



Handbuch

TwinCAT PLC Hydraulics

TwinCAT

Version: 1.2
Datum: 04.10.2016
Bestell-Nr.: TS5810

BECKHOFF

Inhaltsverzeichnis

1	Vorwort	5
1.1	Hinweise zur Dokumentation	5
1.2	Sicherheitshinweise	6
2	Genereller Aufbau	7
2.1	Die Hydraulikbibliothek	7
2.2	Gliederung der Dokumentation	9
2.3	Funktionen, Bausteine und Typen (ab V3.0)	10
3	PLCopen Motion Control	19
3.1	Administrative	19
3.1.1	MC_Power_BkPlcMc (ab V3.0)	19
3.1.2	MC_ReadActualPosition_BkPlcMc (ab V3.0)	21
3.1.3	MC_ReadActualTorque_BkPlcMc (ab V3.0)	22
3.1.4	MC_ReadActualVelocity_BkPlcMc (ab V3.0)	23
3.1.5	MC_ReadAxisError_BkPlcMc (ab V3.0)	24
3.1.6	MC_ReadBoolParameter_BkPlcMc (ab V3.0)	24
3.1.7	MC_ReadDigitalOutput_BkPlcMc (ab V3.0)	25
3.1.8	MC_ReadParameter_BkPlcMc (ab V3.0)	26
3.1.9	MC_ReadStatus_BkPlcMc (ab V3.0)	27
3.1.10	MC_Reset_BkPlcMc (ab V3.0)	29
3.1.11	MC_ResetAndStop_BkPlcMc (ab V3.0)	30
3.1.12	MC_SetOverride_BkPlcMc (ab V3.0)	31
3.1.13	MC_SetPosition_BkPlcMc (ab V3.0)	32
3.1.14	MC_SetReferenceFlag_BkPlcMc (ab V3.0)	33
3.1.15	MC_WriteBoolParameter_BkPlcMc (ab V3.0)	34
3.1.16	MC_WriteDigitalOutput_BkPlcMc (ab V3.0)	35
3.1.17	MC_WriteParameter_BkPlcMc (ab V3.0)	36
3.2	Motion	37
3.2.1	MC_CamIn_BkPlcMc (ab V3.0)	37
3.2.2	MC_CamOut_BkPlcMc (ab V3.0)	39
3.2.3	MC_CamTableSelect_BkPlcMc (ab V3.0)	40
3.2.4	MC_DigitalCamSwitch_BkPlcMc (ab V3.0)	41
3.2.5	MC_EmergencyStop_BkPlcMc (ab V3.0.5)	44
3.2.6	MC_GearIn_BkPlcMc (ab V3.0)	45
3.2.7	MC_GearInPos_BkPlcMc (ab V3.0.33)	47
3.2.8	MC_GearOut_BkPlcMc (ab V3.0)	49
3.2.9	MC_Halt_BkPlcMc (ab V3.0)	50
3.2.10	MC_Home_BkPlcMc (ab V3.0)	51
3.2.11	MC_ImmediateStop_BkPlcMc (ab V3.0.5)	53
3.2.12	MC_MoveAbsolute_BkPlcMc (ab V3.0)	54
3.2.13	MC_MoveJoySticked_BkPlcMc (ab V3.0)	56
3.2.14	MC_MoveRelative_BkPlcMc (ab V3.0)	57
3.2.15	MC_MoveVelocity_BkPlcMc (ab V3.0)	59
3.2.16	MC_Stop_BkPlcMc (ab V3.0)	60
3.3	Datentypen	62
3.3.1	Axis_Ref_BkPlcMc (ab V3.0)	62
3.3.2	CAMSWITCH_REF_BkPlcMc (ab V3.0)	63
3.3.3	E_TcPlcBufferedCmdType_BkPlcMc	64
3.3.4	E_TcMcCurrentStep (ab V3.0)	65
3.3.5	E_TcMcDriveType (ab V3.0)	67
3.3.6	E_TcMcEncoderType (ab V3.0)	70
3.3.7	E_TcMCFbState (ab V3.0)	73
3.3.8	E_TcMcHomingType (ab V3.0)	73
3.3.9	E_TcMCPParameter (ab V3.0)	74
3.3.10	E_TcMcProfileType (ab V3.0)	83
3.3.11	E_TcMcPressureReadingMode (ab V3.0)	84
3.3.12	MC_BufferMode_BkPlcMc (ab V3.0)	84

3.3.13	MC_CAM_ID_BkPlcMc (ab V3.0)	85
3.3.14	MC_CAM_REF_BkPlcMc (ab V3.0)	85
3.3.15	MC_Direction_BkPlcMc (ab V3.0)	85
3.3.16	MC_HomingMode_BkPlcMc (ab V3.0)	86
3.3.17	MC_StartMode_BkPlcMc (ab V3.0)	86
3.3.18	OUTPUT_REF_BkPlcMc (ab V3.0)	87
3.3.19	ST_FunctionGeneratorFD_BkPlcMc (ab V3.0.31)	87
3.3.20	ST_FunctionGeneratorTB_BkPlcMc (ab V3.0.31)	87
3.3.21	ST_TcMcAutoIdent (ab V3.0.4)	88
3.3.22	ST_TcHydAxParam (ab V3.0)	88
3.3.23	ST_TcHydAxRtData (ab V3.0)	94
3.3.24	ST_TcMcAuxDataLabels (ab V3.0)	98
3.3.25	ST_TcPlcDeviceInput (ab V3.0)	98
3.3.26	ST_TcPlcDeviceOutput (ab V3.0)	100
3.3.27	ST_TcPlcMcLogBuffer (ab V3.0)	102
3.3.28	ST_TcPlcMcLogEntry (ab V3.0)	102
3.3.29	ST_TcPlcRegDataItem (ab V3.0.7)	103
3.3.30	ST_TcPlcRegDataTable (ab V3.0.7)	103
3.3.31	TRACK_REF_BkPlcMc (ab V3.0)	103
3.4	System	104
3.4.1	Controller	104
3.4.2	Drive	114
3.4.3	Encoder	123
3.4.4	FunctionGenerator	147
3.4.5	TableFunctions	149
3.4.6	Generatoren	155
3.4.7	Runtime	163
3.4.8	Message Logging	173
3.4.9	Utilitys	177
3.5	Parameter	187
3.5.1	MC_AxAdsCommServer_BkPlcMc (ab V3.0)	187
3.5.2	MC_AxAdsPtrArrCommServer_BkPlcMc	188
3.5.3	MC_AxAdsReadDecoder_BkPlcMc (ab V3.0)	189
3.5.4	MC_AxAdsWriteDecoder_BkPlcMc (ab V3.0)	190
3.5.5	MC_AiParamAuxLabelsLoad_BkPlcMc (ab V3.0)	191
3.5.6	MC_AiParamLoad_BkPlcMc (ab V3.0)	192
3.5.7	MC_AiParamSave_BkPlcMc (ab V3.0)	193
3.5.8	MC_AxUtiReadCoeDriveTerm_BkPlcMc (ab V3.0)	193
3.5.9	MC_AxUtiReadCoeEncTerm_BkPlcMc (ab V3.0)	195
3.5.10	MC_AxUtiReadRegDriveTerm_BkPlcMc (ab V3.0)	196
3.5.11	MC_AxUtiReadRegEncTerm_BkPlcMc (ab V3.0)	197
3.5.12	MC_AxUtiUpdateRegDriveTerm_BkPlcMc (ab V3.0.7)	198
3.5.13	MC_AxUtiUpdateRegEncTerm_BkPlcMc (ab V3.0.7)	200
3.5.14	MC_AxUtiWriteCoeDriveTerm_BkPlcMc (ab V3.0)	201
3.5.15	MC_AxUtiWriteCoeEncTerm_BkPlcMc (ab V3.0)	203
3.5.16	MC_AxUtiWriteRegDriveTerm_BkPlcMc (ab V3.0)	204
3.5.17	MC_AxUtiWriteRegEncTerm_BkPlcMc (ab V3.0)	205
4	Knowledge Base	207
4.1	FAQs (ab V3.0)	208
4.2	Globale Konstanten (ab V3.0)	222
4.3	Ventil	233
4.4	Programm-Beispiele (ab V3.0)	234
4.5	Der PlcMcManager	238
4.6	Setup	241
4.6.1	Achs-Inbetriebnahme (ab V3.0)	241
4.6.2	Achs-Inbetriebnahme: Allgemeines	242
4.6.3	Voreinstellungen	242
4.6.4	Inbetriebnahme	258

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 nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

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.

Deshalb ist die Dokumentation nicht in jedem Fall vollständig auf die Übereinstimmung mit den beschriebenen Leistungsdaten, Normen oder sonstigen Merkmalen geprüft.

Falls sie technische oder redaktionelle Fehler enthält, behalten wir uns das Recht vor, Änderungen jederzeit und ohne Ankündigung vorzunehmen.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® und XTS® 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, DE102004044764, DE102007017835

mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

Die TwinCAT Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:

EP0851348, US6167425 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

© Beckhoff Automation GmbH & Co. KG, Deutschland.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.

Zu widerhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

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!
 Achtung	Schädigung von Umwelt oder Geräten Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.
 Hinweis	Tipp oder Fingerzeig Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

2 Genereller Aufbau

2.1 Die Hydraulikbibliothek

Um den Anforderungen der Hydraulik gerecht zu werden, sind für die Regelung spezielle Algorithmen notwendig. Die SPS Bibliotheken TcPlcHydraulics_30 (für TC2) und TC2_Hydraulics (für TC3) fassen eine Reihe von Bausteinen und Funktionen zum Thema hydraulische Achsen sowie die darin verwendeten Datentypen zusammen. Sie erweitern die Unterstützung dieser Antriebstechnologie indem sie den Betrieb von Achsen ermöglichen, die durch ihre Eigenschaften (Grenzfrequenz, stark streuendes Verhalten) nicht zur Lageregelung geeignet sind oder deren Aufgaben sich von dem bei elektrischen Servo-Achsen stark unterscheiden.

Das hier vorgestellte Produkt umfasst:

- die Softwarebibliothek "TcPlcHydraulics.lib" bzw. "Tc2_Hydraulics.compiled-library"
- das Inbetriebnahme Tool "PlcMcManager.exe"

Um die Nutzung der Bibliothek zu vereinfachen sind die Bausteine entsprechend den Vorgaben der Nutzerorganisation der IEC61131 Anwender (PLCopen) ausgeführt und entsprechend zertifiziert.



❏ HINWEIS! Die Dokumentation der Version V2.1 wird auch weiterhin zur Verfügung stehen.

Themen der Bibliothek:

- Auswertung von [Encodern \[► 123\]](#)
- Auswertung von Druckmessdosen
- diverse Filterfunktionen
 - Pt1-Filter
 - [Gleitender-Mittelwert \[► 180\]](#)
 - [Anstiegsbegrenzung \[► 179\]](#)
- voller Zugriff auf interne Größen
- Bewegungssteuerung
- Regler für
 - Druck/Kraft
 - Position
 - Geschwindigkeit
 - Möglichkeit der eigenen Reglerentwicklung
- Synchronisierung von hydraulischen sowie elektrischen Achsen
- Anpassung der Stellwerte an Ausgabegeräte

- Vollständige Handhabung komplexer Geräte
- Message Logging
- Parameterhandhabung
 - Speicher- und Laderoutinen
 - Autosave
- Kennlinien-Linearisierung
 - abschnittsweise
 - Kompensations-Kennlinie [► 181]

Folgende Bewegungssteuerungen werden unterstützt:

1. Zeitgeführte Bewegungssteuerung:
 - Die steuernde Größe der Profilerzeugung ist die Zeit.
 - Der Generator „kennt“ die Achse nicht.
 - Nur der vorgesteuerte Lageregler stellt die Verbindung her.
2. Weggeführte Bewegungssteuerung:
 - Die steuernde Größe der Profilerzeugung ist der Restweg.
 - Der Generator „kennt“ die Achse.
 - Während der Bewegung ist keine Lageregelung möglich/nötig.
3. Abhängige Bewegungssteuerung:
 - Die Sollwerte werden über eine Abbildungsvorschrift (Getriebeformel, Kurventabelle) aus den Werten einer anderen Achse berechnet.
 - Der Generator „kennt“ die Achse nicht.
 - Nur der vorgesteuerte Lageregler stellt die Verbindung her.

Weggeführte und zeitgeführte Bewegungssteuerung:

Die zeitgeführte Bewegungssteuerung benutzt die Zeit als Bezugsvariable. Die Grundgleichungen sind

$$v=a*t \text{ und}$$

$$s=0.5*a*t*t.$$

Der Sollwertgenerator gibt eine Geschwindigkeit und eine Position vor, diese wird vom Lage- und Geschwindigkeitsregler ausgewertet und mit der aktuellen Position verrechnet.

Bei der weggesteuerten Positionierung wird im Gegensatz zur zeitgesteuerten der Stellwert der Achse als Funktion des Restwegs berechnet. Durch Umstellen der obigen Gleichungen erhält man

$$v=\sqrt{2*a*s}.$$

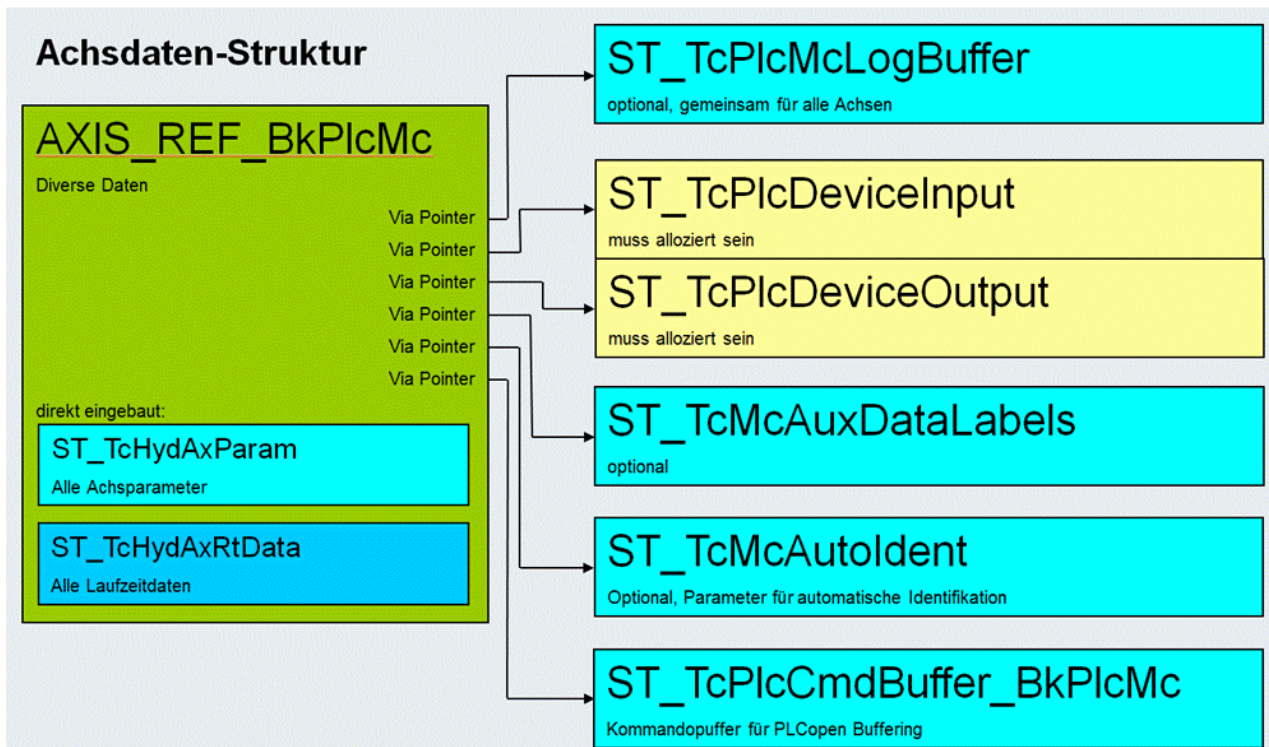
Beide Verfahren haben Vor- und Nachteile.

- Zeitgesteuerte Achsen benötigen gerade für Beschleunigungs- und Verzögerungsvorgänge einen geschlossenen Regelkreis. Nur über die Rückkopplung ist es dem Geschwindigkeitsregler möglich die richtige Ausgabegröße zu generieren. Ein solcher Regelkreis reagiert aber stark auf Stick/Slip-Effekte oder Versorgungsdruckschwankungen, wodurch das System leicht zum Schwingen angeregt wird.
- Weggesteuerte Achsen müssen nicht in einem geschlossenen Regelkreis betrieben werden. Damit ist dieses Verfahren wesentlich robuster gegenüber äußeren Störeinflüssen.
- Da die weggesteuerten Achsen nicht die Zeit sondern den Weg als Basis haben, wird eine Geschwindigkeit gestellt, aber nicht nachgeregelt. Aus diesem Grund ist die Positionierung von hydraulischen Achsen sehr robust.

Beide Verfahren werden von der Hydraulik-Bibliothek unterstützt und können auch kombiniert verwendet werden.

2.2 Gliederung der Dokumentation

Jede Achse besteht aus einer Achsstruktur namens "Axis_ref_BkPlcMc", die aus verschiedenen ausgelagerten Strukturen zusammengesetzt ist. In dieser Achsstruktur befinden sich alle Daten (Laufzeitdaten sowie Parameterdaten) zu dieser Achse.



Bestimmte Bausteine müssen in jeder Applikation vorhanden sein um eine Achse überhaupt verfahren zu können. Zu diesen Bausteinen zählen:

- [MC_AxUtiStandardInit_BkPlcMc \[▶ 171\]](#) : Initialisierung und Überwachung verschiedener Bestandteile der Achse. Ein solcher FB sollte zyklisch aufgerufen werden. Erst nach einer erfolgreichen Initialisierung dürfen Bausteine wie **MC_Power_BkPlcMc** usw. aufgerufen werden.
- [MC_Power_BkPlcMc \[▶ 19\]](#) : Der Funktionsbaustein dient zum Ansteuern eines externen Stellgeräts. Der Baustein gibt Freigaben an z. B. Ventilendstufen oder Frequenzumrichter heraus.
- [MC_AxStandardBody_BkPlcMc \[▶ 170\]](#) : Der Funktionsbaustein ruft jeweils einen Baustein vom Typ [MC_AxRtEncoder_BkPlcMc \[▶ 123\]](#): Ermittlung der Istposition der Achse aus den Eingangsinformationen einer Hardware-Baugruppe.
[MC_AxRuntime_BkPlcMc \[▶ 155\]](#): Übernimmt die Profilerzeugung.
[MC_AxRtFinish_BkPlcMc \[▶ 164\]](#): Anpassung des Stellwertes an die Besonderheiten der Achse (Kennlinienlinearisierung)
[MC_AxRtDrive_BkPlcMc \[▶ 114\]](#): Der Funktionsbaustein übernimmt die Aufbereitung des Stellwertes der Achse für die Ausgabe auf einer Hardware-Baugruppe.
- [MC_AxAdsCommServer_BkPlcMc \[▶ 187\]](#) : Stellt die Verbindung zu PlcMcManager her und überwacht diese. Dieser Baustein muss unabhängig von der Initialisierung aufgerufen werden. Nur so ist eine Inbetriebnahme ohne bereits vorhandene Parameter möglich.

Optional sinnvolle Bausteine sind:

- [MC_AxRtLoggerSpool_BkPlcMc \[▶ 176\]](#) : Der Funktionsbaustein stellt sicher, dass der LogBuffer der Library nicht überläuft.
- [MC_AxParamDelayedSave_BkPlcMc](#): Führt ein automatisches Speichern der Achsparameter durch.

Für die Inbetriebnahme wird der so genannte "PlcMcManager" bereitgestellt. Dieser soll die Inbetriebnahme der Anlage erleichtern, da wesentliche Einstellparameter in diesem Tool zusammengefasst dargestellt werden.

Für die ersten "Gehversuche" eignet sich das erste Beispiel hervorragend.

Funktionsgruppen	Beschreibung
Verwaltungsfunktionen [▶ 12]	Funktionen für die Verwaltung und Kontrolle von Achsen, Zugriff auf Parameter und Zustände.
Einzelachsbewegungsfunktionen [▶ 13]	Auslösen und Überwachen von aktiven Bewegungen für Einzelachsen.
Achsgruppenbewegungsfunktionen [▶ 13]	Auslösen und Überwachen von aktiven Bewegungen für Achsgruppen.
Antriebsanpassungen [▶ 13]	Bausteine zur Vorbereitung von Achsstellwerten für die Ausgabe auf Ausgabegeräte (Klemmen, Steller usw.) in der Peripherie.
Encoderanpassungen [▶ 13]	Bausteine zur Auswertung von Istpositionsdaten, die von Eingabegeräte (Klemmen, Encoder usw.) in der Peripherie eingelesen wurden.
Parameterhandhabung [▶ 14]	Bausteine für das Speichern, Lesen und Kommunizieren von Parametern.
Bewegungsgeneratoren [▶ 13]	Stellwertgeneratoren für aktive Achsbewegungen
Regler [▶ 15]	Regler für verschieden Zustandsgrößen: Position, Geschwindigkeit, Druck.
Tabellenfunktionen [▶ 15]	Tabellenfunktionen für nichtlineare Abbildungen und Kurvenscheiben
Message Logging [▶ 15]	Meldungsaufzeichnung.
Laufzeitfunktionen [▶ 16]	Diverse Laufzeitfunktionen.
Datentypen	In der Bibliothek verwendete Aufzählungen [▶ 17] und Strukturen [▶ 18]

2.3 Funktionen, Bausteine und Typen (ab V3.0)

☐ HINWEIS! Hier werden alle vorhandenen Funktionen, Bausteine und Datentypen dieser Library aufgelistet.

