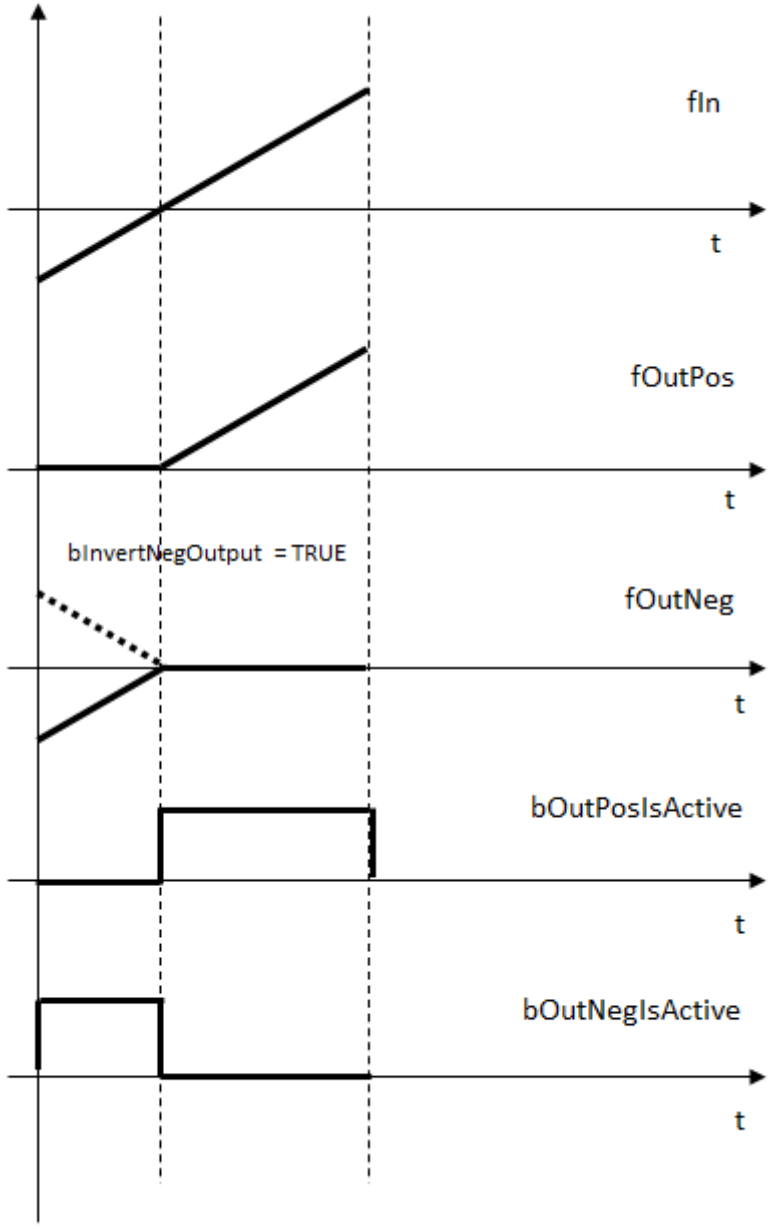
```
    tSignalExtension                              : TIME;
    tAdditionalMoveTimeAtLimits                   : TIME;
    tMinWaitTimeBetweenDirectionChange            : TIME;
    tMinimumPulseTime                             : TIME;
    bMoveOnLimitSwitch                            : BOOL;
    bStopAdditionalMoveTimeIfInputValueIsChanged : BOOL;
    fCtrlOutMax                                   : FLOAT := 100.0;
    fCtrlOutMin                                   : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tMovingTime	TIME	Die Zeit, die benötigt wird, um das Stellglied von einem Anschlag zum anderen zu fahren.
tSignalExtension	TIME	Signalverlängerung, um die jeder Ausgangsimpuls verlängert wird, um eine Totzeit zu kompensieren.
tAdditionalMoveTimeAtLimits	TIME	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	TIME	Minimale Wartezeit zwischen positiven und negativen Ausgangsimpulsen
tMinimumPulseTime	TIME	Minimale Länge eines Ausgangsimpulses
bMoveOnLimitSwitch	BOOL	Wenn TRUE, wird bei einer Stellgröße von `fCtrlOutMin` oder `fCtrlOutMax` solange ein Signal ausgegeben, bis der entsprechende Endschalter erreicht wird.
bStopAdditionalMoveTimeIfInputValueIsChanged	BOOL	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	FLOAT	Stellgröße, bei der das Ventil auf 100% gefahren wird.
fCtrlOutMin	FLOAT	Stellgröße, bei der das Ventil auf 0% gefahren wird.

4.2.1.7.8 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
```

Name	Typ	Beschreibung
fIn	FLOAT	Eingangsgröße des Funktionsbausteins

VAR_OUTPUT

```
VAR_OUTPUT
    fOutPos          : FLOAT;
    fOutNeg          : FLOAT;
    bOutPosIsActive  : BOOL;
    bOutNegIsActive  : BOOL;
    eErrorId         : E_CTRL_ERRORCODES;
    bError           : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fOutPos	FLOAT	Positiver Teil von `fIn`
fOutNeg	FLOAT	Negativer Teil von `fIn`
bOutPosIsActive	BOOL	Ein TRUE signalisiert, das „fIn > 0.0“ ist
bOutNegIsActive	BOOL	Ein TRUE signalisiert, das „fIn < 0.0“ ist
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams        : ST_CTRL_SPLITRANGE_PARAMS;
END_VAR
```

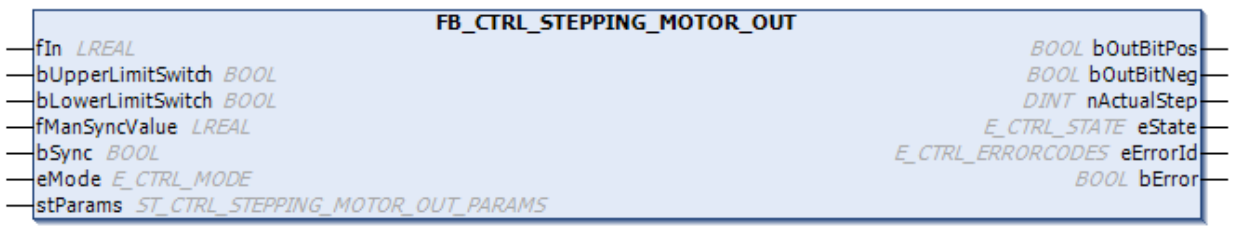
Name	Typ	Beschreibung
stParams	ST_CTRL_SPLITRANGE_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_SPLITRANGE_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    bInvertNegOutput  : BOOL;
    bDisablePosOut    : BOOL;
    bDisableNegOut    : BOOL;
END_STRUCT
END_TYPE
```

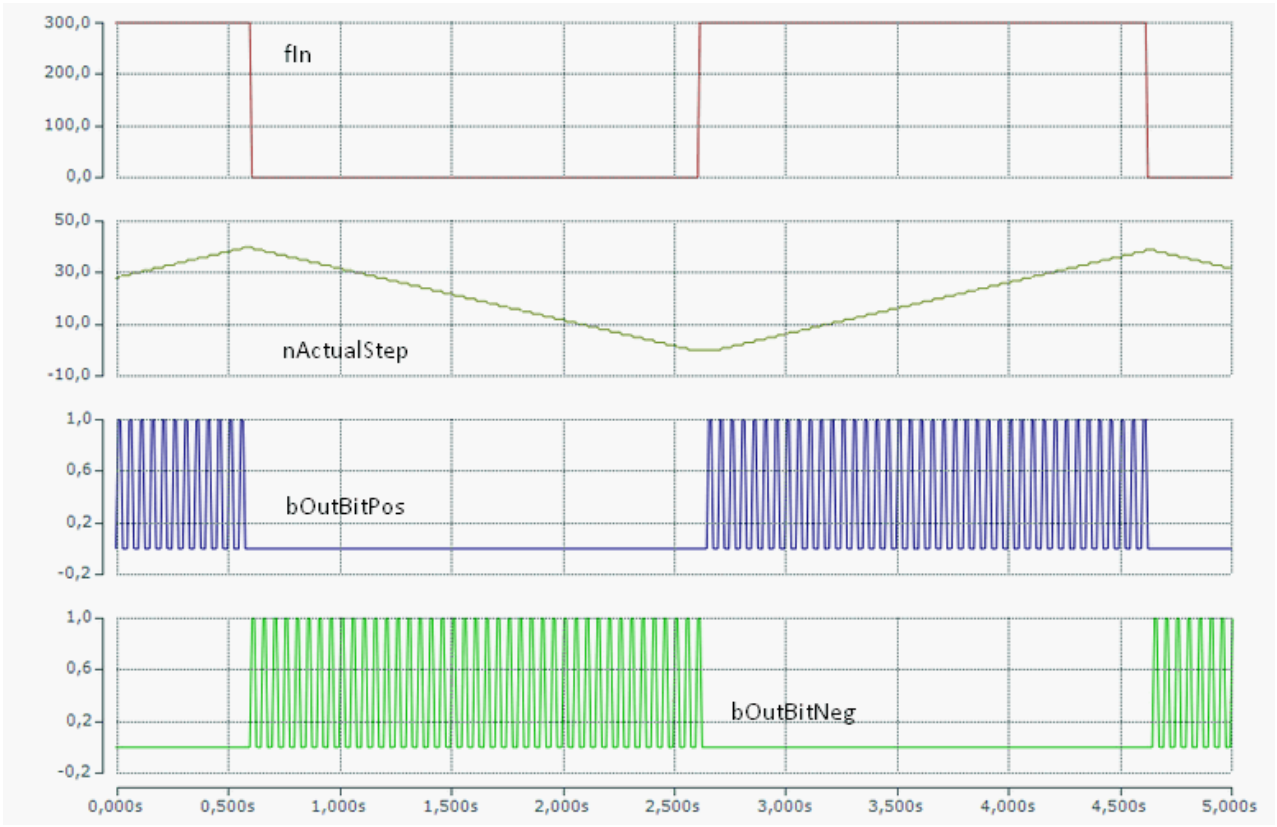
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen.
tTaskCycleTime	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
bInvertNegOutput	BOOL	Wenn dieser Parameter TRUE ist, wird `fOutNeg` invertiert.
bDisablePosOut	BOOL	Der Ausgang `fOutPos` wird deaktiviert und ist immer „0.0“.
bDisableNegOut	BOOL	Der Ausgang `fOutNeg` wird deaktiviert und ist immer „0.0“.

4.2.1.7.9 FB_CTRL_STEPPING_MOTOR_OUT



Dieser Funktionsbaustein erzeugt eine Stellgröße für einen Schrittmotor.

Verhalten des Ausgangs



VAR_INPUT

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

Name	Typ	Beschreibung
fIn	FLOAT	Stellgröße des Reglers (Reglerausgang)
bUpperLimit Switch	BOOL	Endschalter, wird TRUE, wenn der obere Anschlag erreicht ist.
bLowerLimit Switch	BOOL	Endschalter, wird TRUE, wenn der untere Anschlag erreicht ist.
fManSyncValue	FLOAT	Eingang, mit dem der interne Zustand der Motorstellung angepasst werden kann oder auf dessen Wert im Manual-Mode gefahren wird.
bSync	BOOL	Mit einer steigenden Flanke an diesem Eingang wird der interne Schrittzähler auf den Schritt gesetzt, der dem Wert „fManSyncValue“ entspricht.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

Dieser Baustein besitzt zur Synchronisation mit der aktuellen Ventilstellung den Mode **eCTRL_MODE_SYNC_MOVEMENT**. In diesem Mode werden solange 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;
    bOutBitNeg    : BOOL;
    nActualStep   : DINT;
    eState        : E_CTRL_STATE;
    eErrorId      : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bOutBitPos	BOOL	Ausgang, um den Motor in positiver Richtung zu verfahren.
bOutBitNeg	BOOL	Ausgang, um den Motor in negativer Richtung zu verfahren.
nActualStep	DINT	Aktueller Schritt, in dem der Motor steht.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERROR CODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_STEPPING_MOTOR_OUT_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_STEPP ING_MOTOR_ OUT_PARAMS	Parameterstruktur des Funktionsbausteins

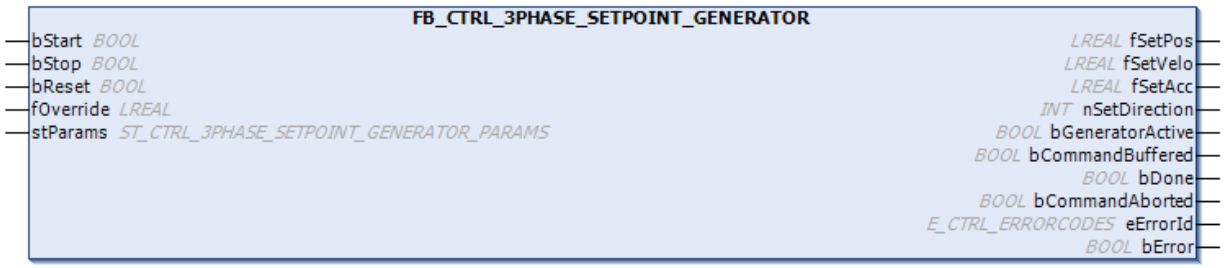
`stParams` besteht aus den folgenden Elementen:

