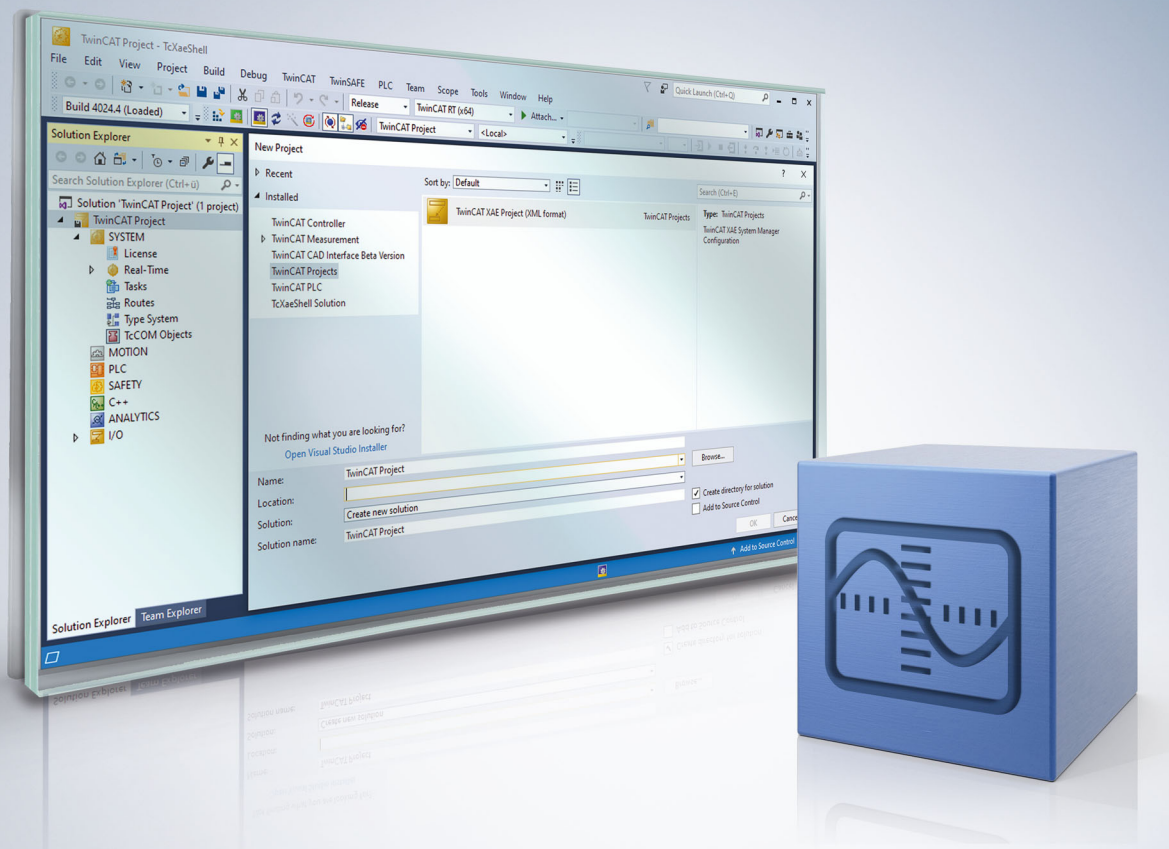
BECKHOFF New Automation Technology

Handbuch | DE

TF3680

TwinCAT 3 | Filter



21.02.2023 | Version: 1.7.0

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

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

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

Disclaimer

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

Marken

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

Patente

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

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

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Erklärung der Symbole

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

⚠ GEFAHR

Akute Verletzungsgefahr!

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

⚠ WARNUNG

Verletzungsgefahr!

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

⚠ VORSICHT

Schädigung von Personen!

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

HINWEIS

Schädigung von Umwelt oder Geräten

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

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

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

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

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

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

2 Übersicht

Die TwinCAT 3 Function TF3680 TC3 Filter stellt in einer SPS-Bibliothek verschiedene Funktionsbausteine zur Realisierung von digitalen Filtern zur Verfügung. Digitale Filter werden verwendet, um digitalisierte (zeitdiskrete und wertquantisierte) Signale zu manipulieren und zum Beispiel bestimmte Bestandteile eines Signals im Frequenzbereich hervorzuheben oder zu unterdrücken.

Anwendungen:

- Unterdrückung von Frequenzbändern, in denen das Nutzsignal im Vergleich zum Rauschsignal eine untergeordnete Rolle einnimmt. Ein Beispiel ist der klassische Tiefpassfilter zur Unterdrückung von hochfrequentem Rauschen.
- Gezielte Eliminierung störender Frequenzbestandteile. Ein Beispiel ist ein zum Nutzsignal überlagertes 50-Hz-Signal, das durch elektromagnetische Kopplung der Netzspannung in das Messsignal einfließt.

Bausteine:

Funktionsbaustein	Filter
FB_FTR_IIRCoeff [▶ 30] FB_FTR_IIRSos [▶ 33]	Ermöglichen die Realisierung eines Filters frei über die Vorgabe von Filterkoeffizienten, sodass prinzipiell alle Filtercharakteristika genutzt werden können.
FB_FTR_IIRSpec [▶ 36]	Ermöglicht die Realisierung eines Filters vom Typ „Butterworth“, vom Typ „Chebyshev“ und vom Typ „Bessel“.
FB_FTR_PT1 [▶ 41] FB_FTR_PT2 [▶ 44] FB_FTR_PT3 [▶ 47] FB_FTR_PTn [▶ 50] FB_FTR_PT2oscillation [▶ 59] FB_FTR_LeadLag [▶ 56] FB_PTt [▶ 62]	Ermöglicht die Realisierung verschiedener Verzögerungsglieder unterschiedlicher Ordnung.
FB_FTR_MovAvg [▶ 39]	Ermöglicht die Realisierung eines Moving-Average-Filters zur Glättung.
FB_FTR_Median [▶ 64]	Median-Filter zur Glättung von Signalen mit Ausreißern.
FB_FTR_Notch [▶ 53]	Ermöglicht die Realisierung eines Bandstopp-Filters zur Unterdrückung eines schmalen Frequenzbandes.
FB_FTR_AcutalValue [▶ 67]	Filter zur Behandlung von Ausreißern.
FB_FTR_Gaussian [▶ 72]	Glättungsfilter mit minimalem Group Delay.

3 Installation

3.1 Systemvoraussetzungen

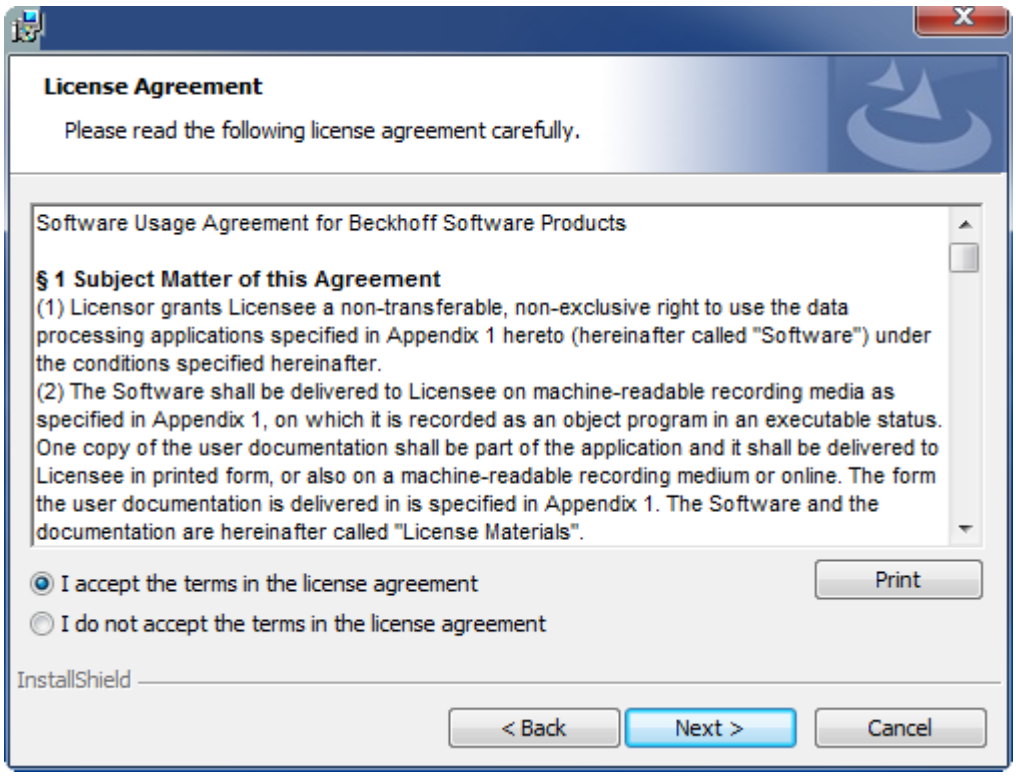
Technische Daten	Beschreibung
Betriebssystem	Windows 7/10, Windows Embedded Standard 7, Windows CE 7
Zielplattform	PC-Architektur (x86, x64 oder ARM)
Minimale TwinCAT-Version	TwinCAT 3.1 Build 4022.25 und höher
Erforderliches TwinCAT-Setup-Level	TwinCAT 3 XAE, XAR
Erforderliche TwinCAT-Lizenz	TF3680 TC3 Filter oder TF3600 TC3 Condition Monitoring

3.2 Installation

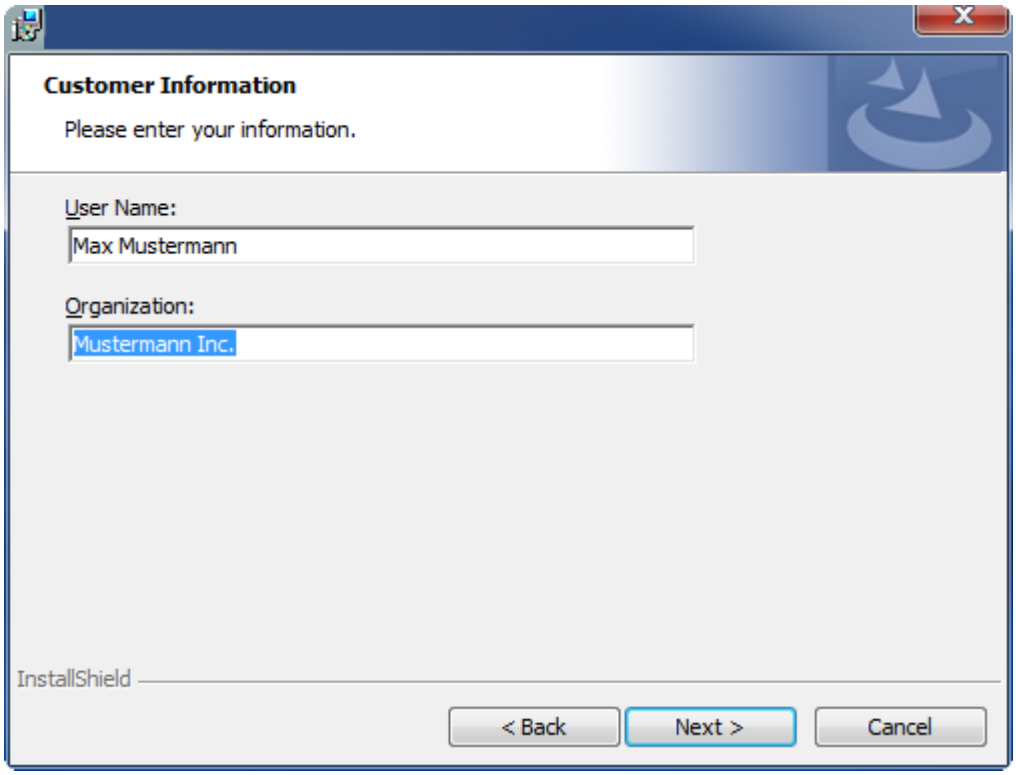
Nachfolgend wird beschrieben, wie die TwinCAT 3 Function für Windows-basierte Betriebssysteme installiert wird.

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

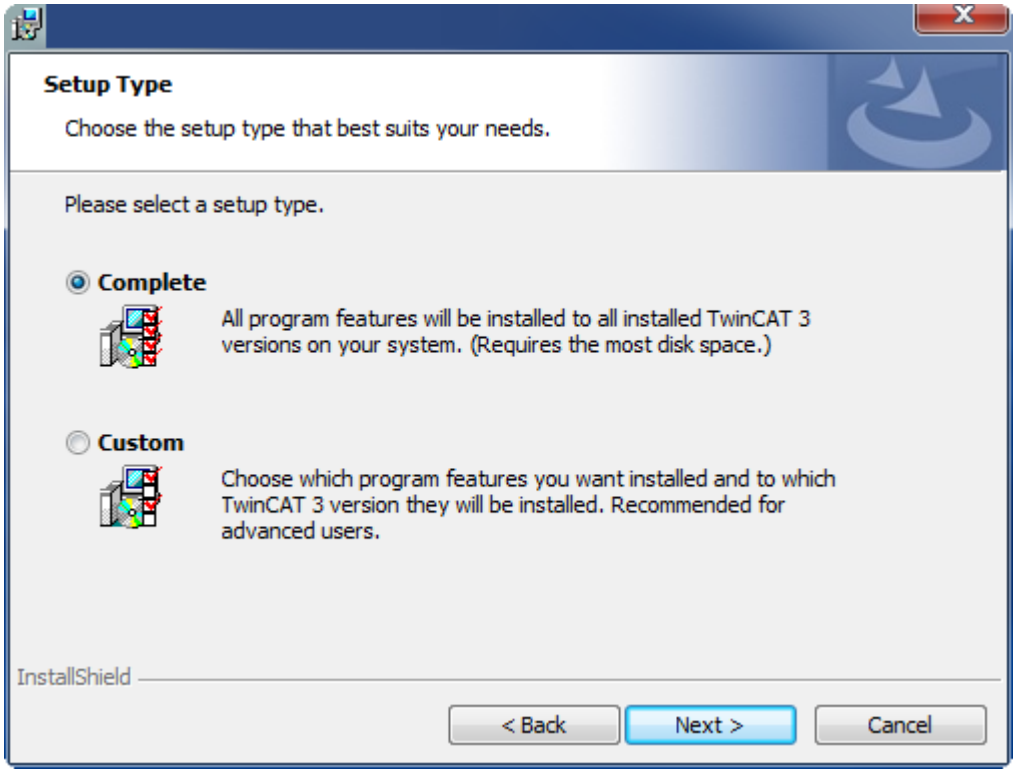
1. Führen Sie die Setup-Datei als Administrator aus. Wählen Sie dazu im Kontextmenü der Datei den Befehl **Als Administrator ausführen**.
   ⇨ Der Installationsdialog öffnet sich.
2. Akzeptieren Sie die Endbenutzerbedingungen und klicken Sie auf **Next**.



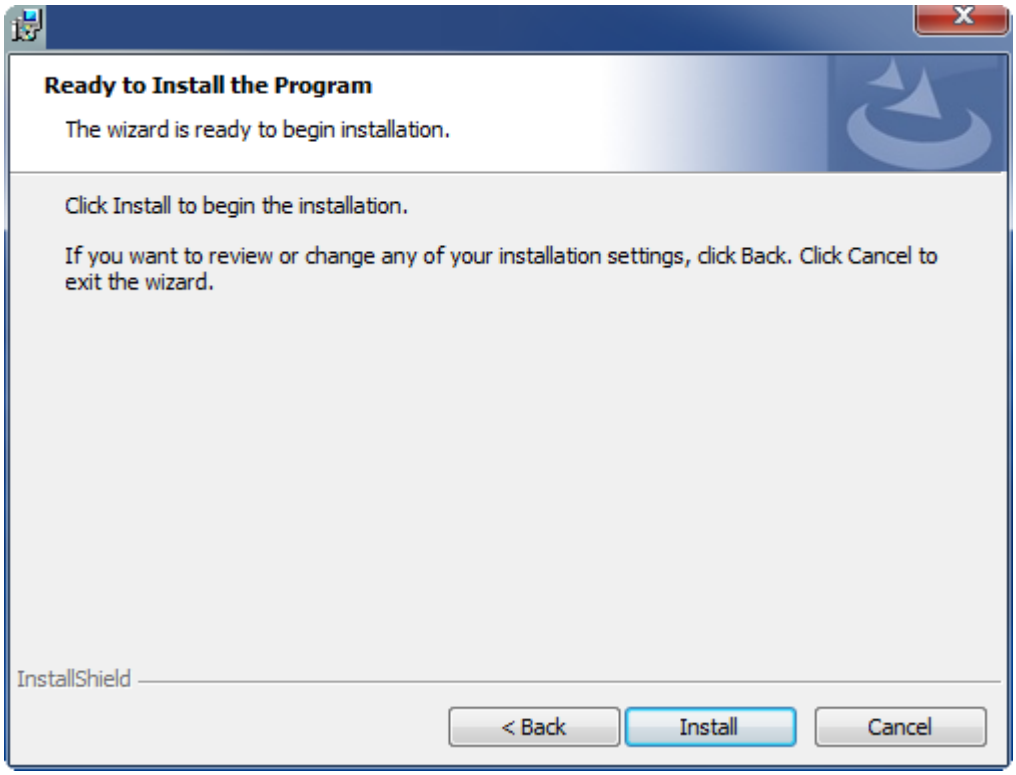
3. Geben Sie Ihre Benutzerdaten ein.



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**.

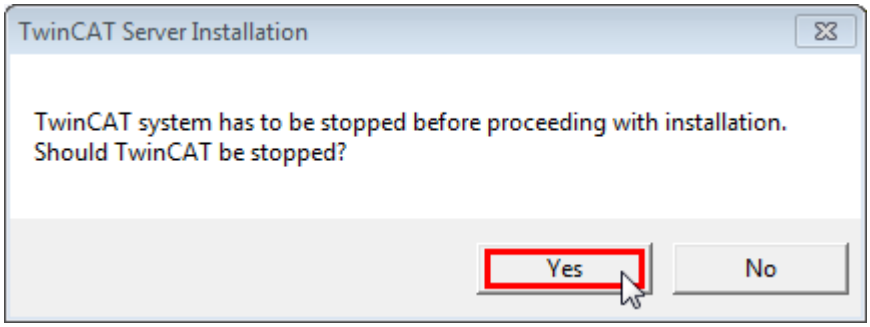


5. Wählen Sie **Next** und anschließend **Install**, um die Installation zu beginnen.

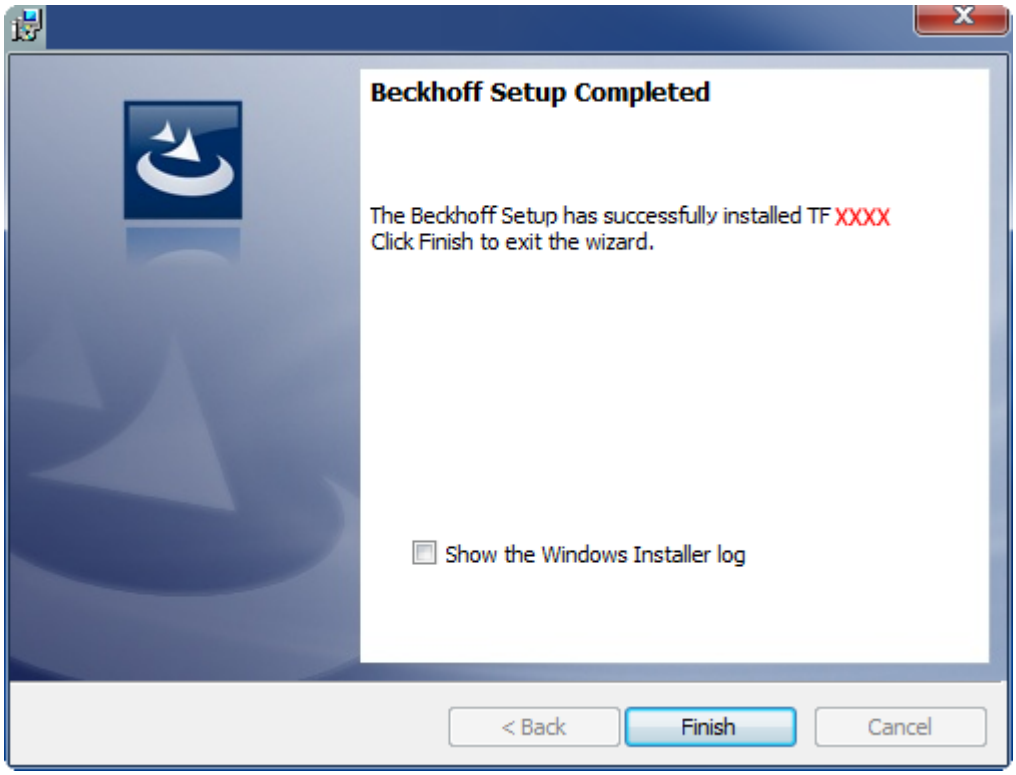


⇨ Ein Dialog weist Sie darauf hin, dass das TwinCAT-System für die weitere Installation gestoppt werden muss.

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 [▶ 12]).

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

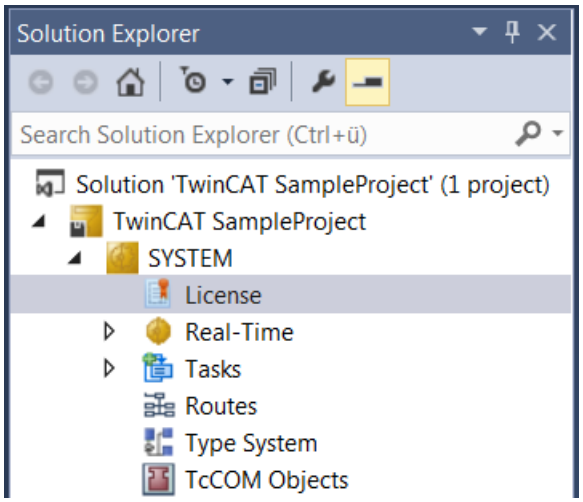
Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT 3 Lizenzierung“.

Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

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

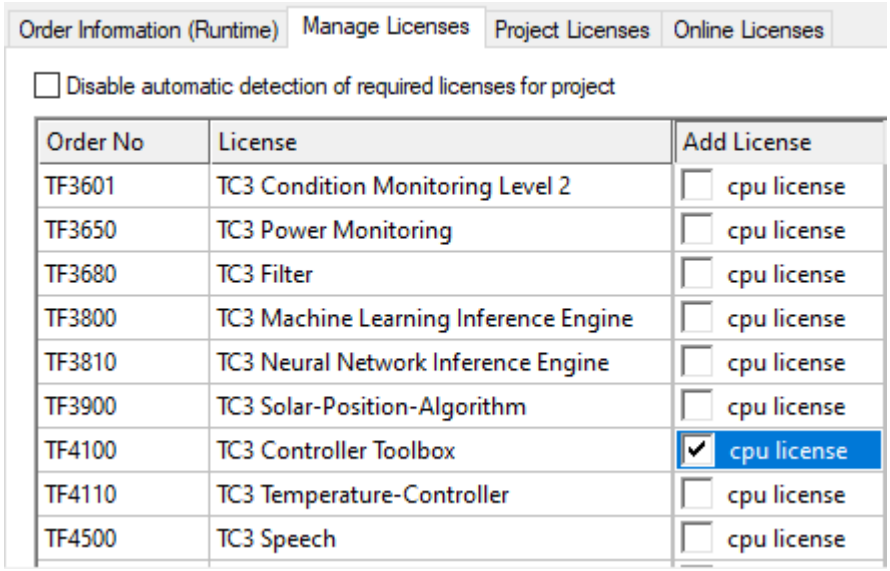
1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.
3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **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.

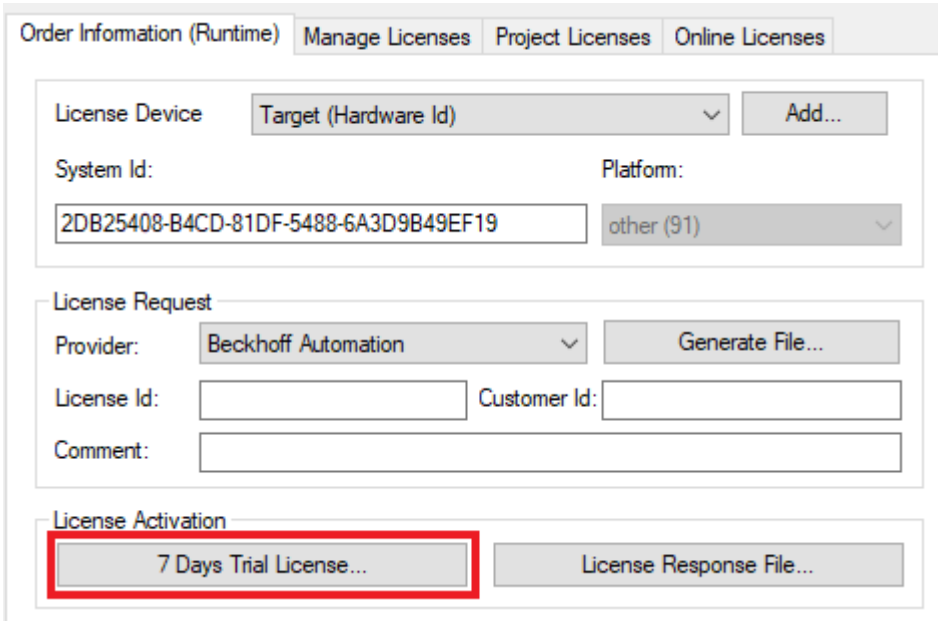
5. Öffnen Sie die Registerkarte **Manage Licenses.** Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (z. B. „TF4100 TC3 Controller Toolbox“).



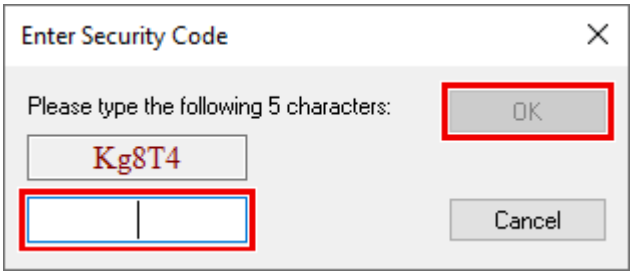
6. Öffnen Sie die Registerkarte **Order Information (Runtime).**

⇨ 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 Technische Einführung

4.1 Digitale Filter

Digitalfilter oder digitale Filter werden verwendet, um digitalisierte (zeitdiskrete und wertquantisierte) Signale zu manipulieren. Die Manipulation zeigt sich dabei im Frequenzbereich, in dem bestimmte Bestandteile eines Signals hervorgehoben oder unterdrückt werden.

Eigenschaften

Digitale Filter können sich u. a. in dem Frequenzbereich unterscheiden, der das Filter passieren darf.

Filtertyp	Beschreibung	Einsatzbereich (Beispiele)
Tiefpass	Frequenzen unterhalb einer Grenzfrequenz können das Filter passieren.	Anti-Aliasing-Filter oder Filter zur Glättung eines Signals.
Hochpass	Frequenzen oberhalb einer Grenzfrequenz können das Filter passieren.	Beseitigung eines störenden Gleichanteils im Signal.
Bandpass	Frequenzen innerhalb eines Frequenzintervalls können das Filter passieren.	Sinnvoll bei amplitudenmodulierten Signalen (Funktechnik, optischen Messsignalen, Ultraschall-Signalen, …), d. h. das Nutzsignal verteilt sich spektral um eine Trägerfrequenz, sodass tiefe und hohe Frequenzen außerhalb des Nutzsignals das SNR (Signal-Rausch-Verhältnis) verschlechtern und unterdrückt werden.
Bandstopp	Frequenzen außerhalb eines Frequenzintervalls können das Filter passieren.	Unterdrückung einer induktiv eingekoppelten Frequenz, z. B. der Netzfrequenz.

Die konkrete Implementierung des Filters bestimmt das Übergangsverhalten vom Durchlassbereich zum Sperrbereich.

Siehe auch: Filtertypen und Parametrierung [▶ 17]

Digitale Signale

Ein analoges Signal x(t) wird durch einen Analog-Digital-Umsetzer, z. B. in einer EL3xxx oder ELM3xxx, in ein zeitdiskretes und wertquantisiertes Signal x[n] überführt. Die Zeitdiskretisierung findet mit der Abtastperiodendauer T statt (Inverse der Abtastfrequenz f_s).

$$x[n] = x(t = nT)$$

Differenzengleichung

Die allgemeine Differenzengleichung für ein Eingangssignal x[n] (Eingang in ein diskretes System, hier ein Filter) und ein zugehöriges Ausgangssignal y[n] lautet:

$$a_0 y[n] + \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

Dabei sind a_k und b_k i. d. R. reellwertige Koeffizienten (Filterkoeffizienten). Der aktuelle Ausgangswert y[n] eines Systems wird also als Linearkombination von vergangenen Filtereingängen x[n-k] mit k > 0, vergangenen Filterausgängen y[n-k] mit k > 0 und dem aktuellen Filtereingang x[n] (k = 0) berechnet.

Die Einbeziehung von vergangenen Filterausgängen in die Berechnung eines aktuellen Ausgangswerts stellt eine Rückkopplung dar und bedarf entsprechend einer Überprüfung hinsichtlich der Stabilität des Systems. Filter mit Rückkopplungen werden „IIR-Filter“ (Infinite-Impulse-Response-Filter) genannt. Filter ohne Rückkopplung werden „FIR-Filter“ (Finite-Impulse-Response-Filter) genannt. Der Vorteil von IIR-Filtern ist, dass mit geringen Filterordnungen bereits „gute“ Manipulationen am Signal x[n] durchgeführt werden können. FIR-Filter hingegen können per Definition nie instabil sein.

Übertragungsfunktion

Durch z-Transformation der Differenzengleichung und unter Ausnutzung der Linearität und der Zeitverschiebungseigenschaft ergibt sich folgende allgemeine Darstellung der Filter-Übertragungsfunktion:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N}}$$

Die Nenner-Koeffizienten a_k gehören zu den Koeffizienten in der Rückkopplung. Damit das Filter mit der Übertragungsfunktion G(z) stabil ist, muss bei der Berechnung dieser Koeffizienten darauf geachtet werden, dass die Polstellen von G(z) innerhalb des Einheitskreises in der komplexen Ebene liegen.

IIR-Filter mit hoher Filter-Ordnung können durch Quantisierungseffekte bei der Berechnung der Koeffizienten instabil werden. Um diese Herausforderung zu umgehen, werden IIR-Filter häufig in kaskadierten Biquad-Filter implementiert, in der Regel bekannt als second-order sections (SOS). Dabei wird die Gesamtübertragungsfunktion durch eine Multiplikation von mehreren Filtern 2ter Ordnung ausgedrückt. Die Übertragungsfunktion G(z) wird dann beschrieben mit:

$$G(z) = \prod_{m=1}^{M} G_m(z) = \prod_{m=1}^{M} \frac{b_{0m} + b_{1m} z^{-1} + b_{2m} z^{-2}}{a_{0m} + a_{1m} z^{-1} + a_{2m} z^{-2}}$$

Aus der Übertragungsfunktion G(z) kann durch Übergang in den Frequenzbereich (Frequenz f) mit

$z = \exp(\mathrm{j} 2\pi f T)$ der Frequenzgang eines Systems bestimmt werden. Der Amplitudengang entspricht dann dem Betrag des Frequenzgangs und der Phasengang dem Argument des Frequenzgangs.

Umsetzung in der SPS-Bibliothek

Die SPS-Bibliothek Tc3_Filter stellt verschiedene Funktionsbausteine zur Realisierung von digitalen Filtern zur Verfügung.

Mit dem Funktionsbaustein FB_FTR_IIRCoeff [▶ 30] können Sie einen freien Filter realisieren. Die Filterkoeffizienten a_k und b_k können Sie dabei individuell berechnen und dem Funktionsbaustein über eine Konfigurationsstruktur übergeben. Für die Stabilität des Filters sind Sie selbst verantwortlich.

Mit dem Funktionsbaustein FB_FTR_IIRSos [▶ 33] können Sie einen freien Filter strukturiert in SOS realisieren. Die Filterkoeffizienten a_k und b_k können Sie dabei individuell berechnen und dem Funktionsbaustein über eine Konfigurationsstruktur übergeben. Für die Stabilität des Filters sind Sie selbst verantwortlich.

Mit dem Funktionsbaustein FB_FTR_IIRSpec [▶ 36] können Sie durch einfache Parametrierung vorgefertigte Filter vom Typ „Butterworth“, „Chebyshev“ und „Bessel“ realisieren. Die Filterkoeffizienten werden dabei intern als Biquads berechnet.

Mit dem Funktionsbaustein FB_FTR_MovAvg [▶ 39] und FB_FTR_Median [▶ 64] können Sie einen Mittelwert- bzw. Median-Filter realisieren, der in vielen Anwendungen zum Glätten von Signalen genutzt wird.

Mit dem Funktionsbaustein FB_FTR_Gaussian [▶ 72] erstellen Sie einen Glättungsfilter mit minimalem Group Delay, sodass die Form ihres Signals beim Durchlaufen des Filters nur minimal beeinflusst wird und nur die störenden Signalanteile entfernt werden.

Mit dem Funktionsbaustein FB_FTR_Notch [▶ 53] können Sie einen Bandstopp-Filter realisieren, der zur Unterdrückung eines schmalen Frequenzbandes genutzt wird.

Mit dem Funktionsbaustein FB_FTR_ActualValue [▶ 67] können Sie eine Plausibilitätskontrolle einer gemessenen Eingangsgröße vornehmen.

Außerdem werden Ihnen weitere in der Systemtheorie und Regelungstechnik gebräuchliche Filter zur Verfügung gestellt: PTt- [▶ 62], PT1- [▶ 41], PT2- [▶ 44], PT3- [▶ 47] , PTn [▶ 50]-, PT2oscillation- [▶ 59] und LeadLag [▶ 56]-Glieder.

Ein PT1-Glied und ein Butterworth-Tiefpassfilter 1. Ordnung können zwar äquivalent ineinander umgerechnet werden, jedoch sind die dargebotenen Parameter der Filter unterschiedlich.

Bilineare Transformation

Die Parametrierung der vordefinierten Filter erfolgt im Laplace-Bereich. Die Umsetzung von der zeitkontinuierlichen Darstellung des Systems in den zeitdiskreten z-Raum erfolgt intern mithilfe der bilinearen Transformation.

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

Beim Filterentwurf wird der Effekt des „Frequency warping“ berücksichtigt.

4.2 Filtertypen und Parametrierung

Diese Beschreibung beschränkt sich auf Tiefpassfilter. Die Konzepte können jedoch auf weitere Filtertypen (Hochpass-, Bandpass- und Stopband-Filter) übertragen werden.

TwinCAT Filter Designer

Die im Folgenden erstellten Grafiken wurden mit dem TwinCAT Filter Designer erstellt. Der Filter Designer ermöglicht es, graphisch Filter zu erstellen und diese dann in der SPS mit TwinCAT 3 Filter (oder als Filter direkt auf dem EtherCAT Terminal oder dem Drive), zu nutzen.

Gebräuchliche Realisierungen eines Digitalfilters sind das Butterworth-Filter, das Chebyshev-Filter und das Bessel-Filter. Jedes Filter besitzt bestimmt Eigenschaften, die in unterschiedlichen Situationen vorteilhaft sind.

Butterworth- vs. Chebyshev Filter:

Der Unterschied beider Realisierungen besteht im Wesentlichen in der Abwägung zwischen der zulässigen Welligkeit des Amplitudengangs im Durchlassbereich und der Steilheit des Amplitudengangs im Übergang vom Durchlass- in den Sperrbereich. Während das Butterworth-Filter einen maximal flachen Amplitudengang im Durchlassbereich besitzt, wird beim Chebyshev-Filter die zulässige Welligkeit des Amplitudengangs im Durchlassbereich als Parameter vorgegeben. Vorteil des Chebyshev-Filters ist eine steilere Abnahme des Amplitudengangs im Übergang vom Durchlass- in den Sperrbereich.

Bessel- vs. Chebyshev- und Butterworth-Filter:

Beim Bessel-Filter wird der Fokus auf eine konstante Gruppenlaufzeit bzw. einen linearen Phasengang im Durchlassbereich des Filters gelegt. Dadurch wird die Form von Signalen mit spektralen Anteilen im Durchlassbereich beim Durchlaufen des Filters nicht verändert. Im Vergleich zum Butterworth- oder Chebyshev-Filter ist beim Bessel-Filter der Übergang vom Durchlassbereich zum Sperrbereich weniger scharf, d. h., Amplituden nahe der Grenzfrequenz werden vergleichsweise weniger stark gedämpft. Im Durchlassbereich selbst weist das Bessel-Filter einen monoton fallenden Amplitudengang auf.

Nachfolgend werden die Filtertypen gegenübergestellt und näher beschrieben. Dazu werden zunächst einige grundlegende Begriffe kurz erläutert.

Übertragungsfunktion im Amplituden-Frequenz-Diagramm

Das Filter wird mathematisch durch die Übertragungsfunktion beschrieben (siehe Digitale Filter [▶ 15]). Die Übertragungsfunktion kann im Amplituden- und Phasengang dargestellt werden.

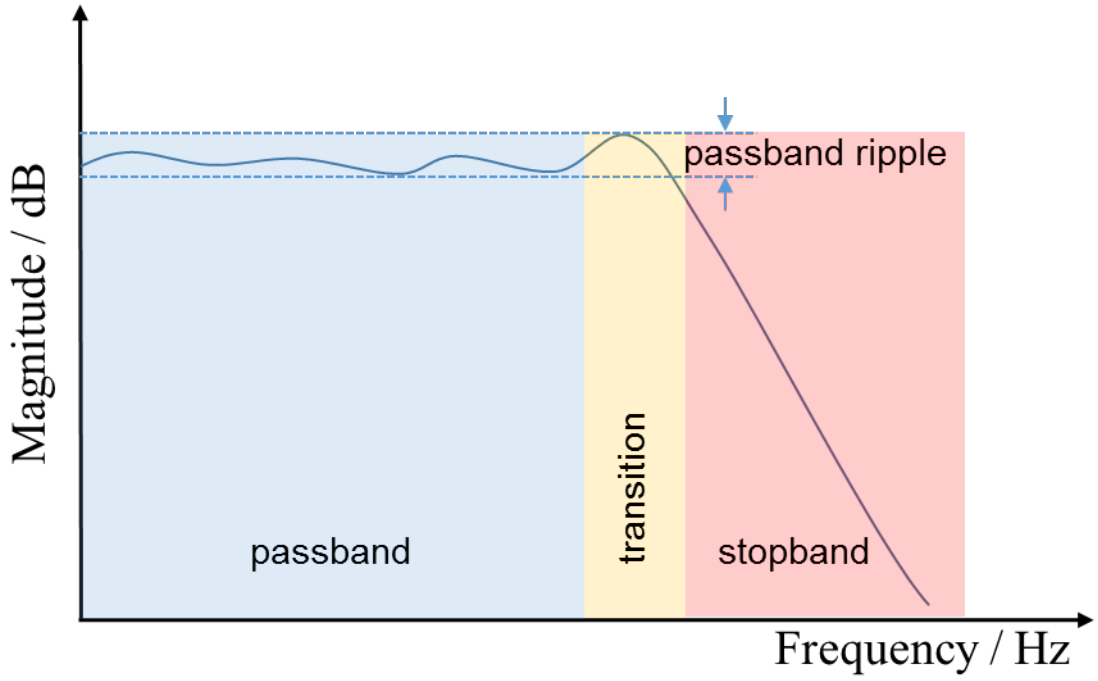


Abb. 1: Grafische Darstellung des Amplitudengangs eines Tiefpassfilters

Durchlassbereich (engl. Passband)

Der Durchlassbereich (blaue Zone) lässt spektrale Anteile eines Signals passieren und soll das Signal in diesem Frequenzbereich nicht möglichst verändern.

Sperrbereich (engl. Stopband)

Im Sperrbereich (rote Zone) dämpft das Filter die entsprechenden Frequenzanteile des Signals.

Übergangsbereich (engl. Transition)

Der Übergangsbereich (gelbe Zone) soll in der Regel möglichst klein sein und trennt Durchlass- und Sperrbereich. Das Design der Übergangsphase ist ein prägendes Kriterium für die Wahl des Filtertyps sowie dessen Parametrisierung.

Welligkeit (engl. Passband ripple)

Die Welligkeit im Durchlassbereich beschreibt den nicht-glatten Verlauf des Amplitudengangs im Durchlassbereich.

Parametrierung des Butterworth-Filters

Eigenschaften

Der Amplitudengang des Butterworth-Filters verläuft im Durchlassbereich maximal flach, sodass das Nutzsignal in diesem Bereich nur minimal manipuliert wird. Außerdem ist der gesamte Verlauf des Amplitudengangs monoton, d. h. ohne Welligkeit. Der Filtertyp ist einer der am häufigsten genutzten Filtertypen.

Parameter

Die Übertragungsfunktion des Butterworth-Filters enthält nur zwei zu definierende Parameter: die Grenzfrequenz (engl. Cutoff frequency) und die Filterordnung.

Filterordnung

Die Filterordnung bestimmt, wie steil der Amplitudengang im Übergangsbereich abnimmt. Je höher die Filterordnung ist, desto steiler nimmt der Amplitudengang ab und desto kleiner ist der Übergangsbereich. Für die Steilheit des Amplitudengangs bei einem Butterworth-Filter gilt: -n * 20 dB/Dekade, mit n = Ordnung, also -20 dB/Dekade für Filterordnung 1, -40 dB/Dekade für Filterordnung 2, usw.

Grenzfrequenz

Die Grenzfrequenz des Butterworth-Filters ist definiert als die Frequenz, bei der der normierte Amplitudengang den Wert 1/sqrt(2) ≈ -3 dB annimmt. Dies gilt für alle Filterordnungen. Entsprechend ist bei der Auslegung des Filters darauf zu achten, dass spektrale Bestandteile eines Signals bei der Grenzfrequenz bereits um 3 dB gedämpft werden. Der Parameter bewirkt eine Parallelverschiebung des Amplitudengangs auf der Frequenzachse (Verzerrung aufgrund der logarithmischen Frequenzachse).