Antworten zu häufig gestellten Fragen sowie Hinweise zum Einsatz der Library, zu Inbetriebnahme und Problemanalyse sowie Beispielprojekte finden Sie in der [Knowledge Base \[▶ 207\]](#).

Einige der hier aufgeführten Komponenten sind nicht für die Benutzung durch eine Applikation vorgesehen. Ihr Vorhandensein, Interface und Verhalten wird dementsprechend nicht garantiert. Da eine TwinCAT PLC Library jedoch strikt offen ist besteht keine Möglichkeit, diese internen Komponenten zu verbergen. Es sollte jedoch unbedingt darauf verzichtet werden, diese entsprechend mit (internal use only) oder (not recommended) gekennzeichneten Komponenten aus einer Applikation heraus direkt zu benutzen. Sollten eine dieser Komponenten für Sie von praktischem Nutzen sein nehmen Sie bitte Kontakt mit unserem Support auf. Wir werden dann die Möglichkeit prüfen, ob Ihnen ein Baustein unabhängig von der Library zur eigenverantwortlichen Verfügung gestellt werden kann.

Sollte die Library Bausteine, Typen oder Konstanten enthalten, die in der Dokumentation nicht aufgeführt werden handelt es sich um nicht freigegebene Elemente, die Gegenstand der aktuellen Softwarepflege und -entwicklung sind. Diese Elemente dürfen auf keinen Fall in einer Applikation direkt verwendet werden, da sie in der Regel noch nicht getestet sind.

☐ HINWEIS! Die Hydraulik Library bietet im Zusammenhang mit elektrischen Antrieben nur einen eingeschränkten Funktionsumfang. Ein erheblich weiteres Spektrum und eine umfassende Unterstützung bei Inbetriebnahme und Diagnose bieten TwinCAT NC PTP, NC I und CNC.

☐ HINWEIS! Es steht eine Reihe von Bibliotheken zur Verfügung, die auf eine typische Anordnung von Achsen oder spezielle Funktionalitäten ausgerichtet sind. Diese Bibliotheken setzen die TcPlcHydraulics Library voraus und sind getrennt zu bestellen.

Name	Beschreibung
TcPlcLibHydraulics_30_2R2Vgantry.LIB	in Vorbereitung
TcPlcLibHydraulics_30_4R3Vgantry.LIB	in Vorbereitung

PLC open Motion Control

Die hier aufgeführten Bausteine orientieren sich an:

Technical Specification

PLCopen - Technical Committee 2 - Task Force

Function blocks for motion control

Part 1 Version 1.1 und Part 2 Version 0.99F (Definition noch nicht endgültig)

Die Namen dieser Bausteine beginnen mit MC_ und enden mit _BkPlcMc.



HINWEIS! Teile der PLCopen Definitionen sind noch nicht endgültig. Es kann hier zu Änderungen in zukünftigen Versionen der Library geben.

Diese Änderungen können betreffen

- Namen, Verhalten oder sogar Existenz von Funktionen, Funktionsblöcken oder abgeleiteten Datentypen
- Namen, Verhalten, Typen oder Vorhandensein von Eingangs- oder Ausgangssignalen

Administrative Function Blocks

Name	Beschreibung
MC_CamTableSelect_BkPlcMc [► 40]	Der Funktionsbaustein initialisiert eine Variable vom Typ ST_TcPlcMcCamId und bereitet dadurch eine Kurvenscheibe für die Kopplung von zwei Achsen vor.
MC_Power_BkPlcMc [► 19]	Funktionsbaustein zum Ansteuern eines externen Stellgeräts.
MC_ReadActualPosition_BkPlcMc [► 21]	Die Istposition einer Achse wird ermittelt.
MC_ReadActualTorque_BkPlcMc [► 22]	Die Istkraft bzw. der Istdruck einer Achse wird ermittelt.
MC_ReadActualVelocity_BkPlcMc [► 23]	Die Istgeschwindigkeit einer Achse wird ermittelt.
MC_ReadAxisError_BkPlcMc [► 24]	Der aktuelle Errorcode einer Achse wird ermittelt.
MC_ReadBoolParameter_BkPlcMc [► 24]	Die boolschen Parameter einer Achse werden ausgelesen.
MC_ReadDigitalOutput_BkPlcMc [► 25]	Der aktuelle Zustand eines digitalen Ausgangs eines Nockenschaltwerks wird ermittelt.
MC_ReadParameter_BkPlcMc [► 26]	Die nicht boolschen Parameter einer Achse werden ausgelesen.
MC_ReadStatus_BkPlcMc [► 27]	Der Zustand der Achse wird dekodiert.
MC_Reset_BkPlcMc [► 29]	Die Achse wird in einen betriebsbereiten Zustand versetzt.
MC_ResetAndStop_BkPlcMc [► 30]	Die Achse wird in einen betriebsbereiten Zustand versetzt und befindet sich im Stillstand.
MC_SetOverride_BkPlcMc [► 31]	Der Override der Achse wird gesetzt.
MC_SetPosition_BkPlcMc [► 32]	Die Istposition der Achse wird gesetzt.
MC_SetReferenceFlag_BkPlcMc [► 33]	Das Referenzier-Flag der Achse wird definiert. (Funktion ist nicht durch PLCopen definiert)
MC_WriteBoolParameter_BkPlcMc [► 34]	Die boolschen Parameter einer Achse werden geschrieben.
MC_WriteDigitalOutput_BkPlcMc [► 35]	Der aktuelle Zustand eines digitalen Ausgangs eines Nockenschaltwerks wird definiert.
MC_WriteParameter_BkPlcMc [► 36]	Die nicht boolschen Parameter einer Achse werden geschrieben.

Motion Function Blocks, Single Axis

Name	Beschreibung
MC_DigitalCamSwitch_BkPlcMc [▶ 41]	Erzeugung von Software-Nocken in Abhängigkeit von Position, Bewegungsrichtung und Geschwindigkeit einer Achse.
MC_EmergencyStop_BkPlcMc [▶ 44]	Stoppen einer Bewegung ohne Erreichen der Zielposition. (Funktion ist nicht durch PLCopen definiert)
MC_Halt_BkPlcMc [▶ 50]	Stoppen einer Bewegung ohne Erreichen der Zielposition.
MC_Home_BkPlcMc [▶ 51]	Auslösung und Überwachung einer Referenzfahrt.
MC_ImmediateStop_BkPlcMc [▶ 53]	Stoppen einer Bewegung ohne Erreichen der Zielposition. (Funktion ist nicht durch PLCopen definiert)
MC_MoveAbsolute_BkPlcMc [▶ 54]	Start und Überwachung einer Positionierung mit angegebener Geschwindigkeit auf eine absolut angegebene Zielkoordinate.
MC_MoveJoySticked_BkPlcMc [▶ 56]	Starten und Kontrollieren einer Achsbewegung mit einem proportionalen Steuergerät. (Funktion ist nicht durch PLCopen definiert)
MC_MoveRelative_BkPlcMc [▶ 57]	Start und Überwachung einer Positionierung mit angegebener Geschwindigkeit um eine absolut angegebene Strecke.
MC_MoveVelocity_BkPlcMc [▶ 59]	Start und Überwachung einer Positionierung mit angegebener Geschwindigkeit, aber ohne Zielangabe.
MC_Stop_BkPlcMc [▶ 60]	Stoppen einer Bewegung ohne Erreichen der Zielposition.

Motion Function Blocks, Multiple Axis

Name	Beschreibung
MC_CamIn_BkPlcMc [▶ 37]	Der Funktionsbaustein startet und überwacht eine Kurvenscheiben-Kopplung zwischen zwei Achsen.
MC_CamOut_BkPlcMc [▶ 39]	Der Funktionsbaustein löst eine Kurvenscheiben-Kopplung zwischen zwei Achsen.
MC_GearIn_BkPlcMc [▶ 45]	Start und Überwachung der Getriebe-Kopplung von zwei Achsen.
MC_GearInPos_BkPlcMc [▶ 47]	Fliegende Getriebe-Kopplung von zwei Achsen.
MC_GearOut_BkPlcMc [▶ 49]	Auflösen der Getriebe-Kopplung von zwei Achsen.

System Function Blocks

Name	Beschreibung
MC_AxRtDrive_BkPlcMc [▶ 114]	Aufbereitung des Stellwerts der Achse für die Ausgabe auf einer Hardware-Baugruppe, Mapping-Hinweise.
MC_AxRtEncoder_BkPlcMc [▶ 123]	Ermittlung der Istposition der Achse aus den Eingangsinformationen einer Hardware-Baugruppe, Mapping-Hinweise.
MC_AxRtFinish_BkPlcMc [▶ 164]	Anpassung des erzeugten Stellwerts an die Besonderheiten der Achse.
MC_AxRtFinishLinear_BkPlcMc [▶ 165]	Anpassung des erzeugten Stellwerts an die Besonderheiten der Achse unter Berücksichtigung einer Kennlinie.
MC_AxRuntime_BkPlcMc [▶ 155]	Stellwertgenerierung der Achse.

System Function Blocks, andere Istwerte

Name	Beschreibung
MC_AxRtReadForceDiff_BkPlcMc [► 139]	Ermittlung der differentiellen Istkraft einer Achse.
MC_AxRtReadForceSingle_BkPlcMc [► 141]	Ermittlung der einseitigen Istkraft einer Achse.
MC_AxRtReadPressureDiff_BkPlcMc [► 143]	Ermittlung des differentiellen Istdrucks einer Achse.
MC_AxRtReadPressureSingle_BkPlcMc [► 145]	Ermittlung des einseitigen Istdrucks einer Achse.

System Function Blocks, Parameter

Name	Beschreibung
MC_AxAdsCommServer_BkPlcMc [► 187]	Die Applikation wird mit den Fähigkeiten eines ADS-Servers ausgestattet.
MC_AxAdsReadDecoder_BkPlcMc [► 189]	Der Funktionsbaustein dekodiert ADS-Read-Zugriffe für einen ADS-Server.
MC_AxAdsWriteDecoder_BkPlcMc [► 190]	Der Funktionsbaustein dekodiert ADS-Write-Zugriffe für einen ADS-Server.
MC_AxAdsPtrArrCommServer_BkPlcMc [► 188]	Die Applikation wird mit den Fähigkeiten eines ADS-Servers ausgestattet.
MC_AiParamAuxLabelsLoad_BkPlcMc [► 191]	Die Beschriftungstexte für die kundenspezifischen Achsparameter aus einer Datei laden.
MC_AiParamLoad_BkPlcMc [► 192]	Parameter einer Achse aus einer Datei laden.
MC_AiParamSave_BkPlcMc [► 193]	Parameter einer Achse in eine Datei schreiben.
MC_AxUtiReadCoeDriveTerm_BkPlcMc [► 193]	Inhalt eines Registers aus der EL-Klemme lesen, die als Antriebsschnittstelle für die Achse dient.
MC_AxUtiReadCoeEncTerm_BkPlcMc [► 195]	Inhalt eines Registers aus der EL-Klemme lesen, die als Encoderschnittstelle für die Achse dient.
MC_AxUtiReadRegDriveTerm_BkPlcMc [► 196]	Inhalt eines Registers aus der KL-Klemme lesen, die als Antriebsschnittstelle für die Achse dient.
MC_AxUtiReadRegEncTerm_BkPlcMc [► 197]	Inhalt eines Registers aus der KL-Klemme lesen, die als Encoderschnittstelle für die Achse dient.
MC_AxUtiUpdateRegDriveTerm_BkPlcMc [► 198]	Einen Parametersatz in die Register einer KL-Klemme schreiben, die als Antriebsschnittstelle für die Achse dient.
MC_AxUtiUpdateRegEncTerm_BkPlcMc [► 200]	Einen Parametersatz in die Register einer KL-Klemme schreiben, die als Encoderschnittstelle für die Achse dient.
MC_AxUtiWriteCoeDriveTerm_BkPlcMc [► 201]	Inhalt eines Registers in der EL-Klemme schreiben, die als Antriebsschnittstelle für die Achse dient.
MC_AxUtiWriteCoeEncTerm_BkPlcMc [► 203]	Inhalt eines Registers in der EL-Klemme schreiben, die als Encoderschnittstelle für die Achse dient.
MC_AxUtiWriteRegDriveTerm_BkPlcMc [► 204]	Inhalt eines Registers in der KL-Klemme schreiben, die als Antriebsschnittstelle für die Achse dient.
MC_AxUtiWriteRegEncTerm_BkPlcMc [► 205]	Inhalt eines Registers in der KL-Klemme schreiben, die als Encoderschnittstelle für die Achse dient.
MC_LinTableExportToAsciiFile_BkPlcMc	in Vorbereitung
MC_LinTableExportToBinFile_BkPlcMc	in Vorbereitung
MC_LinTableImportFromAsciiFile_BkPlcMc	in Vorbereitung
MC_LinTableImportFromBinFile_BkPlcMc	in Vorbereitung

System Function Blocks, Controllers

Name	Beschreibung
MC_AxCtrlAutoZero_BkPlcMc [► 104]	Automatischer Nullpunktabgleich.
MC_AxCtrlPressure_BkPlcMc [► 106]	Regler für eine aufbauend wirkende Druckregelung.
MC_AxCtrlPressureCompensation_BkPlcMc	Anpassung der Ausgabewerte einer Achse an den Ventildruckabfall.
MC_AxCtrlPullbackOnPressure_BkPlcMc	Regler für eine verdrängend wirkende Druckregelung.
MC_AxCtrlSlowDownOnPressure_BkPlcMc [► 110]	Regler für eine ablösende Druckregelung.
MC_AxCtrlStepperDeStall_BkPlcMc [► 112]	Überwachung der Bewegung einer Schrittmotorachse.
MC_AxCtrlVelocity_BkPlcMc	Regler für die Achs-Geschwindigkeit.
MC_AxCtrlVeloMoving_BkPlcMc	Regler für die Achs-Geschwindigkeit.

System Function Blocks, TableFunctions

Name	Beschreibung
MC_AxTableFromAsciiFile_BkPlcMc [► 154]	Lesen des Inhalts einer Tabelle aus einer Textdatei.
MC_AxTableFromBinFile_BkPlcMc [► 153]	Lesen des Inhalts einer Tabelle aus einer binäre Datei.
MC_AxTableReadOutNonCyclic_BkPlcMc [► 151]	Funktionsbaustein zur Ermittlung der einem Master-Wert zugeordneten Slave-Werte mit Hilfe einer Tabelle.
MC_AxTableToAsciiFile_BkPlcMc [► 150]	Schreiben des Inhalts einer Tabelle in eine Textdatei.
MC_AxTableToBinFile_BkPlcMc [► 149]	Schreiben des Inhalts einer Tabelle in eine binäre Datei.

System Function Blocks, Message Logging

Name	Beschreibung
MC_AxRtLogAxisEntry_BkPlcMc [► 173]	Eine achsbezogene Meldung wird in den LogBuffer der Library eingetragen.
MC_AxRtLogClear_BkPlcMc [► 174]	Alle Einträge im LogBuffer löschen und initialisieren.
MC_AxRtLogEntry_BkPlcMc [► 174]	Eine Meldung wird in den LogBuffer der Library eingetragen.
MC_AxRtLoggerDespool_BkPlcMc [► 175]	Minimalanzahl von freien Meldungen im LogBuffer der Library sicherstellen.
MC_AxRtLoggerRead_BkPlcMc [► 175]	Lesen einer Meldung aus dem LogBuffer der Library.
MC_AxRtLoggerSpool_BkPlcMc [► 176]	Übertragung von Meldungen aus dem LogBuffer der Library in die Ereignisanzeige von Windows.

System Function Blocks, Laufzeitfunktionen

Name	Beschreibung
MC_AxRtCheckSyncDistance_BkPlcMc [► 163]	Überwachung der Entfernung zwischen Referenznocken und Nullimpuls.
MC_AxRtCmdBufferExecute_BkPlcMc	in Vorbereitung
MC_AxRtCommandsLocked_BkPlcMc [► 177]	in Vorbereitung
MC_AxRtGoErrorState_BkPlcMc [► 167]	(not recommended) Die Achse wird in einen Stöorzustand versetzt.
MC_AxRtMoveChecking_BkPlcMc [► 167]	Überwachung der Bewegung einer Achse.
MC_AxRtSetDirectOutput_BkPlcMc [► 168]	Direkte Ausgabe eines Stellwerts.
MC_AxRtSetExtGenValues_BkPlcMc [► 169]	Versorgung einer Achse mit Führungsgrößen, die nicht aus dem achseigenen Generator stammen.
MC_AxStandardBody_BkPlcMc [► 170]	Aufruf der üblichen Unterbestandteile einer Achse (Encoder, Generator, Finish, Drive).
MC_AxUtiAutoIdent_BkPlcMc [► 181]	Automatische Ermittlung von Parametern der Achse.
MC_AxUtiAutoldentSlave_BkPlcMc	in Vorbereitung: Automatische Ermittlung von Parametern einer Slave-Achse.
MC_AxUtiAverageDerivative_BkPlcMc [► 177]	Ermittlung der Ableitung eines Wertes durch numerische Differentiation über mehr als einen Zyklus.
MC_AxUtiPT1_BkPlcMc [► 178]	Berechnung eines Tiefpass 1. Ordnung.
MC_AxUtiPT2_BkPlcMc [► 179]	Berechnung eines Tiefpass 2. Ordnung.
MC_AxUtiSlewRateLimiter_BkPlcMc [► 179]	Erzeugung einer ansteigsbegrenzten Rampe.
MC_AxUtiSlidingAverage_BkPlcMc [► 180]	Ermittlung eines gleitenden Mittelwerts.
MC_AxUtiStandardInit_BkPlcMc [► 171]	Initialisierung und Überwachung der Bestandteile einer Achse.
MC_FunctionGeneratorFD_BkPlcMc [► 147]	Ein Funktionsgenerator.
MC_FunctionGeneratorSetFrq_BkPlcMc [► 147]	Aktualisiert die Arbeitsfrequenz einer Zeitbasis für einen oder mehrere Funktionsgeneratoren.
MC_FunctionGeneratorTB_BkPlcMc [► 148]	Aktualisiert eine Zeitbasis für einen oder mehrere Funktionsgeneratoren.

Datentypen: Enumerationen

Name	Beschreibung
E_TcMcCurrentStep [► 65]	Diese Enumeration liefert Codes für die internen Zustände der Stellwertgeneratoren.
E_TcMcDriveType [► 67]	Die Konstanten in dieser Enumeration werden zur Kennzeichnung von für die Stellwertausgabe einer Achse verwendeter Hardware benutzt.
E_TcMcEncoderType [► 70]	Die Konstanten in dieser Enumeration werden zur Kennzeichnung von für die Istwerterfassung einer Achse verwendeter Hardware benutzt.
E_TcMCFbState [► 73]	Diese Enumeration liefert Codes für den aktuellen Zustand einer Achse.
E_TcMcHomingType [► 73]	Diese Enumeration liefert Codes für die Referenziermethode einer Achse.
E_TcMCPParameter [► 74]	Die Konstanten in dieser Auflistung werden zur Parameternummerierung benutzt.
E_TcMcPressureReadingMode [► 84]	Die Konstanten in dieser Auflistung legen fest, welcher Istwert in der ST_TcHydAxRtData Struktur der Achse mit dem Ergebnis einer Druck- oder Krafterfassung zu aktualisieren ist.
E_TcMcProfileType [► 83]	Die Konstanten in dieser Auflistung werden zur Kennzeichnung von Stellwertgeneratoren benutzt.
E_TcPlcBufferedCmdType_BkPlcMc [► 64]	In Vorbereitung: Die Konstanten in dieser Auflistung werden zur Kennzeichnung von bepufferten Achskommandos.
MC_BufferMode_BkPlcMc [► 84]	Die Konstanten in dieser Auflistung werden zur Steuerung des Blendings nach PLC Open verwendet.
MC_Direction_BkPlcMc [► 85]	Diese Enumeration liefert Codes für die Richtung einer Bewegung, wenn diese Information nicht in anderen Daten enthalten oder aus der Situation heraus feststellbar ist.
MC_HomingMode_BkPlcMc [► 86]	Diese Enumeration liefert Codes für die Festlegung der Referenziermethode.
MC_StartMode_BkPlcMc [► 86]	Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Modi beim Starten von Achsen benutzt.

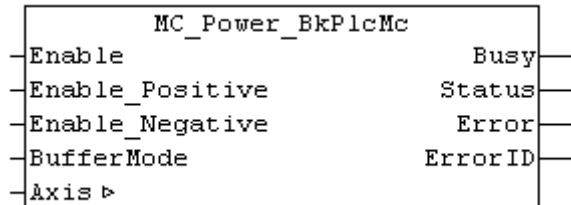
Datentypen: Strukturen

Name	Beschreibung
Axis_Ref_BkPlcMc [► 62]	Eine Variable von diesem Typ enthält alle nötigen Variablen oder Pointer auf Variablen, die zu einer Achse gehören.
CAMSWITCH_REF_BkPlcMc [► 63]	Eine Variable von diesem Typ wird an einen MC_DigitalCamSwitch_BkPlcMc [► 41] Baustein übergeben.
MC_CAM_ID_BkPlcMc [► 85]	Eine Variable von diesem Typ enthält die Beschreibung einer zur Kopplung vorbereiteten Kurvenscheibe.
MC_CAM_REF_BkPlcMc [► 85]	Eine Variable von diesem Typ enthält die Beschreibung einer bereitgestellten Kurvenscheibe.
OUTPUT_REF_BkPlcMc [► 87]	Eine Variable von diesem Typ enthält Ausgabedaten eines MC_DigitalCamSwitch_BkPlcMc [► 41] Bausteins.
ST_FunctionGeneratorFD_BkPlcMc [► 87]	Eine Variable von diesem Typ enthält Parameter für die Definition der Ausgangssignale eines Funktionsgenerators.
ST_FunctionGeneratorTB_BkPlcMc [► 87]	Eine Variable von diesem Typ enthält Parameter für die Definition einer Zeitbasis für einen Funktionsgenerator.
ST_TcMcAutoIdent [► 88]	Eine Variable von diesem Typ enthält die Parameter für einen MC_AxUtiAutoIdent_BkPlcMc [► 181] Baustein.
ST_TcMcAuxDataLabels [► 98]	Eine Variable von diesem Typ enthält Beschriftungstexte der kundenspezifischen Achsparameter.
ST_TcHydAxParam [► 88]	Eine Variable von diesem Typ enthält alle Parameter einer Achse.
ST_TcHydAxRtData [► 94]	Eine Variable von diesem Typ enthält die Laufzeitdaten einer Achse.
ST_TcPlcMcLogBuffer [► 102]	Eine Variable mit dieser Struktur bildet den LogBuffer der Library.
ST_TcPlcMcLogEntry [► 102]	Eine Variable mit dieser Struktur enthält eine Meldung des LogBuffer der Library.
ST_TcPlcDeviceInput [► 98]	Diese Struktur enthält die Eingangsabbild-Variablen einer Achse.
ST_TcPlcDeviceOutput [► 100]	Diese Struktur enthält die Ausgangsabbild-Variablen einer Achse.
ST_TcPlcRegDataItem [► 103]	Diese Struktur enthält einen Parametersatz für eine KL-Klemme.
ST_TcPlcRegDataTable [► 103]	Diese Struktur enthält einen Parameter für eine KL-Klemme.
TRACK_REF_BkPlcMc [► 103]	In Vorbereitung.