```
TYPE
ST_CTRL_STEPPING_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime             : TIME := T#0ms;
    tTaskCycleTime             : TIME := T#0ms;
    tOnTime                    : TIME;
    tOffTime                   : TIME;
    nMaxMovingPulses           : DINT;
    nMinMovingPulses           : UINT := 0;
    bHoldWithOutputOn          : BOOL;
    nAdditionalPulsesAtLimits  : DINT;
    bMoveOnLimitSwitch         : BOOL;
    fCtrlOutMax                : FLOAT := 100.0;
    fCtrlOutMin                : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.
tOnTime	TIME	Impulslänge
tOffTime	TIME	Pausenlänge
nMaxMoving Pulses	DINT	Max. Anzahl Impulse, die benötigt werden, um von einem Limit zum anderen zu fahren.
nMinMoving Pulses	UINT	Min. Anzahl Impulse, die benötigt werden, um von einem Limit zum anderen zu fahren.
bHoldWith OutputOn	BOOL	Wenn dieser Parameter auf TRUE gesetzt ist, bleibt im Stillstand des Antriebs ein Ausgang gesetzt. Dieser wird somit gebremst.
nAdditionalPulse sAtLimits	DINT	Impulse, die zum sicheren Erreichen der Grenzen zusätzlich ausgegeben werden.
bMoveOnLimit Switch	BOOL	Wenn TRUE, werden bei 0% oder 100% Stellgröße solange Impulse ausgegeben, bis ein Endschalter erreicht wird.
fCtrlOutMax	FLOAT	Stellgröße, bei der das Ventil auf 100% gefahren wird.
fCtrlOutMin	FLOAT	Stellgröße, bei der das Ventil auf 0% gefahren wird.