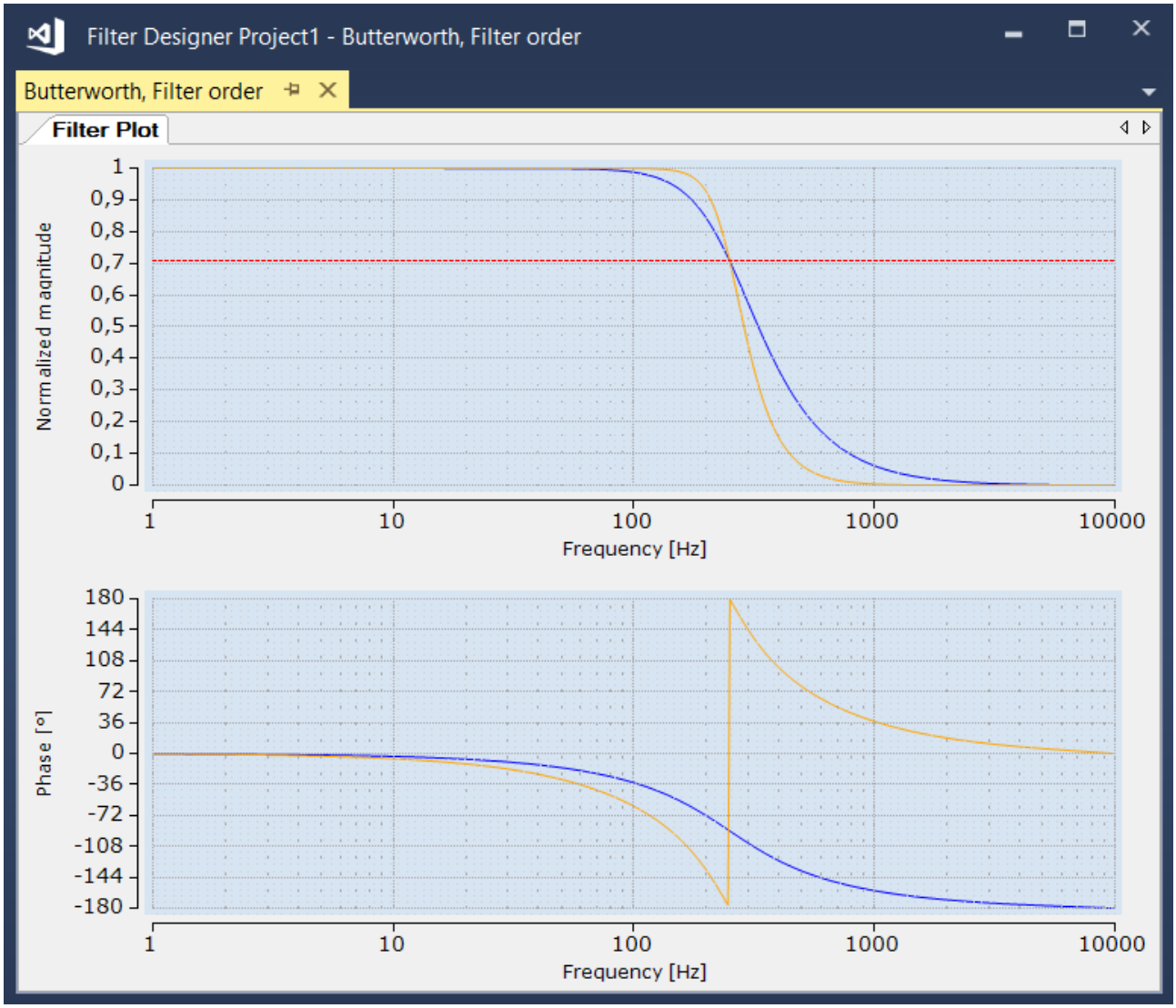


Abb. 2: Grafische Darstellung des Amplituden- und Phasengangs eines Butterworth-Filters mit identischer Grenzfrequenz (blau: Filterordnung 2, gelb: Filterordnung 4)

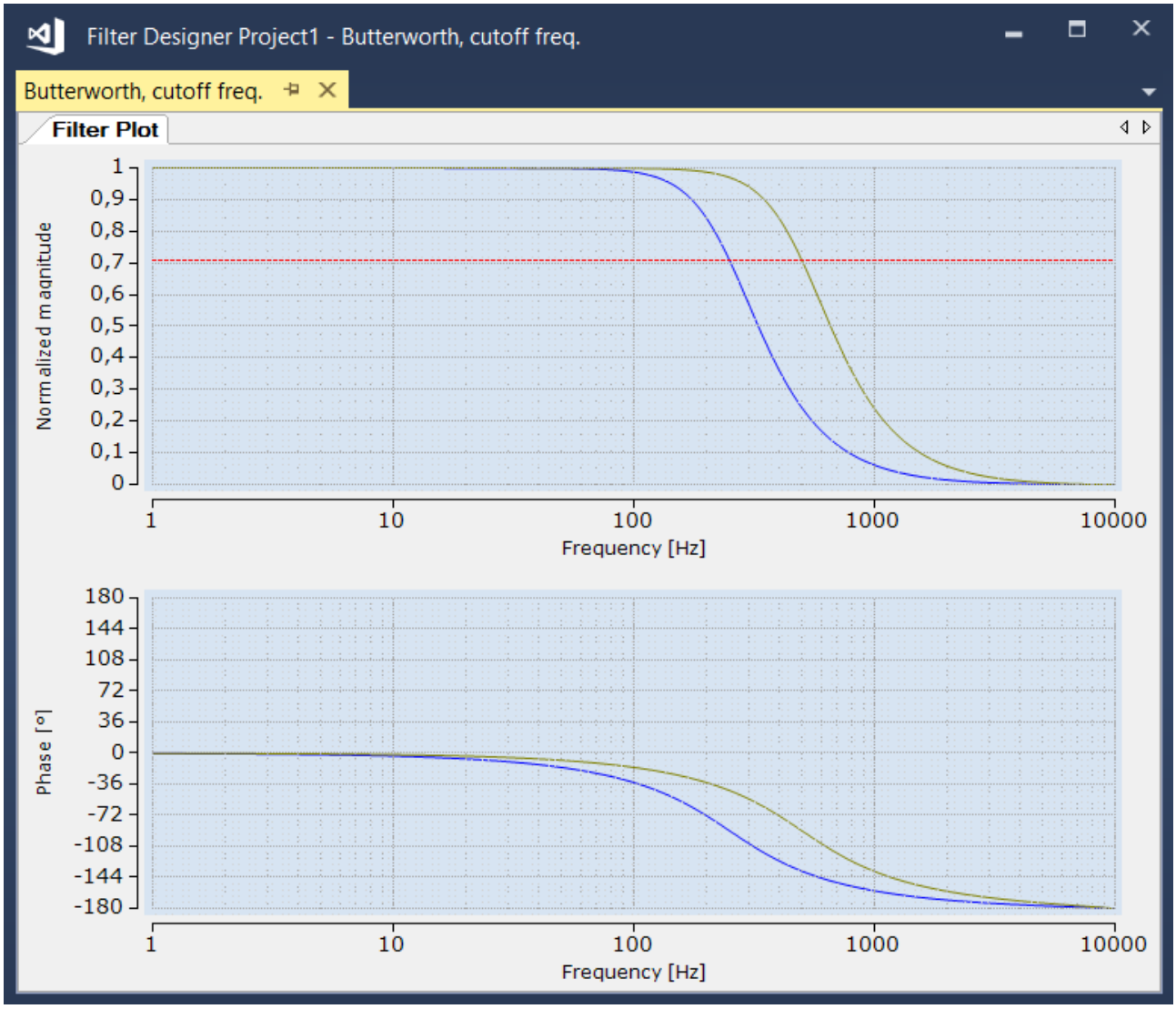


Abb. 3: Grafische Darstellung des Amplituden- und Phasengangs eines Butterworth-Filters identischer Filterordnung (blau: Grenzfrequenz 400 Hz, grün: Grenzfrequenz 700 Hz)

Parametrierung des Chebyshev-Filters

Eigenschaften

Der Amplitudengang des Chebyshev-Filters weist im Durchlassbereich eine parametrierbare Welligkeit auf. Dafür nimmt der Amplitudengang jedoch bereits bei kleiner Filterordnung im Übergangsbereich steil ab. Es gilt: Je größer die zulässige Welligkeit, desto kürzer der Übergangsbereich.

Parameter

Die Übertragungsfunktion des Chebyshev-Filters enthält neben der Filterordnung und der Grenzfrequenz als zu definierende Parameter zusätzlich den Welligkeitsparameter „Passband ripple“.

Welligkeit

Der Parameter gibt die zulässige Welligkeit des Amplitudengangs im Durchlassbereich des Filters an. Durch Zulassen einer Welligkeit kann mit deutlich geringerer Filterordnung ein kurzer Übergangsbereich zwischen Durchlass- und Sperrbereich und somit eine steile Abnahme des Amplitudengangs erreicht werden.

Grenzfrequenz

Die Grenzfrequenz ist beim Chebyshev-Filter definiert als die Frequenz, an der der Amplitudengang die definierte Welligkeit „Passband ripple“ nach unten hin durchstößt. Die Position des Übergangbereichs auf der Frequenzachse geht somit nicht nur mit der Grenzfrequenz, sondern auch mit den Einstellungen zur Filterordnung und Welligkeit einher.

Das nachfolgende Diagramm zeigt drei verschiedene Chebyshev-Filter mit unterschiedlicher Filterordnung und Welligkeit aber gleicher Grenzfrequenz.

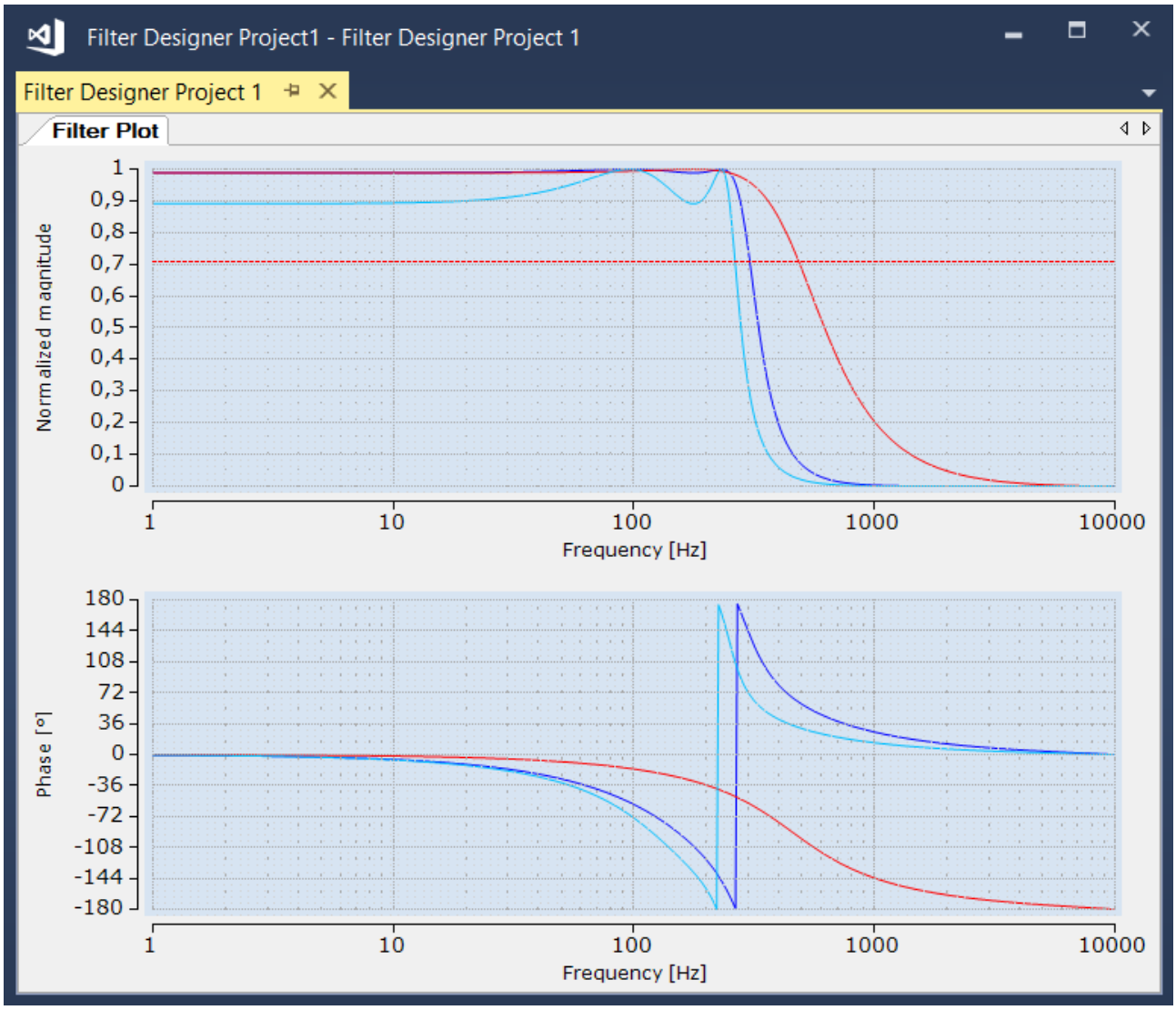


Abb. 4: Grafische Darstellung des Amplituden- und Phasengangs eines Chebyshev-Filters identischer Grenzfrequenz (blau: Filterordnung 4, Passband ripple 0,1 dB, rot: Filterordnung 2, Passband ripple 0,1 dB, cyan: Filterordnung 4, Passband ripple 1 dB)

Filterordnung

Die Filterordnung bestimmt, wie steil der Amplitudengang im Übergangsbereich abnimmt. Je höher die Filterordnung ist, desto steiler nimmt der Amplitudengang ab und desto kleiner ist der Übergangsbereich. Für die Steilheit des Amplitudengangs bei einem Butterworth-Filter gilt: -n * 20 dB/Dekade, mit n = Ordnung, also -20 dB/Dekade für Filterordnung 1, -40 dB/Dekade für Filterordnung 2, usw.

Die Filterordnung beeinfluss die Grenzfrequenz nach obiger Definition nicht, wie in folgender Grafik zu sehen ist. Die Amplitudengang-Verläufe schneiden sich bei der Grenzfrequenz von 250 Hz.

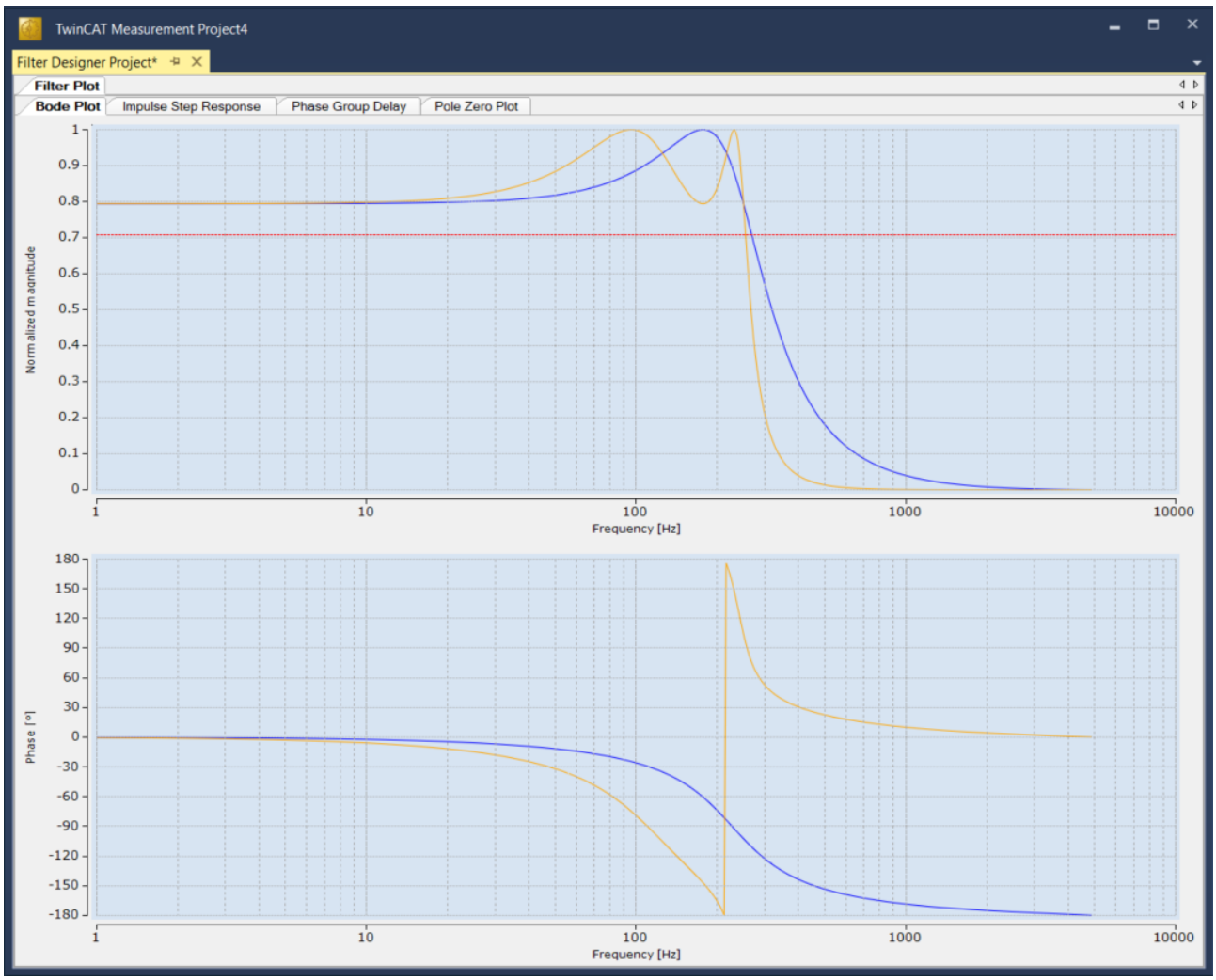


Abb. 5: Grafische Darstellung des Amplituden- und Phasengangs eines Chebyshev-Filters identischer Grenzfrequenz (250 Hz) und identischem Passband ripple (2 dB): (blau: Filterordnung 2, gelb: Filterordnung 4)

Vergleich Butterworth- und Chebyshev-Filter

Die nachfolgende Grafik zeigt einen direkten Vergleich der Amplituden- und Phasengänge eines Butterworth-Filters und eines Chebyshev-Filters. Beide Filter sind so parametriert, dass sich deren Amplitudengänge in der Grenzfrequenz des Butterworth-Filters bei einer normierten Amplitude von 1/sqrt(2) schneiden. Beide Filter sind als Filter fünfter Ordnung angesetzt. Der Welligkeitsparameter (Passband ripple) des Chebyshev-Filters beträgt 0.5 dB.

Es zeigt sich die angesprochene Abwägung zwischen zulässiger Welligkeit des Amplitudengangs im Durchlassbereich und Steilheit im Übergangsbereich bei gleicher Filterordnung. Bei gleicher Filterordnung nimmt beim Chebyshev-Filter der Amplitudengang im Übergangsbereich stärker ab als beim Butterworth-Filter. Dafür ist dessen Amplitudengang im Durchlassbereich nicht glatt, sodass hier das Nutzsignal stärker manipuliert wird als beim Butterworth-Filter.

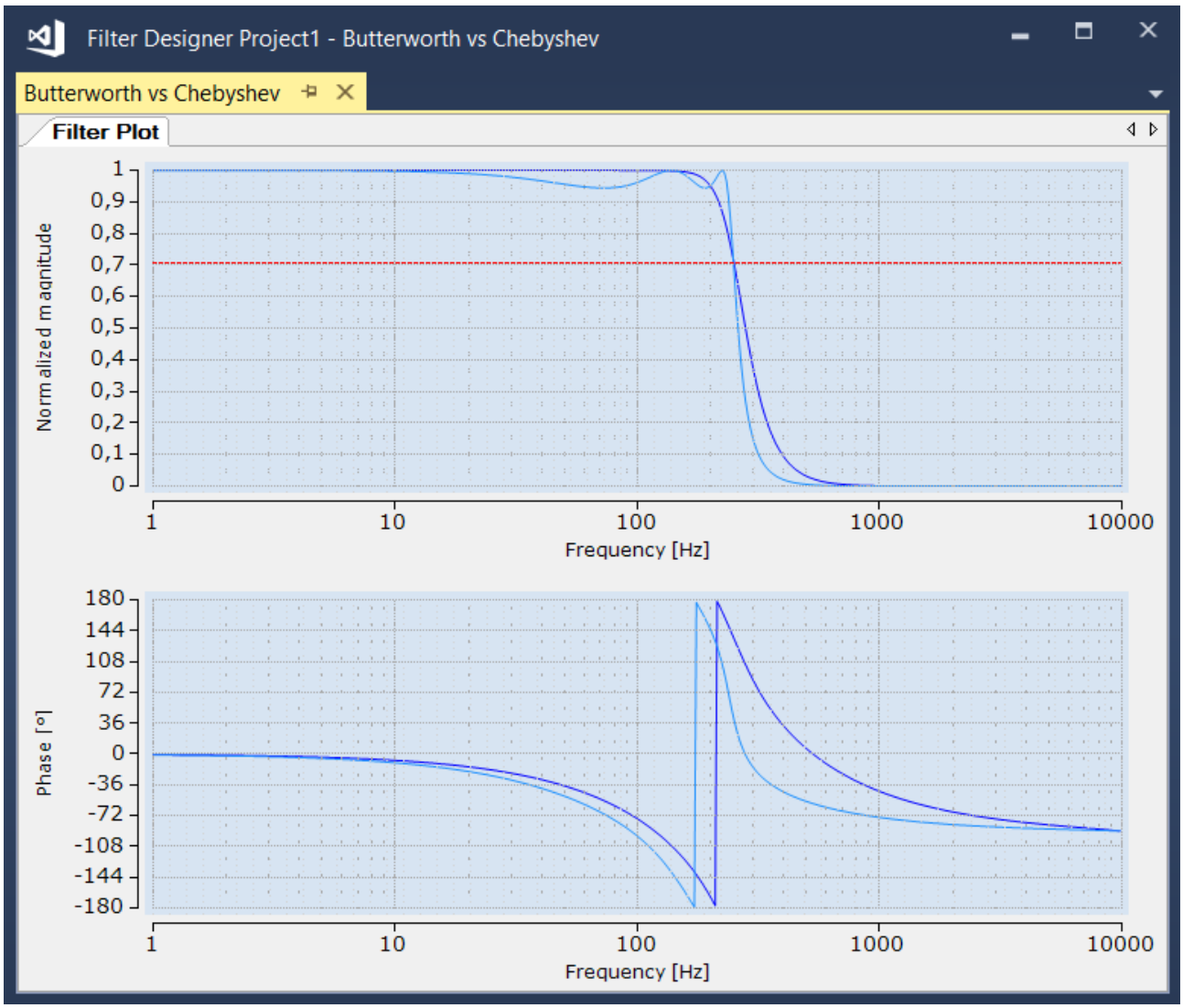


Abb. 6: Grafische Darstellung des Amplituden- und Phasengangs eines Butterworth-Filters (blau) und eines Chebyshev-Filters (cyan)

Parametrierung des Bessel-Filters

Eigenschaften

Das Bessel-Filter weist eine konstante Gruppenverzögerung im Durchlassbereich auf. Der Amplitudengang ist monoton leicht abfallend. Durch diese Eigenschaften wird ein Signal, welches nur spektrale Anteile im Durchlassbereich besitzt, in seiner Signalform beim Durchlaufen des Filters nicht verändert.

Parameter

Das Bessel-Filter wird wie das Butterworth-Filter über die Grenzfrequenz und Filterordnung parametriert.