3 PLCopen Motion Control

3.1 Administrative

3.1.1 MC_Power_BkPlcMc (ab V3.0)



Der Funktionsbaustein dient zum Ansteuern eines externen Stellgeräts. Weitere Informationen zu diesem Thema finden Sie unter [FAQ #9 \[► 216\]](#).

```

VAR_INPUT
  Enable:          BOOL;
  Enable_Positive: BOOL;
  Enable_Negative: BOOL;
  BufferMode:      MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc; (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:    BOOL;
  Status:  BOOL;
  Error:   BOOL;
  ErrorID: UDINT;
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
    
```

Enable: Ein TRUE an diesem Eingang schaltet ein externes Stellgerät einer Achse aktiv.

Enable_Positive: Ein TRUE an diesem Eingang schaltet die richtungsbezogene Freigabe eines externen Stellgeräts einer Achse für Bewegungen in positiver Richtung aktiv.

Enable_Negative: Ein TRUE an diesem Eingang schaltet die richtungsbezogene Freigabe eines externen Stellgeräts einer Achse für Bewegungen in negativer Richtung aktiv.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten Aborting_BkPlcMc belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Status: Hier wird die Betriebsbereitschaft signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlermeldung bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Dieser Baustein dient zur Steuerung von externen Stellgeräten. Dabei kann es sich um Baugruppen zur Ventilansteuerung (Onboard Endstufe des Ventils oder Schaltschrank-Baugruppe), um Frequenzumrichter oder Servoverstärker handeln. Diese Geräte benötigen in der Regel ein digitales Signal zur Freigabe der Energieabgabe durch eine Leistungsstufe. Je nach Ausführung des Geräts können zusätzlich die Bewegungsrichtungen "Positiv" und "Negativ" gezielt aktiv geschaltet werden.

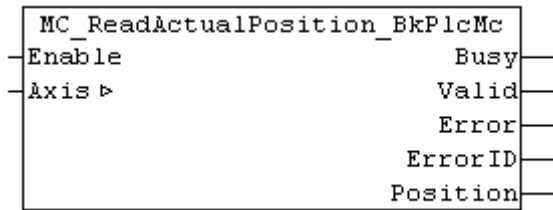
Die Eingangssignale des Bausteins werden in die Interfaces zum Peripherie-Gerät weitergereicht. Zusätzlich wird durch **Enable** eine Fehlerüberwachung aktiviert.

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Ist in `nDrive_Type` in `pStAxParams` der Wert `iTcMc_DriveAx2000_XXXXX` eingestellt, wird wie folgt weiter verfahren:
 - Ist einer der Pointer `pStDeviceOutput` und `pStDeviceInput` in `Axis_Ref_BkPlcMc` [► 62] nicht initialisiert reagiert der Baustein mit **Error** und **ErrorID**:=`dwTcHydErrCdPtrPlcDriveIn` bzw. `dwTcHydErrCdPtrPlcDriveOut`. **Status** ist dann **FALSE**.
 - Wenn in der Kommunikation mit dem AX-Gerät eine Störung erkannt wird oder im `pStDeviceInput` Interface eine Störmeldung des AX-Geräts auftritt reagiert der Baustein mit **Error** und einem **ErrorID**, der in den `Globalen Konstanten` [► 229] der Bibliothek definiert ist. **Status** ist dann **FALSE** und die Achse wird in einen Störzustand mit dem Achsfehler `dwTcHydErrCdDriveNotReady` versetzt.
 - Andernfalls wird am **Status** der Wert von **Enable** zurückgegeben.
- Ist in `nDrive_Type` in `pStAxParams` der Wert `iTcMc_DriveKL2531` oder `iTcMc_DriveKL2541` eingestellt, wird wie folgt weiter verfahren:
 - Die Pointer `pStDeviceOutput` und `pStDeviceInput` in `Axis_Ref_BkPlcMc` [► 62] werden überprüft. Sind diese Pointer nicht initialisiert reagiert der Baustein mit **Error** und **ErrorID**:=`dwTcHydErrCdPtrPlcDriveIn` bzw. `dwTcHydErrCdPtrPlcDriveOut`. **Status** ist dann **FALSE**.
 - Wenn in der Kommunikation mit der E/A-Klemme eine Störung erkannt wird oder im `pStDeviceInput` Interface eine Störmeldung der Klemme auftritt reagiert der Baustein mit **Error** und einem **ErrorID**, der in den `Globalen Konstanten` [► 229] der Bibliothek definiert ist. **Status** ist dann **FALSE** und die Achse wird in einen Störzustand mit dem Achsfehler `dwTcHydErrCdDriveNotReady` versetzt.
 - **Enable** wird benutzt um die Klemmenendstufe durch ein Bit in `pStDeviceOutput.bTerminalCtrl` zu aktivieren. Das Bereit-Signal in `bTerminalCtrl.bTerminalState` wird als **Status** zurückgegeben.
 - Der Wert von **Enable_Positive** wird bei fehlerfreiem Zustand des Antriebsinterfaces mit der Maske `dwTcHydDcDwFdPosEna` in `nDeCtrlDWord` von `pStAxRtData` eingetragen.
 - Der Wert von **Enable_Negative** wird bei fehlerfreiem Zustand des Antriebsinterfaces mit der Maske `dwTcHydDcDwFdNegEna` in `nDeCtrlDWord` von `pStAxRtData` eingetragen.
- Andernfalls wird eine Überprüfung der Pointer `pStDeviceInput` und `pStDeviceOutput` in `Axis_Ref_BkPlcMc` [► 62] vorgenommen. Sind diese Pointer nicht initialisiert reagiert der Baustein mit **Error** und **ErrorID**:=`dwTcHydErrCdPtrPlcDriveIn` bzw. `dwTcHydErrCdPtrPlcDriveOut`. **Status** ist dann **FALSE**.
 - Andernfalls wird am **Status** der Wert von `bPowerOk` aus `pStDeviceInput` zurückgegeben.
- Der Wert von **Enable** wird bei fehlerfreiem Zustand des Antriebsinterfaces mit der Maske `dwTcHydDcDwCtrlEnable` in `nDeCtrlDWord` von `pStAxRtData` eingetragen.
- Der Wert von **Enable_Positive** wird bei fehlerfreiem Zustand des Antriebsinterfaces mit der Maske `dwTcHydDcDwFdPosEna` in `nDeCtrlDWord` von `pStAxRtData` eingetragen.
- Der Wert von **Enable_Negative** wird bei fehlerfreiem Zustand des Antriebsinterfaces mit der Maske `dwTcHydDcDwFdNegEna` in `nDeCtrlDWord` von `pStAxRtData` eingetragen.

ⓘ HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang Busy wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.2 MC_ReadActualPosition_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt die aktuelle Position einer Achse.

```

VAR_INPUT
  Enable:      BOOL;
END_VAR
VAR_OUTPUT
  Busy:        BOOL;
  Valid:       BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
  Position:    LREAL;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR
  
```

Enable: Eine steigende Flanke an diesem Eingang löst eine Aktualisierung des Positionswertes aus.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Valid: Hier wird die erfolgreiche Ermittlung der Istposition signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Position: Die Istposition.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

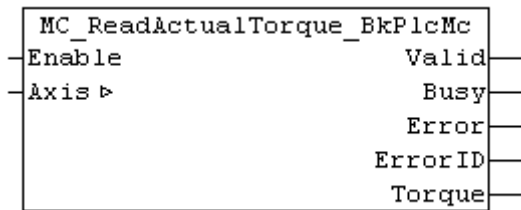
- Befindet sich die Achse in einem gestörten Zustand und ist die Ursache ein Encoder-Problem wird mit **Error** und **ErrorID:=Errorcode** des Encoders reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Istposition ermittelt und **Valid** gemeldet.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

ⓘ HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang **Busy** wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.3 MC_ReadActualTorque_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt die aktuelle Istkraft bzw. den Istdruck einer Achse.

```

VAR_INPUT
  Enable:      BOOL;
END_VAR
VAR_OUTPUT
  Valid:       BOOL;
  Busy:        BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
  Torque:      LREAL;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Eine steigende Flanke an diesem Eingang löst eine Aktualisierung des Istwertes aus.

Valid: Hier wird die erfolgreiche Ermittlung des Istwertes signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Torque: Die Istkraft bzw. der Istdruck.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

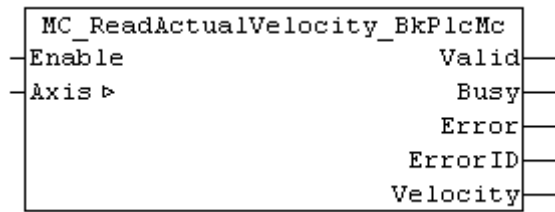
- Befindet sich die Achse in einem gestörten Zustand und ist die Ursache ein Encoder-Problem wird mit **Error** und **ErrorID**=Errorcode des Encoders reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Istkraft bzw. der Istdruck ermittelt und **Valid** gemeldet.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

ⓘ HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang **Busy** wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.4 MC_ReadActualVelocity_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt die aktuelle Geschwindigkeit einer Achse.

```

VAR_INPUT
    Enable:    BOOL;
END_VAR
VAR_OUTPUT
    Valid:    BOOL;
    Busy:    BOOL;
    Error:    BOOL;
    ErrorID:  UDINT;
    Velocity: LREAL;
END_VAR
VAR_INOUT
    Axis:      Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Eine steigende Flanke an diesem Eingang löst eine Aktualisierung des Geschwindigkeitswertes aus.

Valid: Hier wird die erfolgreiche Ermittlung der Geschwindigkeit signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Velocity: Die Istgeschwindigkeit.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

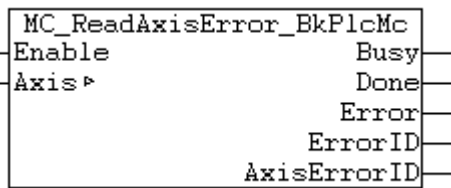
- Befindet sich die Achse in einem gestörten Zustand und ist die Ursache ein Encoder-Problem wird mit **Error** und **ErrorID:=Errorcode** des Encoders reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Geschwindigkeit ermittelt und **Valid** gemeldet.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang **Busy** wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.5 MC_ReadAxisError_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt den aktuellen Errorcode einer Achse.

```

VAR_INPUT
  Enable:      BOOL;
END_VAR
VAR_OUTPUT
  Busy:        BOOL;
  Done:        BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
  AxisErrorID:UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Ein TRUE an diesem Eingang löst eine Aktualisierung des Errorcodes aus.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Ermittlung der Istposition signalisiert.

Error: Hier wird TRUE signalisiert, wenn der Baustein die gewünschte Funktion nicht ausführen konnte.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt wenn der Baustein die gewünschte Funktion nicht ausführen konnte.

AxisErrorID: Hier wird der aktuelle Fehlercode [\[► 224\]](#) der Achse bereitgestellt.

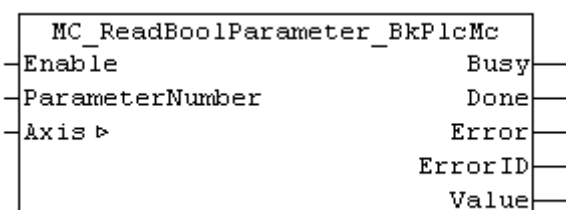
Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Wenn **Enable** TRUE ist untersucht der Baustein das übergebene Achsinterface. Der aktuelle Fehlercode wird als **AxisErrorID** gemeldet. Ist **Enable** FALSE löscht der Baustein alle anstehenden Ausgangssignale.

ⓘ HINWEIS! Dieser Baustein benötigt keine Zeit und keine Vorbedingungen zur Durchführung seiner Aufgaben. Die Ausgänge Error und Busy werden zu keinem Zeitpunkt den Wert TRUE annehmen und sind nur aus Kompatibilitätsgründen vorhanden.

3.1.6 MC_ReadBoolParameter_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest die boolschen Parameter einer Achse aus. Für nicht boolsche Parameter steht der Baustein [MC_ReadParameter_BkPlcMc \[► 26\]](#) zur Verfügung.


```

VAR_INPUT
  Enable:          BOOL;
  ParameterNumber: INT;
END_VAR
VAR_OUTPUT
  Busy:           BOOL;
  Done:           BOOL;
  Error:          BOOL;
  ErrorID:       UDINT;
  Value:         BOOL;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR
    
```

Enable: Eine steigende Flanke an diesem Eingang löst einen Lesevorgang aus.

ParameterNumber: Diese Kennnummer legt den auszulesenden Parameter fest. Es sollten nur benannte Konstanten aus [E_TcMCPParameter \[► 74\]](#) verwendet werden.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Value: Hier wird der Wert des Parameters zur Verfügung gestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

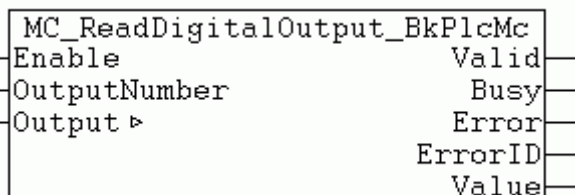
- Wenn an **ParameterNumber** ein nicht unterstützter Wert angelegt wurde wird mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten steht an **Value** der gewünschte Parameterwert zur Verfügung und **Done** wird gemeldet.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

ⓘ HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang Busy wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.7 MC_ReadDigitalOutput_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt den aktuellen Zustand eines digitalen Ausgangs eines Nockenschaltwerks.

```

VAR_INPUT
  Enable:          BOOL;
  OutputNumber:   INT;
END_VAR
    
```

```

VAR_OUTPUT
  Done:      BOOL;
  Busy:      BOOL;
  Error:     BOOL;
  ErrorID:   UDINT;
  Value:     BOOL;
END_VAR
VAR_INOUT
  Output:    OUTPUT_REF_BkPlcMc;
END_VAR

```

Enable: Eine steigende Flanke an diesem Eingang löst eine Aktualisierung des Zustands aus.

OutputNumber: Die Nummer des zu ermittelnden Ausgangs.

Valid: Hier wird die erfolgreiche Ermittlung des Zustands signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Value: Der Zustand des digitalen Ausgangs.

Output: Hier ist die Adresse einer Variablen vom Typ OUTPUT_REF_BkPlcMc [► 87] zu übergeben.

Verhalten des Bausteins

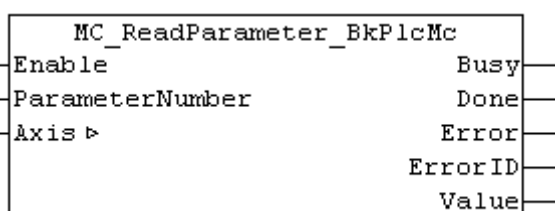
Bei einem TRUE an **Enable** untersucht der Baustein die übergebenen Parameter. Dabei kann ein Problem erkannt und gemeldet werden:

- Wenn der Wert von **OutputNumber** nicht im zulässigen Bereich [0..31] liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdIllegalOutputNumber** reagiert.

Wenn diese Überprüfungen ohne Probleme durchgeführt werden konnten wird der Zustand des digitalen Ausgangs ermittelt und **Valid** gemeldet.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

3.1.8 MC_ReadParameter_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest die nicht boolschen Parameter einer Achse aus. Für boolsche Parameter steht der Baustein MC_ReadBoolParameter_BkPlcMc [► 24] zur Verfügung.

```

VAR_INPUT
  Enable:      BOOL;
  ParameterNumber: INT;
END_VAR
VAR_OUTPUT
  Busy:      BOOL;
  Done:     BOOL;
  Error:    BOOL;
  ErrorID:  UDINT;
  Value:    LREAL;
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Eine steigende Flanke an diesem Eingang löst einen Lesevorgang aus.

ParameterNumber: Diese Kennnummer legt den auszulesenden Parameter fest. Es sollten nur benannte Konstanten aus [E_TcMCPParameter \[► 74\]](#) verwendet werden.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Value: Hier wird der Wert des Parameters zur Verfügung gestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

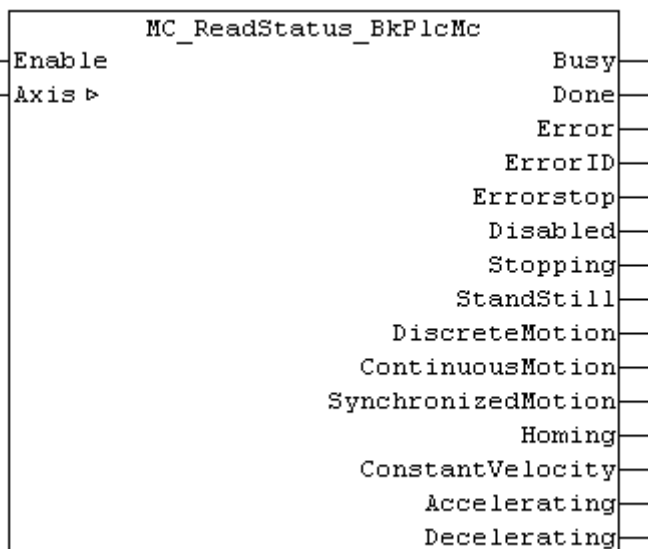
- Wenn an **ParameterNumber** ein nicht unterstützter Wert angelegt wurde wird mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten steht an **Value** der gewünschte Parameterwert zur Verfügung und **Done** wird gemeldet.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang **Busy** wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.9 MC_ReadStatus_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt den aktuellen Zustand einer Achse.

```

VAR_INPUT
  Enable:          BOOL;
END_VAR
VAR_OUTPUT
  Busy:           BOOL;
  Done:          BOOL;

```

```

Error:          BOOL;
ErrorID:       UDINT;
Errorstop:     BOOL;
Disabled:      BOOL;
Stopping:      BOOL;
StandStill:    BOOL;
DiscreteMotion:  BOOL;
ContinuousMotion:  BOOL;
SynchronizedMotion:  BOOL;
Homing:       BOOL;
ConstantVelocity:  BOOL;
Accelerating:  BOOL;
Decelerating:  BOOL;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Ein TRUE Zustand an diesem Eingang aktiviert eine Aktualisierung des Bausteins aus.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Ermittlung der Istposition signalisiert.

Error: Dieser Ausgang meldet Probleme bei der Ausführung der Funktion des Bausteins.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt wenn der Baustein die gewünschte Funktion nicht ausführen konnte.

Errorstop: Dieses Signal zeigt an, dass die Achse eines Fehlers in einen nicht betriebsbereiten Zustand versetzt wurde. Dieser Zustand ist nur durch die Aktivierung eines [MC_Reset_BkPlcMc \[► 29\]](#) oder [MC_ResetAndStop_BkPlcMc \[► 30\]](#) Bausteins aufhebbar.

Disabled: Dieses Signal zeigt an, ob die Achse durch ihren [MC_Power_BkPlcMc \[► 19\]](#) Baustein enabled oder disabled wurde.

Stopping: Dieses Signal zeigt an, dass eine aktive Bewegung der Achse durch einen [MC_Stop_BkPlcMc \[► 60\]](#) oder [MC_ResetAndStop_BkPlcMc \[► 30\]](#) Baustein beendet wird. Sobald die Achse den Stillstand erreicht hat wird dieses Signal gelöscht.

StandStill: Dieses Signal zeigt an, dass die Achse weder gestört noch aktiv ist.

DiscreteMotion: Dieses Signal zeigt an, dass die Achse eine eigenständige (nicht durch eine Kopplung zustande gekommene) Bewegung mit definiertem Ziel ausführt.

ContinuousMotion: Dieses Signal zeigt an, dass die Achse eine eigenständige (nicht durch eine Kopplung zustande gekommene) Bewegung mit definierter Geschwindigkeit, aber ohne festgelegtes Ziel ausführt.

SynchronizedMotion: Dieses Signal zeigt an, dass die Achse durch eine Getriebekopplung kontrolliert wird.

Homing: Dieses Signal zeigt an, dass die Achse eine Referenzfahrt ausführt.

ConstantVelocity: Dieses Signal zeigt an, dass die Achse mit konstanter Geschwindigkeit bewegt wird.

Accelerating: Dieses Signal zeigt an, dass die Achse ihre Geschwindigkeit einem vorgesehenen Wert annähert.

HINWEIS! Dies bedeutet nicht immer eine Betragserhöhung der Geschwindigkeit: beim Starten einer bereits fahrenden Achse kann es vorkommen, dass die Achse entgegen dem aktuellen Geschwindigkeitsvorzeichen beschleunigt, um eine vorgesehene Geschwindigkeit in Gegenrichtung aufzubauen. Dies ist aus Sicht der ursprünglichen Bewegung eine Verzögerung, aus Sicht der aktuellen (neuen) Bewegung jedoch eine Beschleunigung.

Decelerating: Dieses Signal zeigt an, dass die Achse ihre Geschwindigkeit reduziert, um eine Bewegung mit einer geringeren als der aktuellen Geschwindigkeit fortzusetzen oder zu beenden.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

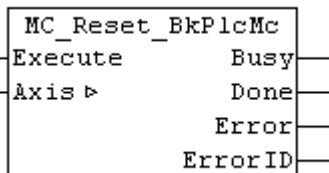
Wenn an **Enable** TRUE anliegt untersucht der Baustein das übergebene Achsinterface und dekodiert die internen Zustandsinformationen. Ein FALSE Zustand an **Enable** löscht alle anstehenden Ausgangssignale.

HINWEIS! Dieser Baustein benötigt keine Zeit und keine Vorbedingungen zur Durchführung seiner Aufgaben. Die Ausgänge **Error** und **Busy** werden zu keinem Zeitpunkt den Wert TRUE annehmen und sind nur aus Kompatibilitätsgründen vorhanden.

 Hinweis	Ausgänge beachten Die Ausgänge Error und ErrorID geben den Zustand des Bausteins und nicht den der Achse wieder.
-----------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

Um den aktuellen Fehlercode der Achse auszulesen ist ein `MC_ReadAxisError_BkPlcMc()` [► 24] Baustein zu verwenden.

3.1.10 MC_Reset_BkPlcMc (ab V3.0)



Der Funktionsbaustein beseitigt einen Störzustand und versetzt die Achse in einen betriebsbereiten Zustand.