4.2.1.8 Setpointgeneration

4.2.1.8.1 FB_CTRL_3PHASE_SETPOINT_GENERATOR



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`

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

Überfahren der Endposition

Bei Vorgabe eines neuen Parametersatzes ist es möglich, dass die alte Endposition überfahren wird. Siehe Beispiel.
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
```

Name	Typ	Beschreibung
bStart	BOOL	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	BOOL	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	BOOL	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	LREAL	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;
    fSetVelo          : LREAL;
    fSetAcc           : LREAL;
    nSetDirection     : INT;
    bCommandBuffered  : BOOL;
    bDone             : BOOL;
    bCommandAborted   : BOOL;
    eErrorId          : E_CTRL_ERRORCODES;
    bError            : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fSetPos	LREAL	Sollposition
fSetVelo	LREAL	Sollgeschwindigkeit
fSetAcc	LREAL	Sollbeschleunigung
nSetDirection	INT	Bewegungsrichtung [-1, 0, 1], 1 --> Bewegungsrichtung positiv 0 --> Generator inaktiv 1 --> Bewegungsrichtung negativ

bCommandBuffered	BOOL	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	BOOL	Dieser Ausgang wird TRUE, wenn die Bewegung abgeschlossen ist und die Zielposition erreicht wurde.
bCommandAborted	BOOL	Dieser Ausgang wird TRUE, wenn die aktuelle Bewegung abgebrochen wurde. Dieses kann beispielsweise durch eine steigende Flanke am Eingang `bStop` verursacht werden.
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams : ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS	Parameterstruktur des Sollwertgenerators

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime       : TIME;
    tCtrlCycleTime       : 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;
EN_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	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.
tCtrlCycleTime	TIME	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	LREAL	Startposition des Bewegungsprofils
fStartVelo	LREAL	Startgeschwindigkeit des Bewegungsprofils
fVelocity	LREAL	Geschwindigkeit in Einheiten / Sekunde
fTargetPos	LREAL	Zielposition des Bewegungsprofils
fTargetVelo	LREAL	Zielgeschwindigkeit des Bewegungsprofils. **Hinweis:** Die Zielgeschwindigkeit bleibt nach dem Erreichen der Zielposition (setzen des *bDone* Flags) bestehen, die Position wird aber ab diesem Zeitpunkt nicht weiter berechnet (konstante Position bei einer Geschwindigkeit ≠ 0.0).
fAcceleration	LREAL	Beschleunigung in Einheiten / Sekunde^2