Grenzfrequenz

Die Grenzfrequenz f_c definiert die Höhe des Group Delay des Bessel-Filters im Durchlassbereich $\tau_{gd,pass}$:

$$\tau_{\mathrm{gd,pass}} = \frac{1}{2 f_s \tan\left(\pi f_c / f_s\right)}$$

Dabei ist f_s die Abtastfrequenz.

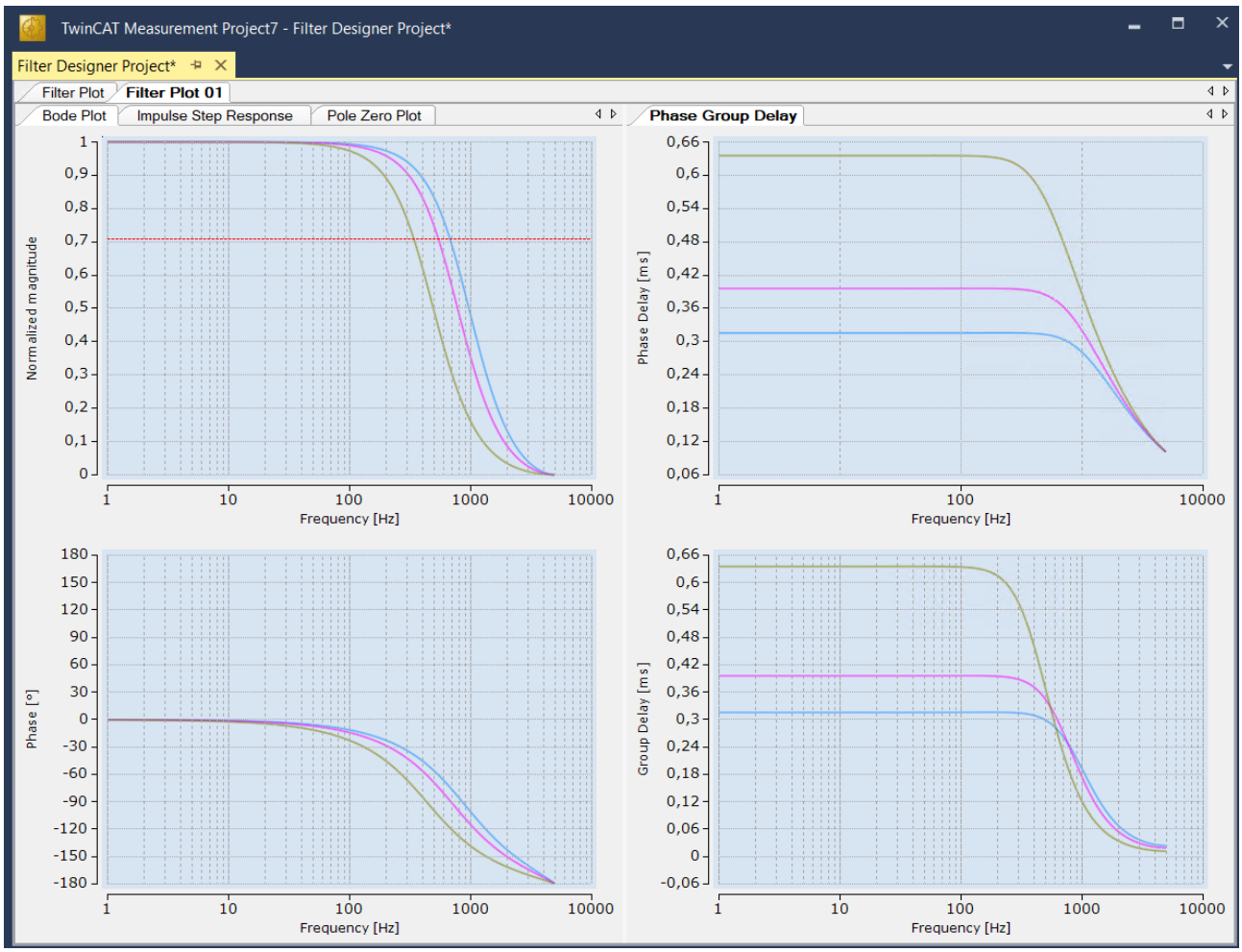


Abb. 7: Grafische Darstellung des Amplituden- und Phasengangs eines Bessel-Filters (blau: Grenzfrequenz 500 Hz, magenta: Grenzfrequenz 400 Hz, olive: Grenzfrequenz 250 Hz)

Filterordnung

Die Filterordnung beeinflusst die Steilheit des Amplitudengangs im Sperrbereich. Zu beachten ist beim Bessel-Filter, dass eine Vergrößerung der Filterordnung einhergeht mit einer Vergrößerung des Durchlassbereichs. Es ist entsprechend ratsam, zunächst die Ordnung zu wählen und dann mit der Grenzfrequenz den Durchlassbereich zu definieren.

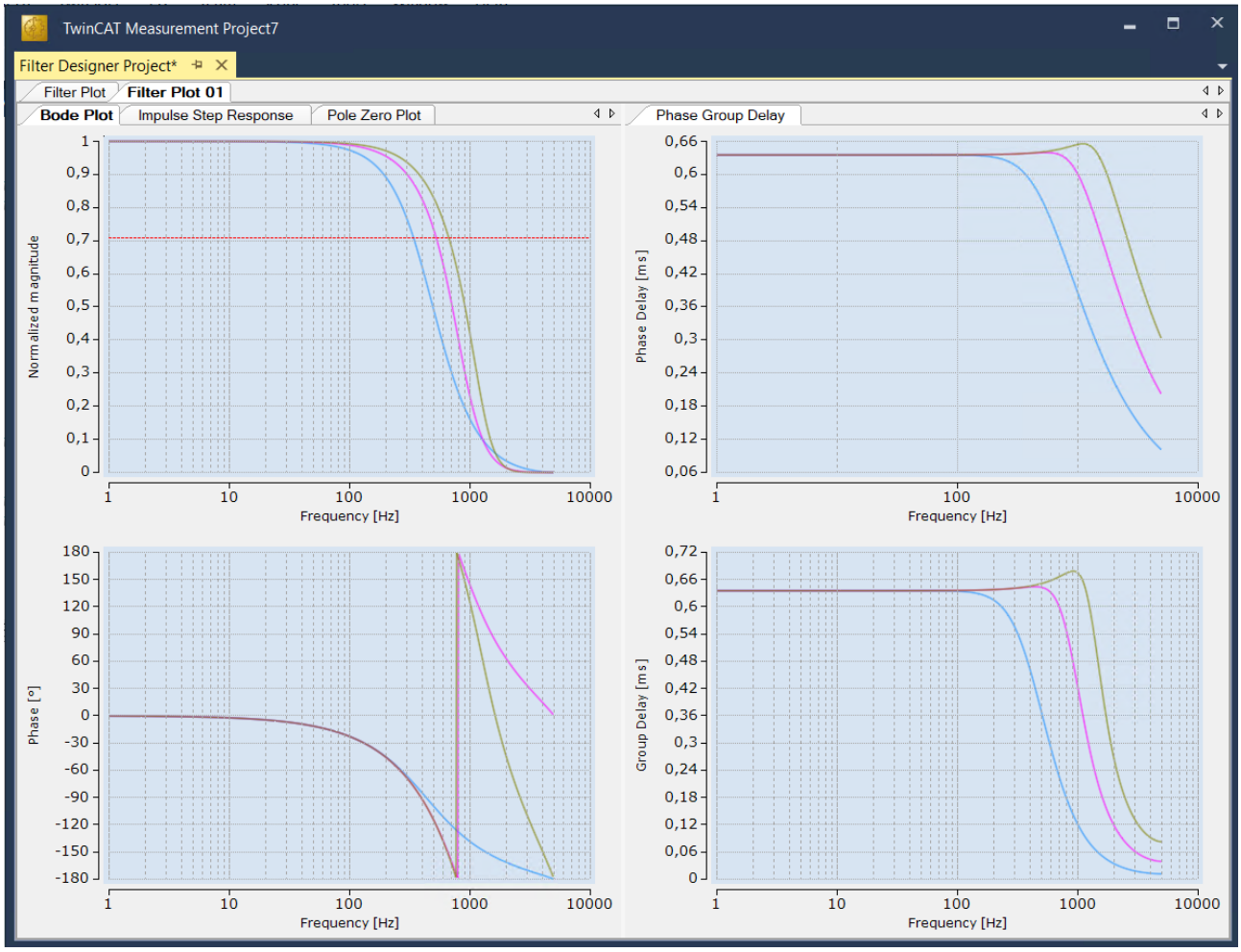


Abb. 8: Grafische Darstellung des Amplituden- und Phasengangs eines Bessel-Filters (blau: Filterordnung 2, magenta: Filterordnung 4, olive: Filterordnung 6)

Vergleich Butterworth-, Chebyshev- und Bessel-Filter

Die wesentlichen Eigenschaften des Bessel-Filters sind im Zeitbereich bzw. im Phase- und Group Delay ersichtlich. Wie in untenstehender Grafik zu sehen, kommt die Impulsantwort und Sprungantwort des Bessel-Filters ohne viel Einschwingen aus. Zudem ist Phase Delay und Group Delay im Durchlassbereich des Filters nahezu konstant, was dazu führt, dass Signale mit spektralen Anteilen im Durchlassbereich nicht in ihrer Form verändert werden.

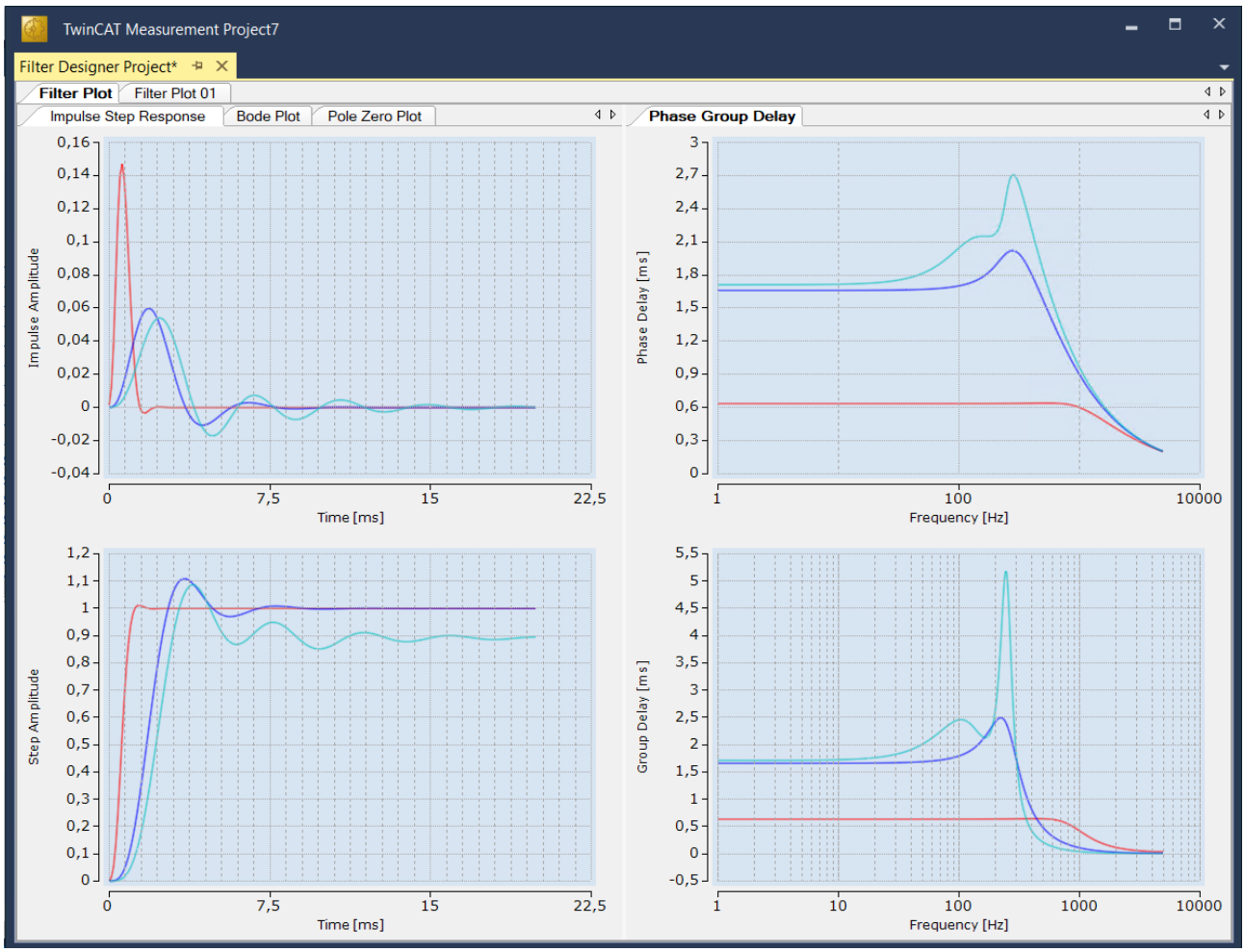
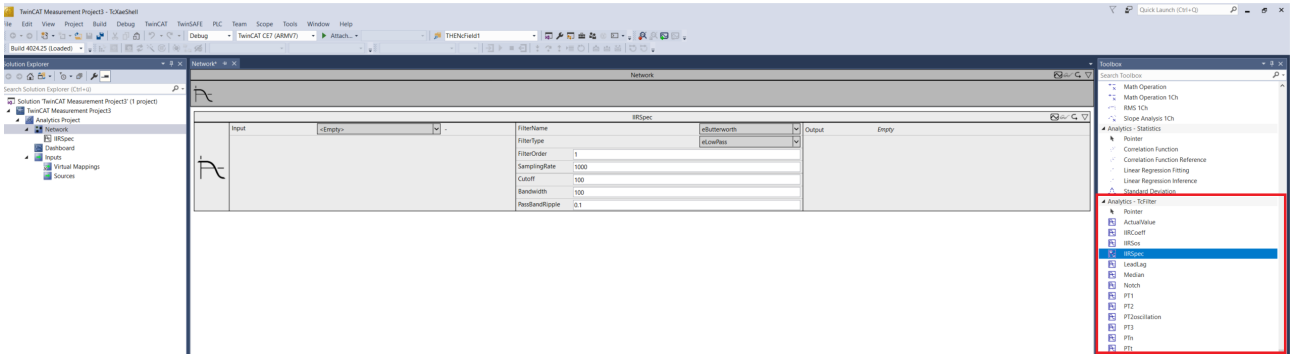


Abb. 9: Grafische Darstellung der Impuls- und Sprungantwort sowie des Phase- und Group Delay eines Butterworth-Filters (blau), eines Chebyshev-Filters (cyan) und eines Bessel-Filters (rot).

4.3 TwinCAT Analytics

Die Bibliothek kann auch in den TwinCAT Analytics Engineering Produkten verwendet werden, um kontinuierliche Datenanalysen aus verschiedenen, dezentralen Maschinensteuerungen zu erstellen. Darüber hinaus lässt sich mit der Analytics Workbench eine SPS-Applikation automatisch erzeugen, der generierte Code kann in die Analytics Runtime heruntergeladen werden, wo er 24/7 parallel zu den Anwendungen im Feld läuft.



Weitere Informationen finden Sie in der TwinCAT Analytics Dokumentation.

5 SPS API

5.1 Funktionsbausteine

Grundstruktur der Funktionsbausteine

Alle Funktionsbausteine der Bibliothek Tc3_Filter basieren auf derselben Grundstruktur. Dies erleichtert das Engineering, wenn von einem Filtertyp zu einem anderen gewechselt wird.

Syntax

```
FUNCTION_BLOCK FB_FTR_<type>
VAR_INPUT
    stConfig        : ST_FTR_<type>;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Zur Konfiguration des Filters wird den Funktionsbausteinen bei der Instanziierung eine Konfigurationsstruktur vom Typ ST_FTR_<type> übergeben. Die Konfigurationsstruktur kann in der Deklaration oder über die Methode `Configure()` zur Laufzeit zugewiesen werden.

Siehe auch: Datentypen [▶ 75] > Konfigurationsstrukturen [▶ 75]

Beispiel zur Konfiguration in der Deklaration:

```
(* define configure structure - exemplary for IIRSpec *)
stParams : ST_FTR_IIRSpec := (
    eFilterName    := E_FTR_Name.eButterworth,
    eFilterType    := E_FTR_Type.eLowPass,
    nFilterOrder   := nFilterOrder,
    fSamplingRate  := fSampleRate,
    fCutoff        := fCutoff,
    nOversamples   := cOversamples,
    nChannels      := cChannels);
```

```
(* create filter instance with configure structure *)
fbFilter : FB_FTR_IIRSpec := (stConfig := stParams);
```

Ausgänge

Alle Funktionsbausteine haben als Ausgangsparameter ein Error-Flag `bError` und ein Flag `bConfigured` vom Typ `BOOL`. Diese zeigen an, ob ein Fehler vorliegt und ob die zugehörige Funktionsbausteininstanz erfolgreich konfiguriert wurde. Der Ausgang `ipResultMessage` vom Typ `I_TcMessage` bietet verschiedene Eigenschaften zur Erläuterung einer Event-Ursache sowie Methoden zur Verarbeitung der Message (Event-Liste [▶ 96]).

Siehe auch: I_TcEventBase und I_TcMessage

Methoden

Alle Funktionsbausteine der Bibliothek Tc3_Filter verfügen über drei Methoden. Diese liefern einen positiven Rückgabewert, wenn sie fehlerfrei ausgeführt wurden.

Configure()

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filterbausteins initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

```
METHOD Configure : BOOL
VAR_INPUT
     stConfig : ST_FTR_<type>;
END_VAR
```

Call()

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL; (*address of input array*)
     nSizeIn     : UDINT;            (*size of input array*)
     pOut        : POINTER TO LREAL; (*address of output array*)
     nSizeOut    : UDINT;            (*size of output array*)
END_VAR
```

Reset()

Die Methode setzt den internen Status eines Filters zurück. Der Einfluss der vergangenen Werte auf den aktuellen Ausgangswert wird eliminiert.

```
METHOD Reset : BOOL
```

Eigenschaften

Die Bibliothek Tc3_Filter referenziert auf den TwinCAT 3 EventLogger und stellt somit sicher, dass Informationen (Ereignisse) über die standardisierte Schnittstelle I_TcMessage bereitgestellt werden.

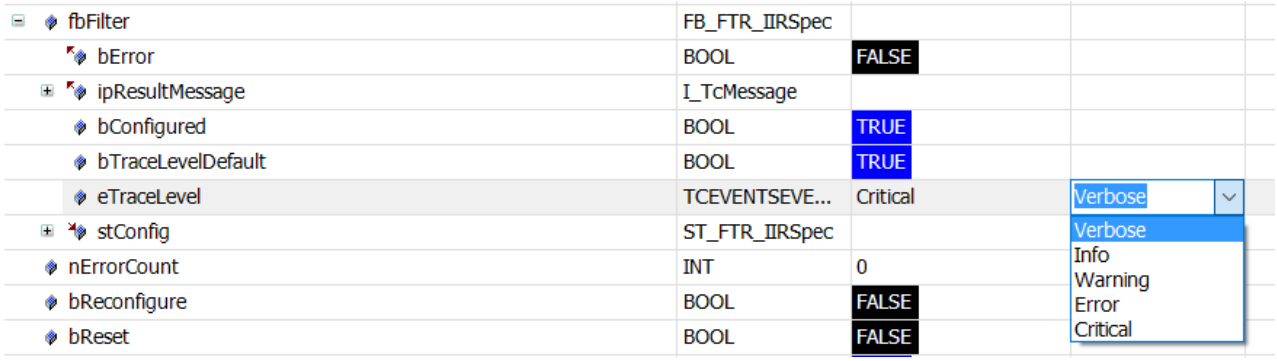
Jeder Funktionsbaustein verfügt über die Eigenschaften `eTraceLevel` vom Typ `TcEventSeverity` und `eTraceLevelDefault` vom Typ `BOOL`.

Das Trace-Level bestimmt die Severity eines Events (Verbose, Info, Warning, Error, Critical) und wird über die Eigenschaft `eTraceLevel` gesetzt.

```
(* Sample of setting fbFilter to trace level info *)
fbFilter.eTraceLevel := TcEventSeverity.Info;
```

Über die Eigenschaft `eTraceLevelDefault` kann das Trace-Level wieder auf den Standardwert (`TcEventSeverity.Critical`) gesetzt werden. Auf die Eigenschaft kann lesend und schreibend zugegriffen werden, d. h. über die Eigenschaft `eTraceLevelDefault` kann abgefragt werden, ob der Standardwert gesetzt ist.

Die Eigenschaften können auch im Online View gesetzt werden.



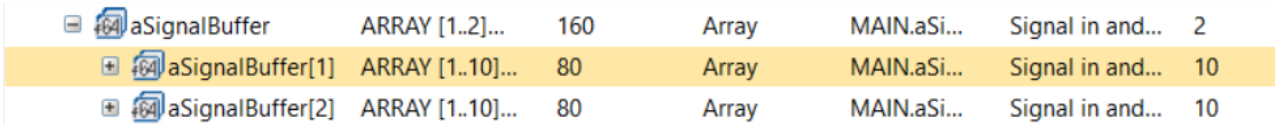
Umgang mit Oversampling und mehreren Kanälen

Alle Funktionsbausteine sind Oversampling- und Multi-Channel-fähig, wobei unterschiedliche Arten der Nutzung möglich sind. Die Deklaration der Filter-Funktionsbausteininstanz `fbFilter` ist hier immer gleich.

Multi-Channel mit 2-dimensionalen Signal-Arrays

Die Definition eines 2-dimensionalen Arrays hat den Vorteil, dass die Definition allgemeingültig ist und der Parameter `cChannels` in anderen Projekten auch auf 1 gesetzt werden kann. Ebenso lassen sich die Kanäle im TwinCAT 3 Scope einzeln selektieren, sodass alle Kanäle unabhängig voneinander betrachtet werden können.