```

VAR_INPUT
    Execute:    BOOL;
END_VAR
VAR_OUTPUT
    Busy:      BOOL;
    Done:     BOOL;
    Error:    BOOL;
    ErrorID:  UDINT;
END_VAR
VAR_INOUT
    Axis:     Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang löst einen Achsreset aus.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung des Achsreset signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

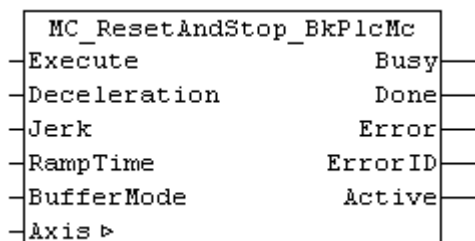
Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin wird der Achsreset durchgeführt. Dadurch wird die Achse soweit möglich in einem betriebsbereiten Zustand versetzt und **Done** gemeldet. Ist dies nicht möglich wird mit **Error** und **ErrorID:=Achs-ErrorCode** reagiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale.

HINWEIS! Bei einigen Antriebstypen ist zur Behebung bestimmter Fehler ein Signalaustausch mit einem externen Gerät erforderlich. Während der dafür benötigten Zeit kann der Baustein kein endgültiges Ergebnis (Done oder Error) melden, sondern signalisiert mit Busy die andauernde Bearbeitung der Funktion.

3.1.11 MC_ResetAndStop_BkPlcMc (ab V3.0)



Der Funktionsbaustein versetzt eine gestörte Achse in einen betriebsbereiten Zustand. Wenn die Achse einen Fahrauftrag abarbeitet wird dieser abgebrochen und die dazu nötige Stopp-Operation wird überwacht.

```

VAR_INPUT
  Execute:      BOOL;
  Deceleration: LREAL; (ab/from V3.0.5)
  Jerk:         LREAL; (ab/from V3.0.5)
  RampTime:     LREAL; (ab/from V3.0.5)
  BufferMode:    MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc; (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:         BOOL;
  Done:         BOOL;
  Error:        BOOL;
  ErrorID:      UDINT;
END_VAR
VAR_INOUT
  Axis:         Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang löst einen Achsreset und eine Stop-Operation aus.

Deceleration: Die anzuwendende Verzögerung.

Jerk: Der anzuwendende Ruck.

RampTime: Die geforderte Anhaltezeit.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten Aborting_BkPlcMc belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung des Achsreset signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [▶ 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Ist ein anstehender Störszustand der Achse durch die Reset-Operation nicht erfolgreich zu beheben wird mit **Error** und **ErrorID:=Achs-ErrorCode** reagiert.

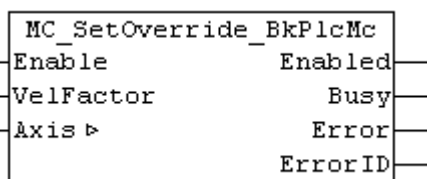
- Wird die Achse im Verlauf einer eventuell erforderlichen Stop-Operation in einen Stöorzustand versetzt wird mit **Error** und **ErrorID:=Achs-ErrorCode** reagiert.

Nach erfolgreichem Abschluss beider Operationen wird **Done** gemeldet. Die Achse ist dann störungsfrei und im Stillstand.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale.

HINWEIS! Wenn die Achse eine Bewegung ausführt wird sie bis zum Stillstand abgebremst. Zusätzlich ist bei einigen Antriebstypen zur Behebung bestimmter Fehler ein Signalaustausch mit einem externen Gerät erforderlich. Während der dafür benötigten Zeit kann der Baustein kein endgültiges Ergebnis (**Done** oder **Error**) melden, sondern signalisiert mit **Busy** die andauernde Bearbeitung der Funktion.

3.1.12 MC_SetOverride_BkPlcMc (ab V3.0)



Der Funktionsbaustein setzt den Override einer Achse.

HINWEIS! Dieser Baustein wirkt nur bei Verwendung des Profiltyps **iTcMc_ProfileCtrlBased**.

```

VAR_INPUT
  Enable:      BOOL;
  VelFactor:   LREAL;
END_VAR
VAR_OUTPUT
  Enabled:     BOOL;
  Busy:        BOOL;
  Error:       BOOL;
  ErrorID:    UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR
    
```

Enable: Ein aktiver Zustand an diesem Eingang setzt den Override der Achse.

VelFactor: Der neue Override der Achse.

Enabled: Hier wird der aktive Zustand des Bausteins signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

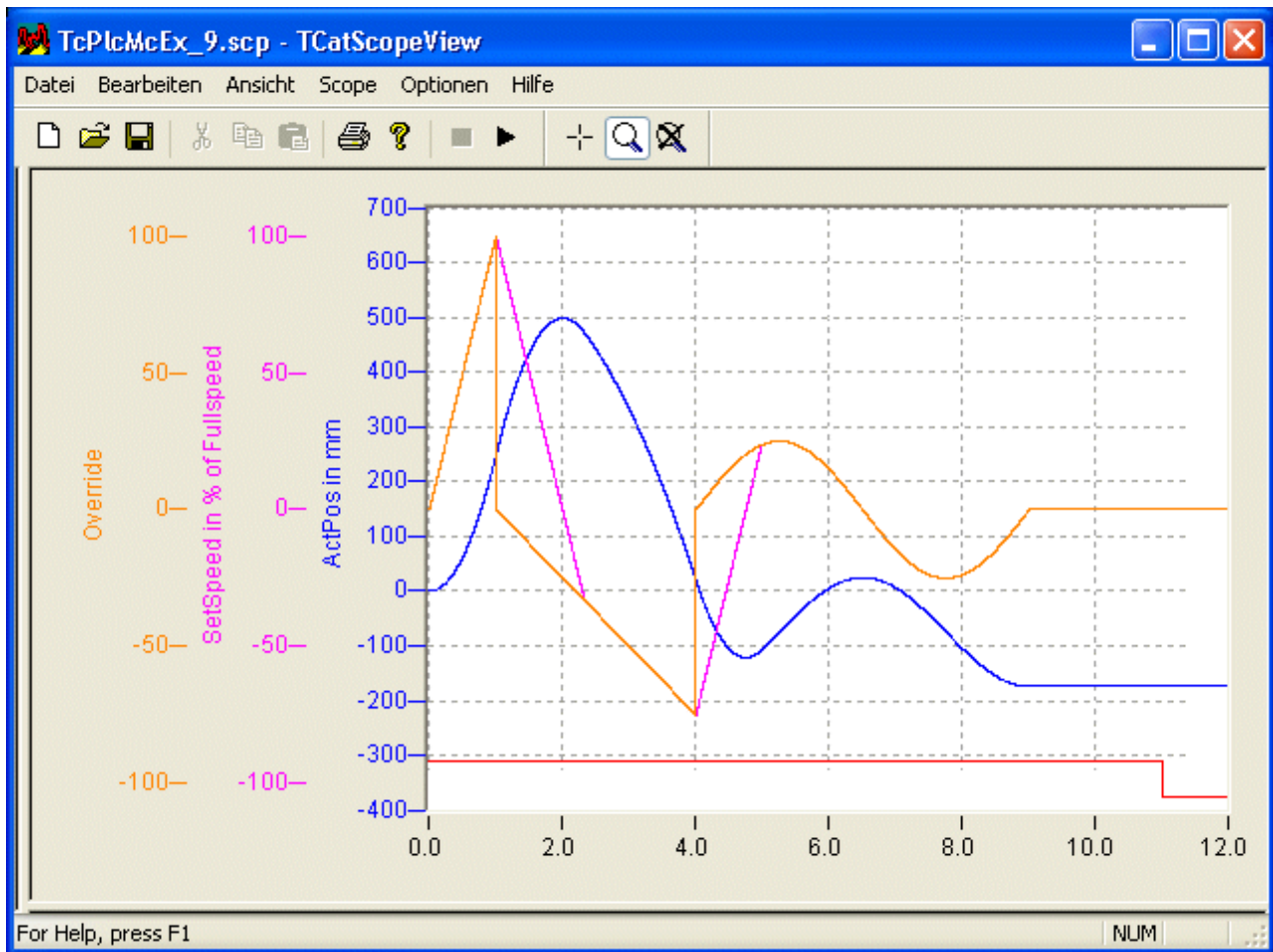
ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Bei einem aktiven Zustand an **Enable** wird der als **VelOverride** übergebene Wert auf den Bereich 0.0 bis 1.0 begrenzt und in **Axis.pStAxParams^.fOverride** eingetragen. **Enabled** wird auf TRUE gesetzt.

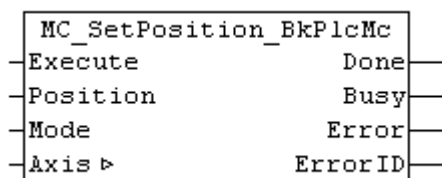
Eine fallende Flanke an **Enable** löscht alle Ausgänge.



Alle durch eine Override-Änderung hervorgerufenen Geschwindigkeitsänderungen werden entsprechend der maximal zulässigen Beschleunigungen und Verzögerungen begrenzt.

HINWEIS! Um ein reproduzierbares Verhalten bei der Zielfahrt zu garantieren wird durch den Override die Fahrgeschwindigkeit nur bis auf `pStAxParams.fCreepSpeed` reduziert. Dadurch ist es nicht möglich, die Achse durch einen Override von 0.0 in der Bewegung zu stoppen.

3.1.13 MC_SetPosition_BkPlcMc (ab V3.0)



Der Funktionsbaustein setzt die Istposition einer Achse.

```

VAR_INPUT
  Execute:    BOOL;
  Position:   LREAL;
  Mode:       BOOL;
END_VAR
VAR_OUTPUT
  Done:       BOOL;
  Busy:       BOOL;
  Error:      BOOL;
  ErrorID:    UDINT;
END_VAR
    
```



```
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
```

Execute: Eine steigende Flanke an diesem Eingang setzt die Istposition der Achse.

Position: Die neue Istposition der Achse.

Mode: Dieser Parameter legt den Arbeitsmodus fest. Bei **Mode=TRUE** wird die Istposition um **Position** geändert, bei **Mode=FALSE** wird die Istposition auf **Position** gesetzt.

Done: Hier wird die erfolgreiche Abarbeitung des Befehls signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

Verhalten des Bausteins

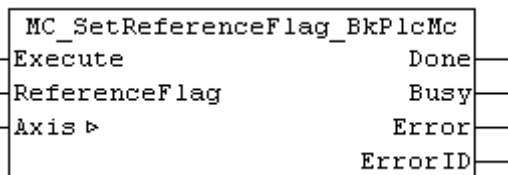
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- In Abhängigkeit vom in **Axis.pStAxParams^.nEnc_Type** festgelegten Encodertyp wird entweder ST_TcHydAxRtData.fEnc_RefShift oder ST_TcHydAxParam.fEnc_ZeroShift so aktualisiert, dass die Istposition der Achse den geforderten Wert annimmt. Ist der Encodertyp nicht bekannt oder der Encoder lässt ein Setzen des Istwerts nicht zu wird mit **Error** und **ErrorID:=dwTcHydErrCdEncType** reagiert.
- Wird ST_TcHydAxParam.fEnc_ZeroShift dabei erkennbar verändert wird Axis_Ref_BkPlcMc [▶ 62].ST_TcHydAxRtData [▶ 94].bParamsUnsave gesetzt.

HINWEIS! Durch diesen Baustein können die Istposition und/oder die Zielposition einer aktuell abgearbeiteten Bewegung hinter einen aktiven Software-Endschalter verschoben werden. Dies wird nicht durch den Baustein überwacht.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten werden alle ebenfalls betroffenen Elemente in ST_TcHydAxRtData automatisch aktualisiert. Dadurch ist dieser Baustein auch bei Achsen aktivierbar, die eine aktive Bewegung ausführen. Die erfolgreiche Ausführung der Funktion wird mir **Done** signalisiert. Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale.

3.1.14 MC_SetReferenceFlag_BkPlcMc (ab V3.0)



(Funktion ist nicht durch PLCopen definiert) Der Funktionsbaustein definiert das Referenzier-Flag einer Achse.

```
VAR_INPUT
  Execute:      BOOL;
  ReferenceFlag:  BOOL;
END_VAR
VAR_OUTPUT
  Done:        BOOL;
  Busy:        BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
END_VAR
```

```

VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang setzt das Referenzier-Flag der Achse.

ReferenceFlag: Der neue Zustand des Referenzier-Flags der Achse.

Done: Hier wird die erfolgreiche Abarbeitung des Befehls signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

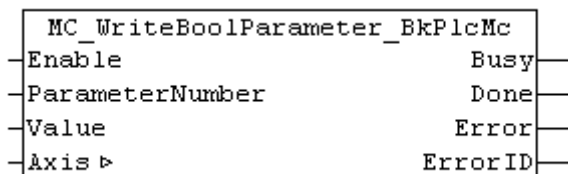
ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin wird das Referenzier-Flag in `ST_TcHydAxRtData.nStateDWord` [► 223] aktualisiert. Dazu wird abhängig von **ReferenceFlag** das entsprechende Bit mit `dwTcHydNsDwReferenced` gelöscht oder gesetzt. Die erfolgreiche Ausführung der Funktion wird mir **Done** signalisiert. Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale.

3.1.15 MC_WriteBoolParameter_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt die booleschen Parameter einer Achse aus. Für nicht boolesche Parameter steht der Baustein `MC_WriteParameter_BkPlcMc` [► 36] zur Verfügung.

```

VAR_INPUT
  Enable:      BOOL;
  ParameterNumber: INT;
  Value:      BOOL;
END_VAR
VAR_OUTPUT
  Busy:      BOOL;
  Done:      BOOL;
  Error:      BOOL;
  ErrorID:   UDINT;
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Eine steigende Flanke an diesem Eingang löst einen Schreibvorgang aus.

ParameterNumber: Diese Kennnummer legt den auszulesenden Parameter fest. Es sollten nur benannte Konstanten aus `E_TcMCPParameter` [► 74] verwendet werden.

Value: Hier ist der Wert des Parameters bereit zu stellen.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung des Schreibvorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

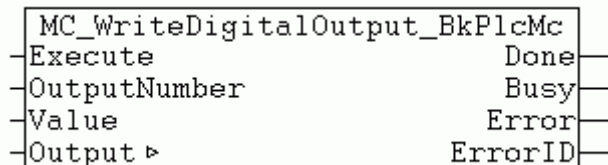
- Wenn an **ParameterNumber** ein nicht unterstützter Wert angelegt wurde wird mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.

Wenn diese Überprüfungen ohne Probleme durchgeführt werden konnten wird **Value** in den gewünschten Parameterwert eingetragen und **Done** wird gemeldet. Wird der Parameter dabei verändert wird `Axis_Ref_BkPlcMc` [► 62].`ST_TcHydAxRtData` [► 94].`bParamsUnsave` gesetzt.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang **Busy** wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.1.16 MC_WriteDigitalOutput_BkPlcMc (ab V3.0)



Der Funktionsbaustein definiert den Zustand eines digitalen Ausgangs eines Nockenschaltwerks.

```

VAR_INPUT
    Execute:      BOOL;
    OutputNumber: INT;
    Value:        BOOL;
END_VAR
VAR_OUTPUT
    Done:        BOOL;
    Busy:        BOOL;
    Error:       BOOL;
    ErrorID:     UDINT;
END_VAR
VAR_INOUT
    Output:      OUTPUT_REF_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang löst eine Aktualisierung des Zustands aus.

OutputNumber: Die Nummer des zu ermittelnden Ausgangs.

Value: Der Zustand des digitalen Ausgangs.

Done: Hier wird die erfolgreiche Ermittlung des Zustands signalisiert.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Output: Hier ist die Adresse einer Variablen vom Typ `OUTPUT_REF_BkPlcMc` [► 87] zu übergeben.

Verhalten des Bausteins

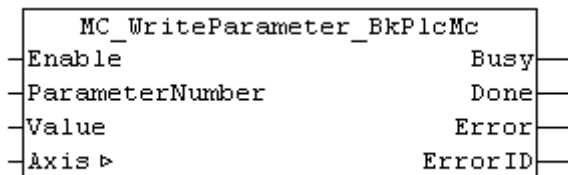
Bei einer steigenden Flanke an **Execute** untersucht der Baustein die übergebenen Parameter. Dabei kann ein Problem erkannt und gemeldet werden:

- Wenn der Wert von **OutputNumber** nicht im zulässigen Bereich [0..31] liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdIllegalOutputNumber** reagiert.

Wenn diese Überprüfungen ohne Probleme durchgeführt werden konnten wird der Zustand des digitalen Ausgangs entsprechend dem Wert von **Value** definiert und **Done** gemeldet.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale.

3.1.17 MC_WriteParameter_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt die nicht boolschen Parameter einer Achse aus. Für boolsche Parameter steht der Baustein MC_WriteBoolParameter_BkPlcMc [► 34] zur Verfügung.

```

VAR_INPUT
  Enable:          BOOL;
  ParameterNumber: INT;
  Value:          LREAL;
END_VAR
VAR_OUTPUT
  Busy:          BOOL;
  Done:          BOOL;
  Error:         BOOL;
  ErrorID:       UDINT;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Eine steigende Flanke an diesem Eingang löst einen Schreibvorgang aus.

ParameterNumber: Diese Kennnummer legt den auszulesenden Parameter fest. Es sollten nur benannte Konstanten aus E_TcMCPParameter [► 74] verwendet werden.

Value: Hier ist der Wert des Parameters bereit zu stellen.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung des Schreibvorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Enable** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn an **ParameterNumber** ein nicht unterstützter Wert angelegt wurde wird mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.

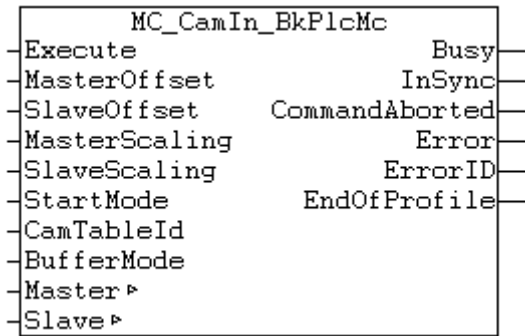
Wenn diese Überprüfungen ohne Probleme durchgeführt werden konnten wird **Value** in den gewünschten Parameterwert eingetragen und **Done** wird gemeldet. Wird der Parameter dabei erkennbar verändert wird Axis_Ref_BkPlcMc [► 62].ST_TcHydAxRtData [► 94].bParamsUnsave gesetzt.

Eine fallende Flanke an **Enable** löscht alle anstehenden Ausgangssignale.

HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang Busy wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.2 Motion

3.2.1 MC_CamIn_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht eine Kurvenscheiben-Kopplung zwischen zwei Achsen. Zum Lösen der Kopplung ist ein [MC_CamOut_BkPlcMc](#) [► 39] Baustein zu verwenden.

```

VAR_INPUT
  Execute:          BOOL;
  MasterOffset:     LREAL:=0.0;
  SlaveOffset:      LREAL:=0.0;
  MasterScaling:    LREAL:=0.0;
  SlaveScaling:     LREAL:=0.0;
  StartMode:        MC_StartMode_BkPlcMc:=MC_StartMode_Absolute;
  CamTableId:       MC_CAM_ID_BkPlcMc;
  BufferMode:        MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;    (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:             BOOL;
  InSync:           BOOL;
  CommandAborted:  BOOL;
  Error:            BOOL;
  ErrorID:          UDINT;
  EndOfProfile:    BOOL;
END_VAR
VAR_INOUT
  Master:           Axis_Ref_BkPlcMc;
  Slave:           Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet die Kopplung.

MasterOffset, MasterScaling: Diese Werte werden mit der Istposition des Masters verrechnet, bevor der sich ergebende Wert in der Master-Spalte der Tabelle gesucht wird.

SlaveOffset, SlaveScaling: Diese Werte werden mit der Slave-Position aus der Tabelle verrechnet.

StartMode: Ein Wert aus [MC_StartMode_BkPlcMc](#) [► 86], der das Verhalten der Slave-Achse beim Aktivieren der Kopplung festlegt.