fDeceleration	LREAL	Verzögerung in Einheiten / Sekunde^2

eNewParameterType	E_CTRL_NEW_PARAMETER_TYPE	`eCTRL_NEW_PARAMETER_TYPE_Instant`
		und
		`eCTRL_NEW_PARAMETER_TYPE_NotInstant`

Besonderheiten für eNewParameterType

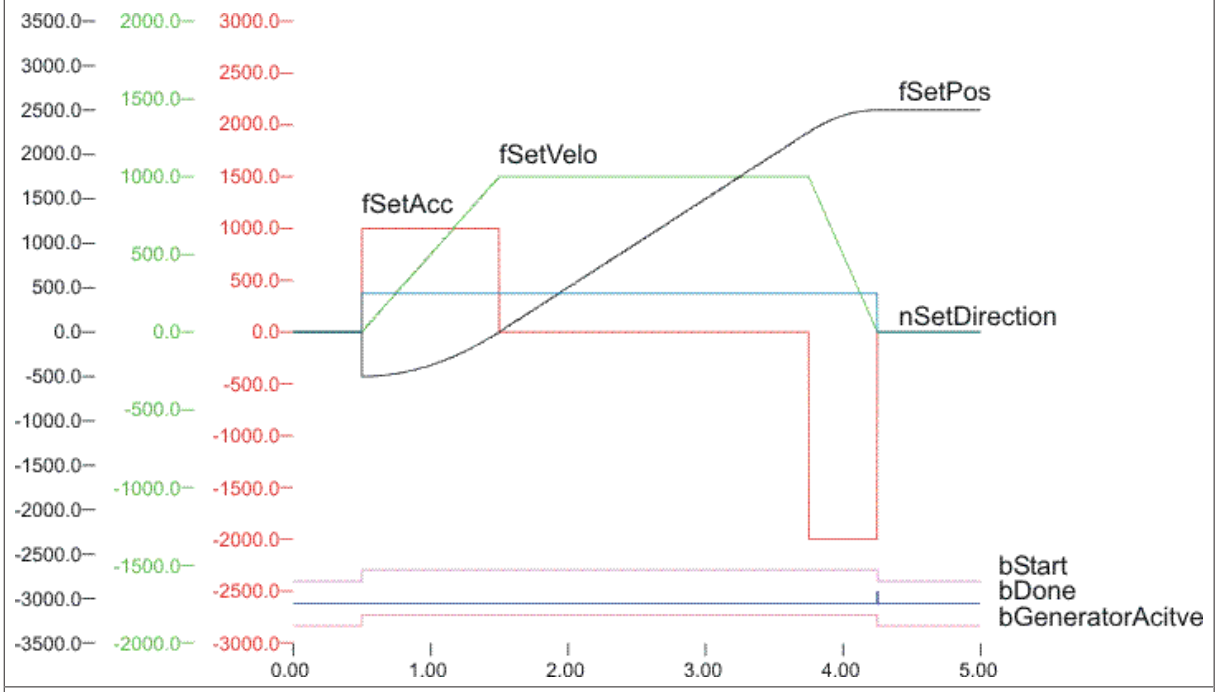
```
TYPE E_CTRL_NEW_PARAMETER_TYPE :
(    eCTRL_NEW_PARAMETER_TYPE_NotInstant := 0,
    eCTRL_NEW_PARAMETER_TYPE_Instant := 1);
END_TYPE
```

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.

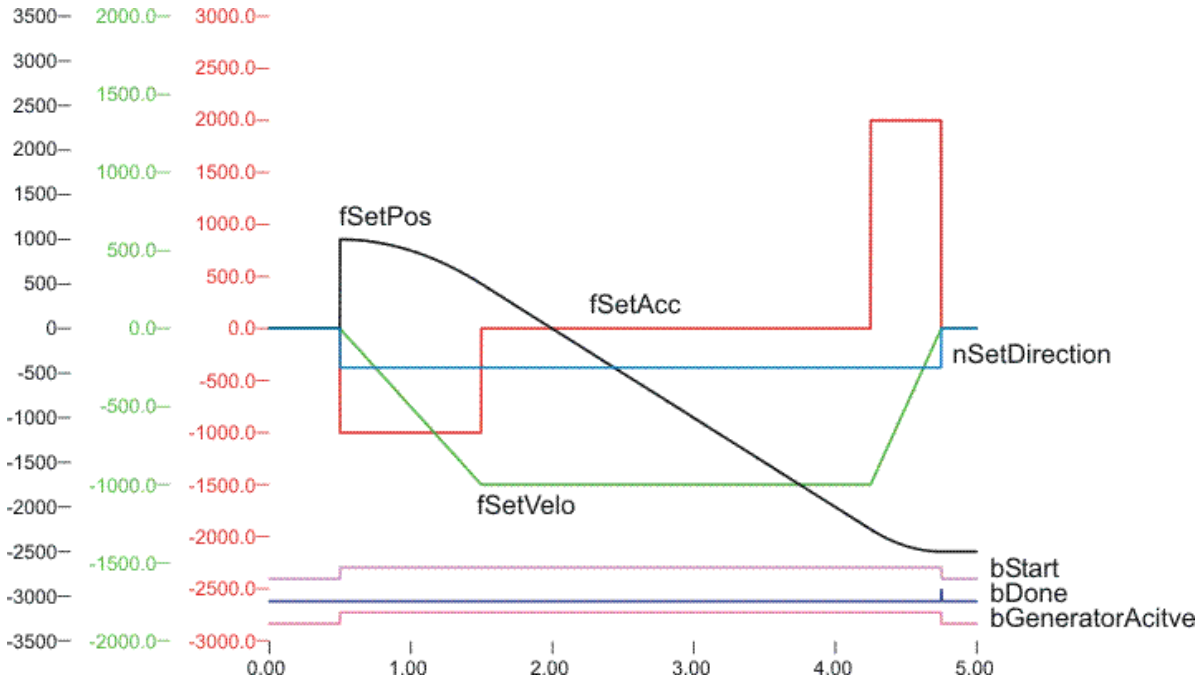
eCTRL_NEW_PARAMETER_TYPE_NotInstant: Wenn ein Nachstartauftrag mit einem neuen Parametersatz erfolgt, wird dieser nicht sofort übernommen. D. h., die aktuelle Bewegung wird erst zu Ende ausgeführt und danach wird mit den neuen Parametern auf das neue Ziel positioniert. Ein TRUE an dem Ausgang **bCommandBuffered** 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

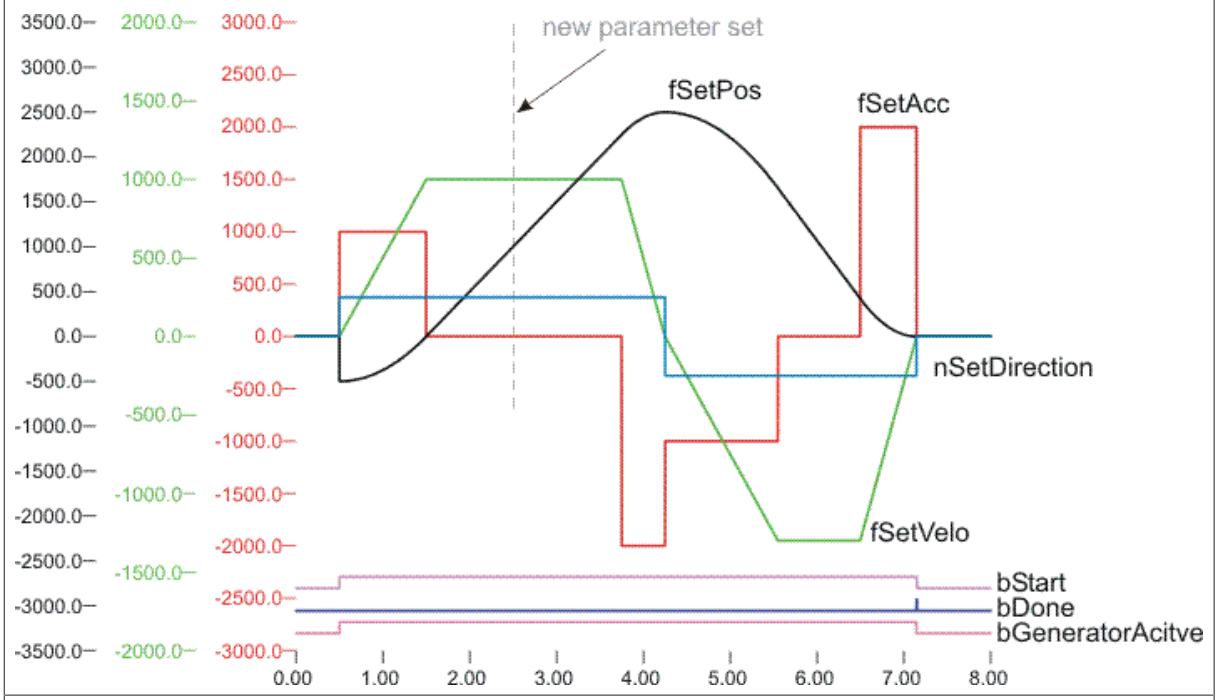
Beispielpositionierung 1:	stParams.fStartPos := -500.0;
	stParams.fTargetPos := 2500.0;
	stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;
	stParams.fAcceleration := 1000.0;
	stParams.fDeceleration := 2000.0; fOverride := 100.0;



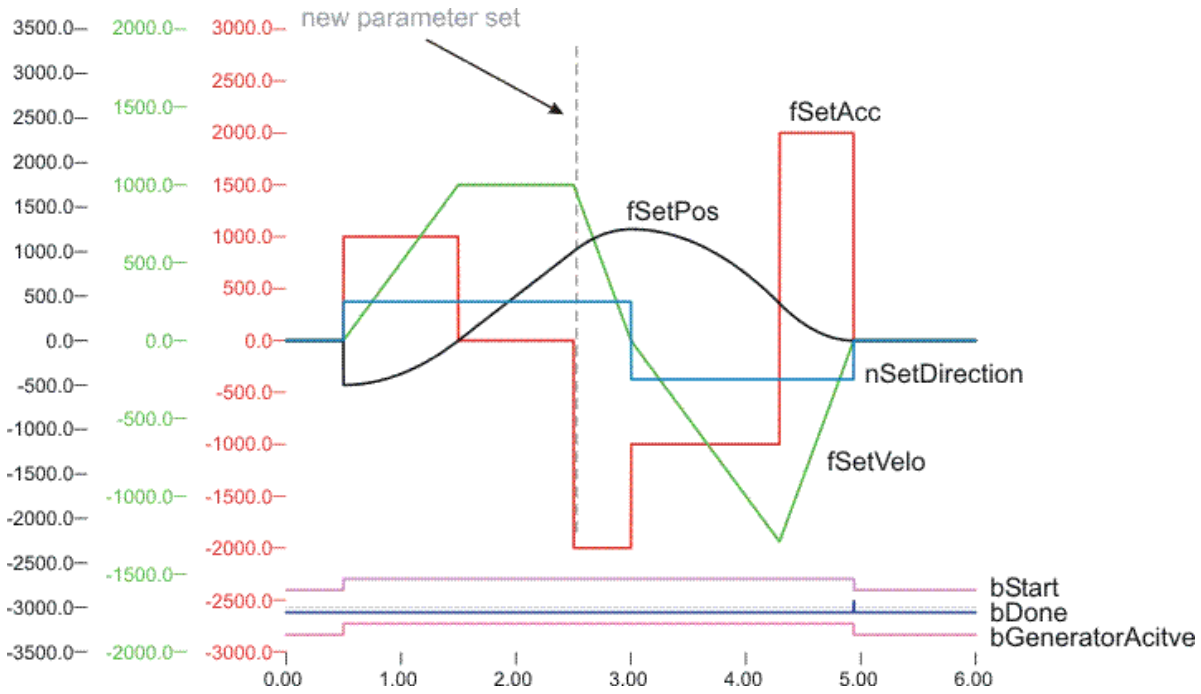
Beispielpositionierung 2:	stParams.fStartPos := 1000.0;
	stParams.fTargetPos := -2500.0;
	stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;
	stParams.fAcceleration := 1000.0;
	stParams.fDeceleration := 2000.0; fOverride := 100.0;



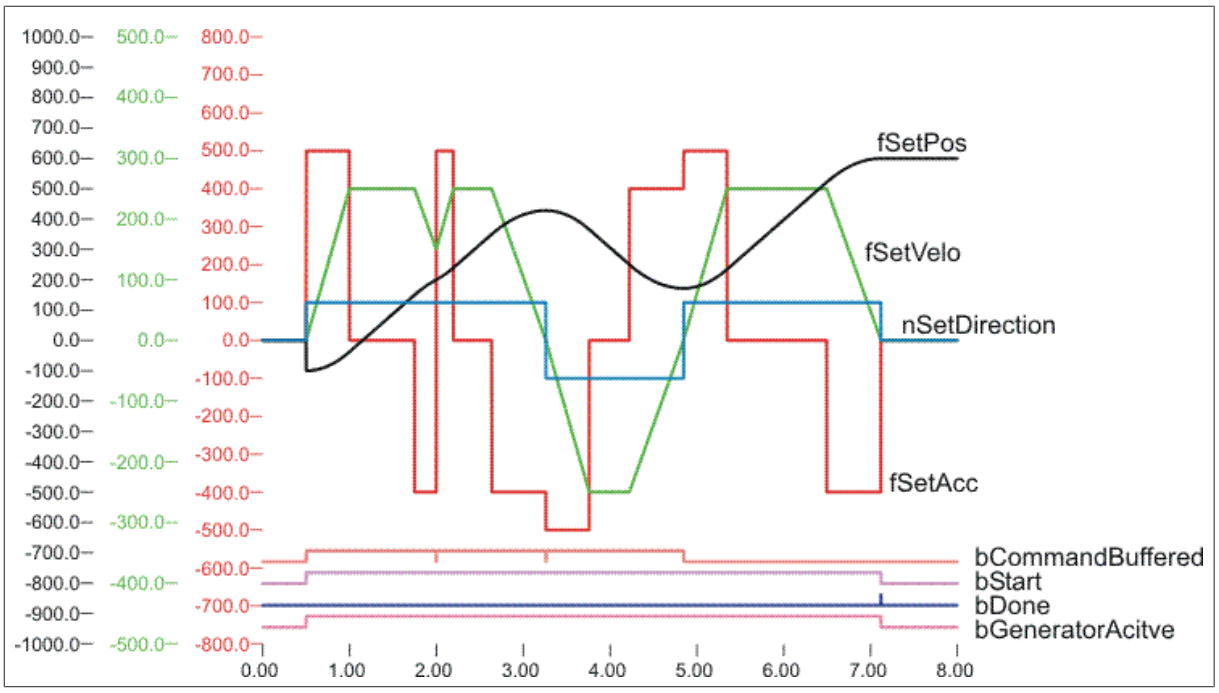
Beispielpositionierung 3:	stParams.fStartPos := -500.0;
	stParams.fTargetPos := 2500.0;
	stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;
	stParams.fAcceleration := 1000.0;
	stParams.fDeceleration := 2000.0;
	stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;
	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**Nach starten 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

Überfahren der Zielposition!

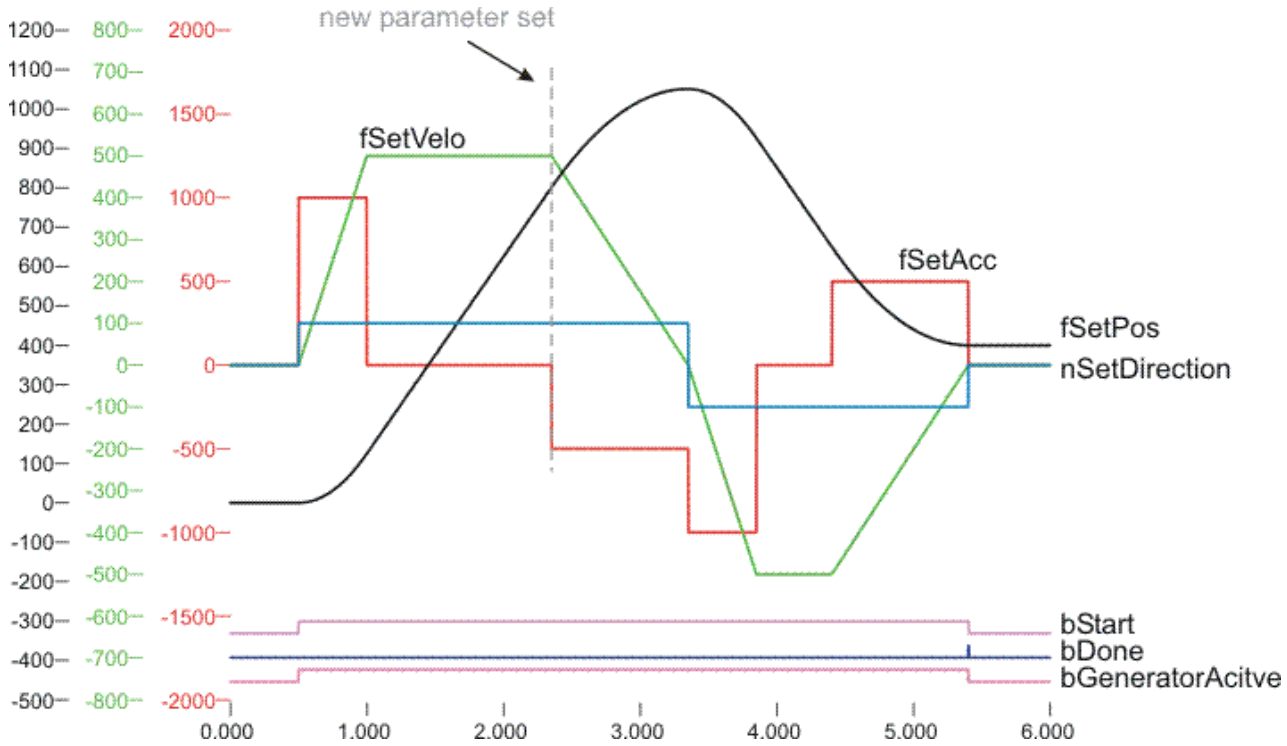
Wenn ein neuer Parametersatz vom Typ "eCTRL_NEW_POS_TYPE_Instant" an den Baustein übergeben wird, in dem die Verzögerung verringert wird, ist es möglich, dass die alte Zielposition überfahren wird.

Beispiel:

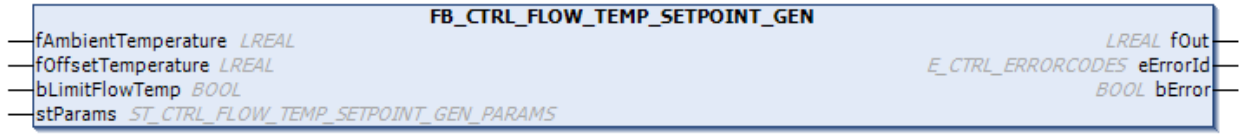
```
stParams.fTargetPos := 1000.0;
stParams.fStartPos := 0.0;
stParams.fVelocity := 500.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 1000.0;
IF fSetPos > 800.0 THEN
stParams.fTargetPos := 400.0;
stParams.fVelocity := 500.0;
stParams.fAcceleration := 1_000.0;
stParams.fDeceleration := 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 zurückzuführen ist, dass die Verzögerung in dem neuen Parametersatz