Darstellung des mehrdimensionalen Arrays im „Target Browser“:



Beispiel:

```
VAR CONSTANT
    cChannels    : UINT := 2;
    cOversamples : UINT := 10;
END_VAR
VAR
    aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
    aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
END_VAR
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

Multi-Channel mit 1-dimensionalen Signal-Arrays

Alternativ können die Abtastwerte der verschiedenen Kanäle auch in einem eindimensionalen Array liegen. Die Aufnahme der einzelnen Kanäle mit dem TwinCAT 3 Scope ist dann jedoch nicht so einfach möglich.

Beispiel:

```
VAR CONSTANT
     cChannels    : UINT := 2;
     cOversamples : UINT := 10;
END_VAR
VAR
     aInput  : ARRAY [1..cChannels*cOversamples] OF LREAL;
     aOutput : ARRAY [1..cChannels*cOversamples] OF LREAL;
END_VAR
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

Einkanalige Anwendung mit Oversamples

Wenn nur ein einzelner Kanal betrachtet wird, können die Ein- und Ausgangsarrays als eindimensionale Größen deklariert werden.

```
VAR CONSTANT
     cChannels    : UINT := 1;
     cOversamples : UINT := 10;
END_VAR
VAR
     aInput  : ARRAY [1..cOversamples] OF LREAL;
     aOutput : ARRAY [1..cOversamples] OF LREAL;
END_VAR
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

Einkanalige Anwendung ohne Oversamples

Wenn nur ein einzelner Kanal betrachtet und auch kein Oversampling angewendet wird, können die Ein- und Ausgangsgrößen auch als `LREAL` deklariert werden.

```
VAR CONSTANT
     cChannels    : UINT := 1;
     cOversamples : UINT := 1;
END_VAR
VAR
     fInput  : LREAL;
     fOutput : LREAL;
END_VAR
```

```
bSucceed := fbFilter.Call(ADR(fInput), SIZEOF(fInput), ADR(fOutput), SIZEOF(fOutput));
```

5.1.1 FB_FTR_IIRCoeff

Der Funktionsbaustein realisiert einen IIR-Filter (Infinite-Impulse-Response-Filter). Die allgemeine Übertragungsfunktion lautet:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N}}$$

Die Koeffizienten a_k und b_k sind in der Konfigurationsstruktur frei definierbar. Zählergrad und Nennergrad dürfen unterschiedlich sein. Der Nenner kann auch zu $a_0 = 1$ und $a_k = 0$ (für alle $k > 0$) gesetzt werden, sodass ein FIR-Filter konfiguriert ist. Die Filterspezifikation wird mit der Struktur ST_FTR_IIRCoeff übergeben.

Siehe auch: Digitale Filter [▶ 15]

Syntax

Deklaration:

```
fbFilter : FB_FTR_IIRCoeff(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_IIRCoeff
VAR_INPUT
    stConfig        : ST_FTR_IIRCoeff;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_IIRCoeff [▶ 77]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.

Name	Definitionsort	Beschreibung
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.1.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_IIRCoeff;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_IIRCoeff [▶ 77]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_IIRCoeff();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.pCoefficientArrayAdr_A := ADR(aNewArray); (*change coefficients of denominator*)
    stParams.nCoefficientArraySize_A := SIZEOF(aNewArray);
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.1.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn        : POINTER TO LREAL;
     nSizeIn    : UDINT;
     pOut       : POINTER TO LREAL;
     nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.1.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

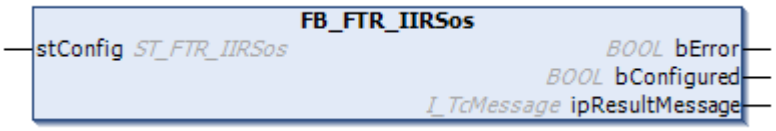
Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.2 FB_FTR_IIRSos



Der Funktionsbaustein realisiert einen IIR-Filter (**I**nfinite-**I**mpulse-**R**esponse-Filter). Die Übertragungsfunktion lautet:

$$G(z) = \prod_{m=1}^{M} G_m(z) = \prod_{m=1}^{M} \frac{b_{0m} + b_{1m}z^{-1} + b_{2m}z^{-2}}{a_{0m} + a_{1m}z^{-1} + a_{2m}z^{-2}}$$

Die Koeffizienten der Übertragungsfunktion sind frei definierbar und können mit der Struktur ST_FTR_IIRSos übergeben werden.

Siehe auch:

Digitale Filter [▶ 15]

Syntax

Deklaration:

```
fbFilter : FB_FTR_IIRSos(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_IIRSos
VAR_INPUT
    stConfig        : ST_FTR_IIRSos;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_IIRSos [▶ 77]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

Siehe auch:

FB_FTR_IIRSpec [▶ 36]

Filtertypen und Parametrierung [▶ 17]

5.1.2.1 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.2.2 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.2.3 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_IIRSos;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_IIRSos [▶ 77]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_IIRSos();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.pCoefficientArrayAdr_Sos := ADR(aNewArray); (*change coefficients*)
    stParams.nCoefficientArraySize_Sos := SIZEOF(aNewArray);
```

```
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.3 FB_FTR_IIRSpec

Der Funktionsbaustein realisiert einen IIR-Filter (Infinite-Impulse-Response-Filter).

Die Filterkoeffizienten der Übertragungsfunktion werden intern auf Basis der übergebenen Filterspezifikation in Form von Biquads berechnet. Die Filterspezifikation wird mit der Struktur ST_FTR_IIRSpec übergeben. Spezifiziert werden können Filter vom Typ „Butterworth“, „Chebyshev“ und „Bessel“. Dabei können jeweils Tief- und Hochpässe sowie Bandpass- und Bandstopp-Filter definiert werden.

Siehe auch:

Digitale Filter [▶ 15], Filtertypen und Parametrierung [▶ 17]

Syntax

Deklaration:

```
fbFilter : FB_FTR_IIRSpec(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_IIRSpec
VAR_INPUT
    stConfig        : ST_FTR_IIRSpec;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_IIRSpec [▶ 78]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.

Name	Definitionsort	Beschreibung
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

Siehe auch:

Filtertypen und Parametrierung [▶ 17]

FB_FTR_IIRSos [▶ 33]

5.1.3.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_IIRSpec;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_IIRSpec [▶ 78]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_IIRSpec();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fCutoff := 50; (*change cutoff freq.*)
```

```
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.3.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.3.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.4 FB_FTR_MovAvg

Der Funktionsbaustein realisiert einen Moving-Average-Filter. Das Filter bildet den Mittelwert aus M Eingangswerten x[n-k] mit k = 0 .. M - 1.

$$y[n] = \frac{1}{M}(x[n] + x[n-1] + ...x[n-M+1])$$

Die Filterspezifikation wird mit der Struktur ST_FTR_MovAvg übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_MovAvg(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_MovAvg
VAR_INPUT
    stConfig        : ST_FTR_MovAvg;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_MovAvg [▶ 79]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.4.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_MovAvg;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_MovAvg [▶ 79]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_MovAvg ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.nSamplesToFilter := 11; (*change filter order*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.4.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn        : POINTER TO LREAL;
     nSizeIn    : UDINT;
     pOut       : POINTER TO LREAL;
     nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.4.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.5 FB_FTR_PT1

Der Funktionsbaustein FB_FTR_PT1 realisiert ein Verzögerungsglied 1. Ordnung (PT1-Glied) mit der komplexen Übertragungsfunktion (Laplace-Bereich):

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

Die Filterspezifikation wird mit der Struktur ST_FTR_PT1 übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_PT1(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_PT1
VAR_INPUT
    stConfig        : ST_FTR_PT1;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT1 [▶ 79]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.5.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_PT1;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT1 [▶ 79]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PT1 ();
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fKp := 2; (*change gain*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.5.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pIn        : POINTER TO LREAL;
    nSizeIn    : UDINT;
    pOut       : POINTER TO LREAL;
    nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.5.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.6 FB_FTR_PT2

Der Funktionsbaustein FB_FTR_PT2 realisiert ein Verzögerungsglied 2. Ordnung (PT2-Glied) mit der komplexen Übertragungsfunktion (Laplace-Bereich):

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

Die Filterspezifikation wird mit der Struktur ST_FTR_PT2 übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_PT2(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_PT2
VAR_INPUT
    stConfig        : ST_FTR_PT2;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT2 [▶ 80]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.6.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_PT2;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT2 [▶ 80]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PT2 ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fKp := 2; (*change gain*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.6.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pIn        : POINTER TO LREAL;
    nSizeIn    : UDINT;
    pOut       : POINTER TO LREAL;
    nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.6.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.7 FB_FTR_PT3

Der Funktionsbaustein FB_FTR_PT3 realisiert ein Verzögerungsglied 3. Ordnung (PT3-Glied) mit der komplexen Übertragungsfunktion (Laplace-Bereich):

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

Die Filterspezifikation wird mit der Struktur ST_FTR_PT3 übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_PT3(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_PT3
VAR_INPUT
    stConfig        : ST_FTR_PT3;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT3 [▶ 80]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.7.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_PT3;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT3 [▶ 80]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PT3 ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fKp := 2; (*change gain*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.7.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.7.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.8 FB_FTR_PTn

Der Funktionsbaustein FB_FTR_PTn realisiert ein Verzögerungsglied n-ter Ordnung mit gleichen Zeitkonstanten. Die komplexe Übertragungsfunktion (Laplace-Bereich):

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

Die Filterspezifikation wird mit der Struktur ST_FTR_PTn übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_PTn(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_PTn
VAR_INPUT
    stConfig        : ST_FTR_PTn;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PTn [▶ 81]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.8.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_PTn;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PTn [▶ 81]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PTn ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fKp := 2; (*change gain*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.8.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.8.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

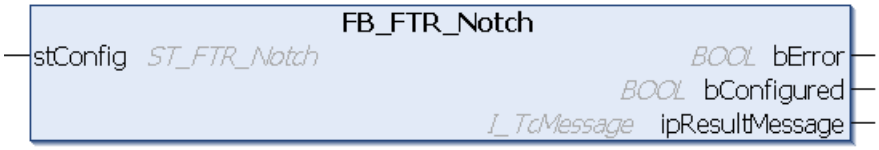
Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.9 FB_FTR_Notch



Der Funktionsbaustein FB_FTR_Notch realisiert einen Bandstopp-Filter mit schmaler Bandbreite. Die komplexe Übertragungsfunktion (Laplace-Bereich):

$$G(s) = \frac{s^2 + \Omega_{\text{Notch}}^2}{(s^2 + \Omega_{\text{Notch}}/Q)^2 + \Omega_{\text{Notch}}^2}$$

$$\Omega_{\text{Notch}} = \frac{2}{T} \tan\left(\pi \frac{f_{\text{Notch}}}{f_{\text{sample}}}\right)$$

Die Filterspezifikation wird mit der Struktur ST_FTR_Notch übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_Notch(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_Notch
VAR_INPUT
    stConfig        : ST_FTR_Notch;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Notch [▶ 81]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.9.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_Notch;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Notch [▶ 81]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_Notch ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE;
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fQ := 20; (*change quality factor*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.9.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn        : POINTER TO LREAL;
     nSizeIn    : UDINT;
     pOut       : POINTER TO LREAL;
     nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.9.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

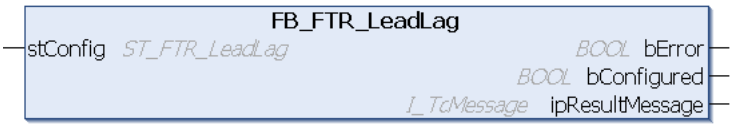
Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.10 FB_FTR_LeadLag



Der Funktionsbaustein FB_FTR_LeadLag realisiert ein Phasenkorrekturglied 1. Ordnung mit der komplexen Übertragungsfunktion (Laplace-Bereich):

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

Die Filterspezifikation wird mit der Struktur ST_FTR_LeadLag übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_LeadLag(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_LeadLag
VAR_INPUT
    stConfig        : ST_FTR_LeadLag;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_LeadLag [▶ 82]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.10.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_LeadLag;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Notch [▶ 82]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_LeadLag ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE;
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
```

```
    stParams.fT1      := 0.0002; (*change time constant T1*)
    bSucceed          := fbFilter.Configure(stConfig := stParams);
    bReconfigure      := FALSE;
END_IF
```

5.1.10.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.10.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

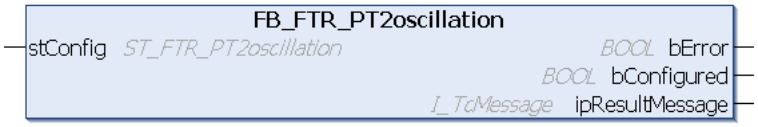
Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.11 FB_FTR_PT2oscillation



Der Funktionsbaustein FB_FTR_PT2oscillation realisiert ein schwingungsfähiges Verzögerungsglied 2. Ordnung mit der komplexen Übertragungsfunktion (Laplace-Bereich):

$$G(s) = \frac{K_p}{1 + 2\vartheta T_1 s + (T_1 s)^2}$$

Die Filterspezifikation wird mit der Struktur ST_FTR_PT2oscillation übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_PT2oscillation(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_PT2oscillation
VAR_INPUT
    stConfig        : ST_FTR_PT2oscillation;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT2oscillation [▶ 83]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.11.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_PT2oscillation;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT2oscillation [▶ 83]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PT2oscillation ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE;
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fTheta  := 0.7; (*change damping factor*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.11.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn        : POINTER TO LREAL;
     nSizeIn    : UDINT;
     pOut       : POINTER TO LREAL;
     nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.11.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.12 FB_FTR_PTt

Der Funktionsbaustein FB_FTR_PTt realisiert ein Totzeitglied (PTt-Glied) mit der komplexen Übertragungsfunktion (Laplace-Bereich):

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

Die Filterspezifikation wird mit der Struktur ST_FTR_PTt übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_PTt(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_PTt
VAR_INPUT
    stConfig        : ST_FTR_PTt;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PTt [▶ 83]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.12.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
    stConfig : ST_FTR_PTt;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PTt [▶ 83]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PTt ();
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fKp := 2; (*change gain*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.12.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn        : POINTER TO LREAL;
     nSizeIn    : UDINT;
     pOut       : POINTER TO LREAL;
     nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.12.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.13 FB_FTR_Median

Der Funktionsbaustein realisiert einen Median-Filter. 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.

Die Filterspezifikation wird mit der Struktur ST_FTR_Median übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_Median(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_Median
VAR_INPUT
    stConfig        : ST_FTR_Median;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Median [▶ 84]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.13.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
     stConfig : ST_FTR_Median;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Median [▶ 84]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_Median ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.nSamplesToFilter := 11; (*change filter order*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.13.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn        : POINTER TO LREAL;
     nSizeIn    : UDINT;
     pOut       : POINTER TO LREAL;
     nSizeOut   : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays

Name	Typ	Beschreibung
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.13.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.14 FB_FTR_ActualValue

Der Funktionsbaustein ermöglicht eine Plausibilitätskontrolle und Filterung einer gemessenen Eingangsgröße.

Wenn die Differenz zweier aufeinander folgender Abtastwerte 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 linear extrapoliert. Wenn die vorgegebene Schranke länger als 3 Zyklen überschritten wird, folgt der Ausgang wieder der Eingangsgröße.

$$y_k = \begin{cases} x_k & : \quad |x_k - x_{k-1}| <= fDeltaMax \\ y_{temp} & : \quad |x_k - x_{k-1}| > fDeltaMax \end{cases}$$

$$y_{temp} = \begin{cases} y_{k-1} + \frac{(y_{k-1} - y_{k-2}) + (y_{k-2} - y_{k-3})}{2} & : \quad n <= 3 \\ y_{k-1} + \frac{(x_k - y_{k-1})}{2} & : \quad n = 4 \\ y_k & : \quad n = 5 \end{cases}$$

Variable n wird inkrementiert, wenn $|x_k - x_{k-1}| > fDeltaMax$.

Die Filterspezifikation wird mit der Struktur ST_FTR_ActualValue [▶ 84] übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_ActualValue(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_ActualValue
VAR_INPUT
    stConfig        : ST_FTR_ActualValue;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_ActualValue [▶ 84]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt.

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.
GetFilterActive()	Lokal	Zeigt an, welche Elemente des Eingangssignals verändert wurden.

Name	Definitionsort	Beschreibung
GetFilterActiveTimestamps()	Lokal	Zeigt an, wann der Filter zuletzt aktiv war.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical.
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events.
nTimestamp	UINT	Set	Lokal	0	Zeitstempel des ältesten Eingangswert des nächsten Call()-Aufrufs.
tSamplePeriod	LTIME	Set	Lokal	0	Zeitdifferenz zwischen zwei Eingangswerten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.14.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
     stConfig : ST_FTR_Median;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Median [▶ 84]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_Median ();
```

```
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
```

```
IF bReconfigure THEN
    stParams.nSamplesToFilter := 11; (*change filter order*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.14.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.14.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.1.14.4 GetFilterActive

Die Methode errechnet ausgehend vom letzten Call()-Aufruf, welche Elemente des Eingangssignals verändert wurden.

Syntax

```
METHOD GetFilterActive: BOOL
VAR_INPUT
     pFilterActive        : POINTER TO BOOL;
     nSizeGetFilterActive : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pFilterActive	POINTER TO BOOL	Adresse des Eingangsarrays
nSizeGetFilterActive	UDINT	Größe des Eingangsarrays

Rückgabewert

Name	Typ	Beschreibung
GetFilterActive	BOOL	Liefert TRUE, wenn Aktivzustand berechnet wurde.

Beispiel

```
aFilterActive : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF BOOL;
bSucceed := fbFilter.GetFilterActive(ADR(aFilterActive),SIZEOF(aFilterActive));
```

5.1.14.5 GetFilterActiveTimestamps

Die Methode errechnet ausgehend vom letzten Call()-Aufruf, wann das Filter zuletzt aktiv war.

Syntax

```
METHOD GetFilterActiveTimestamps: BOOL
VAR_INPUT
     pFilterActiveTimestamps : POINTER TO BOOL;
     nSizeFilterActiveTimestamps : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pFilterActiveTimestamps	POINTER TO ULINT	Adresse des Eingangsarrays
nSizeFilterActiveTimestamps	UDINT	Größe des Eingangsarrays

Rückgabewert

Name	Typ	Beschreibung
GetFilterActiveTimestamps	BOOL	Liefert TRUE, wenn die Zeitstempel berechnet wurden.

Beispiel

```
aFilterActiveTimeStamps : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF ULINT;
```

```
bSucceed := fbFilter. GetFilterActiveTimestamps
(ADR(aFilterActiveTimeStamps),SIZEOF(aFilterActiveTimeStamps));
```

5.1.15 FB_FTR_Gaussian

Der Funktionsbaustein FB_FTR_Gaussian realisiert einen Gauß-Filter. Das Filter berechnet den gewichteten Mittelwert aus M Eingangswerten. Die Gewichte werden über eine Gaußfunktion berechnet, wobei die Standardabweichung σ die Grenzfrequenz des Filters bestimmt.

Die Grenzfrequenz f_c ist über die Standardabweichung σ und die Abtastrate f_s so definiert, dass der normierte Amplitudengang den Wert 1/sqrt(2) ≈ -3dB an der Grenzfrequenz annimmt.

$$f_c = \frac{f_s}{2\pi\sigma} \cdot \sqrt{\ln(2)}$$

Die Filterspezifikation wird mit der Struktur ST_FTR_Gaussian [▶ 85] übergeben.

Syntax

Deklaration:

```
fbFilter : FB_FTR_Gaussian(stConfig := ...)
```

Definition:

```
FUNCTION_BLOCK FB_FTR_Gaussian
VAR_INPUT
    stConfig        : ST_FTR_Gaussian;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    bConfigured     : BOOL;
    ipResultMessage : I_TCMessage;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_Gaussian [▶ 85]	Struktur zur Konfiguration des Filterverhaltens

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn ein Fehler auftritt.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war.

Name	Typ	Beschreibung
ipResultMessage	I_TCMessage	Schnittstelle, die Eigenschaften und Methoden zum Messagehandling bereitstellt

Methoden

Name	Definitionsort	Beschreibung
Configure()	Lokal	Lädt eine neue (oder initiale) Konfigurationsstruktur.
Call()	Lokal	Berechnet das Ausgangssignal bei gegebenem Eingangssignal und Konfiguration des Filters.
Reset()	Lokal	Setzt interne Zustände zurück.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Initialwert	Beschreibung
bTraceLevelDefault	BOOL	Get, Set	Lokal	TRUE	TRUE, wenn eTraceLevel = Critical
eTraceLevel	TcEventSeverity	Get, Set	Lokal	Critical	Severity eines Events

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.25	PC oder CX (x64, x86)	Tc3_Filter

5.1.15.1 Configure

Die Methode kann zur Laufzeit verwendet werden, um die Instanz eines Filters initial zu konfigurieren (wenn nicht schon in der Deklaration geschehen) oder zu rekonfigurieren.

Wenn eine Filterinstanz nicht konfiguriert ist, können die Methoden `Call()` und `Reset()` nicht verwendet werden.