CamTableId: Hier ist eine Variable des Typs [MC_CAM_ID_BkPlcMc](#) [► 85] zu übergeben, die von einem Baustein des Typs [MC_CamTableSelect_BkPlcMc](#) [► 40] initialisiert wurde.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten [Aborting_BkPlcMc](#) belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

InSync: Hier wird das erstmalige erfolgreiche Aufsynchronisieren der Achsen signalisiert. Das Signal bleibt anschließend auch dann anstehen, wenn die Synchronisierung zu einem späteren Zeitpunkt zeitweise oder bleibend aussetzt.

CommandAborted: Hier wird ein Abbruch der Kopplung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

EndOfProfile: Hier wird signalisiert, ob der Master das Ende des definierten Bereichs erreicht hat.

Master, Slave: Hier ist jeweils die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc](#) [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **CamTableId.bValidated** nicht durch einen Baustein vom Typ `MC_CamTableSelect_BkPlcMc` gesetzt wurde wird mit **Error** und **ErrorID:=dwTcHydErrCdTblNoInit** reagiert.
- Wenn sich entweder Master oder Slave nicht im Ruhezustand befinden wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** reagiert.
- Wenn als **StartMode** der Wert `MC_StartMode_Rampln` vorgegeben wird reagiert der Baustein mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Kopplung initiiert. In Abhängigkeit von **StartMode** wird die Bezugsposition für **Slave** entweder auf den Wert 0.0 oder mit der aktuellen Istposition von **Slave** festgelegt. Die Achse befindet sich ab jetzt im Zustand [McState_Synchronizedmotion](#) [► 73] und der Baustein beginnt mit der Berechnung und Überwachung der Kopplung.

Sollposition und Sollgeschwindigkeit von **Slave** werden in Abhängigkeit von Istposition und Sollgeschwindigkeit des Masters und der Tabelle berechnet.

Wenn bei aktiver Kopplung die Slaveachse erstmalig die durch die Kopplung geforderte Geschwindigkeit erreicht wird dies am Ausgang `InGear` signalisiert. Da die Kopplung derzeit nur im Stillstand aktiviert werden kann ist dies unmittelbar der Fall. Sollte während bei aktiver Kopplung die Slaveachse aus einem beliebigen Grund den Vorgaben nicht folgen können bleibt `InGear` unverändert.

Tritt bei aktiver Kopplung im Bewegungsgenerator ein Fehlercode auf wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

Eine fallende Flanke an **Execute** beendet weder die Berechnung noch die Überwachung der Kopplung. Dies kann nur dadurch erreicht werden, dass die Kopplung durch einen `MC_CamOut_BkPlcMc` Baustein oder das Auftreten eines Fehlers bewirkt werden. Erst dann werden alle anstehenden Ausgangssignale gelöscht.

ⓘ HINWEIS! Dieser Baustein übernimmt zeitweise die Aufgabe der Sollwertgenerierung. Um dies zu signalisieren wird Busy nicht nur bis zum Übergang in die Synchronität TRUE, sondern bleibt bis zum Auflösen der Kopplung anstehen.

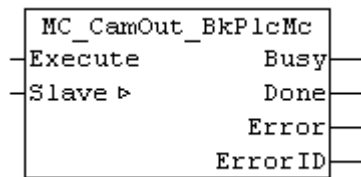


Hinweis

Bausteinaufruf

Es ist zwingend erforderlich, diesen Baustein zyklisch aufzurufen wenn **Busy** auf TRUE steht. Anschließend ist der Baustein noch mindestens einmal mit **Execute:=FALSE** aufzurufen.

3.2.2 MC_CamOut_BkPlcMc (ab V3.0)



Der Funktionsbaustein löst eine durch einen [MC_CamIn_BkPlcMc \[▶ 37\]](#) Baustein gestartete Kurvenscheiben-Kopplung zwischen zwei Achsen.

```

VAR_INPUT
    Execute:      BOOL;
ND_VAR
VAR_OUTPUT
    Busy:         BOOL;
    Done:         BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
END_VAR
VAR_INOUT
    Slave:        Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet die Kopplung.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung des Kommandos signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Slave: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[▶ 62\]](#) zu übergeben.

Verhalten des Bausteins

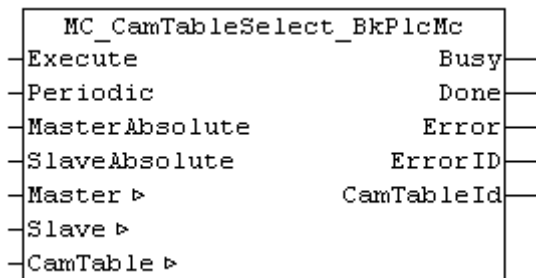
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn der Pointer pStAxParams in [Axis_Ref_BkPlcMc \[▶ 62\]](#) nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn der Pointer pStAxRtData in [Axis_Ref_BkPlcMc \[▶ 62\]](#) nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrMcPlc** reagiert.
- Wenn sich die Achse nicht gekoppelt ist reagiert der Baustein ohne weitere Prüfungen oder Aktivitäten mit **Done**.
- Ist die aktuelle Sollgeschwindigkeit der Achse kleiner als die von pStAxParams.fCreepSpeed festgelegte Geschwindigkeit geht die Achse unmittelbar in McState_Standstill über und baut die Restgeschwindigkeit ab. Es wird **Done** signalisiert und alle weiteren Überprüfungen oder Aktivitäten werden unterlassen.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten und nicht bereits aus einem der genannten Gründe **Done** signalisiert wird erfolgt eine Umwandlung der von der Kurvenscheibenkopplung kontrollierten Bewegung in eine vom Master unabhängige endlose Bewegung mit gleicher Geschwindigkeit und Richtung. Wenn diese Umwandlung erfolgreich ausgeführt wurde wird **Done** signalisiert, andernfalls wird mit **Error** und **ErrorID:=Fehlercode** reagiert.

ⓘ HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang Busy wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.2.3 MC_CamTableSelect_BkPlcMc (ab V3.0)



Der Funktionsbaustein initialisiert eine Variable vom Typ MC_CAM_ID_BkPlcMc [▶ 85] und bereitet dadurch eine Kurvenscheibe für die Kopplung von zwei Achsen vor.

```

VAR_INPUT
  Execute:          BOOL;
  Periodic:         BOOL;
  MasterAbsolute:  BOOL;
  SlaveAbsolute:   BOOL;
ND_VAR
VAR_OUTPUT
  Busy:             BOOL;
  Done:            BOOL;
  Error:           BOOL;
  ErrorID:         UDINT;
  CamTableId:     MC_CAM_ID_BkPlcMc;
END_VAR
VAR_INOUT
  Master:          Axis_Ref_BkPlcMc;
  Slave:          Axis_Ref_BkPlcMc;
  CamTable:       MC_CAM_REF_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet das Kommando.

Periodic: Nicht unterstützt: Hier ist derzeit FALSE zu übergeben.

MasterAbsolute: Nicht unterstützt: Hier ist derzeit TRUE zu übergeben.

SlaveAbsolute: Nicht unterstützt: Hier ist derzeit TRUE zu übergeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Initialisierung von CamTableId signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

CamTableId: Hier wird eine Variable des Typs MC_CAM_ID_BkPlcMc [▶ 85] zurückgegeben, die an einen Baustein des Typs MC_CamIn_BkPlcMc [▶ 37] weitergegeben werden kann.

Master, Slave: Hier ist jeweils die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

CamTable: Hier ist eine Variable des Typs MC_CAM_REF_BkPlcMc [▶ 85] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **CamTable.pTable** nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn **CamTable.nLastIdx** nicht grösser als **CamTable.nFirstIdx** ist wird mit **Error** und **ErrorID:=dwTcHydErrCdTblEntryCount** reagiert.

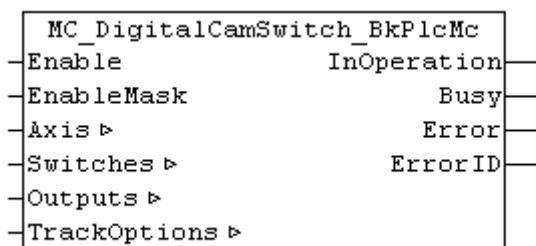
- Wenn **CamTable.nFirstIdx** und **CamTable.nLastIdx** eine Tabelle mit mehr als 100 Zeilen definieren wird mit **Error** und **ErrorID:=dwTcHydErrCdTblLineCount** reagiert.
- Wenn **MasterAbsolute** oder **SlaveAbsolute** nicht gesetzt oder **Periodic** gesetzt ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird **CamTableId** initialisiert. Dazu werden die Daten aus **CamTable** und die Eingangsdaten des Bausteins übernommen. **CamTableId** wird als gültig und verändert markiert. Mit **Done** wird die Abarbeitung des Kommandos gemeldet.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale.

HINWEIS! Dieser Baustein benötigt keine Zeit zur Durchführung seiner Aufgaben. Der Ausgang **Busy** wird zu keinem Zeitpunkt den Wert TRUE annehmen und ist nur aus Kompatibilitätsgründen vorhanden.

3.2.4 MC_DigitalCamSwitch_BkPlcMc (ab V3.0)



Der Funktionsbaustein erzeugt in Abhängigkeit von Position, Bewegungsrichtung und Geschwindigkeit einer Achse Software-Nocken.

```

VAR_INPUT
    Enable:          BOOL;
    EnableMask:     DWORD;
END_VAR
VAR_OUTPUT
    InOperation:    BOOL;
    Busy:           BOOL;
    Error:         BOOL;
    ErrorID:       UDINT;
END_VAR
VAR_INOUT
    Axis:          Axis_Ref_BkPlcMc;
    Switches:     CAMSWITCH_REF_BkPlcMc;
    Outputs:     OUTPUT_REF_BkPlcMc;
    TrackOptions: TRACK_REF_BkPlcMc;
END_VAR
    
```

Enable: Dieser Eingang kontrolliert alle Aktivitäten des Bausteins.

EnableMask: Eine Maske, deren Bits die Freigabe der Ausgänge in **Outputs** festlegen.

InOperation: Hier wird signalisiert, ob der Baustein aktiv ist.

Busy: Dieser Ausgang ist für die Dauer der Abarbeitung des Kommandos auf TRUE.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

Switches: Hier ist ein Array vom Typ CAMSWITCH_REF_BkPlcMc [▶ 63] zu übergeben.

Outputs: Hier ist die Adresse einer Variablen vom Typ OUTPUT_REF_BkPlcMc [▶ 87] zu übergeben.

TrackOptions: Hier ist ein Array vom Typ TRACK_REF_BkPlcMc [▶ 103] zu übergeben.

Verhalten des Bausteins

Gesteuert durch die Istposition einer Achse werden Nockensignale (Switches) geschaltet. Es stehen positionsgesteuerte (mit Anfangs- und Endposition) und zeitgesteuerte (mit Triggerposition und Dauer) zur Verfügung. Dabei kann die Bewegungsrichtung der Achse berücksichtigt werden.

Die Nockensignale werden Spuren (Tracks) mit parametrierbaren Eigenschaften zugeordnet. Durch Ein- und Ausschaltverzögerung kann das Zeitverhalten festgelegt werden. Durch negative Werte kann hier auch eine vorausschauende Signalgabe erreicht werden. Eine Hysterese ermöglicht die Unterdrückung von unerwünschten Signalgaben, wenn die Achse in der Nähe eines Schaltpunkts steht und die Istposition nicht völlig konstant ist.

Beispiel

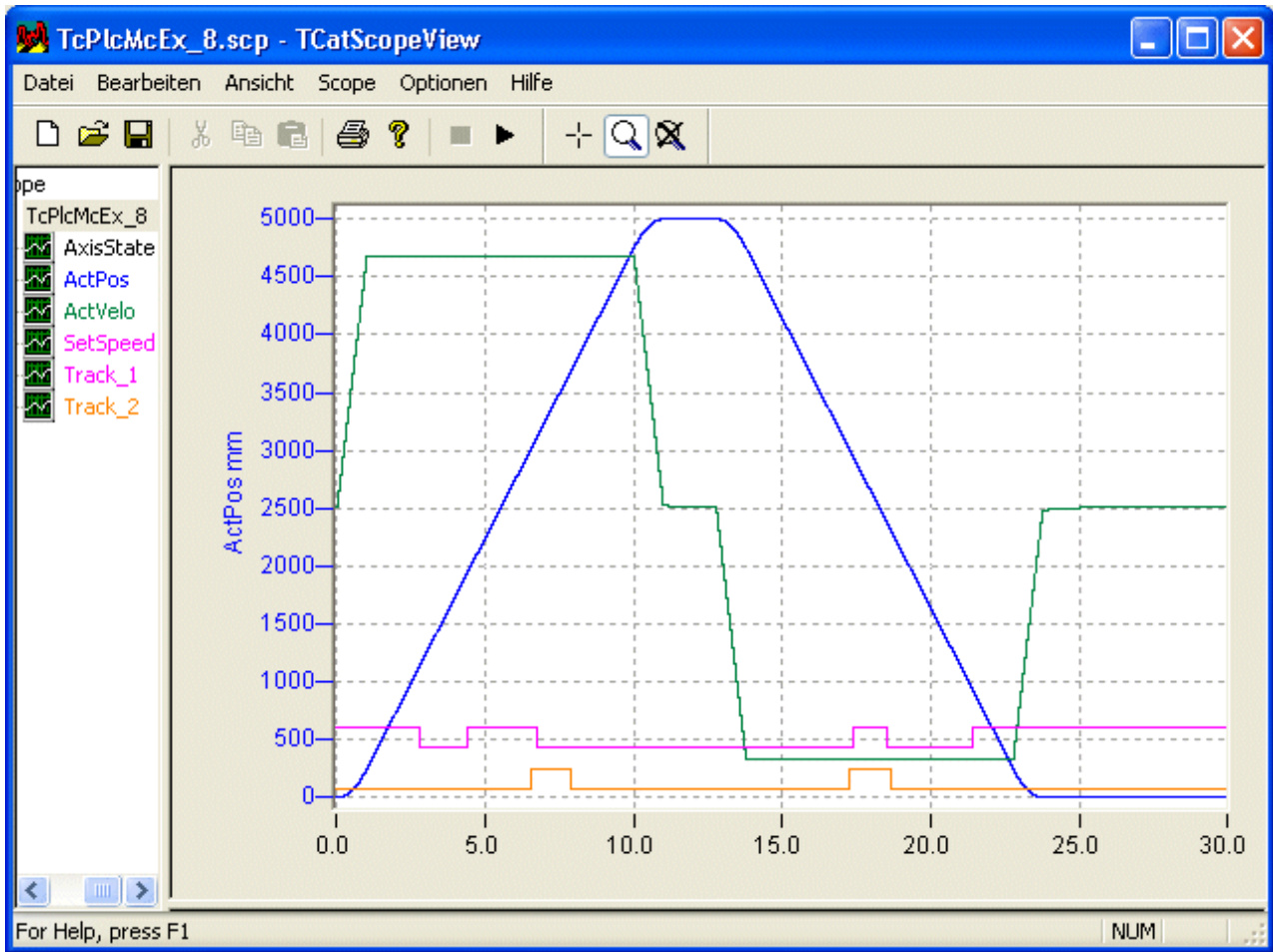
Verwendete CAMSWITCH_REF BkPlcMc [► 63]:

Parameter	Switch[1]	Switch[2]	Switch[3]	Switch[4]	...	Switch[n]
TrackNumber	1	1	1	2		
FirstOnPosition	2000.0	2500.0	-1000.0	3000.0		
LastOnPosition	3000.0	3000.0	1000.0			
AxisDirection	1	2	0	0		
CamSwitchMode	0	0	0	1		
Duration				1.35		
.....						

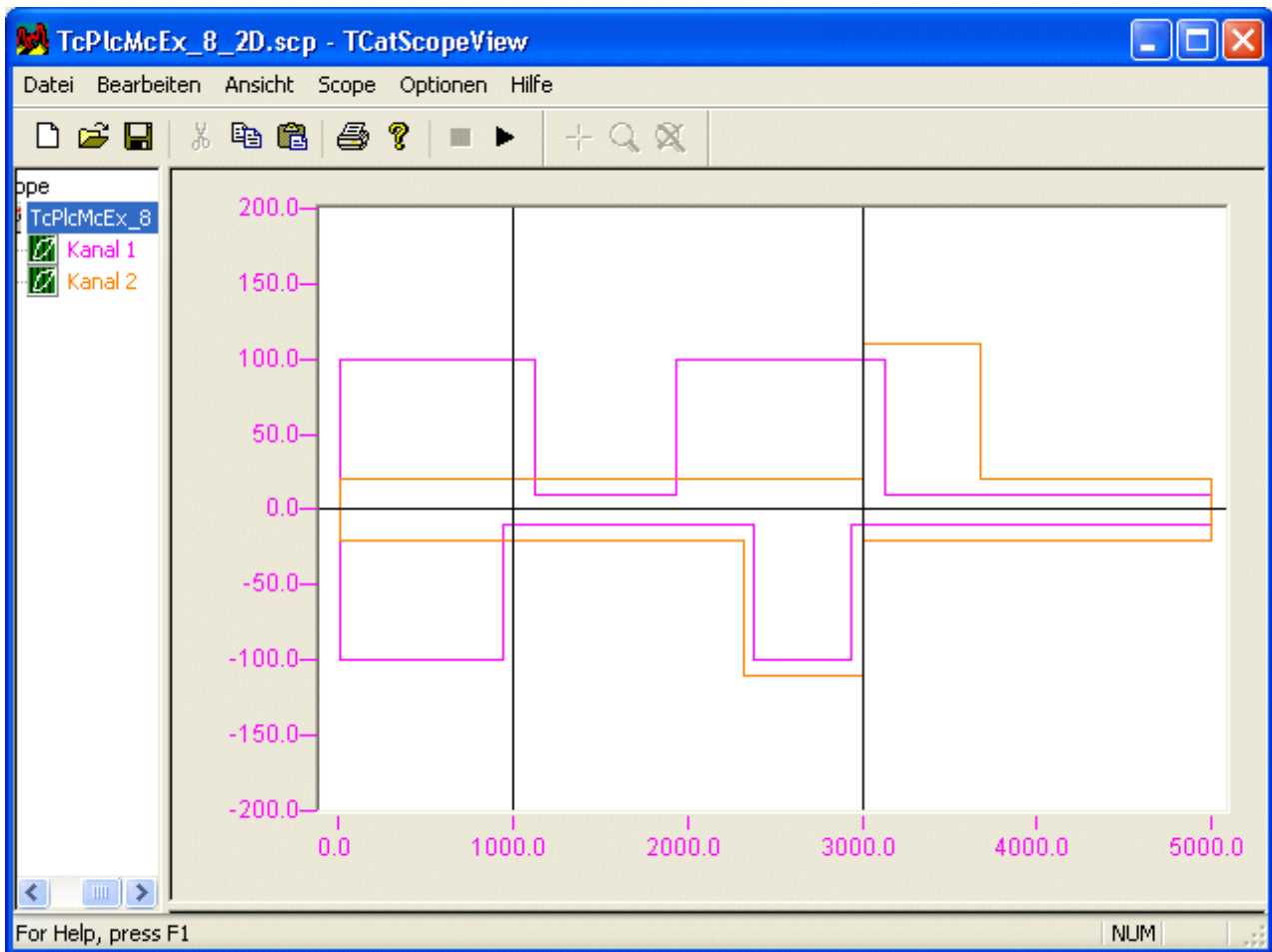
Verwendete TRACK_REF BkPlcMc [► 103]:

Parameter	Track[1]	Track[2]	...	Track[n]
OnCompensation	-0.125	0.0		
OffCompensation	0.250	0.0		
Hysteresis	0.0	0.0		

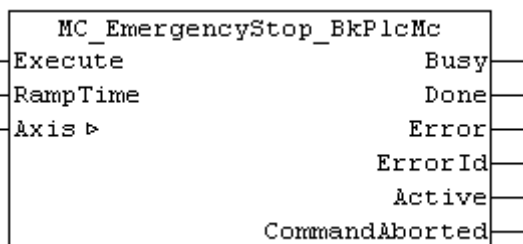
Verlauf der Signale bei einer Fahrbewegung der Achse von 0.0 nach 5000.0 und zurück:



Die nachfolgende Darstellung zeigt den Verlauf der Signale über die Position. Dabei werden die Signale bei positiver Bewegungsrichtung normal (nach oben), bei negativer Bewegungsrichtung jedoch negativ = 'nach unten' dargestellt. Die senkrechten Cursor-Linien markieren die Positionen 1000 bzw. 3000 mm.