verringert wird.



4.2.1.8.2 FB_CTRL_FLOW_TEMP_SETPOINT_GEN

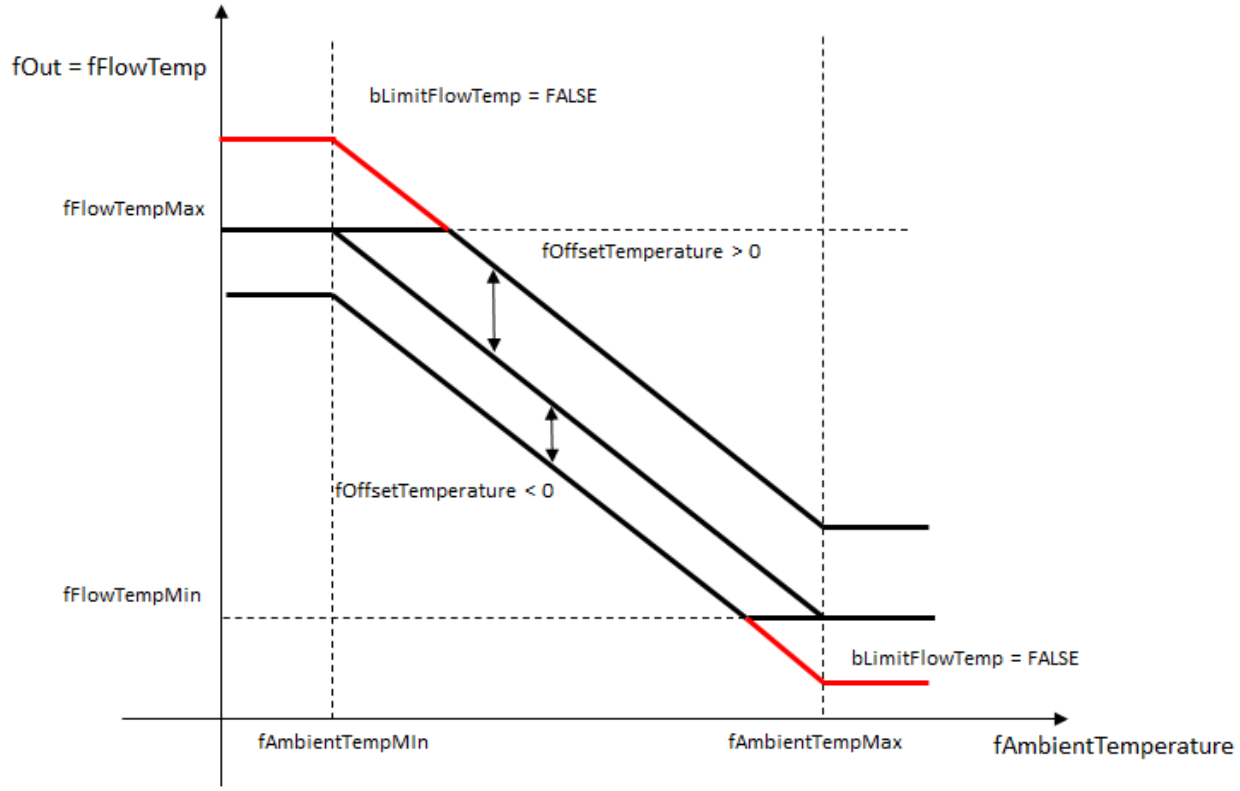


Der Funktionsbaustein ermöglicht die Vorgabe einer Vorlauftemperatur in Abhängigkeit der Außentemperatur.

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 eine Nachtabsenkung oder eine Vorsteuerung durchgeführt werden.

Verhalten der Ausgangsgröße



VAR_INPUT

```
VAR_INPUT
    fAmbientTemperature   : FLOAT;
    fOffsetTemperature    : FLOAT;
    bLimitFlowTemp        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fAmbient Temperature	FLOAT	Start der Rampengenerierung
fOffset Temperature	FLOAT	Startwert der Rampe
bLimitFlowTemp	BOOL	Zielwert der Rampe

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Sollwert der Vorlauftemperatur
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS;
END_VAR
```

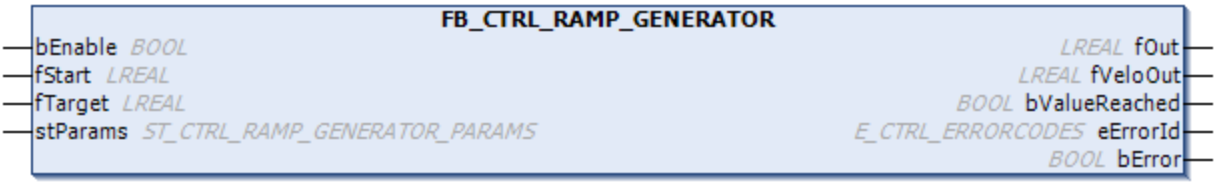
Name	Typ	Beschreibung
stParams	ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS	Parameterstruktur des Rampengenerators

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS:
STRUCT
    tTaskCycleTime    : TIME;
    tCtrlCycleTime    : TIME;
    fForeRunTempMax   : FLOAT;
    fForeRunTempMin   : FLOAT;
    fAmbientTempMax   : FLOAT;
    fAmbientTempMin   : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	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	TIME	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.
fForeRunTempMax	FLOAT	Maximale Vorlauftemperatur (siehe Diagramm)
fForeRunTempMin	FLOAT	Minimale Vorlauftemperatur (siehe Diagramm)
fAmbientTempMax	FLOAT	Außentemperatur, bei der die minimale Vorlauftemperatur vorgegeben wird.
fAmbientTempMin	FLOAT	Außentemperatur, bei der die maximale Vorlauftemperatur vorgegeben wird.

4.2.1.8.3 FB_CTRL_RAMP_GENERATOR

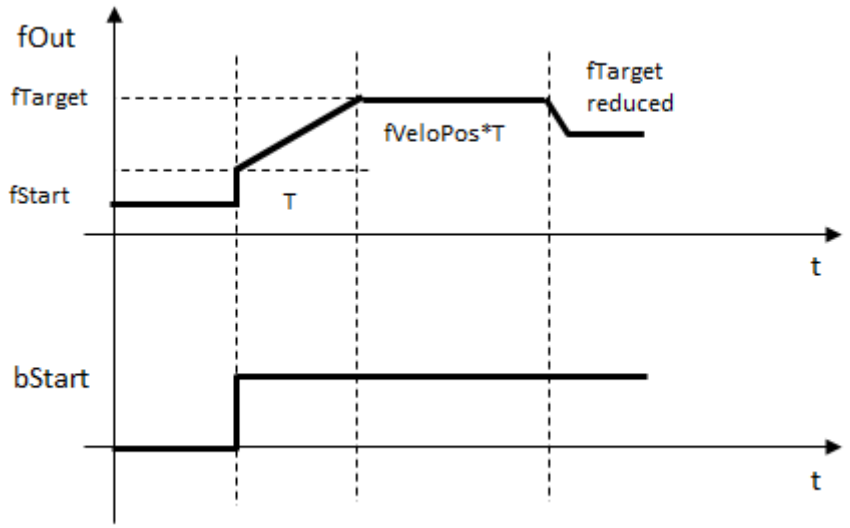


Der Funktionsbaustein stellt einen parametrierbaren Rampengenerator dar.

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;
END_VAR
```

Name	Typ	Beschreibung
bEnable	BOOL	Start der Rampengenerierung
fStart	FLOAT	Startwert der Rampe
fTarget	FLOAT	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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Rampengenerators
fVeloOut	FLOAT	Aktuelle Geschwindigkeit des Rampengenerators
bValueReached	BOOL	Der Ausgang signalisiert mit einem TRUE, das der Ausgang `fOut` den Wert `fTarget` erreicht hat.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams     : ST_CTRL_RAMP_GENERATOR_PARAMS
END_VAR
```

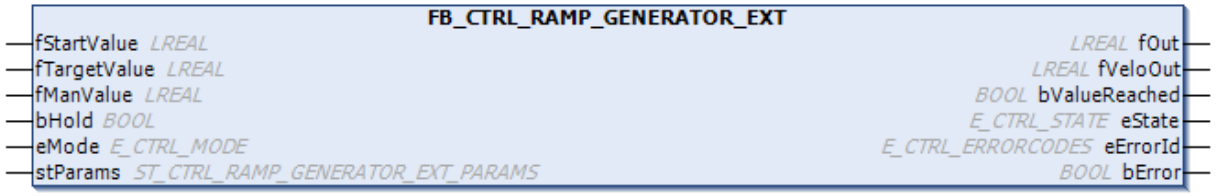
Name	Typ	Beschreibung
stParams	ST_CTRL_RAMP_GENERATOR_PARAMS	Parameterstruktur des Rampengenerators

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime  : TIME;
    tCtrlCycleTime  : TIME;
    fVeloPos        : FLOAT;
    fVeloNeg        : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	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.
tCtrlCycleTime	TIME	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	FLOAT	Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem niedrigeren Wert auf einen höheren überführt wird.
fVeloNeg	FLOAT	Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem höheren Wert auf einen niedrigeren überführt wird.

4.2.1.8.4 FB_CTRL_RAMP_GENERATOR_EXT

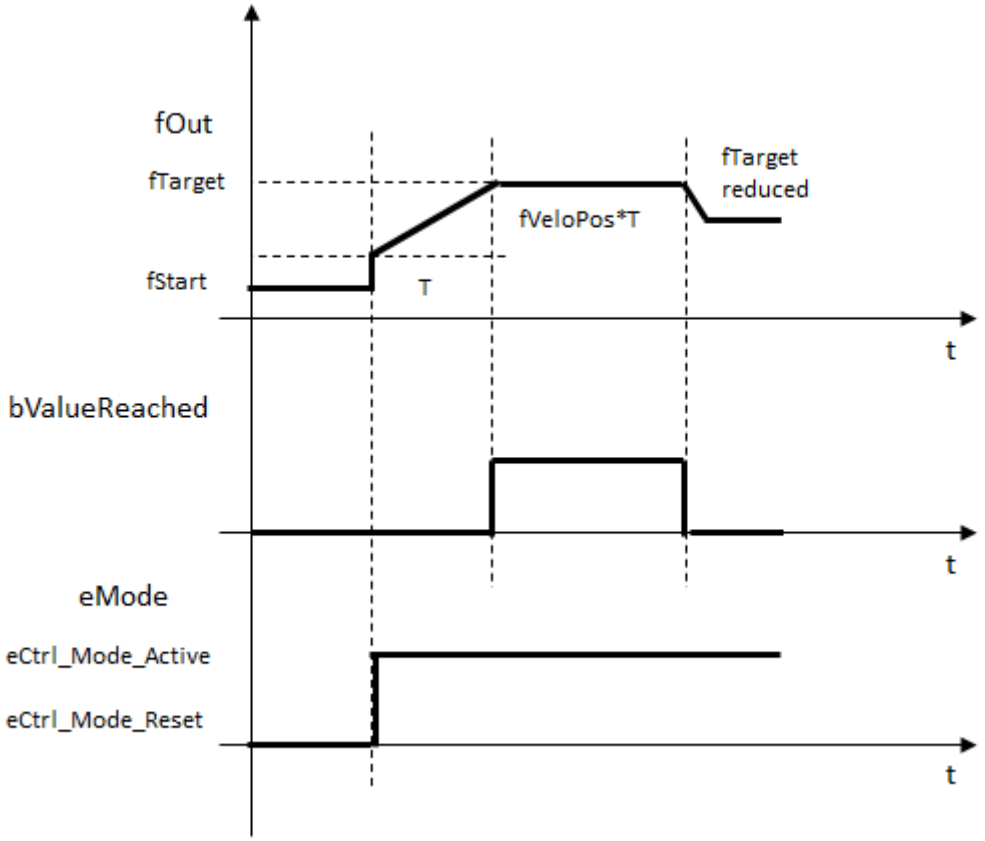


Der Funktionsbaustein stellt einen parametrierbaren Rampengenerator dar, welcher gegenüber dem **FB_CTRL_RAMP_GENERATOR** die Modi **E_CTRL_MODE** unterstützt.