Syntax

```
METHOD Configure : BOOL
VAR_INPUT
     stConfig : ST_FTR_PT1;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stConfig	ST_FTR_PT1 [▶ 79]	Struktur zur Konfiguration des Filterverhaltens

Rückgabewert

Name	Typ	Beschreibung
Configure	BOOL	TRUE, wenn die Filterinstanz erfolgreich konfiguriert wurde.

Beispiel

```
(*Declaration without configuration*)
fbFilter : FB_FTR_PT1 ();
(* initial configuration of fbFilter *)
IF bInit THEN
    bSucceed := fbFilter.Configure(stConfig := stParams);
```

```
    bInit    := FALSE
END_IF

(* reconfigure fbFilter on bReconfigure = TRUE *)
IF bReconfigure THEN
    stParams.fKp := 2; (*change gain*)
    bSucceed         := fbFilter.Configure(stConfig := stParams);
    bReconfigure     := FALSE;
END_IF
```

5.1.15.2 Call

Die Methode berechnet aus einem Eingangssignal, das in Form eines Pointers übergeben wird, ein manipuliertes Ausgangssignal.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
     pIn         : POINTER TO LREAL;
     nSizeIn     : UDINT;
     pOut        : POINTER TO LREAL;
     nSizeOut    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pIn	POINTER TO LREAL	Adresse des Eingangsarrays
nSizeIn	UDINT	Größe des Eingangsarrays
pOut	POINTER TO LREAL	Adresse des Ausgangsarrays
nSizeOut	UDINT	Größe des Ausgangsarrays

Rückgabewert

Name	Typ	Beschreibung
Call	BOOL	Liefert TRUE, wenn ein manipuliertes Ausgangssignal berechnet wurde.

Beispiel

```
aInput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
aOutput : ARRAY [1..cChannels] OF ARRAY [1..cOversamples] OF LREAL;
```

```
bSucceed := fbFilter.Call(ADR(aInput), SIZEOF(aInput), ADR(aOutput), SIZEOF(aOutput));
```

5.1.15.3 Reset

Die Methode setzt den internen Status des Filters zurück. Der aktuelle Ausgangswert y[n] setzt sich aus dem aktuellen Eingangswert x[n] sowie vergangenen Eingangs- und Ausgangswerten (x[n-k] und y[n-k], k > 0) zusammen. Durch das Reset des Funktionsbausteins wird der Filter in seinen Ursprungszustand, d. h. ohne Vergangenheitseinfluss, versetzt. Der Filter wird also wieder auf den letzten Konfigurationszustand zurückgesetzt.

Siehe auch: Digitale Filter [▶ 15]

Syntax

```
METHOD Reset : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Reset	BOOL	Liefert TRUE, wenn der interne Status des Filters erfolgreich zurückgesetzt wurde.

5.2 Datentypen

5.2.1 Konfigurationsstrukturen

Allgemeine Beschreibung

Für jeden Funktionsbaustein FB_FTR_<type> existiert eine individuelle Konfigurationsstruktur ST_FTR_<type>. In der Konfigurationsstruktur werden alle Parameter definiert, die zur Berechnung der Übertragungsfunktion, der Ein- und Ausgangsgrößen (Größe und Form der Arrays) sowie der internen Zustände benötigt werden.

Gemeinsame Parameter

Alle Konfigurationsstrukturen ST_FTR_<type> enthalten folgende Parameter:

Parameter	Typ	Beschreibung
`nOversamples`	UDINT	Anzahl der Oversamples > 0
`nChannels`	UDINT	Anzahl der Kanäle > 0 & < 101
`pInitialValues`	POINTER TO LREAL	Pointer auf Array mit Initialwerten (optional)
`nInitialValuesSize`	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

nOversamples und nChannles

Die Parameter `nOversamples` und `nChannels` beschreiben die Größe der Ein- und Ausgangsarrays, die beim Aufruf der Methode `Call()` übergeben werden. Wenn z. B. mit 10 Oversamples und 3 Kanälen gearbeitet wird, erwartet die Methode `Call()` ein Array mit 30 Elementen am Eingang und am Ausgang (siehe OverSampling und Kanäle [▶ 28]).

Während der Parameter `nChannels` die Anzahl der mit einem Aufruf zu verarbeitenden parallelen Signalkanäle beschreibt, beschreibt der Parameter `nOversamples` die für jeden einzelnen Kanal auftretende Anzahl von Abtastwerten pro Aufruf der Methode `Call()`.

Inital Values

Die optionalen Parameter `pInitialValues` und `nInitialValuesSize` werden verwendet, um den internen Zustand des Filters nach dessen Konfiguration zu definieren. Praktisch dienen die Parameter dazu, die Einschwingzeit des konfigurierten Filters durch das Einbringen von Vorwissen zu reduzieren. Dazu müssen die Vergangenheitswerte y[n-k] und x[n-k] mit k > 0 in der Differenzengleichung gezielt gesetzt werden.

Für die Funktionsbausteine FB_FTR_IIRCoeff [▶ 30], FB_FTR_MovAvg [▶ 39], FB_FTR_PT1 [▶ 41], FB_FTR_PT2 [▶ 44] und FB_FTR_PT3 [▶ 47] gilt die folgende Struktur der Differenzengleichung:

$$a_0 y[n] + \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

Zur kompakteren Darstellung werden folgende Schreibweisen für die Vergangenheitswerte verwendet: $x_{i,k}$ und $y_{i,k}$ mit Kanal i und zeitlicher Verzögerung k > 0. Beispiel: Kanal i=2 und Verzögerung k=3 entspricht $y_{2,3}$ = y_2[n-3].

Verwendungsmöglichkeiten:

- `InitialValues = nullptr`
  Alle Vergangenheitswerte $y_{i,k}$ und $x_{i,k}$ mit k > 0 werden für alle Kanäle i auf Null gesetzt. Dies entspricht dem Verhalten, wenn die Einträge der Struktur nicht genutzt werden.
- `InitialValues = P`
  Alle Vergangenheitswerte $x_{i,k}$ mit k > 0 werden für alle Kanäle i auf den Wert P gesetzt. Alle Vergangenheitswerte $y_{i,k}$ mit k > 0 werden für alle Kanäle i auf V*P gesetzt, wobei V der Gleichspannungsverstärkung (DC-Gain) des Filters entspricht.
  Das Filter befindet sich entsprechend für alle Kanäle im eingeschwungenen Zustand bezüglich eines konstanten Eingangssignals x[n] = P.
- `InitialValues = [`$x_{1,1}, x_{1,2}, \dots, x_{1,M}, y_{1,1}, y_{1,2}, \dots, y_{1,N}$`]`
  Alle Vergangenheitswerte $x_{i,k}$ mit k > 0 und $y_{i,k}$ mit k > 0 werden individuell, aber für alle Kanäle i gleich, gesetzt.
- `InitialValues = [`$x_{1,1}, x_{1,2}, \dots, x_{1,M}, y_{1,1}, y_{1,k}, \dots, y_{1,N}$,
  $x_{2,1}, x_{2,2}, \dots, x_{2,M}, y_{2,1}, y_{2,2}, \dots, y_{2,N}$,
  .....
  $x_{I,1}, x_{I,2}, \dots, x_{I,M}, y_{I,1}, y_{I,2}, \dots, y_{I,N}$ `]`
  Alle Vergangenheitswerte $x_{i,k}$ und $y_{i,k}$ werden für jeden Kanal i=1..I individuell gesetzt.

Für die Funktionsbausteine FB_FTR_IIRSos [▶ 33] und FB_FTR_IIRSpec [▶ 36] wird intern die Übertragungsfunktion in second-order sections (SOS) berechnet:

$$G(z) = \prod_{m=1}^{M} G_m(z) = \prod_{m=1}^{M} \frac{b_{0m} + b_{1m}z^{-1} + b_{2m}z^{-2}}{a_{0m} + a_{1m}z^{-1} + a_{2m}z^{-2}}$$

Die kompakte Darstellung wird entsprechend um m erweitert, welche den Biquad angibt: $x^{(m)}_{i,k}$ und $y^{(m)}_{i,k}$.
Beispiel: Kanal i=2 und Verzögerung k=1 des vierten Biquad (m=4) entspricht
$y^{(4)}_{2,1} = y^{(4)}_{2}[n-1]$. Jeder Biquad kann hier nur 2 Vergangenheitswerte nutzen, entsprechend ist k = 1,2.

Daher ist dort die Strukturierung für die Ein- und Ausgangswerte anders.

Verwendungsmöglichkeiten:

- `InitialValues = nullptr`
  Gleiches Verhalten wie oben beschreiben.
- `InitialValues = P`
  Gleiches Verhalten wie oben beschreiben.
- `InitialValues = [`$x^{(1)}_{1,1}, x^{(1)}_{1,2}, x^{(2)}_{1,1}, x^{(2)}_{1,2}, \dots, x^{(M)}_{1,1}, x^{(M)}_{1,2}$,
  $y^{(1)}_{1,1}, y^{(1)}_{1,2}, y^{(2)}_{1,1}, y^{(2)}_{1,2}, \dots, y^{(M)}_{1,1}, y^{(M)}_{1,2}$`]`
  Alle Vergangenheitswerte $x^{(m)}_{i,k}$ und $y^{(m)}_{i,k}$ werden individuell, aber für alle Kanäle i gleich, gesetzt.
- `InitialValues = [`$x^{(1)}_{1,1}, x^{(1)}_{1,2}, x^{(2)}_{1,1}, x^{(2)}_{1,2}, \dots, x^{(M)}_{1,1}, x^{(M)}_{1,2}$,
  $y^{(1)}_{1,1}, y^{(1)}_{1,2}, y^{(2)}_{1,1}, y^{(2)}_{1,2}, \dots, y^{(M)}_{1,1}, y^{(M)}_{1,2}$,
  $x^{(1)}_{2,1}, x^{(1)}_{2,2}, x^{(2)}_{2,1}, x^{(2)}_{2,2}, \dots, x^{(M)}_{2,1}, x^{(M)}_{2,2}$,
  $y^{(1)}_{2,1}, y^{(1)}_{2,2}, y^{(2)}_{2,1}, y^{(2)}_{2,2}, \dots, y^{(M)}_{2,1}, y^{(M)}_{2,2}$,
  .....
  $x^{(1)}_{I,1}, x^{(1)}_{I,2}, x^{(2)}_{I,1}, x^{(2)}_{I,2}, \dots, x^{(M)}_{I,1}, x^{(M)}_{I,2}$,
  $y^{(1)}_{I,1}, y^{(1)}_{I,2}, y^{(2)}_{I,1}, y^{(2)}_{I,2}, \dots, y^{(M)}_{I,1}, y^{(M)}_{I,2}$`]`
  Alle Vergangenheitswerte $x^{(m)}_{i,k}$ und $y^{(m)}_{i,k}$ werden für jeden Kanal i=1..I individuell gesetzt.

Der Funktionsbaustein FB_FTR_PTn [▶ 50] wird ebenfalls als kaskadierter Filter beschrieben, jedoch als kaskadierter Filter erster Ordnung. Entsprechend gilt hier die Betrachtung wie für die SOS Bausteine, jedoch mit k = 1. Für einen PTn Filter mit n = M gilt dann:

- `InitialValues = nullptr`
  Gleiches Verhalten wie oben beschreiben.
- `InitialValues = P`
  Gleiches Verhalten wie oben beschreiben.

- InitialValues = $[x^{(1)}_{1,1}, x^{(2)}_{1,1}, \ldots, x^{(M)}_{1,1}, y^{(1)}_{1,1}, y^{(2)}_{1,1}, \ldots, y^{(M)}_{1,1}]$
  Alle Vergangenheitswerte $x^{(m)}_{i,k}$ und $y^{(m)}_{i,k}$ werden individuell, aber für alle Kanäle i gleich, gesetzt.
- InitialValues = $[x^{(1)}_{1,1}, x^{(2)}_{1,1}, \ldots, x^{(M)}_{1,1}, y^{(1)}_{1,1}, y^{(2)}_{1,1}, \ldots, y^{(M)}_{1,1}, x^{(1)}_{2,1}, x^{(2)}_{2,1}, \ldots, x^{(M)}_{2,1}, y^{(1)}_{2,1}, y^{(2)}_{2,1}, \ldots, y^{(M)}_{2,1}, \ldots\ldots, x^{(1)}_{I,1}, x^{(2)}_{I,1}, \ldots, x^{(M)}_{I,1}, y^{(1)}_{I,1}, y^{(2)}_{I,1}, \ldots, y^{(M)}_{I,1}]$
  Alle Vergangenheitswerte $x^{(m)}_{i,k}$ und $y^{(m)}_{i,k}$ werden für jeden Kanal i=1..I individuell gesetzt.

5.2.1.1 ST_FTR_IIRCoeff

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_IIRCoeff [▶ 30].

Syntax

Definition:

```
TYPE ST_FTR_IIRCoeff :
STRUCT
    pCoefficientArrayAdr_A  : POINTER TO LREAL;
    nCoefficientArraySize_A : UDINT;
    pCoefficientArrayAdr_B  : POINTER TO LREAL;
    nCoefficientArraySize_B : UDINT;
    bReset                  : BOOL := TRUE;
    nOversamples            : UDINT;
    nChannels               : UDINT;
    pInitialValues          : POINTER TO LREAL;
    nInitialValuesSize      : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
pCoefficientArrayAdr_A	Pointer to LREAL	Pointer auf Array mit Filterkoeffizienten a_k (Nenner) [a_0, a_1, a_2, …, a_N]
nCoefficientArraySize_A	UDINT	Größe des Arrays [a_0, a_1, a_2, …, a_N] in BYTE
pCoefficientArrayAdr_B	Pointer to LREAL	Pointer auf Array mit Filterkoeffizienten b_k (Zähler) [b_0, b_1, b_2, …, b_M]
nCoefficientArraySize_B	UDINT	Größe des Arrays [b_0, b_1, b_2, …, b_M] in BYTE
bReset	BOOL	Wenn TRUE, wird Reset ausgeführt, wenn der Filter konfiguriert wird. Wenn FALSE, werden die Vergangenheitswerte x[n-k] und y[n-k] nicht zurückgesetzt.
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.2 ST_FTR_IIRSos

Konfigurationsstruktur für den FunktionsbausteinFB_FTR_IIRSos [▶ 33].

Syntax

Definition:

```
TYPE ST_FTR_IIRSos :
STRUCT
    pCoefficientArrayAdr_Sos  : POINTER TO LREAL;
    nCoefficientArraySize_Sos : UDINT;
    bReset                    : BOOL := TRUE;
    nOversamples              : UDINT;
    nChannels                 : UDINT;
    pInitialValues            : POINTER TO LREAL;
    nInitialValuesSize        : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
pCoefficientArrayAdr_Sos	Pointer to LREAL	Pointer auf Array mit Filterkoeffizienten [b_{01}, b_{11}, b_{21},a_{01}, a_{11}, a_{21}, b_{02}, b_{12}, b_{22},a_{02}, a_{12}, a_{22}, …. b_{0M}, b_{1M}, b_{2M},a_{0M}, a_{1M}, a_{2M}]
nCoefficientArraySize_Sos	UDINT	Größe des Arrays mit Filterkoeffizienten in BYTE
bReset	BOOL	Wenn TRUE, wird Reset ausgeführt, wenn der Filter konfiguriert wird. Wenn FALSE, werden die Vergangenheitswerte x[n-k] und y[n-k] nicht zurückgesetzt.
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

Ein Beispiel für ein Filter mit 3 Biquads kann so aussehen:

```
aCoeffs_Sos : ARRAY [1..3] OF ARRAY [1..6] OF LREAL :=
              [ [1, 1, 1, 1, 1, 1],   // [b01,b11,b21,a01,a11,a21]
              [ [1, 1, 1, 1, 1, 1],   // [b02,b12,b22,a02,a12,a22]
              [ [1, 1, 1, 1, 1, 1]];  // [b03,b13,b23,a03,a13,a23
```

5.2.1.3 ST_FTR_IIRSpec

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_IIRSpec [▶ 36].

Syntax

Definition:

```
TYPE ST_FTR_IIRSpec :
STRUCT
    eFilterName        : E_FTR_Name;
    eFilterType        : E_FTR_Type;
    nFilterOrder       : UDINT;
    fCutoff            : LREAL;
    fBandwidth         : LREAL;
    fPassBandRipple    : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
eFilterName	E_FTR_Name [▶ 85]	Beschreibt die Filterrealisierung (Butterworth, Chebyshev, Bessel)
eFilterType	E_FTR_Type [▶ 85]	Beschreibt den Filtertyp (Hochpass, Tiefpass, …)

Name	Typ	Beschreibung
nFilterOrder	UDINT	Filterordnung (max. 20 für Hoch- und Tiefpass, max 10 für Bandpass und Bandstopp)
fCutoff	LREAL	Grenzfrequenz in Hz (größer 0 und kleiner fSamplingRate/2)
fBandwidth	LREAL	Bandbreite in Hz bezüglich Bandpass und Bandstopp.
fPassbandRipple	LREAL	Welligkeit des Amplitudengangs im Durchlassbereich des Filters in dB (größer 0)
fSamplingRate	LREAL	Abtastrate f_s in Hz
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

Hinweise zum Einfluss der Parameter auf die Filter vom Typ Butterworth, Chebyshev und Bessel finden Sie hier: Filtertypen und Parametrierung [▶ 17].

5.2.1.4 ST_FTR_MovAvg

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_MovAvg [▶ 39].

Syntax

Definition:

```
TYPE ST_FTR_MovAvg :
STRUCT
    nSamplesToFilter   : UDINT:
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
nSamplesToFilter	UDINT	Anzahl der Samples zur Bildung des gleitenden Mittelwerts (oft als Fenstergröße bezeichnet)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.5 ST_FTR_PT1

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PT1 [▶ 41].

Syntax

Definition:

```
TYPE ST_FTR_PT1 :
STRUCT
    fKp                : LREAL;
    fT1                : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
```

```
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fKp	LREAL	Verstärkungsfaktor (größer Null)
fT1	LREAL	Zeitkonstante T_1 in Sekunden (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.6 ST_FTR_PT2

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PT2 [▶ 44].

Syntax

Definition:

```
TYPE ST_FTR_PT2 :
STRUCT
    fKp                : LREAL;
    fT1                : LREAL;
    fT2                : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fKp	LREAL	Verstärkungsfaktor (größer Null)
fT1	LREAL	Zeitkonstante T_1 in Sekunden (größer Null)
fT2	LREAL	Zeitkonstante T_2 in Sekunden (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.7 ST_FTR_PT3

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PT3 [▶ 47].

Syntax

Definition:

```
TYPE ST_FTR_PT3 :
STRUCT
    fKp                : LREAL;
```

```
    fT1                : LREAL;
    fT2                : LREAL;
    fT3                : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fKp	LREAL	Verstärkungsfaktor (größer Null)
fT1	LREAL	Zeitkonstante T_1 in Sekunden (größer Null)
fT2	LREAL	Zeitkonstante T_2 in Sekunden (größer Null)
fT3	LREAL	Zeitkonstante T_3 in Sekunden (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.8 ST_FTR_PTn

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PTn [▶ 50].

Syntax

Definition:

```
TYPE ST_FTR_PTn :
STRUCT
    fKp                : LREAL;
    fT1                : LREAL;
    nOrder             : UDINT;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fKp	LREAL	Verstärkungsfaktor (größer Null)
fT1	LREAL	Zeitkonstante T_1 in Sekunden (größer Null)
nOrder	UDINT	Ordnung des Filters (1..10)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.9 ST_FTR_Notch

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_Notch [▶ 53].

Syntax

Definition:

```
TYPE ST_FTR_Notch :
STRUCT
    fNotchFrequency    : LREAL;
    fQ                 : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fNotchFrequency	LREAL	Notchfrequenz in Hz (größer 0 und kleiner fSamplingRate/2)
fQ	LREAL	Q-Faktor = fNotchFrequency/Bandbreite (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

Beispiel zur Konfiguration

Sie möchten beispielsweise einen Notch-Filter erstellen, der die Frequenz 100 Hz stark dämpft. Sie erlauben dabei eine einseitige Bandbreite um diese 100 Hz von 10 Hz, d.h. Sie erwarten die -3 dB-Grenze bei 90 Hz und 110 Hz. Definieren dazu:

```
fNotchFrequency := 100;
fQ := 10; (* B = fNotchFrequency/fQ = 10 Hz *)
```

5.2.1.10 ST_FTR_LeadLag

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PT2 [▶ 44].

Syntax

Definition:

```
TYPE ST_FTR_PT2 :
STRUCT
    fT1                : LREAL;
    fT2                : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fT1	LREAL	Zeitkonstante T_1 in Sekunden (größer Null)
fT2	LREAL	Zeitkonstante T_2 in Sekunden (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)

Name	Typ	Beschreibung
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.11 ST_FTR_PT2oscillation

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PT2oscillation [▶ 59].

Syntax

Definition:

```
TYPE ST_FTR_PT2 :
STRUCT
    fKp                : LREAL;
    fT1                : LREAL;
    fTheta             : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fKp	LREAL	Verstärkungsfaktor (größer Null)
fT1	LREAL	Zeitkonstante T_1 in Sekunden (größer Null)
fTheta	LREAL	Dämpfungsfaktor (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.12 ST_FTR_PTt

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_PTt [▶ 62].

Syntax

Definition:

```
TYPE ST_FTR_PTt :
STRUCT
    fKp                : LREAL;
    fTt                : LREAL;
    fSamplingRate      : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fKp	LREAL	Verstärkungsfaktor (größer Null)
fTt	LREAL	Totzeit T_t in Sekunden (größer Null)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)

Name	Typ	Beschreibung
nOversamples	UDINT	Anzahl der Oversamples (größer Null und Vielfaches von 1/fSamplingRate)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)

5.2.1.13 ST_FTR_Median

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_Median [▶ 64].

Syntax

Definition:

```
TYPE ST_FTR_Median :
STRUCT
    nSamplesToFilter   : UDINT;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
nSamplesToFilter	UDINT	Anzahl der Samples zur Bildung des gleitenden Mittelwerts (oft als Fenstergröße bezeichnet)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.14 ST_FTR_ActualValue

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_ActualValue [▶ 67].

Syntax

Definition:

```
TYPE ST_FTR_ActualValue :
STRUCT
    fDeltaMax            : LREAL;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fDeltaMax	LREAL	Maximale Differenz zweier aufeinander folgender Eingangswerte (größer oder gleich Null)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.1.15 ST_FTR_Gaussian

Konfigurationsstruktur für den Funktionsbaustein FB_FTR_Gaussian [▶ 72].