3.2.5 MC_EmergencyStop_BkPlcMc (ab V3.0.5)



Der Funktionsbaustein bricht eine aktuell ausgeführte Bewegung einer Achse ab und überwacht die Notstopoperation.

```

VAR_INPUT
    Execute:      BOOL;
    RampTime:    LREAL; (ab/from V3.0.5)
END_VAR
VAR_OUTPUT
    Busy:        BOOL;
    Done:        BOOL;
    Error:       BOOL;
    ErrorID:     UDINT;
    Active:      BOOL;
    CommandAborted: BOOL;
END_VAR
VAR_INOUT
    Axis:        Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang beendet eine Bewegung der Achse.

RampTime: Die geforderte Anhaltezeit.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Operation signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Active: Hier wird angezeigt, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird angezeigt, dass die Abarbeitung dieses Kommandos durch ein anderes Kommando abgebrochen wurde.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

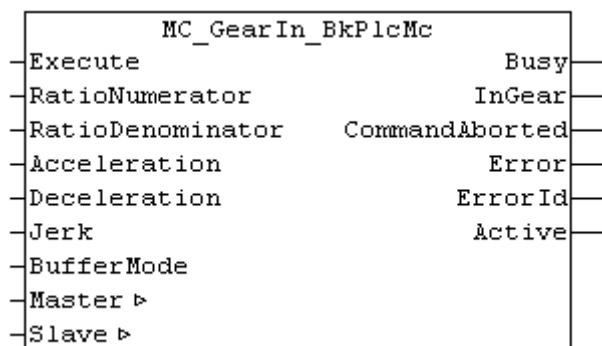
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Ein Stopp kann nur ausgeführt werden, wenn die Achse eine aktive Bewegung ausführt. Befindet sie sich im Stillstand meldet der Baustein sofort **Done**.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Befindet sich die Achse in einem Zustand, in dem sie durch eine Kopplung mit einer anderen Achse oder einen vergleichbaren Mechanismus kontrolliert wird reagiert sie mit **Error** und **ErrorID:=dwTcHydErrCdNotReady**.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Stop-Operation initiiert. Dabei wird **RampTime** verwendet, um unter Berücksichtigung der Bezugsgeschwindigkeit eine Verzögerung zu berechnen. Ist ein ruckbegrenzender Stellwert-Generator ausgewählt wird **MaxJerk** verwendet. Ist für **RampTime** kein Wert vorgegeben, der erkennbar größer als 0 ist wird der Achsparameter fEmergencyRamp verwendet.

Zum Abbremsen der Achse wird intern ein MC_Stop_BkPlcMc [► 60] Baustein verwendet. Nachdem die Stellwertausgabe auf 0 abgebaut ist werden alle Ausgaben von Steuer- oder Regelspannungen unterdrückt, solange **Execute** auf TRUE gesetzt ist.

3.2.6 MC_GearIn_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht eine Kopplung zwischen zwei Achsen. Zum Lösen der Kopplung ist ein MC_GearOut_BkPlcMc [► 49] Baustein zu verwenden.

```

VAR_INPUT
  Execute:          BOOL;
  RatioNumerator:   INT;
  RatioDenominator: INT;
  Acceleration:     LREAL;
  Deceleration:     LREAL;

```

```

    Jerk:                LREAL;    (ab/from V3.0.5)
    BufferMode:          MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;    (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
    Busy:               BOOL;
    InGear:             BOOL;
    CommandAborted:    BOOL;
    Error:              BOOL;
    ErrorID:           UDINT;
    Active:            BOOL;
END_VAR
VAR_INOUT
    Master:             Axis_Ref_BkPlcMc;
    Slave:            Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet die Kopplung.

RatioNumerator, RatioDenominator: Diese Parameter beschreiben den Kopplungsfaktor in der Form eines Getriebes.

Acceleration: Die für das Aufsynchronisieren zulässige Beschleunigung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Deceleration: Die für das Aufsynchronisieren zulässige Verzögerung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Jerk: Der anzuwendende Ruck.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten Aborting_BkPlcMc belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

InGear: Hier wird das erstmalige erfolgreiche Aufsynchronisieren der Achsen signalisiert. Das Signal bleibt anschließend auch dann anstehen, wenn die Synchronisierung zu einem späteren Zeitpunkt zeitweise oder bleibend aussetzt.

CommandAborted: Hier wird ein Abbruch der Kopplung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Active: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Master, Slave: Hier ist jeweils die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc](#) [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Als nächstes wird überprüft, ob **RatioDenominator** gleich 0 ist. In diesem Fall wird mit **Error** und **ErrorID:=dwTcHydErrCdIllegalGearFactor** reagiert.
- Derzeit kann die Kopplung nur dann aktiviert werden, wenn sowohl der Master als auch der Slave im Stillstand sind. Andernfalls wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** reagiert.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn der Bewegungsalgorithmus bereits einen Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Kopplung initiiert. Die Achse befindet sich ab jetzt im Zustand [McState_Synchronizedmotion](#) [► 73] und der Baustein beginnt mit der Überwachung der Kopplung.

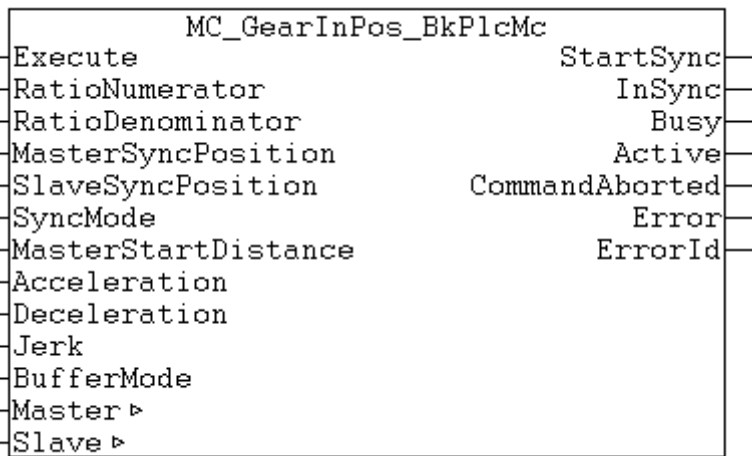
Wenn bei aktiver Kopplung die Slaveachse erstmalig die durch die Kopplung geforderte Geschwindigkeit erreicht wird dies am Ausgang InGear signalisiert. Da die Kopplung derzeit nur im Stillstand aktiviert werden kann ist dies unmittelbar der Fall. Sollte während bei aktiver Kopplung die Slaveachse aus einem beliebigen Grund den Vorgaben nicht folgen können bleibt InGear unverändert.

Tritt bei aktiver Kopplung im Bewegungsgenerator ein Fehlercode auf wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Kopplung auf FALSE gesetzt bleibt die bestehende Kopplung unbeeinflusst wirksam.

ⓘ HINWEIS! Der Ausgang Active ist derzeit mit dem Ausgang Busy identisch.

3.2.7 MC_GearInPos_BkPlcMc (ab V3.0.33)



Der Funktionsbaustein startet und überwacht eine fliegende Kopplung zwischen zwei Achsen. Zum Lösen der Kopplung ist ein MC_GearOut_BkPlcMc [► 49] Baustein zu verwenden.

```

VAR_INPUT
  Execute:          BOOL;
  RatioNumerator:   INT;
  RatioDenominator: INT;
  MasterSyncPosition: LREAL;
  SlaveSyncPosition: LREAL;
  SyncMode:         INT;
  MasterStartDistance: LREAL;
  Acceleration:     LREAL;
  Deceleration:     LREAL;
  Jerk:             LREAL;    (ab/from V3.0.5)
  BufferMode:       MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;
END_VAR
VAR_OUTPUT
  StartSync:        BOOL;
  InSync:           BOOL;
  Busy:             BOOL;
  Active:           BOOL;
  CommandAborted:  BOOL;
  Error:            BOOL;
  ErrorID:          UDINT;
END_VAR
VAR_INOUT
  Master:          Axis_Ref_BkPlcMc;
  Slave:          Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet die Kopplung.

RatioNumerator, RatioDenominator: Diese Parameter beschreiben den Kopplungsfaktor in der Form eines Getriebes.

MasterSyncPosition: Ab dieser Master-Position ist die Kopplung vollständig wirksam.

SlaveSyncPosition: Ab dieser Slave-Position ist die Kopplung vollständig wirksam.

SyncMode: derzeit nicht unterstützt.

MasterStartDistance: Über diesen Weg des Masters wird die Kopplung aufgebaut.

Acceleration: Die für das Aufsynchronisieren zulässige Beschleunigung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Deceleration: Die für das Aufsynchronisieren zulässige Verzögerung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Jerk: Der anzuwendende Ruck.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten `Aborting_BkPlcMc` belegt werden. (ab V3.0.8)

StartSync: Hier wird die Übergangsphase zwischen Ruhezustand und vollständig wirksamer Kopplung signalisiert.

InSync: Hier wird das erstmalige erfolgreiche Aufsynchronisieren der Achsen signalisiert. Das Signal bleibt anschließend auch dann anstehen, wenn die Synchronisierung zu einem späteren Zeitpunkt zeitweise oder bleibend aussetzt.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Active: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird ein Abbruch der Kopplung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Master, Slave: Hier ist jeweils die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Als nächstes wird überprüft, ob **RatioDenominator** gleich 0 ist. In diesem Fall wird mit **Error** und **ErrorID:=dwTcHydErrCdIllegalGearFactor** reagiert.
- Ist der **RatioDenominator** kleiner als 0 wird mit **Error** und **ErrorID:=dwTcHydErrCdNotSupport** reagiert.
- Die Kopplung kann nur dann aktiviert werden, wenn der Slave im Stillstand ist. Andernfalls wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** reagiert.
- Wenn der Absolutwert des **MasterStartDistance** zu klein ist wird mit **Error** und **ErrorID:=dwTcHydErrCdCannotSynchronize** reagiert.
- Wenn die Istposition des Masters nicht zwischen **MasterSyncPosition** und dem von **MasterStartDistance** festgelegten Ende der Synchronisationsstrecke liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdCannotSynchronize** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Kopplung initiiert. Die Slave-Achse befindet sich zunächst weiter im Zustand `McState_Standstill` [► 73]. Erst wenn die Master-Achse erstmalig den Anfang der Synchronisationsstrecke erreicht meldet die Slave-Achse `McState_Synchronizedmotion` [► 73], signalisiert **StartSync** und der Baustein beginnt mit der Überwachung der Kopplung. Sobald die Achse erstmalig das Ende der Synchronisationsstrecke erreicht signalisiert die Slave-Achse **InSync**. Sollte die Master-Achse zu einem späteren Zeitpunkt den Anfang der Synchronisationsstrecke rückwärts passieren, wird die Kopplung nicht wieder gelöst.

Tritt bei aktiver Kopplung im Bewegungsgenerator ein Fehlercode auf wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

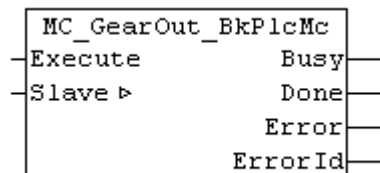
Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Kopplung auf FALSE gesetzt bleibt die bestehende Kopplung unbeeinflusst wirksam.

Es steht unter [#103 \[▶ 234\]](#) ein Beispiel zur Verfügung.

VORSICHT! Der Baustein unterstützt nicht den Funktionsumfang der TwinCAT NC.

HINWEIS! Der Ausgang Active ist derzeit mit dem Ausgang Busy identisch.

3.2.8 MC_GearOut_BkPlcMc (ab V3.0)



Der Funktionsbaustein löst eine Kopplung zwischen zwei Achsen. Diese Kopplung muss zuvor mit einem [MC_GearIn_BkPlcMc \[▶ 45\]](#) oder einem [MC_GearInPos_BkPlcMc \[▶ 47\]](#) Baustein hergestellt worden sein.

```

VAR_INPUT
    Execute:      BOOL;
END_VAR
VAR_OUTPUT
    Busy:         BOOL;
    Done:         BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
END_VAR
VAR_INOUT
    Slave:        Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang löst die Kopplung.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Bewegung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Slave: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[▶ 62\]](#) zu übergeben.

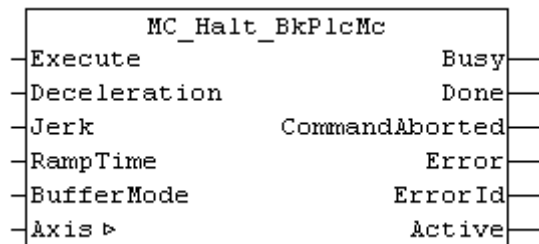
Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Achse nicht in einer Getriebekopplung betrieben wird signalisiert der Baustein unmittelbar **Done** und unterlässt alle weiteren Überprüfungen oder Aktivitäten.
- Ist die aktuelle Sollgeschwindigkeit der Achse kleiner als die von `pStAxParams.fCreepSpeed` festgelegte Geschwindigkeit geht die Achse unmittelbar in `McState_Standstill` über und baut die Restgeschwindigkeit ab. Es wird **Done** signalisiert und alle weiteren Überprüfungen oder Aktivitäten werden unterlassen.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten und nicht bereits aus einem der genannten Gründe **Done** signalisiert wird erfolgt eine Umwandlung der von der Getriebekopplung kontrollierten Bewegung in eine vom Master unabhängige endlose Bewegung mit gleicher Geschwindigkeit und Richtung. Wenn diese Umwandlung erfolgreich ausgeführt wurde wird **Done** signalisiert, andernfalls wird mit **Error** und **ErrorID:=Fehlercode** reagiert.

3.2.9 MC_Halt_BkPlcMc (ab V3.0)



Der Funktionsbaustein bricht eine aktuell ausgeführte Bewegung einer Achse ab und überwacht die Stoppoperation.

HINWEIS! Die von diesem Baustein eingeleitete Stoppoperation ist durch andere Bausteine unterbrechbar. Soll eine Achse während der Stoppoperation nicht durchstartbar sein ist ein **MC_Stop_BkPlcMc** Baustein zu verwenden.

```

VAR_INPUT
  Execute:          BOOL;
  Deceleration:    LREAL;  (ab/from V3.0.5)
  Jerk:            LREAL;  (ab/from V3.0.5)
  RampTime:        LREAL;  (ab/from V3.0.5)
  BufferMode:       MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;  (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:            BOOL;
  Done:            BOOL;
  Error:           BOOL;
  ErrorID:         UDINT;
  Active:          BOOL;
  CommandAborted: BOOL;
END_VAR
VAR_INOUT
  Axis:            Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang beendet eine Bewegung der Achse.

Deceleration: Die anzuwendende Verzögerung.

Jerk: Der anzuwendende Ruck.

RampTime: Die geforderte Anhaltezeit.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten `Aborting_BkPlcMc` belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Operation signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Active: Hier wird angezeigt, dass ein Kommando abgearbeitet wird.

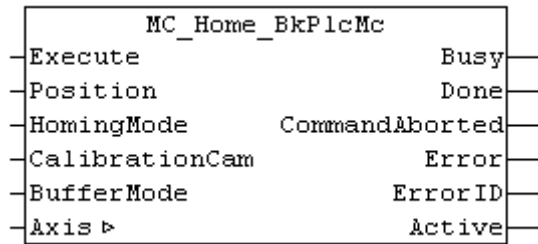
CommandAborted: Hier wird angezeigt, dass die Abarbeitung dieses Kommandos durch ein anderes Kommando abgebrochen wurde.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Das Verhalten des Bausteins ist identisch mit dem des `MC_Stop_BkPlcMc` [► 60]() Bausteins. Der einzige Unterschied ist, dass die Abarbeitung des Kommandos durch andere Bausteine abbrechbar ist.

3.2.10 MC_Home_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht die Referenzfahrt einer Achse.

```

VAR_INPUT
    Execute:          BOOL;
    Position:         LREAL;
    HomingMode:       MC_HomingMode_BkPlcMc;
    CalibrationCam:   BOOL;
    BufferMode:        MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;    (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
    Busy:             BOOL;
    Done:             BOOL;
    CommandAborted:  BOOL;
    Error:            BOOL;
    ErrorID:          UDINT;
END_VAR
VAR_INOUT
    Axis:             Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet die Referenzfahrt.

Position: Die Referenzposition.

HomingMode: Hier wird die zu verwendende Methode [▶ 86] festgelegt.

CalibrationCam: Hier kann der Referenzierindex (Nocken) direkt übergeben werden.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten Aborting_BkPlcMc belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Referenzfahrt signalisiert.

CommandAborted: Hier wird ein Abbruch der Referenzfahrt signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Eine Referenzfahrt kann nur aus einem fehlerfreien Stillstand begonnen werden. Ist dies nicht der Fall wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** bzw. dem vorliegenden Errorcode reagiert.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn eine der in den Achs-Parametern angegebene Geschwindigkeiten zu klein (weniger als 1% der Referenzgeschwindigkeit) ist wird mit **Error** und **ErrorID:=dwTcHydErrCdSetVelo** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Referenzfahrt initiiert. Der genaue Ablauf wird von [HomingMode \[► 86\]](#) festgelegt. Sollte während der Abarbeitung der Referenzfahrt vom Bewegungsalgorithmus ein Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert. Wird die vollständige Referenzfahrt durch die Aktivität eines anderen Bausteins verhindert, wird mit **CommandAborted** reagiert. Ein erfolgreicher Abschluss der Referenzfahrt wird mit **Done** gemeldet.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Referenzfahrt auf FALSE gesetzt wird die eingeleitete Referenzfahrt unbeeinflusst weiter bearbeitet. Die Signale am Ende der Bewegung (**Error**, **ErrorID**, **CommandAborted**, **Done**) werden für einen Zyklus gegeben.

VORSICHT! Wenn die Applikation nicht selbst eine Referenzposition festlegt, sondern ein mit den Maschinendaten abgespeicherter und geladener Wert Verwendung finden soll ist die Benutzung von fEnc_DefaultHomePosition in pStAxParams vorgesehen. Sollten situationsabhängig verschiedene Werte benötigt werden sollte auf fCustomerData[] in pStAxParams zurückgegriffen werden.

Ist als Encoder-Typ iTcMc_EncoderSim eingestellt wird unabhängig von **HomingMode** und [Axis_Ref_BkPlcMc \[► 62\].stAxParams.nEnc_HomingType](#) der Modus **MC_Direct_BkPlcMc** wirksam.

MC_DefaultHomingMode_BkPlcMc

Die Referenziermethode wird nicht durch die Applikation sondern durch [Axis_Ref_BkPlcMc \[► 62\].stAxParams.nEnc_HomingType](#) festgelegt. Dabei gelten die folgenden Regeln:

nEnc_HomingType	MC_HomingMode_BkPlcMc
iTcMc_HomingOnBlock	MC_Block_BkPlcMc
iTcMc_HomingOnIndex	MC_AbsSwitch_BkPlcMc
iTcMc_HomingOnSync	MC_RefPulse_BkPlcMc
iTcMc_HomingOnExec	MC_Direct_BkPlcMc

MC_AbsSwitch_BkPlcMc

Die Achse wird mit [Axis_Ref_BkPlcMc \[► 62\].stAxParams.fEnc_RefIndexVelo](#) in der durch [bEnc_RefIndexPositive](#) festgelegten Richtung bewegt. Nimmt **CalibrationCam** den Zustand TRUE an oder wird in [Axis_Ref_BkPlcMc \[► 62\].stAxRtData.nDeCtrlDWord](#) die Referenznocke (Bit 5, [dwTcHydDcDwRefIndex](#)) erkannt stoppt die Achse. Anschließend wird sie mit [fEnc_RefSyncVelo](#) in der durch [bEnc_RefSyncPositive](#) festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

MC_LimitSwitch_BkPlcMc

Derzeit nicht unterstützt.

MC_RefPulse_BkPlcMc

Die Achse wird mit [Axis_Ref_BkPlcMc \[► 62\].stAxParams.fEnc_RefIndexVelo](#) in der durch [bEnc_RefIndexPositive](#) festgelegten Richtung bewegt. Nimmt **CalibrationCam** den Zustand TRUE an oder wird in [Axis_Ref_BkPlcMc \[► 62\].stAxRtData.nDeCtrlDWord](#) die Referenznocke (Bit 5, [dwTcHydDcDwRefIndex](#)) erkannt stoppt die Achse. Anschließend wird sie mit [fEnc_RefSyncVelo](#) in der durch [bEnc_RefSyncPositive](#) festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Dann wird das Hardware-Latch des Encoders aktiviert und die Achse so lange weiter bewegt, bis das Latch gültig wird. Nach dem Stoppen der Achse wird der Istwert der Achse auf einen Wert gesetzt, der aus der Referenzposition und der seit dem Sync-Puls des Encoders zurückgelegten Strecke errechnet wird.

MC_Direct_BkPlcMc

Der Istwert der Achse wird unmittelbar auf den Wert der Referenzposition gesetzt.

MC_Absolute_BkPlcMc

Derzeit nicht unterstützt.

MC_Block_BkPlcMc

Die Achse wird mit `Axis_Ref_BkPlcMc [► 62].stAxParams.fEnc_RefIndexVelo` in der durch `bEnc_RefIndexPositive` festgelegten Richtung bewegt. Wird für eine Zeitdauer von 2 Sekunden keine Bewegung festgestellt gilt der Festanschlag (Block) als erreicht. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

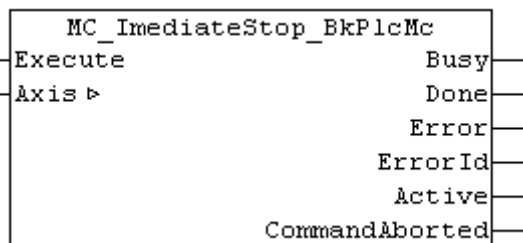
MC_FlyingSwitch_BkPlcMc

Derzeit nicht unterstützt.

MC_FlyingRefPulse_BkPlcMc

Derzeit nicht unterstützt.

3.2.11 MC_ImmediateStop_BkPlcMc (ab V3.0.5)



Der Funktionsbaustein bricht eine aktuell ausgeführte Bewegung einer Achse ab.

```

VAR_INPUT
    Execute:      BOOL;
END_VAR
VAR_OUTPUT
    Busy:         BOOL;
    Done:         BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
    Active:       BOOL;
    CommandAborted: BOOL;
END_VAR
VAR_INOUT
    Axis:         Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang beendet eine Bewegung der Achse.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Operation signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Active: Hier wird angezeigt, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird angezeigt, dass die Abarbeitung dieses Kommandos durch ein anderes Kommando abgebrochen wurde.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc [► 62]` zu übergeben.

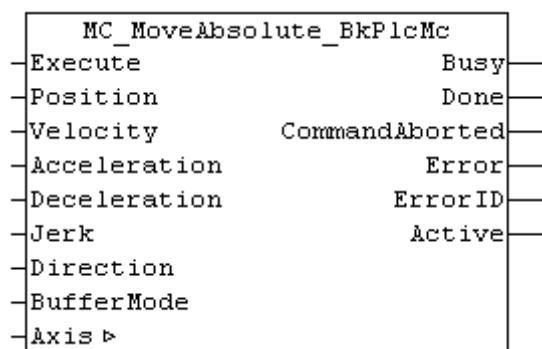
Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Ein Stopp kann nur ausgeführt werden, wenn die Achse eine aktive Bewegung ausführt. Befindet sie sich im Stillstand meldet der Baustein sofort **Done**.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Befindet sich die Achse in einem Zustand, in dem sie durch eine Kopplung mit einer anderen Achse oder einen vergleichbaren Mechanismus kontrolliert wird reagiert sie mit **Error** und **ErrorID:=dwTcHydErrCdNotReady**.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Stop-Operation initiiert. Dabei wird der Stellwert der Achse unmittelbar und ohne jede Rampe auf 0 gesetzt. Anschließend werden alle Ausgaben von Steuer- oder Regelspannungen unterdrückt, solange **Execute** auf TRUE gesetzt ist.

3.2.12 MC_MoveAbsolute_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht die Bewegung einer Achse.