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
```

Name	Typ	Beschreibung
fStartValue	FLOAT	Startwert der Rampe
fTargetValue	FLOAT	Zielwert der Rampe
fManValue	FLOAT	Eingangsgröße, auf die der Ausgang im `eCTRL_MODE_MANUAL` gesetzt wird.
bHold	BOOL	Die Berechnung der Rampe wird auf dem aktuellen Wert angehalten.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] 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
```

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Rampengenerators
fVeloOut	FLOAT	Aktuelle Geschwindigkeit des Rampengenerators
bValueReached	BOOL	Der Ausgang signalisiert mit einem TRUE, dass der Ausgang `fOut` den Wert „fTargetValue“ erreicht hat.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_RAMP_GENERATOR_EXT_PARAMS;
END_VAR
```

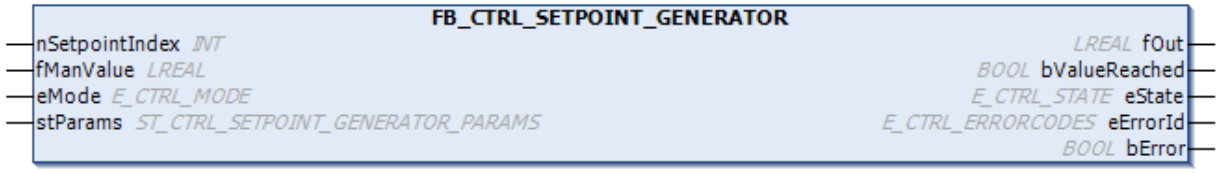
Name	Typ	Beschreibung
stParams	ST_CTRL_RAMP_GENERATOR_EXT_PARAMS	Parameterstruktur des Rampengenerators

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_RAMP_GENERATOR_EXT_PARAMS :
STRUCT
    tTaskCycleTime  : TIME;
    tCtrlCycleTime  : TIME;
    fVeloPos        : FLOAT;
    fVeloNeg        : FLOAT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tTaskCycleTime	TIME	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.
tCtrlCycleTime	TIME	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	FLOAT	Geschwindigkeit (>0.0) in Einheiten / s, mit der der Ausgang von einem niedrigeren Wert auf einen höheren überführt wird.
fVeloNeg	FLOAT	Geschwindigkeit (>0.0) in Einheiten / s, mit der der Ausgang von einem höheren Wert auf einen niedrigeren überführt wird.

4.2.1.8.5 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

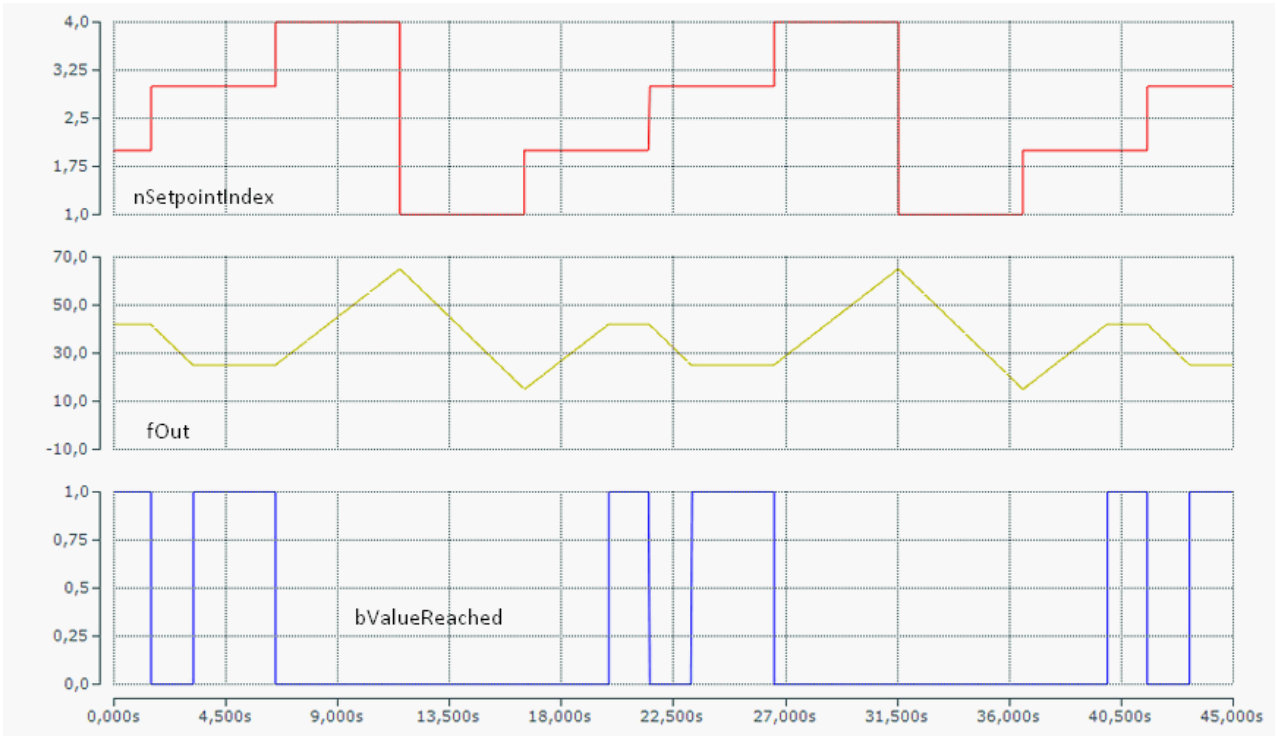


Tabelle des Beispiels:

Index	
1	12
2	42
3	25
…	73

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

Name	Typ	Beschreibung
nSetpointIndex	INT	Index des angewählten Sollwertes
fManValue	FLOAT	Eingang, der im Manual-Mode ausgegeben wird.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] 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
```

Name	Typ	Beschreibung
fOut	SETPOINT_TABLE_ELEMENT	Ausgang des Sollwert-Generators
bValueReached	BOOL	Der Ausgang ist TRUE, wenn der angewählte Sollwert erreicht worden ist, also wenn das Rampen auf den angewählten Wert beendet ist.
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams    : ST_CTRL_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

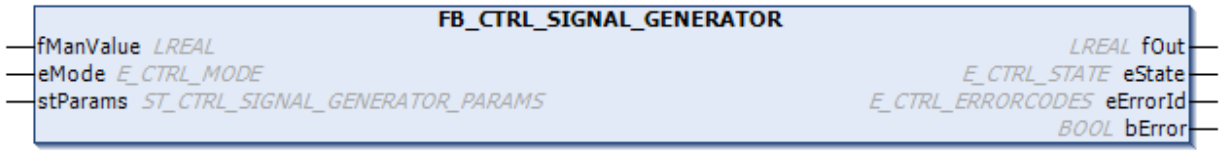
Name	Typ	Beschreibung
stParams	ST_CTRL_SETPOINT_GENERATOR_PARAMS	Parameterstruktur des Rampengenerators