Syntax

Definition:

```
TYPE ST_FTR_Gaussian :
STRUCT
    fCutoff            : LREAL;
    fSamplingRate      : LREAL;
    nSamplesToFilter   : UDINT;
    nOversamples       : UDINT;
    nChannels          : UDINT;
    pInitialValues     : POINTER TO LREAL;
    nInitialValuesSize : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
fCutoff	LREAL	Grenzfrequenz in Hz (größer 0 und kleiner fSamplingRate/2)
fSamplingRate	LREAL	Abtastrate f_s in Hz (größer Null)
nSamplesToFilter	UDINT	Anzahl der Samples zur Bildung des gleitenden Mittelwerts (oft als Fenstergröße bezeichnet). Bei Null wird die Anzahl automatisch berechnet (optional)
nOversamples	UDINT	Anzahl der Oversamples (größer Null)
nChannels	UDINT	Anzahl der Kanäle (größer Null und kleiner 101)
pInitialValues	Pointer to LREAL	Pointer auf Array mit Initialwerten (optional)
nInitialValuesSize	UDINT	Größe des Array mit Initialwerten in BYTE (optional)

5.2.2 E_FTR_Name

ENUM für Filternamen.

Syntax

Definition:

```
TYPE ST_FTR_Name :
 (
    eButterworth := 1,
    eChebyshev   := 2,
    eBessel      := 3
 ) UDINT
END_TYPE
```

5.2.3 E_FTR_Type

ENUM für Filtertypen.

Syntax

Definition:

```
TYPE ST_FTR_Type :
 (
    eLowPass    := 1,
    eHighPass   := 2,
    eBandPass   := 3,
```

```
    eBandStop    := 4
 ) UDINT
END_TYPE
```

6 Beispiele

6.1 Konfiguration eines Filters mit FB_FTR_IIRSpec

In diesem Beispiel wird exemplarisch gezeigt, wie ein Filter vom Typ „Butterworth“ mit dem Funktionsbaustein FB_FTR_IIRSpec konfiguriert wird.

Download: https://infosys.beckhoff.com/content/1031/TF3680_TC3_Filter/Resources/5906979595.zip (*.tnzip)

Beschreibung:

- Das Beispielprojekt besteht aus einem TwinCAT-SPS-Projekt und einem Measurement-Projekt.
- Im Measurement-Projekt sind zwei Eingangssignale und zwei Ausgangssignale in zwei Achsen konfiguriert.
- Die beiden Eingangssignale werden über einen in der SPS aufgerufenen Funktionsgenerator synthetisch erzeugt. Es handelt sich um harmonische Signale (Sinus) mit einer Frequenz von 250 Hz für den ersten Kanal und mit einer Frequenz von 300 Hz für den zweiten Kanal. Die Signale werden von einem Filter vom Typ „Butterworth“ verarbeitet. Der realisiert Filter hat eine Grenzfrequenz von 250 Hz.
- Das SPS-Programm MAIN wird von einer Task mit 1 ms Zykluszeit aufgerufen.

Implementierung:

- Zunächst wird eine Struktur `stParams` vom Typ `ST_FTR_IIRSpec` deklariert und initialisiert. Die Struktur wird bei Parametrierung des Funktionsbausteins `FB_FTR_IIRSpec` bzw. zur Konfiguration des Filters verwendet.

```
stParams : ST_FTR_IIRSpec := ( … )
```

- Anschließend wird eine Instanz `fbFilter` des Funktionsbausteins `FB_FTR_IIRSpec` erzeugt. Bei der Instanziierung wird die Struktur `stParams` übergeben.

```
fbFilter : FB_FTR_IIRSpec := (stConfig := stParams);
```

- Im Quellcode werden die Methode `Call()` aufgerufen, um den Filter auszuführen, und Varianten zur Fehlerprüfung aufgeführt.

```
IF NOT fbFilter.bError THEN
    fbFilter.Call(ADR(aSignalBuffer), SIZEOF(aSignalBuffer),
                  ADR(aFilteredSignal), SIZEOF(aFilteredSignal));
ELSE
    // if on error state, do something
    nErrorCount := nErrorCount + 1;
    // e.g. check if filter is configured
    // IF fbFilter.bConfigured THEN
    // or use ipResultMessage to check root cause
    // fbFilter.ipResultMessage
END_IF;
```

- Außerdem werden die Methoden `Configure()` und/oder `Reset()` aufgerufen, um während der Laufzeit die Filterkonfiguration zu verändern (`fbFilter.Configure`) oder ein Reset durchzuführen (`fbFilter.Reset`). Der Aufruf der Methoden wird über die Flags `bReconfigure` und `bReset` gesteuert. Diese können Sie im Online Watch manuell auf `TRUE` oder `FALSE` setzen.

Beobachtung:

Die Aufnahme mit dem TwinCAT 3 Scope zeigt, dass die Ausgangssignale durch den Filter in ihrer Amplitude gedämpft und in ihrer Phase verschoben sind. Das Signal im zweiten Kanal (grün) wird stärker gedämpft als das im ersten Kanal (blau, Dämpfung um -3 dB).

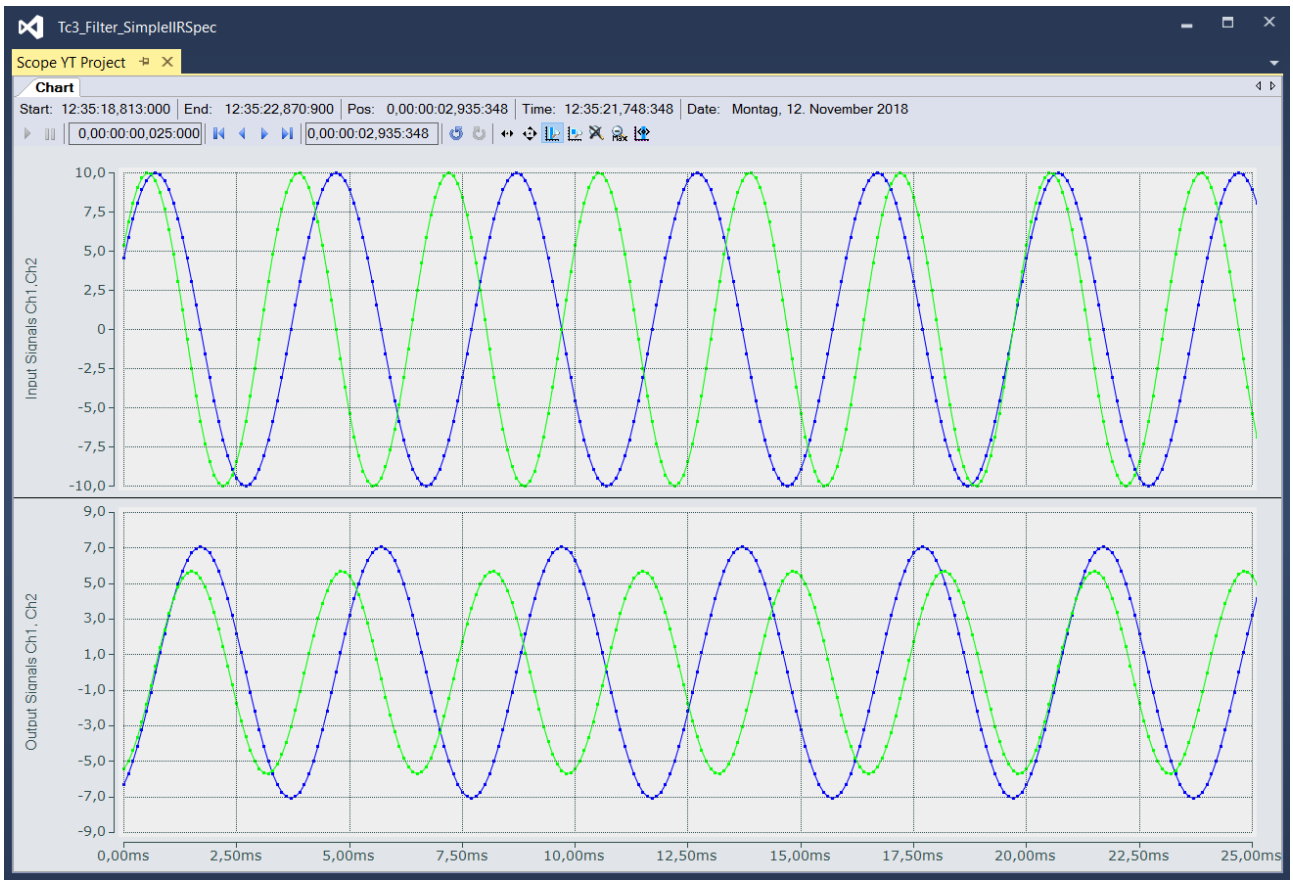


Abb. 10: Signalverläufe der Eingangssignale (oben) und der Ausgangsignale (unten) (blau: Kanal 1, grün: Kanal 2)

Wenn Sie das Flag `bReconfigure` aktivieren, wird die Grenzfrequenz auf 50 Hz verschoben, wodurch die Amplitude beider Ausgangssignale noch kleiner wird.

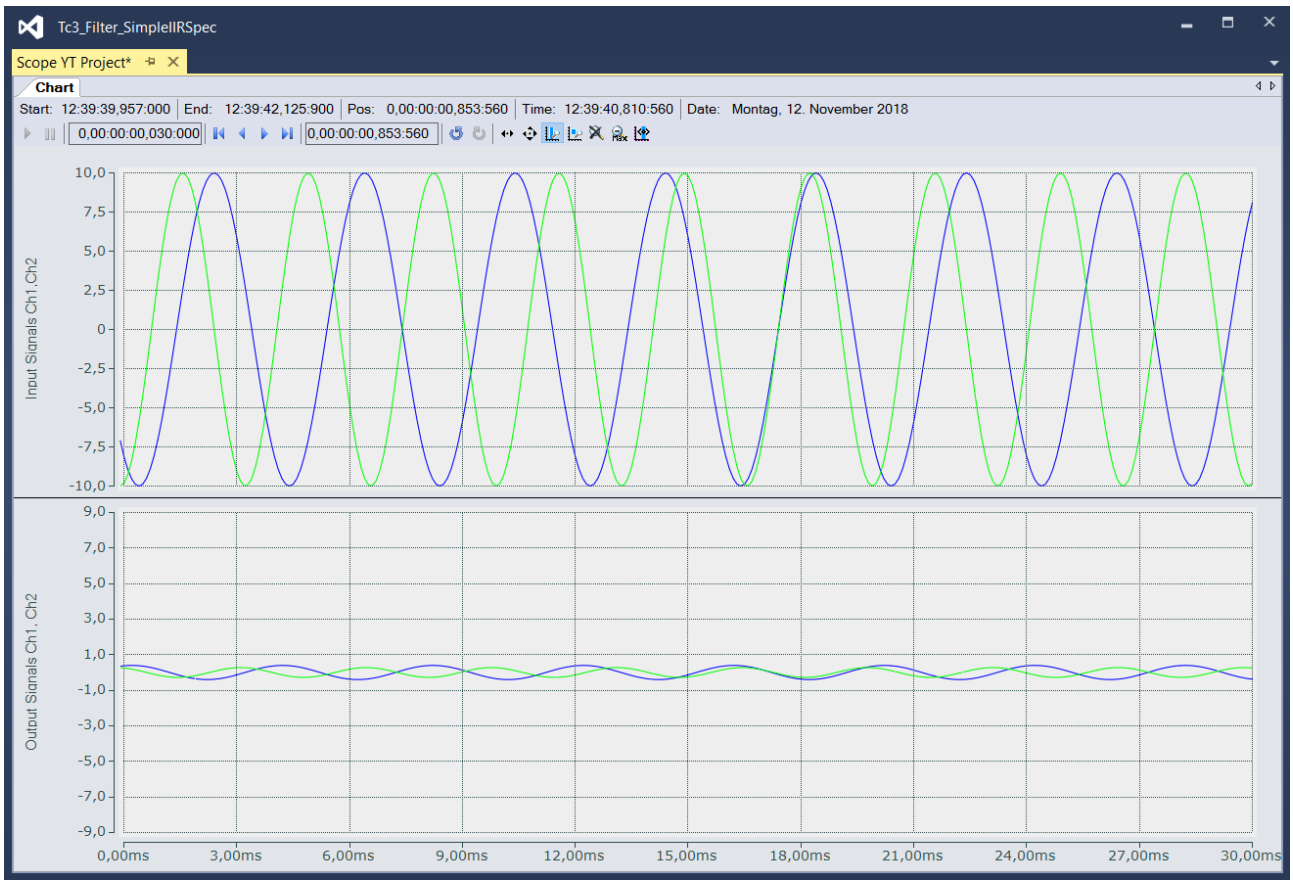


Abb. 11: Signalverlauf der Eingangssignale (oben) und der Ausgangssignale (unten), wenn das Flag bReconfigure aktiviert ist (blau: Kanal 1, grün: Kanal 2)

Wenn Sie das Flag `bReset` aktiveren, wird der interne Zustand des Filters gelöscht, d. h. vergangene Ein- und Ausgangswerte werden nicht mehr berücksichtigt. Der stetige Verlauf des Signals wird unterbrochen (bei ca. 27 ms) und das Filter schwingt sich neu ein.

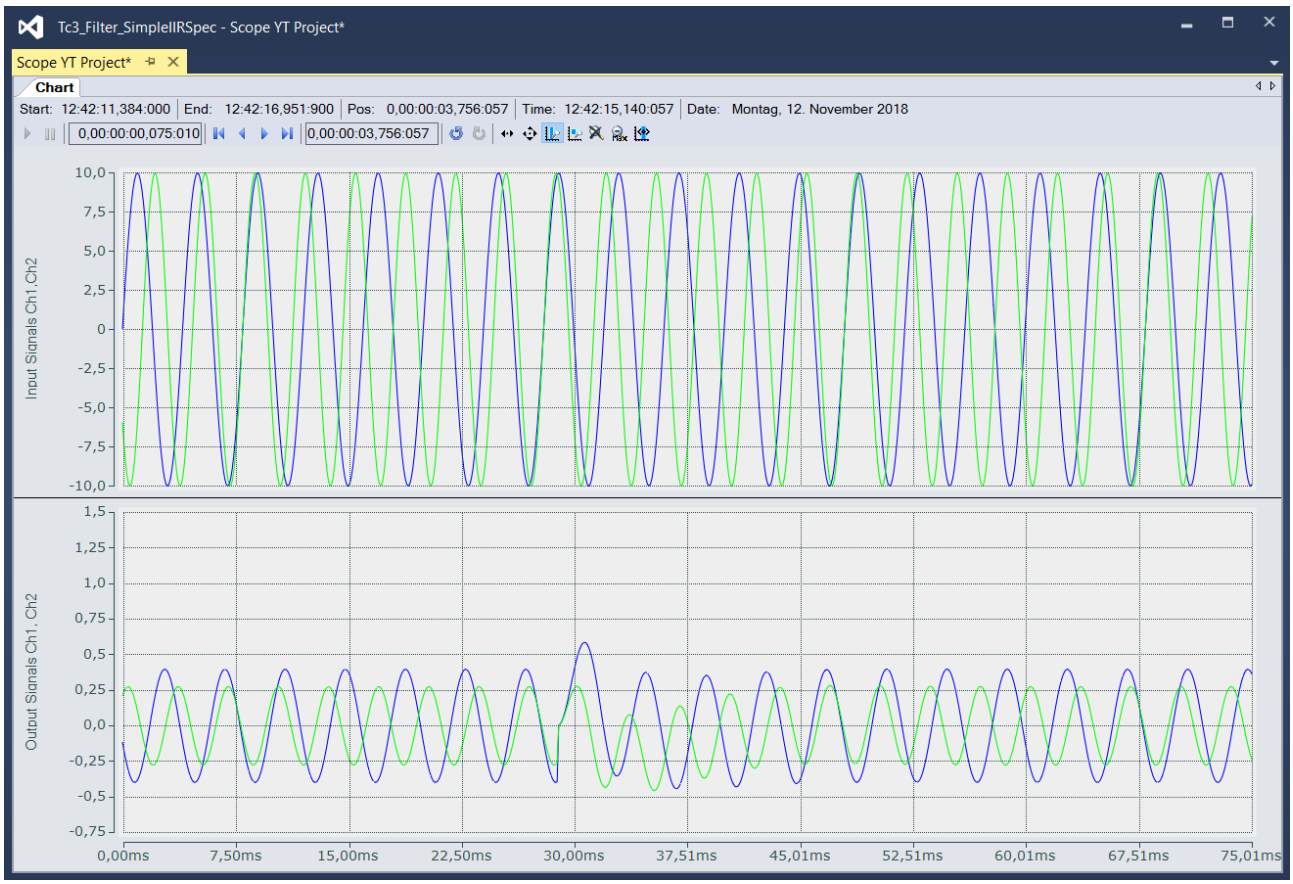


Abb. 12: Signalverläufe der Eingangssignale (oben) und der Ausgangssignale (unten), wenn das Flag bReset aktiviert ist (blau: Kanal 1, grün: Kanal 2)

Siehe auch:

- Filtertypen und Parametrierung [▶ 17]
- FB_FTR_IIRSpec [▶ 36]

6.2 Deklaration und Aufruf von Filtern mit FB_FTR_<type>

In diesem Beispiel wird exemplarisch gezeigt, wie die einzelnen Filterfunktionsbausteine der SPS-Bibliothek Tc3_Filter deklariert und aufgerufen werden.

Download: https://infosys.beckhoff.com/content/1031/TF3680_TC3_Filter/Resources/5906974603.zip (*.tnzip)

Beschreibung:

- Das Beispielprojekt besteht aus einem TwinCAT-SPS-Projekt.
- Im SPS-Programm MAIN wird über einen Funktionsgenerator ein harmonisches Signal erzeugt, das als Eingangssignal für die verschiedenen Filter genutzt wird.
- Jeder Filter wird in einem separaten Funktionsbaustein beschrieben, der im SPS-Programm MAIN instanziiert wird.
- Die SPS-Programm MAIN wird von einer Task mit 1 ms Zykluszeit aufgerufen.

Implementierung:

- In den Funktionsbausteinen werden exemplarisch die Deklarationen und Aufrufe der in der Bibliothek Tc3_Filter zur Verfügung gestellten Filterbausteine dargestellt.
- Über die Variable `nExampleSelector` können Sie den verwendenden Filter während der Laufzeit verändern.

Grundprinzip:

```
FUNCTION_BLOCK FB_PT1
VAR_INPUT
    aBuffer    : ARRAY [1..MAIN.cChannels] OF ARRAY [1..MAIN.cOversamples] OF LREAL;
END_VAR
VAR_OUTPUT
    aOutput    : ARRAY [1..MAIN.cChannels] OF ARRAY [1..MAIN.cOversamples] OF LREAL;
END_VAR
VAR
    stParams: ST_FTR_PT1 := (fSamplingRate := 10000,
    fKp := 1,
    fT1 := 6.366197723675813430755350534900 6E-4,
    nOversamples := MAIN.cOversamples,
    nChannels := MAIN.cChannels);
// fT1 correlates to fc=250 Hz
    fbFilter : FB_FTR_PT1:=(stConfig:=stParams);
END_VAR
```

Siehe auch:

- FB_FTR_IIRCoeff [▶ 30]
- FB_FTR_IIRSpec [▶ 36]
- FB_FTR_MovAvg [▶ 39]
- FB_FTR_PT1 [▶ 41]
- FB_FTR_PT2 [▶ 44]
- FB_FTR_PT3 [▶ 47]
- FB_FTR_PTn [▶ 50]
- FB_FTR_Notch [▶ 53]
- FB_FTR_LeadLag [▶ 56]
- FB_FTR_PT2oscillation [▶ 59]
- FB_FTR_PTt [▶ 62]
- FB_FTR_Median [▶ 64]
- FB_FTR_ActualValue [▶ 67]
- FB_FTR_Gaussian [▶ 72]

6.3 Rekonfiguration mit Initialwerten

In diesem Beispiel wird exemplarisch gezeigt, wie ein Filter während der Laufzeit rekonfiguriert bzw. dynamisch angepasst werden kann. Dabei wird auch beschrieben, wie das Vorwissen über ein betrachtetes Signal genutzt werden kann, um die Einschwingzeit eines Filters nach der Rekonfiguration zu verkürzen.

Download: https://infosys.beckhoff.com/content/1031/TF3680_TC3_Filter/Resources/5906977931.zip (*.tnzip)

Beschreibung:

- Das Beispielprojekt besteht aus einem TwinCAT-SPS-Projekt und einem Measurement-Projekt.
- Im Measurement-Projekt sind zwei Ausgangssignale und eine Counter-Variable konfiguriert.
- Das Eingangssignal wird über einen in der SPS aufgerufenen Funktionsgenerator synthetisch erzeugt. Das Signal wird von zwei unterschiedlich konfigurierten Filtern vom Typ „Butterworth-Tiefpass“ verarbeitet.
- Jeder Filter wird in einem separaten Funktionsbaustein beschrieben, der im SPS-Programm MAIN instanziiert wird.
- Das SPS-Programm MAIN wird von einer Task mit 1 ms Zykluszeit aufgerufen.

Beobachtung:

Die Aufnahme des TwinCAT 3 Scope zeigt einen Signalverläufe, die stark verrauscht sind und zwischen zwei Plateaus springen.

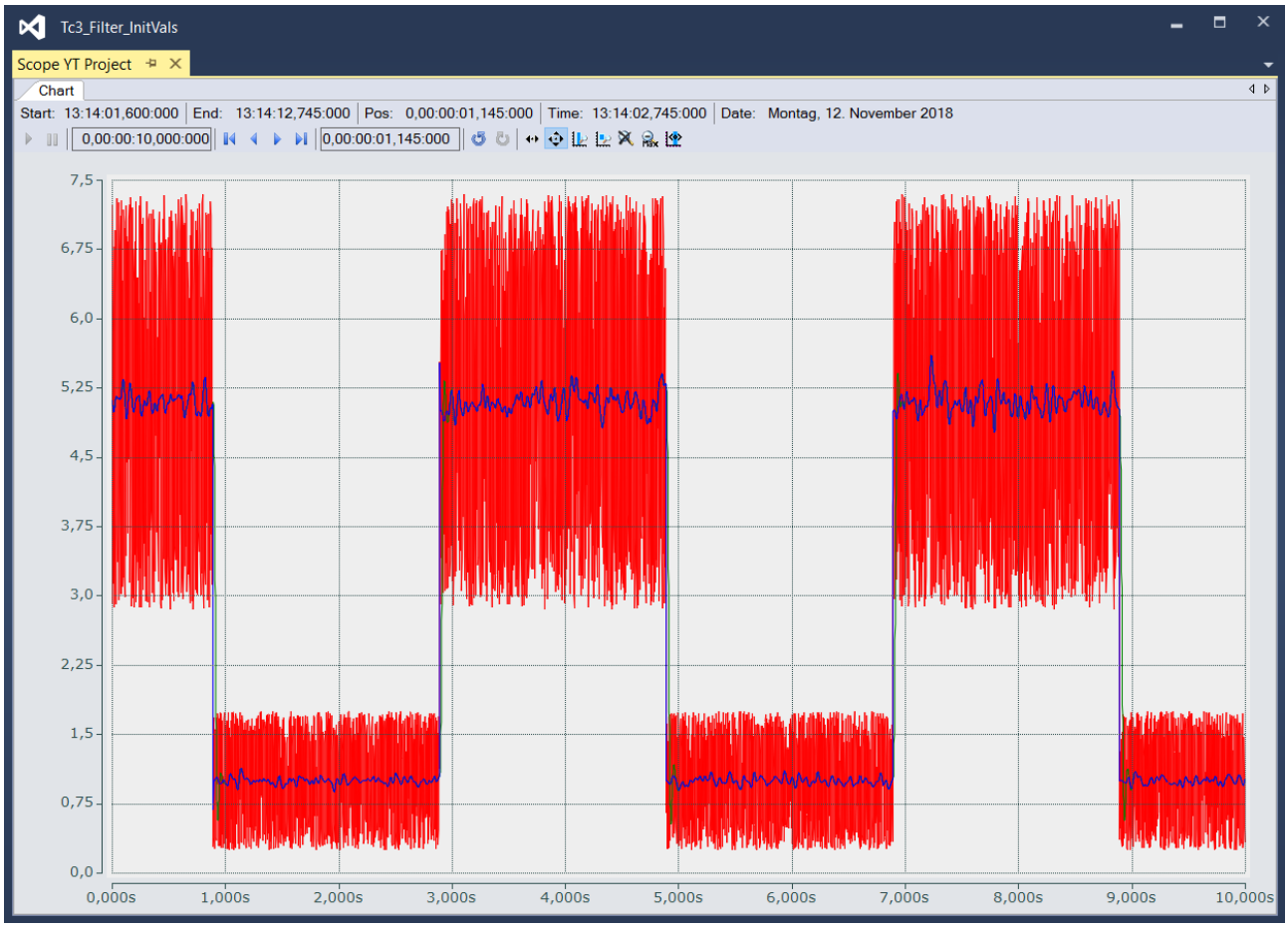


Abb. 13: Signalverlauf des Eingangssignals (rot) und Signalverläufe der Ausgangssignale (grün FB_ContinuousFilter, blau: FB_DynamicFiltering)

Das erste Filter, das im Funktionsbaustein `FB_ContinousFilter` beschrieben ist, bleibt während der Laufzeit unverändert. Das Ausgangsignal dieses Filters ist in seiner Dynamik wegen der Tiefpasswirkung beschränkt.

Das zweite Filter, das im Funktionsbaustein `FB_DynamicFiltering` beschrieben ist, reagiert während der Laufzeit dynamisch auf die Sprungstellen des Eingangssignals.

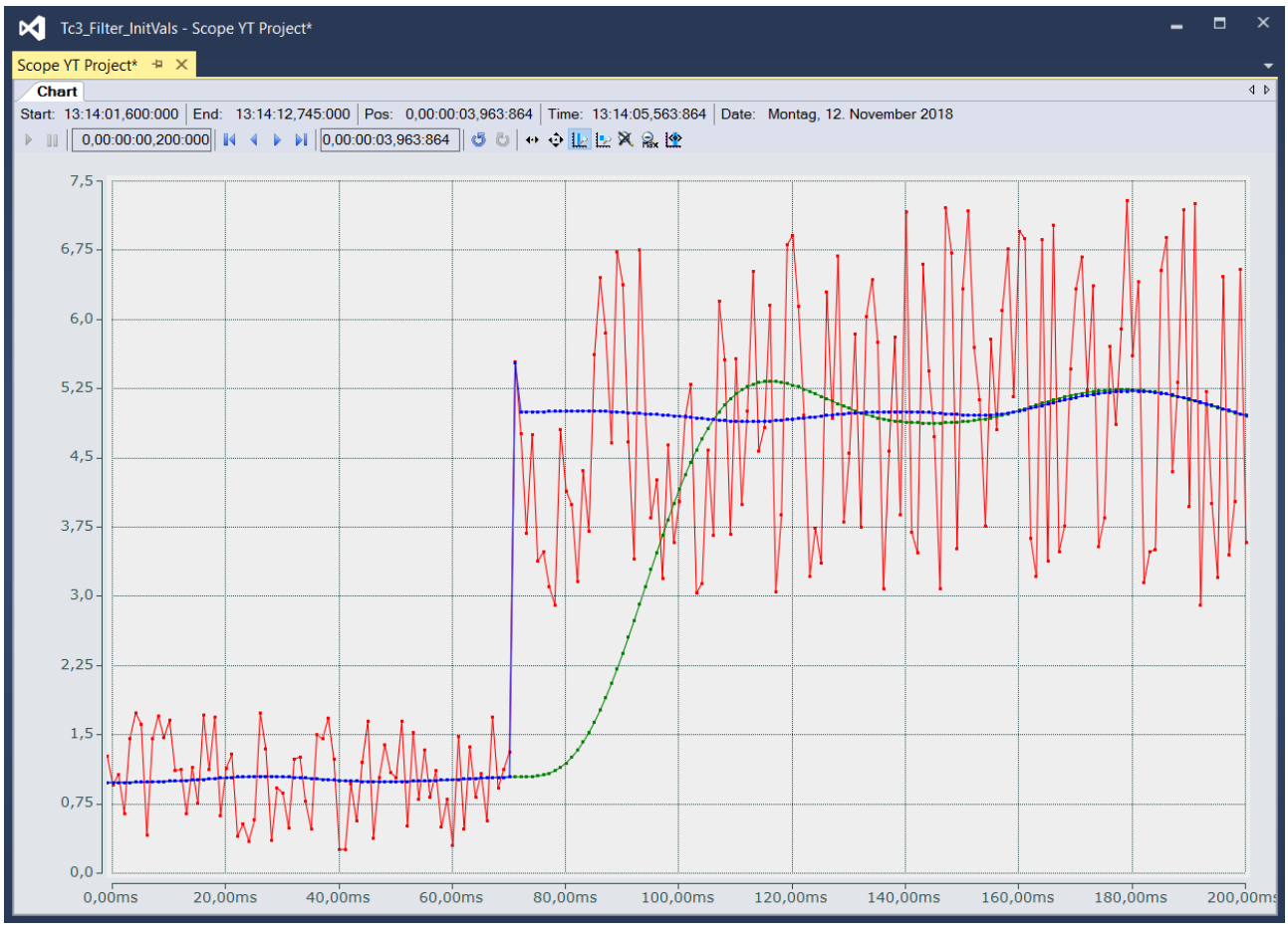


Abb. 14: Signalverlauf des Eingangssignals (rot) und Signalverläufe der Ausgangssignale (grün FB_ContinuousFilter, blau: FB_DynamicFiltering) (Zoom in eine Flanke)

Über den Eingang `bTrigger` wird der Instanz `fbDynamicLowPass` des Funktionsbausteins `FB_DynamicFiltering` mitgeteilt, dass ein Sprung im Signal vorliegt. In der Praxis kann dies anhand des Signals selbst oder auf Basis einer durch die SPS eingeleiteten Aktion angezeigt werden.

Wenn `bTrigger` auf `TRUE` gesetzt wird, wird die Methode `Configure()` aufgerufen, welche die Grenzfrequenz des Tiefpasses auf den maximal zulässigen Wert setzt. Dadurch geht beim nächsten Aufruf der Methode `Call()` die Tiefpasswirkung verloren. Das Ausgangsignal kann dem Sprung im Eingangssignal jedoch schnell folgen.

Schließlich wird die Filterwirkung wieder aktiviert, indem die Methode `Configure()` aufgerufen und die Grenzfrequenz stark verringert wird.

Um außerdem die Einschwingzeit des Filters stark zu reduzieren, wird das Vorwissen über die beiden Plateaus genutzt. Dazu wird dem Parameter `pInitialValues` der Konfigurationsstruktur `stParams` (Typ `ST_FTR_IIRSpec`) der Initialwert 1 bzw. 5 zugewiesen. Die Werte werden im Programm MAIN festgelegt (`fPreKnowledgeHigh`, `fPreKnowledgeLow`) und spiegeln das Wissen des Anwenders wider, dass das zu bewertende Signal ungefähr zwischen 1 und 5 springen wird. Das Filter befindet sich so beim ersten Aufruf der Methode `Call()` bereits im eingeschwungenen Zustand um diesen Wert.

```
stParams.fCutoff := Main.fCutOff; // reduce cutoff freq.

IF bRisingEdge THEN
    // on rising edge apply preknowledge (Signal around 5)
    bRisingEdge := FALSE;
    stParams.pInitialValues := ADR(fInitValHighLevel);
    stParams.nInitialValuesSize := SIZEOF(fInitValHighLevel);
ELSE
    // on falling edge
    stParams.pInitialValues := ADR(fInitValBaseLevel);
    stParams.nInitialValuesSize := SIZEOF(fInitValBaseLevel);
END_IF;

fbFilter.Configure(stConfig := stParams);
```

Durch das event-basierte Umschalten der Filtercharakteristik kann die Einschwingzeit auf das neue Plateau deutlich reduziert werden.

Siehe auch:

- FB_FTR_IIRSpec [▶ 36]

6.4 Rekonfiguration mit und ohne Reset

In diesem Beispiel exemplarisch gezeigt, wie ein Filter währen der Laufzeit rekonfiguriert werden kann und welche Möglichkeiten es gibt, dabei ein Reset zu unterdrücken.

Download: https://infosys.beckhoff.com/content/1031/TF3680_TC3_Filter/Resources/5907283339.zip (*.tnzip)

Beschreibung:

- Das Beispielprojekt besteht aus einem TwinCAT-SPS-Projekt und einem Measurement-Projekt
- Im Measurement-Projekt sind zwei Ausgangssignale und eine Counter-Variable konfiguriert.
- Über einen Signalgenerator wird ein harmonisches Signal mit einer Frequenz von 700 Hz erzeugt. Das Signal wird von zwei Filtern mit identischen Filterkoeffizienten verarbeitet.
- Während der Laufzeit wird zwischen zwei Filterkonfigurationen (Parametrierung der Filterkoeffizienten) hin und her geschaltet. Für die eine Filterinstanz (`fbFilterReset`) wird im Gegensatz zur anderen Filterinstanz (`fbFilterNoReset`) die Methode `Reset()` bei der Rekonfiguration des Filters aufgerufen, um den internen Status des Filters zurückzusetzen.
- Das SPS-Programm MAIN wird von einer Task mit 1 ms Zykluszeit aufgerufen.

Implementierung:

- Über den Parameter `bReset` der Konfigurationsstruktur ST_FTR_IIRCoeff können Sie steuern, ob ein Reset beim Ausführen der Methode `Configure()` ausgeführt wird (Standardwert ist `TRUE`).

```
stParams : ST_FTR_IIRCoeff := ( … )
```

```
stParams.bReset := FALSE;
fbFilterNoReset.Configure(stConfig := stParams);
```

Beobachtung:

Wenn Sie das Projekt aktivieren, können Sie im Measurement-Projekt die Ausgangssignale der beiden Filterinstanzen beobachten. Das rote Signal zeigt das Ausgangssignal der Funktionsbausteininstanz `fbFilterNoReset`. Im zeitlichen Signalverlauf ist kein Sprung durch die Rekonfiguration zu erkennen. Dagegen springt das Ausgangssignal der Funktionsbausteininstanz `fbFilterReset` bei Ausführung der Rekonfiguration, da sich das Filter zum Zeitpunkt der Rekonfiguration ausgehend vom Wert Null wieder einschwingt (blaues Signal).

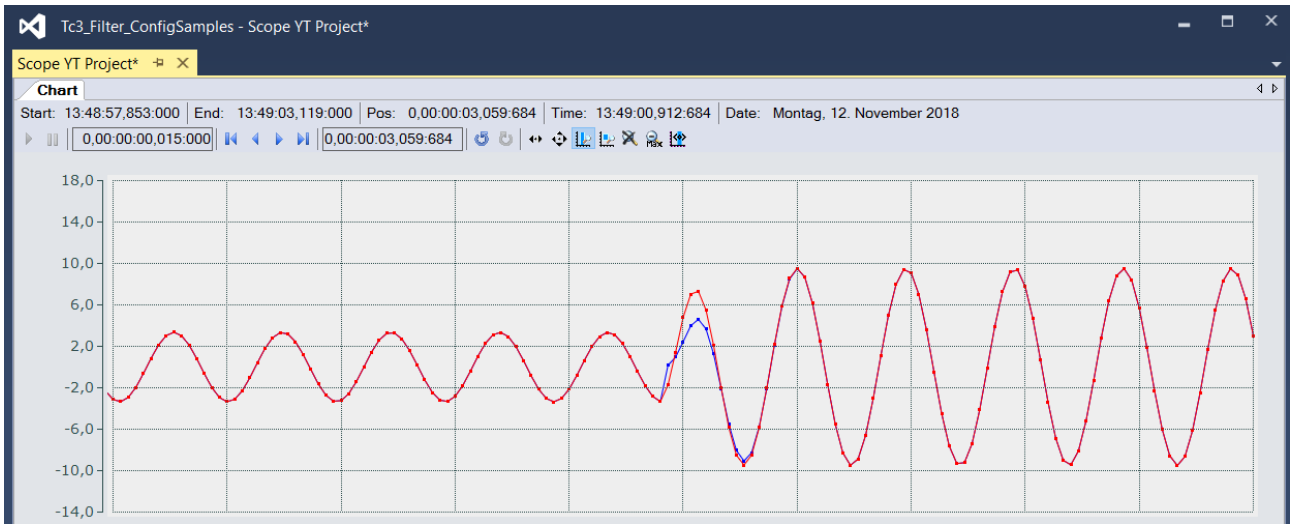


Abb. 15: Signalverläufe der Ausgangssignale bei einer Rekonfiguration ohne Reset (rot) und mit Reset (blau)

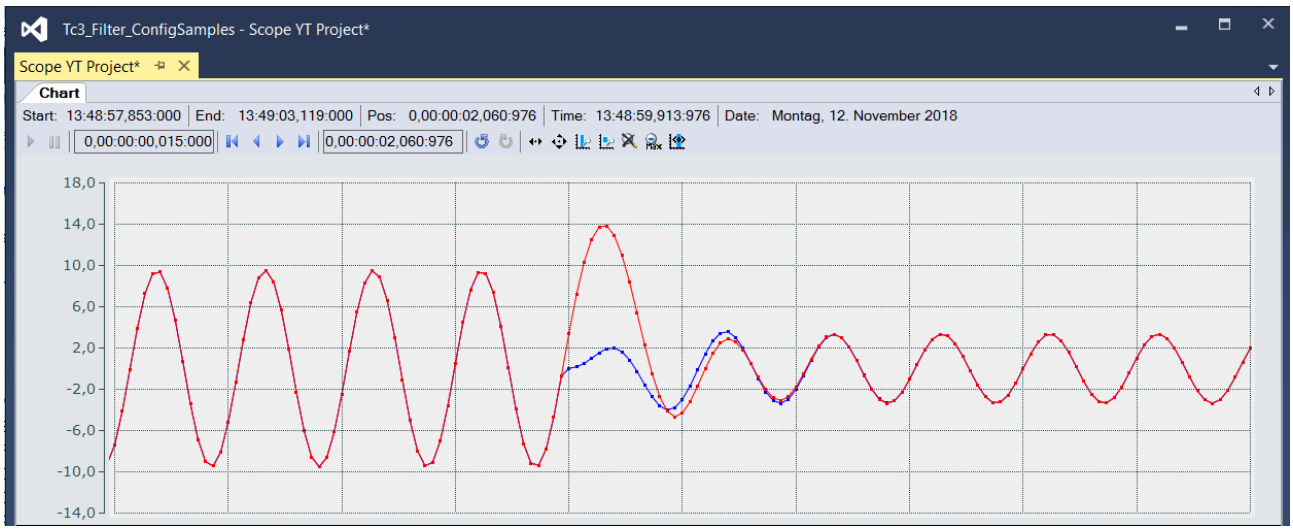


Abb. 16: Signalverläufe der Ausgangssignale bei einer Rekonfiguration ohne Reset (rot) und mit Reset (blau)

Siehe auch:

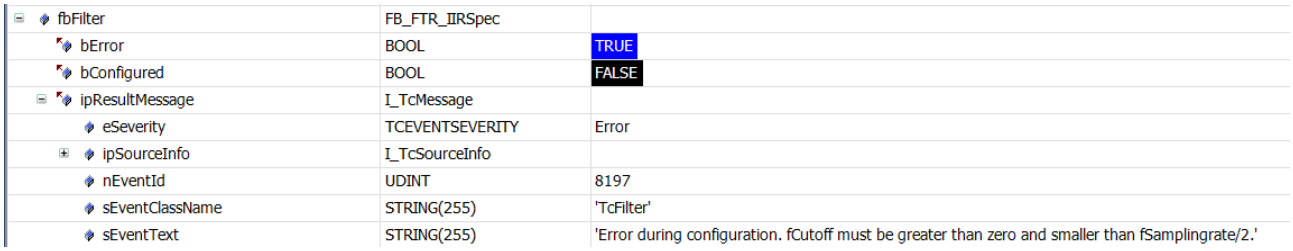
- FB_FTR_IIRCoeff [▶ 30]

7 Anhang

7.1 Rückgabecodes

Rückgabecodes des ipResultMessage.

Online Watch:



fbFilter	FB_FTR_IIRSpec	
bError	BOOL	TRUE
bConfigured	BOOL	FALSE
ipResultMessage	I_TcMessage	
eSeverity	TCEVENTSEVERITY	Error
ipSourceInfo	I_TcSourceInfo	
nEventId	UDINT	8197
sEventClassName	STRING(255)	'TcFilter'
sEventText	STRING(255)	'Error during configuration. fCutoff must be greater than zero and smaller than fSamplingrate/2.'

Definierte Events:

nEventId (hex)	sEventText
16#1001	Error during initialization. No router memory available. Check size of router memory.
16#1002	Error during access to object archive.
16#1004	Error in Transition PREOP->SAFEOP.
16#1005	Error in Transition SAFEOP->OP
16#1006	Error in Transition SAFEOP->OP: No Task assigned. Module will not be executed cyclically.
16#1007	Error in Transition OP->SAFEOP
16#1008	Error in Transition SAFEOP->PREOP
16#2001	Error during configuration. A nullpointer has been allocated.
16#2002	Error during configuration. A nullpointer has been allocated.
16#2003	Error during configuration. fT1 must be greater than zero.
16#2004	Error during configuration. fSamplingrate must be greater than zero.
16#2005	Error during configuration. fCutoff must be greater than zero and smaller than fSamplingrate/2.
16#2006	Error during configuration. fBandwidth must be greater than zero and smaller than fSamplingrate/2 - fCutoff.
16#2007	Error during configuration. fPassBandRipple must be greater than zero.
16#2008	Error during configuration. fStopBandRipple must be greater than zero.
16#2009	Error during configuration. nChannels must be greater than zero and smaller than 101.
16#200A	Error during configuration. nOversamples must be greater than zero.
16#200B	Error during configuration. nFilterOrder has to be between 1 and 10/20 (band pass and band stop/ low pass and high pass)
16#200C	Error during configuration. nCoefficientArraySize_A and nCoefficientArraySize_B must be equal and greater than seven.
16#200D	Error during configuration. pCoefficientArrayAdr_A is an invalid pointer.
16#200E	Error during configuration. A0 coefficient cant be zero due zero devision.
16#200F	Error during configuration. pCoefficientArrayAdr_B is an invalid pointer.
16#2010	Error during configuration. fKp must be greater than zero.
16#2011	Error during configuration. nSamplesToFilter must be greater than zero.
16#2012	Error during configuration. fT2 must be greater than zero.
16#2013	Error during configuration. fT3 must be greater than zero.
16#2014	Error during configuration.nOrder must be greater than zero and smaller than eleven.

nEventId (hex)	sEventText
16#2015	Error during configuration. nInitialValuesSize must be 0, 8, (OrderB+OrderA)*8 or (OrderB+OrderA)*nChannels*8.
16#2016	Error during configuration. Invalid FilterName.
16#2017	Error during configuration. Invalid FilterType.
16#2018	Error during configuration. bReset = false is only allowed if the following filter structure member don't change: nCoefficientArraySize_A, nCoefficientArraySize_B, nChannels and nOversamples.
16#2019	Error during configuration. nCoefficientArraySize_Sos must be a multiple of 48.
16#201A	Error during configuration. pCoefficientArrayAdr_Sos is an invalid pointer.
16#201B	Error during configuration. Filter parameter lead to an unstable filter, please choose other filter parameter.
16#201C	Error during configuration. bReset = false is only allowed if the following filter structure member don't change: nCoefficientArraySize_Sos, nChannels and nOversamples.
16#201D	Error during configuration. nInitialValuesSize must be 0, 8, (number of biquads)*4*8 or (number of biquads)*4*nChannels*8.
16#201E	Error during configuration. fNotchfrequency must be greater than zero and smaller than fSamplingrate/2.
16#201F	Error during configuration. fQ must be greater than zero.
16#2020	Error during configuration. fTheta must be greater than zero.
16#2021	Error during configuration. fTt must be greater than zero and a multiple of 1/ fSamplingRate.
16#2022	Error during configuration. fDeltaMax must be equal or greater than zero.
16#3001	Error during runtime. No router memory available. Check size of router memory.
16#3002	Error during runtime. Missing configuration.
16#3003	Error during runtime. Cyclic caller is assigned. Methods can not be called.
16#3004	Error during runtime. Size of fIn Array doesnt match with nOversamples*nChannels.
16#3005	Error during runtime. Size of fOut Array cant be smaller than fIn Array Size.
16#3006	Error during runtime. A nullpointer has been allocated @ pIn.
16#3007	Error during runtime. A nullpointer has been allocated @ pOut.

7.2 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Mehr Informationen:
www.beckhoff.de/TF3680

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