```

VAR_INPUT
  Execute:          BOOL;
  Position:         LREAL;
  Velocity:         LREAL;
  Acceleration:    LREAL;
  Deceleration:    LREAL;
  Jerk:            LREAL;
  Direction:       MC_Direction_BkPlcMc:=MC_Shortest_Way_BkPlcMc;    (ab/from V3.0.8)
  BufferMode:       MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;        (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:            BOOL;
  Done:           BOOL;
  CommandAborted: BOOL;
  Error:          BOOL;
  ErrorID:       UDINT;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet die Bewegung.

Position: Die Zielposition der Bewegung in Istwert-Einheiten der Achse.

Velocity: Die geforderte Geschwindigkeit der Bewegung in Istwert-Einheiten der Achse pro Sekunde.

Acceleration: Die geforderte Beschleunigung in Istwert-Einheiten der Achse pro Quadrat-Sekunde. Ist dieser Parameter 0.0 wird er durch einen Defaultwert aus den Achsparametern ersetzt.

Deceleration: Die geforderte Verzögerung in Istwert-Einheiten der Achse pro Quadrat-Sekunde. Ist dieser Parameter 0.0 wird er durch einen Defaultwert aus den Achsparametern ersetzt.

Jerk: reserviert.

Direction: reserviert. Dieser Eingang wurde nur aus Kompatibilitätsgründen ergänzt und sollte nicht oder mit der Konstanten `MC_Shortest_Way_BkPlcMc` belegt werden. (ab V3.0.8)

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten `Aborting_BkPlcMc` belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Bewegung signalisiert.

CommandAborted: Hier wird ein Abbruch der Bewegung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Als nächstes wird überprüft, ob **Position** hinter einem aktiven Software-Endschalter liegt. In diesem Fall wird mit **Error** und **ErrorID:=dwTcHydErrCdSoftEnd** reagiert.
- In Abhängigkeit vom in `Axis.pStAxParams^.nProfile` festgelegten Bewegungsalgorithmus kann die Achse entweder nur aus dem Stillstand oder auch aus einer anderen noch nicht abgeschlossenen Bewegung heraus die hier gestartete Bewegung beginnen. Sollte sie derzeit nicht in der Lage sein diesen neuen Auftrag zu akzeptieren wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** reagiert.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Velocity** zu klein (weniger als 1% der Referenzgeschwindigkeit) ist wird mit **Error** und **ErrorID:=dwTcHydErrCdSetVelo** reagiert.
- Wenn **Acceleration** zu klein ist (**Velocity** kann nicht innerhalb von 100 Sekunden erreicht werden) wird mit **Error** und **ErrorID:=dwTcHydErrCdAcc** reagiert.
- Wenn **Deceleration** zu klein ist (**Velocity** kann nicht innerhalb von 100 Sekunden abgebaut werden) wird mit **Error** und **ErrorID:=dwTcHydErrCdDec** reagiert.
- Wenn der Bewegungsalgorithmus bereits einen Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Bewegung initiiert. Dazu werden die Parameter **Position**, **Velocity**, **Acceleration** und **Deceleration** auf die maximal zulässigen Werte begrenzt und an den Bewegungsalgorithmus übergeben. Die Achse befindet sich ab jetzt im Zustand `McState_DiscreteMotion` [► 73] und der Baustein beginnt mit der Überwachung der Bewegung.

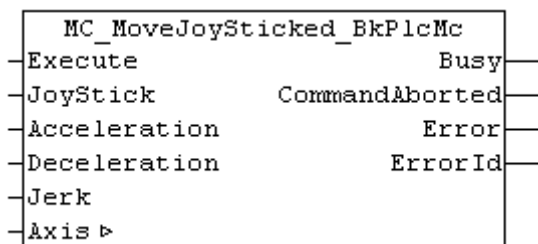
Sollte während der Abarbeitung der Bewegung vom Bewegungsalgorithmus ein Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert. Wird die vollständige Bewegung durch die Aktivität eines anderen Bausteins verhindert, wird mit **CommandAborted** reagiert. Erreicht der Bewegungsalgorithmus die Zielbedingungen der Achse wird mit **Done** reagiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Bewegung auf FALSE gesetzt wird die eingeleitete Bewegung unbeeinflusst weiter bearbeitet. Die Signale am Ende der Bewegung (**Error**, **ErrorID**, **CommandAborted**, **Done**) werden für einen Zyklus gegeben.

Sehen Sie dazu auch

- 📖 `MC_BufferMode_BkPlcMc` (ab V3.0) [► 84]
- 📖 `MC_Direction_BkPlcMc` (ab V3.0) [► 85]

3.2.13 MC_MoveJoySticked_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht die Bewegung einer Achse.

HINWEIS! Diese Funktion wird derzeit nur durch Achsen unterstützt, die durch einen Baustein vom Typ **MC_AxRuntimeCtrlBased_BkPlcMc** kontrolliert werden (in Vorbereitung: **MC_AxRunTimeTimeRamp_BkPlcMc**). Die Auswahl eines solchen Bausteins erfolgt dadurch, dass in **nProfileType** in **ST_TcHydAxParam** die entsprechende Konstante aus **E_TcMcProfileType** vorgegeben wird.

```

VAR_INPUT
    Execute:          BOOL;
    JoyStick:         LREAL;
    Acceleration:     LREAL;
    Deceleration:     LREAL;
    Jerk:             LREAL;
END_VAR
VAR_OUTPUT
    Busy:            BOOL;
    CommandAborted: BOOL;
    Error:           BOOL;
    ErrorID:        UDINT;
END_VAR
VAR_INOUT
    Axis:           Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet die Bewegung.

JoyStick: Die auf den Wertebereich ± 1.0 normierte Geschwindigkeitsvorgabe durch das Kontrollgerät.

Acceleration: Die geforderte Beschleunigung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Deceleration: Die geforderte Verzögerung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Jerk: reserviert.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird ein Abbruch der Bewegung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

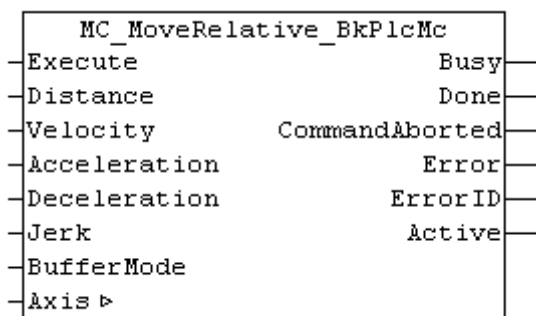
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn der Bewegungsalgorithmus bereits einen Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

- Als nächstes wird überprüft, ob der Generator der Achse die geforderte Funktion unterstützt. Ist dies nicht der Fall wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Bewegung initiiert. Dazu wird der Bewegungsalgorithmus in den Zustand iTcHydStateExtGenerated und die Achse in den Zustand McState_Synchronizedmotion versetzt. Die Geschwindigkeit der Achse wird durch **JoyStick** und **ST_TcHydAxParam [▶ 88].fRefVelo** festgelegt. Bei einer Änderung der Geschwindigkeit wird eine Rampenbegrenzung auf **ST_TcHydAxParam [▶ 88].fMaxAcc** vorgenommen. Fährt die Achse in Richtung auf einen aktiven Software-Endschalter, wird die Geschwindigkeit in Abhängigkeit von der verbleibenden Entfernung so begrenzt, dass eine korrekte Zielfahrt auf den Endschalter zustande kommt.

Eine fallende Flanke an **Execute** versetzt den Bewegungsalgorithmus in den Zustand iTcHydStateTcDecP oder iTcHydStateTcDecM und die Achse in den Zustand McState_Standstill. Falls sich die Achse zu diesem Zeitpunkt bewegt wird sie mit einer Stopprampe abgebremst und geht in den Zustand iTcHydStateIdle über.

3.2.14 MC_MoveRelative_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht die Bewegung einer Achse.

```

VAR_INPUT
  Execute:      BOOL;
  Distance:     LREAL;
  Velocity:     LREAL;
  Acceleration: LREAL;
  Deceleration: LREAL;
  Jerk:         LREAL;
  BufferMode:   MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;    (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:        BOOL;
  Done:        BOOL;
  CommandAborted: BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet die Bewegung.

Distance: Die Entfernung zur Zielposition der Bewegung in Istwert-Einheiten der Achse.

Velocity: Die geforderte Geschwindigkeit der Bewegung in Istwert-Einheiten der Achse pro Sekunde.

Acceleration: Die geforderte Beschleunigung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Deceleration: Die geforderte Verzögerung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Jerk: reserviert.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten Aborting_BkPlcMc belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Bewegung signalisiert.

CommandAborted: Hier wird ein Abbruch der Bewegung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Als nächstes wird überprüft, ob eine Bewegung um **Distance** zu einem Konflikt mit einem aktiven Software-Endschalter liegt. In diesem Fall wird mit **Error** und **ErrorID:=dwTcHydErrCdSoftEnd** reagiert.
- In Abhängigkeit vom in **Axis.pStAxParams^.nProfile** festgelegten Bewegungsalgorithmus kann die Achse entweder nur aus dem Stillstand oder auch aus einer anderen noch nicht abgeschlossenen Bewegung heraus die hier gestartete Bewegung beginnen. Sollte sie derzeit nicht in der Lage sein diesen neuen Auftrag zu akzeptieren wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** reagiert.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Velocity** zu klein (weniger als 1% der Referenzgeschwindigkeit) ist wird mit **Error** und **ErrorID:=dwTcHydErrCdSetVelo** reagiert.
- Wenn **Acceleration** zu klein ist (**Velocity** kann nicht innerhalb von 100 Sekunden erreicht werden) wird mit **Error** und **ErrorID:=dwTcHydErrCdAcc** reagiert.
- Wenn **Deceleration** zu klein ist (**Velocity** kann nicht innerhalb von 100 Sekunden abgebaut werden) wird mit **Error** und **ErrorID:=dwTcHydErrCdDec** reagiert.
- Wenn der Bewegungsalgorithmus bereits einen Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Bewegung initiiert. Dazu werden die Parameter **Distance**, **Velocity**, **Acceleration** und **Deceleration** auf die maximal zulässigen Werte begrenzt und an den Bewegungsalgorithmus übergeben. Die Achse befindet sich ab jetzt im Zustand `McState_DiscreteMotion` [► 73] und der Baustein beginnt mit der Überwachung der Bewegung.

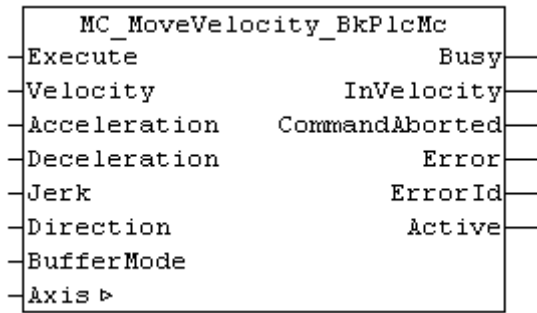
Sollte während der Abarbeitung der Bewegung vom Bewegungsalgorithmus ein Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert. Wird die vollständige Bewegung durch die Aktivität eines anderen Bausteins verhindert, wird mit **CommandAborted** reagiert. Erreicht der Bewegungsalgorithmus die Zielbedingungen der Achse wird mit **Done** reagiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Bewegung auf FALSE gesetzt wird die eingeleitete Bewegung unbeeinflusst weiter bearbeitet. Die Signale am Ende der Bewegung (**Error**, **ErrorID**, **CommandAborted**, **Done**) werden für einen Zyklus gegeben.

Sehen Sie dazu auch

📖 `MC_BufferMode_BkPlcMc` (ab V3.0) [► 84]

3.2.15 MC_MoveVelocity_BkPlcMc (ab V3.0)



Der Funktionsbaustein startet und überwacht die Bewegung einer Achse.

```

VAR_INPUT
    Execute:          BOOL;
    Velocity:         LREAL;
    Acceleration:     LREAL;
    Deceleration:     LREAL;
    Direction:        MC_Direction_BkPlcMc;
    BufferMode:        MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc;    (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
    Busy:            BOOL;
    InVelocity:      BOOL;
    CommandAborted: BOOL;
    Error:           BOOL;
    ErrorID:         UDINT;
END_VAR
VAR_INOUT
    Axis:            Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet die Bewegung.

Velocity: Die geforderte Geschwindigkeit der Bewegung in Istwert-Einheiten der Achse pro Sekunde.

Acceleration: Die geforderte Beschleunigung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Deceleration: Die geforderte Verzögerung in Istwert-Einheiten der Achse pro Quadrat-Sekunde.

Direction: Eine entsprechend [MC_Direction_BkPlcMc \[► 85\]](#) codierte Richtungsangabe.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten `Aborting_BkPlcMc` belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

InVelocity: Dieser Ausgang wird TRUE, wenn die Achse das erste Mal die geforderte Geschwindigkeit erreicht.

CommandAborted: Hier wird ein Abbruch der Bewegung signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- In Abhängigkeit vom in **Axis.pStAxParams^.nProfile** festgelegten Bewegungsalgorithmus kann die Achse entweder nur aus dem Stillstand oder auch aus einer anderen noch nicht abgeschlossenen Bewegung heraus die hier gestartete Bewegung beginnen. Sollte sie derzeit nicht in der Lage sein diesen neuen Auftrag zu akzeptieren wird mit **Error** und **ErrorID:=dwTcHydErrCdNotStartable** reagiert.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Velocity** zu klein (weniger als 1% der Referenzgeschwindigkeit) ist wird mit **Error** und **ErrorID:=dwTcHydErrCdSetVelo** reagiert.
- Wenn der Bewegungsalgorithmus bereits einen Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Bewegung initiiert. Dazu wird in Abhängigkeit von **Direction** und den Parametern der Softwareendschalter ein Wert für die Zielposition gewählt. Dann werden die Parameter **Velocity**, **Acceleration** und **Deceleration** auf die maximal zulässigen Werte begrenzt und an den Bewegungsalgorithmus übergeben. Die Achse befindet sich ab jetzt im Zustand McState Continousmotion [▶ 73] und der Baustein beginnt mit der Überwachung der Bewegung.

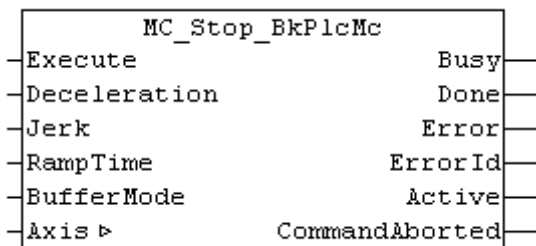
Sollte während der Abarbeitung der Bewegung vom Bewegungsalgorithmus ein Fehlercode signalisiert wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert. Wird die vollständige Bewegung durch die Aktivität eines anderen Bausteins verhindert, wird mit **CommandAborted** reagiert. Erreicht der Bewegungsalgorithmus die geforderte Geschwindigkeit wird **InVelocity** gesetzt.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Bewegung auf FALSE gesetzt wird die eingeleitete Bewegung unbeeinflusst weiter bearbeitet. Die Signale am Ende der Bewegung (**Error**, **ErrorID**, **CommandAborted**, **InVelocity**) werden für einen Zyklus gegeben.

Sehen Sie dazu auch

📖 MC_BufferMode_BkPlcMc (ab V3.0) [▶ 84]

3.2.16 MC_Stop_BkPlcMc (ab V3.0)



Der Funktionsbaustein bricht eine aktuell ausgeführte Bewegung einer Achse ab und überwacht die Stoppoperation.

⚠ HINWEIS! Die von diesem Baustein eingeleitete Stoppoperation ist nicht durch andere Bausteine unterbrechbar. Soll eine Achse während der Stoppoperation durchstartbar sein ist ein MC_Halt_BkPlcMc Baustein zu verwenden.

```

VAR_INPUT
  Execute:          BOOL;
  Deceleration:    LREAL; (ab/from V3.0.5)
  Jerk:            LREAL; (ab/from V3.0.5)
  RampTime:        LREAL; (ab/from V3.0.5)
  BufferMode:       MC_BufferMode_BkPlcMc:=Aborting_BkPlcMc; (ab/from V3.0.8)
END_VAR
VAR_OUTPUT
  Busy:            BOOL;
  Done:            BOOL;
  Error:           BOOL;
  ErrorID:         UDINT;
    
```

```

Active:          BOOL;
CommandAborted: BOOL;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang beendet eine Bewegung der Achse.

Deceleration: Die anzuwendende Verzögerung.

Jerk: Der anzuwendende Ruck.

RampTime: Die geforderte Anhaltezeit.

BufferMode: reserviert. Dieser Eingang wurde vorbereitend ergänzt und sollte derzeit nicht oder mit der Konstanten `Aborting_BkPlcMc` belegt werden. (ab V3.0.8)

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Operation signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Active: Hier wird angezeigt, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird angezeigt, dass die Abarbeitung dieses Kommandos durch ein anderes Kommando abgebrochen wurde.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Ein Stopp kann nur ausgeführt werden, wenn die Achse eine aktive Bewegung ausführt. Befindet sie sich im Stillstand meldet der Baustein sofort **Done**.
- Befindet sich die Achse in einem gestörten Zustand oder führt sie gerade eine Stop-Operation durch wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Befindet sich die Achse in einem Zustand, in dem sie durch eine Kopplung mit einer anderen Achse oder einen vergleichbaren Mechanismus kontrolliert wird reagiert sie mit **Error** und **ErrorID:=dwTcHydErrCdNotReady**.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Stop-Operation initiiert. Dabei wird **Deceleration** angewendet, wenn dieser Parameter erkennbar größer als 0 ist. Andernfalls wird **RampTime** verwendet, um unter Berücksichtigung der Bezugsgeschwindigkeit eine Verzögerung zu berechnen. Ist ein ruckbegrenzender Stellwert-Generator ausgewählt wird **Jerk** verwendet, wenn dieser Parameter erkennbar größer als 0 ist. Ist keiner der genannten Parameter erkennbar größer als 0 werden die Achsparameter `MaxDec` und `MaxJerk` verwendet.

Unter Berücksichtigung der aktuellen Sollgeschwindigkeit und der jetzt geltenden Parameter wird die nächsterreichbare Position ermittelt und als neue Zielposition übernommen. Nachdem diese Position erreicht wurde geht die Achse in ihr reguläres Verhalten im Ruhezustand über.

ⓘ HINWEIS! Die RampTime legt die Zeit fest, in der die Achse von ihrer Bezugsgeschwindigkeit zum Stillstand abgebremst werden soll. Fährt die Achse mit einer anderen Geschwindigkeit wird sich die Bremszeit auf einen entsprechenden Anteil verringern. Bei Stellwert-Generatoren mit Schleichfahrt addiert sich der dabei entstehende Zeitbedarf zur Bremszeit.

Sollte während der Abarbeitung der Bewegung vom Bewegungsalgorithmus ein Fehlercode signalisiert werden wird mit **Error** und **ErrorID:=Bewegungsalgorithmus-Fehlercode** reagiert. Wird die vollständige Abarbeitung durch die Aktivität eines anderen Bausteins verhindert, wird mit **CommandAborted** reagiert. Ein erfolgreicher Abschluss der Operation wird mit **Done** gemeldet.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktiver Operation auf FALSE gesetzt wird der eingeleitete Stopp unbeeinflusst weiter bearbeitet. Die Signale am Ende der Bewegung (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

❏ **HINWEIS!** Der Ausgang **Active** ist derzeit mit dem Ausgang **Busy** identisch.