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_SETPOINT_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    pDataTable_ADR          : POINTER TO
     INTERPOLATION_TABLE_ELEMENT := 0;
    nDataTable_SIZEOF       : UINT := 0;
    nDataTable_NumberOfRows : UINT := 0;
    fVeloPos                : FLOAT;
    fVeloNeg                : FLOAT;
    bDisableRamping         : BOOL := FALSE;
END_STRUCT
END_TYPE
```

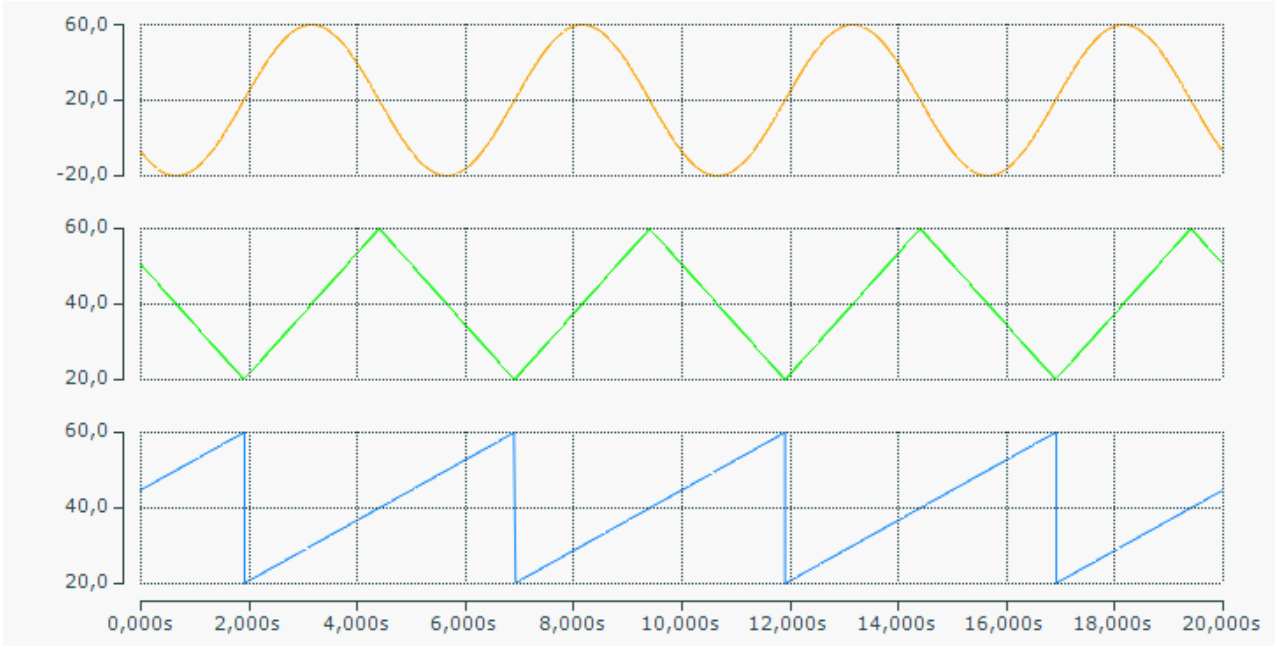
Name	Typ	Beschreibung
tCtrlCycleTime	TIME	Zykluszeit, mit der der Regelkreis bearbeitet wird. Diese muss größer oder gleich der **TaskCycleTime** sein. Der Funktionsbaustein berechnet mit dieser Eingangsgröße intern, ob die Zustands- und Ausgangsgrößen im aktuellen Zyklus aktualisiert werden müssen. Indent: -103; margin-left: 108">
tTaskCycleTime	TIME	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	POINTER TO INTERPOLATION_TABLE_ELEMENT	Adresse des Daten-Arrays
nDataTable_SIZEOF	UINT	Größe des Daten-Arrays
nDataTable_NumberOfRows	UINT	Anzahl der Zeilen des Daten-Arrays
fVeloPos	FLOAT	Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem niedrigeren Wert auf einen höheren überführt wird.
fVeloNeg	FLOAT	Geschwindigkeit in Einheiten / s, mit der der Ausgang von einem höheren Wert auf einen niedrigeren überführt wird.
bDisableRamping	BOOL	Wenn dieser Parameter TRUE ist, wird keine stetige Ausgangsgröße berechnet. Es wird sprungförmig zwischen den Ausgangswerten umgeschaltet.

4.2.1.8.6 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
```

Name	Typ	Beschreibung
fManValue	FLOAT	Eingang, dessen Wert im Manual-Mode am Ausgang anliegt.
eMode	E_CTRL_MODE	Eingang, der die Betriebsart [▶ 174] des Bausteins festlegt.

VAR_OUTPUT

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

Name	Typ	Beschreibung
fOut	FLOAT	Ausgang des Signalgenerators
eState	E_CTRL_STATE	State des Funktionsbausteins
eErrorId	E_CTRL_ERRORCODES	Liefert bei einem gesetzten `bError`-Ausgang die Fehlernummer [▶ 174].
bError	BOOL	Wird TRUE, sobald ein Fehler eintritt.

VAR_IN_OUT

```
VAR_IN_OUT
    stParams     : ST_CTRL_SIGNAL_GENERATOR_PARAMS;
END_VAR
```

Name	Typ	Beschreibung
stParams	ST_CTRL_SIGNAL_GENERATOR_PARAMS	Parameterstruktur des Funktionsbausteins

`stParams` besteht aus den folgenden Elementen:

```
TYPE ST_CTRL_SIGNAL_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    eSignalType     : E_CTRL_SIGNAL_TYPE;
    tCylceDuration  : TIME;
    fAmplitude      : FLOAT;
    fOffset         : FLOAT := 0.0;
    tStart          : TIME := T#0s;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
tCtrlCycleTime	TIME	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	TIME	Zykluszeit, mit der der Funktionsbaustein aufgerufen wird. Diese entspricht der Task-Zykluszeit der aufrufenden Task, wenn der Baustein in jedem Zyklus aufgerufen wird.

eSignalType	E_CTRL_SIGNAL_TYPE	Anwahl der Signalform.
		`TYPE E_CTRL_SIGNAL_TYPE :`
		`( eCTRL_TRIANGLE := 0,`
		`eCTRL_SINUS := 1,`
		`eCTRL_SAWTOOTH := 2 );`
		`END_TYPE`

tCylceDuration	TIME	Periodendauer des erzeugten Signalverlaufs
fAmplitude	FLOAT	Amplitude des erzeugten Signalverlaufs
fOffset	FLOAT	Offset, der auf den Signalverlauf addiert wird
tStart	TIME	Zeitpunkt innerhalb einer Periode, bei dem der Signalverlauf startet, wenn in den **eCTRL_MODE_ACTIVE** geschaltet wird.

4.2.2 Globale Konstanten

4.2.2.1 Version der Bibliothek

Alle Bibliotheken haben eine kennzeichnende Version. Diese Version ist auch im SPS Bibliotheks-Repository ersichtlich.
Eine globale Konstante beinhaltet die Information der Bibliotheksversion:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_ControllerToolbox : ST_LibVersion;
END_VAR
```

ST_LibVersion

Um die existierende Version mit einer benötigten Version zu vergleichen, wird die Funktion F_CmpLibVersion (definiert in der Bibliothek Tc2_System) angeboten.

Veraltete Funktionen

Alle anderen aus TwinCAT 2 bekannten Möglichkeiten zur Abfrage der Bibliotheksversion sind veraltet und sollten nicht mehr verwendet werden.

4.2.3 Datenstrukturen

4.2.3.1 Definition der Strukturen und Enums

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

FLOAT:

In den Funktionsbausteinen der Bibliothek wird 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
```

E_CTRL_MODE

```
TYPE E_CTRL_MODE :
(
    eCTRL_MODE_IDLE     := 0,
    eCTRL_MODE_PASSIVE     := 1,
    eCTRL_MODE_ACTIVE     := 2,
    eCTRL_MODE_RESET     := 3,
    eCTRL_MODE_MANUAL     := 4,
    eCTRL_MODE_TUNE     := 5,
    eCTRL_MODE_SELFTEST    := 6,
    eCTRL_MODE_SYNC_MOVEMENT := 7
)
END_TYPE
```

E_CTRL_STATE

```
TYPE E_CTRL_STATE :
(
    eCTRL_STATE_IDLE        := 0,
    eCTRL_STATE_PASSIVE := 1,
    eCTRL_STATE_ACTIVE := 2,
    eCTRL_STATE_RESET     := 3,
    eCTRL_STATE_MANUAL := 4,
    eCTRL_STATE_TUNING := 5,
    eCTRL_STATE_TUNED     := 6,
    eCTRL_STATE_SELFTEST := 7,
    eCTRL_STATE_ERROR     := 8,
    eCTRL_STATE_SYNC_MOVEMENT := 9
);
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

```
TYPE E_CTRL_SIGNAL_TYPE :
(
eCTRL_TRIANGLE := 0,
eCTRL_SINUS := 1,
eCTRL_SAWTOOTH := 2
);
END_TYPE
```

```
TYPE E_CTRL_STEP_SENSORTYPE
(
eSENSOR_NONE := 0,
eSENSOR_PT100 := 1,
eSENSOR_THERMO_J := 2,
eSENSOR_THERMO_K := 3
);
END_TYPE
```

5 Beispielprojekt

Die Funktionsweise der Bausteine der TF4100 Controller Toolbox 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.

5.1 Beispielprojekt installieren und starten

Die TwinCAT Controller Toolbox bietet dem Programmierer Übertragungsglieder für die Realisierung verschiedenster Regelungen.

1. Speichern Sie das Beispielprogramm https://infosys.beckhoff.com/content/1031/TF4100_TC3_Controller_Toolbox/Resources/1461955979.zip und entpacken Sie es.
   ⇨ Das Beispielprogramm *TcControllerToolbox_Examples.*sln wird in das TwinCAT XAE geladen, übersetzt und gestartet.
2. Laden Sie die Solution Datei.
3. Übersetzen Sie das SPS-Projekt mit Hilfe des Menüs *Project - Rebuild All*.
4. Laden Sie das SPS-Projekt über das Menü *Online - Login* in das Laufzeitsystem.
5. Starten Sie das Programm über das Menü *Online - Run*.

TwinCAT ScopeView-Beispiele verwenden

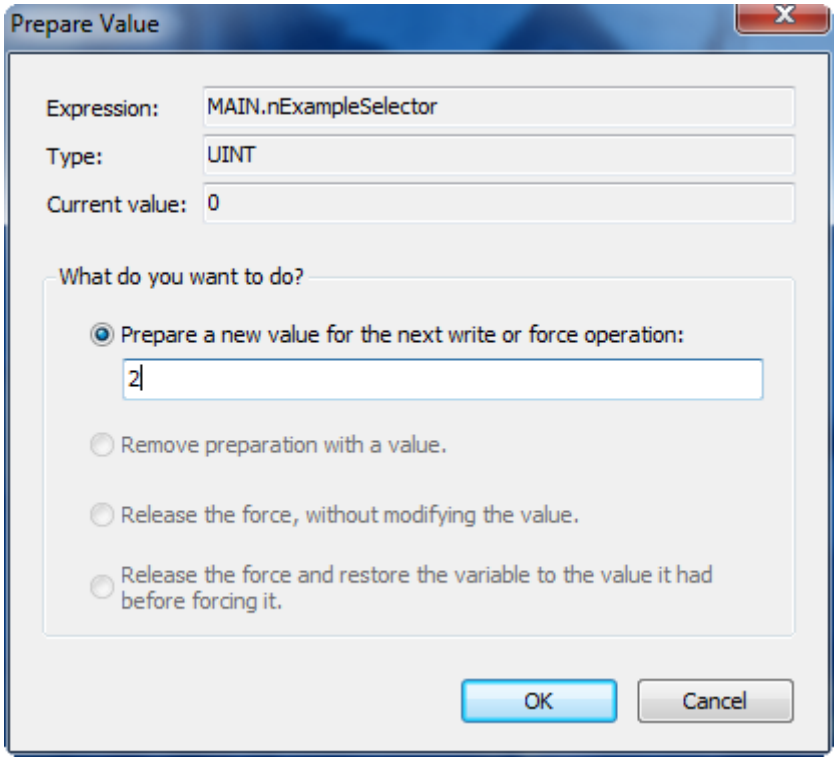
Mit dem TwinCAT ScopeView können die Signalverläufe der einzelnen Beispiele graphisch dargestellt werden. Zu diesem Zweck können die mitgelieferten Einstellungen *TcControllerToolboxExamples_Scope_x.sv* verwendet werden. Die Angabe des bei dem jeweiligen Beispiel zu verwendenden Scope-Files kann den Kommentaren in dem Programm **MAIN** des Projekts entnommen werden.

6. Laden Sie die Scope Configuration-Datei in das TwinCAT ScopeView.
7. Starten Sie die Aufzeichnung über das Menü oder über die Taste **Record**.

Gewünschtes Beispielprojekt auswählen

Da das Beispielprojekt mehrere unterschiedliche Beispiele enthält, muss in dem Programm **MAIN** die Variable `nExampleSelector` auf die entsprechende Beispielnummer gesetzt werden.

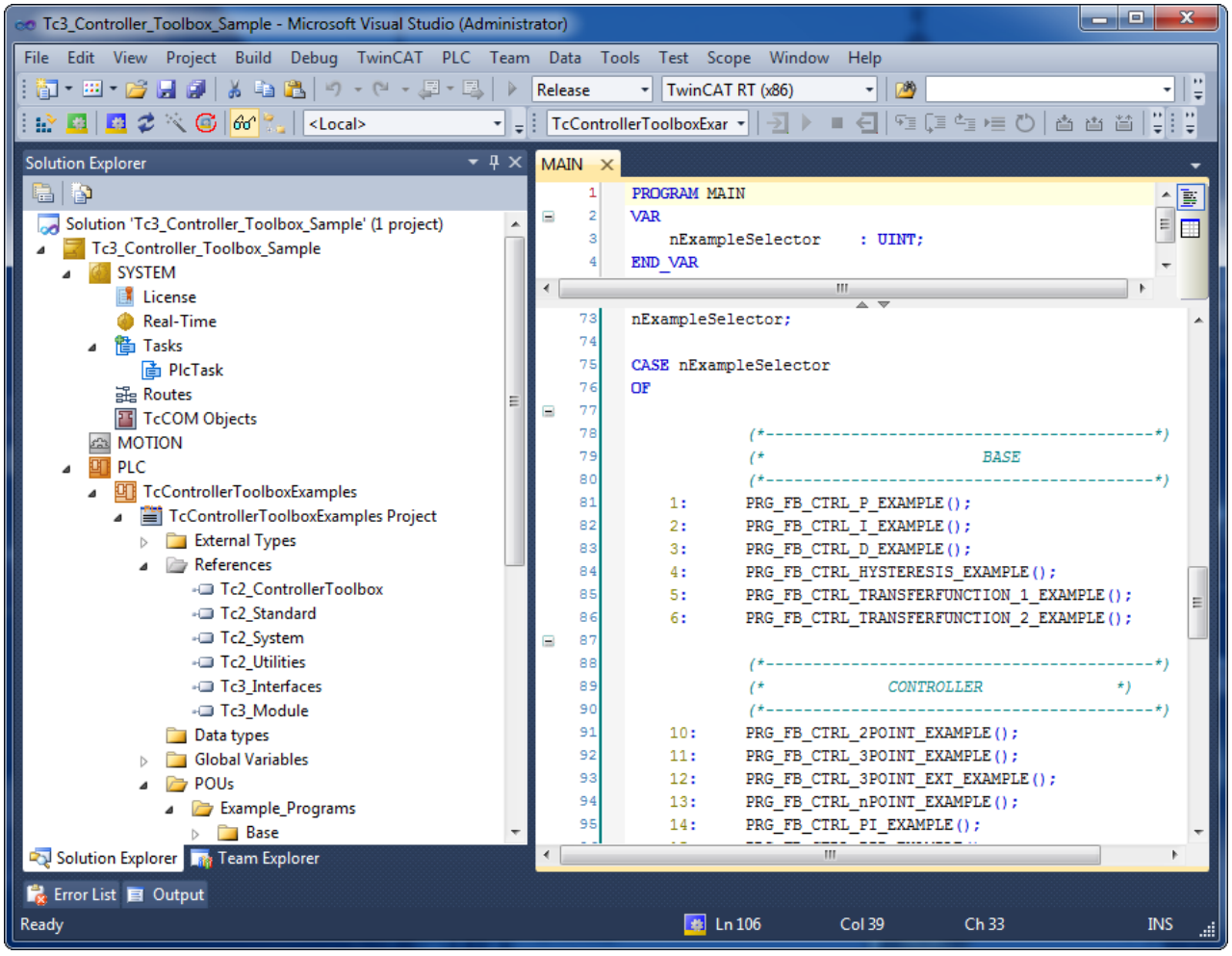
8. Klicken Sie doppelt auf die Variable `nExampleSelector`.



9. Tragen Sie die Nummer des Beispiels ein.
10. Klicken Sie auf die Schaltfläche **OK**.
11. Drücken Sie die Taste **[F7]** oder klicken Sie **Force Values** im Menü **Online** an.

5.2 Programmstruktur

Das Hauptmodul **MAIN** ruft in Abhängigkeit der Variable `nExampleSelector` das entsprechende Beispielprogramm auf.



Für die Verständlichkeit und Nachvollziehbarkeit befinden sich Kommentare in den einzelnen Beispielprogrammen.

6 Anhang

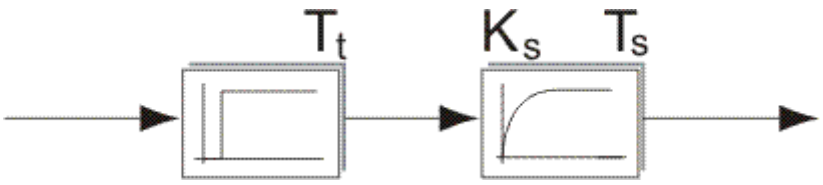
6.1 Einstellregeln für die P-, PI- und PID-Regler

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

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	$1.2 \cdot \frac{T_s}{K_s \cdot T_t}$	$2.0 \cdot T_t$	$0.5 \cdot T_t$

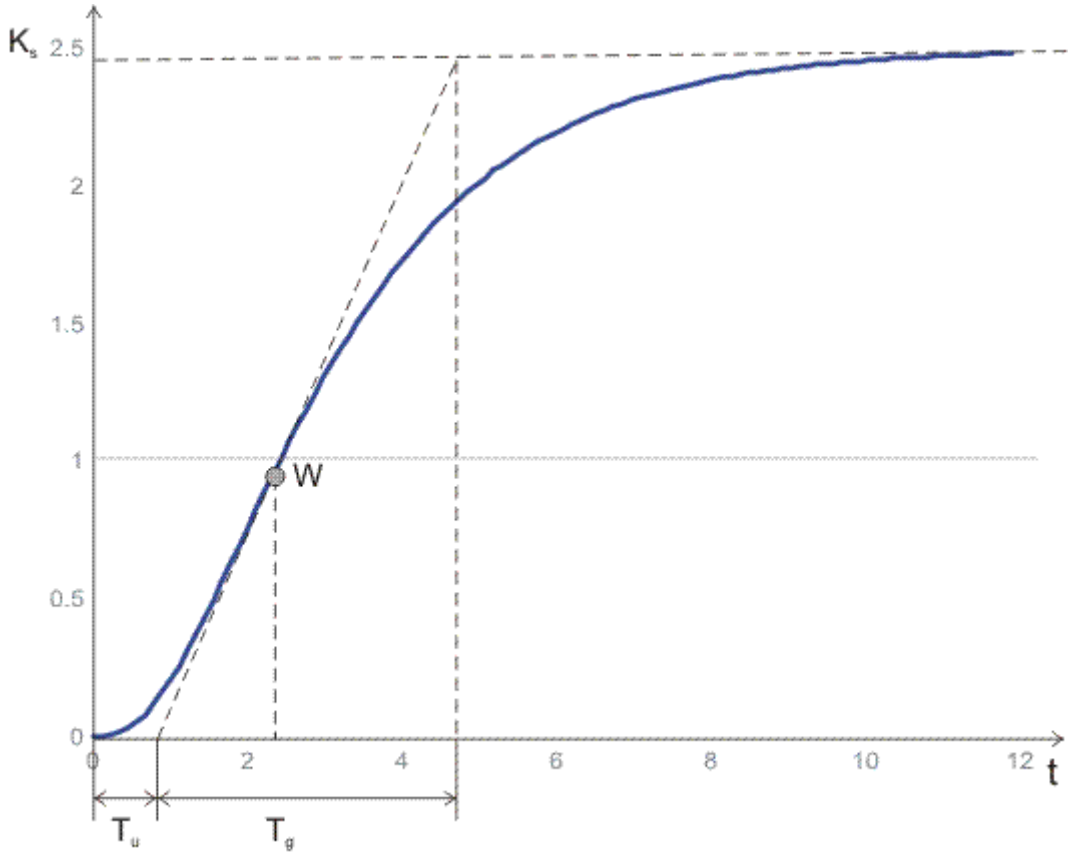
Chien, Hrones und Reswick

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:

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



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	$0.50 \cdot T_u$	$0.47 \cdot T_u$

Mehr Informationen:
www.beckhoff.de/tf4100

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com