3.3 Datentypen

3.3.1 Axis_Ref_BkPlcMc (ab V3.0)

Die Variablen in dieser Struktur fassen die Unterbestandteile der Achse zusammen. Eine Variable dieses Typs wird an die meisten Bausteine der Bibliothek übergeben und somit entspricht dieser Typ dem **AXIS_REF** Datentyp der PlcOpen.

```

TYPE Axis_Ref_BkPlcMc:
STRUCT
  sAxisName:          STRING(83) := 'NoName';
  pStAxLogBuffer:     POINTER TO ST_TcMcLogBuffer:=0;
  pStDeviceInput:     POINTER TO ST_TcPlcDeviceInput:=0;
  pStDeviceOutput:    POINTER TO ST_TcPlcDeviceOutput:=0;
  pStAxAuxLabels:     POINTER TO ST_TcMcAuxDataLabels:=0;
  pStAxAutoParams:    POINTER TO ST_TcMcAutoIdent:=0;
  pStAxCommandBuf:    POINTER TO ST_TcPlcCmdBuffer_BkPlcMc:=0;
  nActiveRequest:     UDINT := 0;
  nNextRequest:       UDINT := 1;
  bParamsEnable:      BOOL:=FALSE;
  nState:             E_TcMCFbState:=McState_Standstill;
  nInitState:         INT:=0;
  nInitError:         DINT:=0;
  nInterfaceType:     UINT := 16#FFFF;
  nDeviceInType:      UINT := 16#FFFF;
  nDeviceOutType:     UINT := 16#FFFF;
  nRtDataType:        UINT := 16#FFFF;
  nParamType:         UINT := 16#FFFF;
  nLogBufferType:     UINT := 16#FFFF;
  nAxAutoIdentType:   UINT := 16#FFFF;
  nCmdBufferType:     UINT := 16#FFFF;
  nLogLevel:          DINT := 0;
  nDebugTag:          UDINT := 16#00000000;
  stAxParams:         ST_TcHydAxParam;
  stAxRtData:         ST_TcHydAxRtData;
END_STRUCT
END_TYPE

```

sAxisName: Der Text-Name der Achse.

pStAxLogBuffer: Die Adresse einer Variablen vom Typ [ST_TcMcLogBuffer](#) [► 102]. Diese Variable enthält den LogBuffer der Library.

pStDeviceInput: Die Adresse einer Variablen vom Typ [ST_TcPlcDeviceInput](#) [► 98]. Diese Variable enthält alle Eingangsinterfaces der Achse.

pStDeviceOutput: Die Adresse einer Variablen vom Typ [ST_TcPlcDeviceOutput](#) [► 100]. Diese Variable enthält alle Ausgangsinterfaces der Achse.

pStAxAuxLabels: Die Adresse einer Variable vom Typ [ST_TcMcAuxDataLabels](#) [► 98]. Diese Variable enthält optional die Bezeichnungen der Applikations-Parameter in [ST_TcHydAxParam.fCustomerData\[..\]](#).

pStAxAutoParams: Die Adresse einer Variable vom Typ [ST_TcMcAutoIdent](#) [► 88]. Diese Variable enthält optional die Parameter für einen [MC_AxUtiAutoIdent_BkPlcMc](#) [► 181] Baustein.

pStAxCommandBuf: Ab V3.0.8 ist bei diversen Bausteinen der durch die PLCopen definierte Eingang **BufferMode** vorhanden. Die damit steuerbare Funktionalität wird derzeit vorbereitet. In diesem Zusammenhang ist dieser Befehls-Puffer ergänzt worden.

nActiveRequest: Hier stellt jeder Baustein eine Kennzahl ein, der eine Funktion auf dieser Achse startet. Anschließend überwacht der Baustein diese Variable auf Veränderung durch einen anderen Baustein, der durch eine andere Funktion die Kontrolle übernimmt. So kann jeder Baustein feststellen, ob die von ihm gestartete Funktion durch einen anderen Baustein abgebrochen wurde und entsprechende Signale erzeugen.

nNextRequest: Reserviert. Wird für die Erzeugung von neuen Werten für **nActiveRequest** benutzt.

bParamsEnable: Diese Variable ist nur auf TRUE, wenn die Parameter durch Laden aus einer Datei in einen gültigen Zustand versetzt wurden. Ein Abspeichern der Parameter setzt ebenfalls dieses Signal, da die Übereinstimmung der Daten in der Parameterstruktur und in der Datei ebenfalls sichergestellt ist. Solange diese Variable nicht auf TRUE steht ist die Achse nicht betriebsbereit.

! **HINWEIS!** Zur Laufzeit wird bei schreibenden Zugriffen auf die Parameterstruktur diese Variable temporär auf FALSE und anschließend wieder auf ihren Vorzustand gesetzt.

nState: Hier ist in einer entsprechend [E_TcMCFbState](#) [▶ 73] codierten Form der aktuelle Zustand der Achse hinterlegt.

nInitState: Der aktuelle Zustand der Initialisierung.

nInitError: Ein eventuell beim Initialisieren festgestellte Fehlercode.

nInterfaceType: Der Typecode des derzeit gültigen [Axis_Ref_BkPlcMc](#) Variablentyps.

nDeviceInType: Der Typecode des derzeit gültigen [ST_TcPlcDeviceInput](#) [▶ 98] Variablentyps.

nDeviceOutType: Der Typecode des derzeit gültigen [ST_TcPlcDeviceOutput](#) [▶ 100] Variablentyps.

nRtDataType: Der Typecode des derzeit gültigen [ST_TcHydAxRtData](#) [▶ 94] Variablentyps.

nParamType: Der Typecode des derzeit gültigen [ST_TcHydAxParam](#) [▶ 88] Variablentyps.

nLogBufferType: Der Typecode des derzeit gültigen [ST_TcMcLogBuffer](#) [▶ 102] Variablentyps.

nAxAutoIdentType: Der Typecode des derzeit gültigen [ST_TcMcAutoIdent](#) [▶ 88] Variablentyps.

nCmdBufferType: reserviert. Der Typecode des derzeit gültigen Befehls-Puffer-Variablentyps.

nLogLevel: Der [Message Level](#) [▶ 232], ab dem ein Eintrag in den Logging Puffer erfolgen soll.

nDebugTag: Viele Bausteine der Bibliothek tragen hier für die Dauer ihrer Durchführung eine Debug-Kennung ein.

stAxParams: Diese Variable vom Typ [ST_TcHydAxParam](#) [▶ 88] enthält die Parameter der Achse.

stAxRtData: Diese Variable vom Typ [ST_TcHydAxRtData](#) [▶ 94] enthält die Laufzeitdaten der Achse.



Hinweis

Um die Datenstrukturen der Bibliothek unabhängig von der Architektur der CPU (I86, Strong ARM) zu machen ist es erforderlich an einigen Stellen Daten in der Reihenfolge umzuordnen oder Platzhalter einzufügen. Diese Platzhalter erhalten einen Namen der Form "bAlign_1" wobei die Zahlenangabe keinerlei Bedeutung besitzt. Weder Vorhandensein, Benennung, Typ oder Dimensionierung werden garantiert.

3.3.2 CAMSWITCH_REF_BkPlcMc (ab V3.0)

Eine Variable dieses Typs wird an einen [MC_DigitalCamSwitch_BkPlcMc](#) [▶ 41] Baustein übergeben.

```
TYPE CAMSWITCH_REF_BkPlcMc:
STRUCT
    Switch:      ARRAY [ciBkPlcMc_CamSwitchRef_MinIdx..ciBkPlcMc_CamSwitchRef_MaxIdx] OF CAMS-
WITCH_REFTYPE_BkPlcMc;
END_STRUCT
END_TYPE
```

```

TYPE CAMSWITCH_REFTYPE_BkPlcMc:
STRUCT
  TrackNumber:      INT;
  FirstOnPosition:  LREAL;
  LastOnPosition:   LREAL;
  AxisDirection:    INT;
  CamSwitchMode:    INT;
  Duration:         LREAL;
  (* private members, do not touch *)
  nCurrentState:    SINT:=0;
  bTriggered:       BOOL:=FALSE;
  fTimer:          LREAL;
  (**)
END_STRUCT
END_TYPE

```

TrackNumber: Dies ist ein Index in einem ARRAY

[ciBkPlcMc_TrackRef_MinIdx..ciBkPlcMc_TrackRef_MaxIdx] OF TRACK_REF_BkPlcMc [► 103] , das an einen Baustein vom Typ MC_DigitalCamSwitch_BkPlcMc [► 41] übergeben wird.

FirstOnPosition: Der Beginn der Nockenspur. Bei zeitgesteuerten Nocken ist dies die Triggerposition.

LastOnPosition: Das Ende der Nockenspur. Nicht wirksam bei zeitgesteuerten Nocken.

AxisDirection: Hier wird festgelegt, bei welcher Bewegungsrichtung der Nocken aktiv wird: 0 = beide Richtung, 1 = positive Richtung, 2 = negative Richtung.

CamSwitchMode: Der Arbeitsmodus des Nockens: Für weggesteuerte Nocken ist hier der Wert 0, für zeitgesteuerte Nocken der Wert 1 einzutragen.

Duration: Bei zeitgesteuerten Nocken ist hier die Einschaltdauer in Sekunden einzutragen.

nCurrentState, bTriggered, fTimer: Diese Elemente sind Laufzeitvariablen und dürfen nicht von der Applikation beeinflusst oder benutzt werden.

3.3.3 E_TcPlcBufferedCmdType_BkPlcMc

Die Konstanten in dieser Auflistung werden zur Kennzeichnung von gepufferten Achskommandos. Siehe hierzu MC_BufferMode_BkPlcMc [► 84].

```

TYPE E_TcPlcBufferedCmdType_BkPlcMc : (
(* last modification: xx.xx.2009 *)
iBufferedCmd_NoOperation,
iBufferedCmd_MoveAbsolute,
iBufferedCmd_MoveRelative,
iBufferedCmd_MoveVelocity,
(**)
iBufferedCmd_Stop,
iBufferedCmd_ResetAndStop,
iBufferedCmd_Halt,
iBufferedCmd_CamIn,
iBufferedCmd_GearIn,
iBufferedCmd_Power,
iBufferedCmd_Home,
iBufferedCmd_StepAbsSwitch,
iBufferedCmd_StepLimitSwitch,
iBufferedCmd_StepBlock,
iBufferedCmd_StepDirect,
iBufferedCmd_FinishHoming,
(**)
iBufferedCmdEx_Jerk:=100,
iBufferedCmdEx_Acc,
iBufferedCmdEx_Velo,
iBufferedCmdEx_Creep,
(**)
iBufferedCmd_
);
END_TYPE

```

iBufferedCmd_NoOperation: Diese Konstante wird als Initial-Wert für Aufruf-Parameter von Funktionsbausteinen und in Variablen benutzt.

iBufferedCmd_MoveAbsolute: Das gepufferte Kommando ist durch einen MC_MoveAbsolute_BkPlcMc-Baustein eingetragen worden. Siehe Hinweis #1.

iBufferedCmd_MoveRelative: Das gepufferte Kommando ist durch einen MC_MoveRelative_BkPlcMc-Baustein eingetragen worden. Siehe Hinweis #1.

iBufferedCmd_MoveVelocity: Das gepufferte Kommando ist durch einen MC_MoveVelocity_BkPlcMc-Baustein eingetragen worden. Siehe Hinweis #1.

iBufferedCmd_Stop: reserviert, nicht implementiert.

iBufferedCmd_ResetAndStop: reserviert, nicht implementiert.

iBufferedCmd_Halt: reserviert, nicht implementiert.

iBufferedCmd_CamIn: reserviert, nicht implementiert.

iBufferedCmd_GearIn: reserviert, nicht implementiert.

iBufferedCmd_Power: reserviert, nicht implementiert.

iBufferedCmd_Home: reserviert, nicht implementiert.

iBufferedCmd_StepAbsSwitch: reserviert, nicht implementiert.

iBufferedCmd_StepLimitSwitch: reserviert, nicht implementiert.

iBufferedCmd_StepBlock: reserviert, nicht implementiert.

iBufferedCmd_StepDirect: reserviert, nicht implementiert.

iBufferedCmd_FinishHoming: reserviert, nicht implementiert.

iBufferedCmdEx_Jerk: Der mit konstanter Ruck gefahrene Anteil eines Kommandos ist durch einen Baustein eingetragen worden. Siehe Hinweis #2.

iBufferedCmdEx_Acc: Der mit konstanter Beschleunigung oder Verzögerung gefahrene Anteil eines Kommandos ist durch einen Baustein eingetragen worden. Siehe Hinweis #2.

iBufferedCmdEx_Velo: Der mit konstanter Geschwindigkeit gefahrene Anteil eines Kommandos ist durch einen Baustein eingetragen worden. Siehe Hinweis #2.

iBufferedCmdEx_Creep: reserviert, nicht implementiert.

ⓘ HINWEIS! #1 Wenn die Achse einen Sollwertgenerator-Typ nutzt, der keinen Look Ahead realisiert werden vollständige Kommandos als ein Puffer-Element eingetragen.

ⓘ HINWEIS! #2 Wenn die Achse einen Sollwertgenerator-Typ nutzt, der einen Look Ahead realisiert werden Kommandos in Teil-Abschnitte zerlegt und als Paket von typischerweise sieben Puffer-Elemente (Ruck, Beschleunigung, Ruck, Geschwindigkeit, Ruck, Verzögerung, Ruck) eingetragen.

3.3.4 E_TcMcCurrentStep (ab V3.0)

Die Konstanten in dieser Auflistung werden für die Kennzeichnung der internen Zustände der Stellwertgeneratoren benutzt.

ⓘ HINWEIS! Nicht jeder dieser Zustände wird von allen Stellwertgeneratoren benutzt.

```

TYPE E_TcMcCurrentStep : (
  iTcHydStateIdle,
  iTcHydStateTcAccP,
  iTcHydStateTcAccM,
  iTcHydStatePcAccP,
  iTcHydStatePcAccM,
  iTcHydStateConstVeloP,
  iTcHydStateConstVeloM,
  iTcHydStatePcDecP,
  iTcHydStatePcDecM,
  iTcHydStateCreepVeloP,

```

```

iTcHydStateCreepVeloM,
iTcHydStateTcDecP,
iTcHydStateTcDecM,
iTcHydStateFeedStopPos,
iTcHydStateFeedStopNeg,
iTcHydStateDoBrake,
iTcHydStateCoupling := 1000,
iTcHydStateCoupled,
iTcHydStateExtCoupled,
iTcHydStateExtGenerated := 2000,
iTcHydStateEmergencyBreak := 9000,
iTcHydStateFault := 9999
);
END_TYPE

```

iTcHydStateIdle: Die Achse führt keine aktive Fahrbewegung aus. Ihr Verhalten wird von ST_TcHydAxParam.fLagAmp, ST_TcHydAxParam.fTargetClamping und ST_TcHydAxParam.fReposDistance bestimmt.

iTcHydStateTcAccP: Die Achse baut entsprechend ST_TcHydAxRtData.fDestAcc einen positiven Stellwert auf. Dieser Wert wird durch einen der Startbausteine entsprechend den Daten des Fahrauftrags gesetzt. Erreicht der Stellwert den vorgesehenen Fahrstellwert wird in den Zustand gewechselt. Wird vorher festgestellt, dass der Bremsvorgang für die Zielanfahrt eingeleitet werden muss wird in den Zustand **iTcHydStatePcDecP** gewechselt. Bei Wegfall der Vorschubfreigabe wird in den Zustand **iTcHydStateFeedStopPos** gewechselt.

iTcHydStateTcAccM: Die Achse baut entsprechend ST_TcHydAxRtData.fDestAcc einen negativen Stellwert auf. Dieser Wert wird durch einen der Startbausteine entsprechend den Daten des Fahrauftrags gesetzt. Erreicht der Stellwert den vorgesehenen Fahrstellwert wird in den Zustand gewechselt. Wird vorher festgestellt, dass der Bremsvorgang für die Zielanfahrt eingeleitet werden muss wird in den Zustand **iTcHydStatePcDecM** gewechselt. Bei Wegfall der Vorschubfreigabe wird in den Zustand **iTcHydStateFeedStopNeg** gewechselt.

iTcHydStatePcAccP: Die Achse befindet sich in der weggesteuerten Beschleunigungsphase einer Fahrbewegung in positiver Richtung. Dabei wird der Stellwert entsprechend ST_TcHydAxRtData.fDestAcc auf einen vom Fahrauftrag festgelegten Wert aufgebaut. Anschließend wird in den Zustand **iTcHydStateConstVeloP** gewechselt.

iTcHydStatePcAccM: Die Achse befindet sich in der weggesteuerten Beschleunigungsphase einer Fahrbewegung in negativer Richtung. Dabei wird der Stellwert entsprechend ST_TcHydAxRtData.fDestAcc auf einen vom Fahrauftrag festgelegten Wert aufgebaut. Anschließend wird in den Zustand **iTcHydStateConstVeloM** gewechselt.

iTcHydStateConstVeloP: Die Achse fährt mit konstantem Stellwert in positiver Richtung. Der Stellwert wird durch den Fahrauftrag festgelegt.

iTcHydStateConstVeloM: Die Achse fährt mit konstantem Stellwert in negativer Richtung. Der Stellwert wird durch den Fahrauftrag festgelegt.

iTcHydStatePcDecP: Die Achse befindet sich in der weggesteuerten Bremsphase einer Fahrbewegung in positiver Richtung. Dabei wird der Stellwert bis auf ST_TcHydAxParam.fCreepSpeed abgebaut. Anschließend wird in den Zustand **iTcHydStateCreepVeloP** gewechselt.

iTcHydStatePcDecM: Die Achse befindet sich in der weggesteuerten Bremsphase einer Fahrbewegung in negativer Richtung. Dabei wird der Stellwert bis auf ST_TcHydAxParam.fCreepSpeed abgebaut. Anschließend wird in den Zustand **iTcHydStateCreepVeloM** gewechselt.

iTcHydStateCreepVeloP: Die Achse fährt mit konstantem Stellwert in positiver Richtung. Der Stellwert wird durch ST_TcHydAxParam.fCreepSpeed festgelegt.

iTcHydStateCreepVeloM: Die Achse fährt mit konstantem Stellwert in negativer Richtung. Der Stellwert wird durch ST_TcHydAxParam.fCreepSpeed festgelegt.

iTcHydStateTcDecP: Die Achse führt ausgehend von einer Fahrbewegung in positiver Richtung einen regulären Stopp aus. Dabei wird der Stellwert mit ST_TcHydAxParam.fStopRamp abgebaut. Anschließend wird in den Zustand **iTcHydStateIdle** gewechselt.

iTcHydStateTcDecM: Die Achse führt ausgehend von einer Fahrbewegung in negativer Richtung einen regulären Stopp aus. Dabei wird der Stellwert mit ST_TcHydAxParam.fStopRamp abgebaut. Anschließend wird in den Zustand **iTcHydStateIdle** gewechselt.

iTcHydStateFeedStopPos: Die Achse führt wegen nicht anstehender Vorschubfreigabe in positiver Richtung (in ST_TcHydAxRtData.nDeCtrIDWord ist dwTcHydDcDwFdPosEna nicht gesetzt) einen Zwischenstopp aus. Dabei wird der Stellwert mit ST_TcHydAxParam.fStopRamp abgebaut. Anschließend wird auf eine Vorschubfreigabe gewartet.

iTcHydStateFeedStopNeg: Die Achse führt wegen nicht anstehender Vorschubfreigabe in negativer Richtung (in ST_TcHydAxRtData.nDeCtrIDWord ist dwTcHydDcDwFdNegEna nicht gesetzt) einen Zwischenstopp aus. Dabei wird der Stellwert mit ST_TcHydAxParam.fStopRamp abgebaut. Anschließend wird auf eine Vorschubfreigabe gewartet.

iTcHydStateDoBrake: Die Achse führt eine Wartezeit aus. Dies ist erforderlich, wenn wegen einer Bremse oder ein Schaltventils geschaltet werden muss.

iTcHydStateCoupling: Die Achse befindet sich im Übergang in den Zustand **iTcHydStateCoupled**.

iTcHydStateCoupled: Der Stellwert der Achse wird nach dem Prinzip des elektronischen Getriebes vom Stellwert einer anderen Achse abgeleitet.

iTcHydStateExtCoupled: Der Stellwert der Achse wird nach dem Prinzip des stufenlosen Getriebes berechnet.

iTcHydStateExtGenerated: Der Stellwert der Achse wird durch einen externen Baustein erzeugt. dabei kann es sich um einen Baustein der Library oder um einen applikationsspezifischen Baustein handeln.

iTcHydStateEmergencyBreak: Die Achse führt einen Notstopp aus. Dabei wird der Stellwert mit ST_TcHydAxParam.fEmergencyRamp abgebaut. Anschließend wird untersucht, ob sich die Achse in einem Stöorzustand (ST_TcHydAxRtData.nErrorCode ungleich 0) befindet. Wenn ja wird in den Zustand **iTcHydStateFault** gewechselt, andernfalls in den Zustand **iTcHydStateIdle**.

iTcHydStateFault: Die Achse ist in einem Stöorzustand. Sie führt keine aktiv kontrollierten Bewegungen aus und nimmt auch keine Bewegungsaufträge an. Um die Achse wieder in einen ungestörten Zustand zu versetzen ist ein Baustein des Typs MC_Reset_BkPlcMc oder MC_ResetAndStop_BkPlcMc aufzurufen.

3.3.5 E_TcMcDriveType (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung von für die Stellwertausgabe einer Achse verwendeter Hardware benutzt.

```

TYPE E_TcMcDriveType : (
  (*
    The sequence below must not be changed!
    New types have to be added at the end.
    In case a type becomes obsolete it has to be replaced by a dummy
    to ensure the numerical meaning of the other codes.
  *)
  (*
    Die bestehende Reihenfolge darf nicht veraendert werden.
    Neue Typen muessen am Ende eingefuegt werden.
    Wenn ein Typ wegfallen sollte, muss er durch einen Dummy
    ersetzt werden, um die numerische Zuordnung zu garantieren.
  *)
  (* last modification: 26.02.2016 *)
  iTcMc_Drive_Customized,
  iTcMc_DriveLowCostStepper,
  iTcMc_DriveKL2521,
  iTcMc_DriveKL4032,
  iTcMc_DriveAx2000_B900R,
  iTcMc_DriveM2400_D1,
  iTcMc_DriveM2400_D2,
  iTcMc_DriveM2400_D3,
  iTcMc_DriveM2400_D4,
  iTcMc_DriveLowCostStepperHS,
  iTcMc_DriveLowCostStepperFS,
  iTcMc_DriveIx2512_1Coil,
  iTcMc_DriveIx2512_2Coil,
  iTcMc_DriveKL2531,

```

```

iTcMc_DriveKL2541,
iTcMc_DriveEL4132,
iTcMc_DriveAx2000_B200R,
iTcMc_DriveAx2000_B110R,
iTcMc_DriveKL2532,
iTcMc_DriveKL2552,
iTcMc_DriveKL2535_1Coil,
iTcMc_DriveKL2535_2Coil,
iTcMc_DriveKL2545_1Coil,
iTcMc_DriveKL2545_2Coil,
iTcMc_DriveLowCostInverter,
iTcMc_Drive_CoE_DS408,
iTcMc_DriveAx2000_B110A,
iTcMc_DriveAx5000_B110A,
iTcMc_DriveAx2000_B750A,
iTcMc_Drive_CoE_DS402,
iTcMc_DriveAx5000_B110SR,
iTcMc_DriveEL4x22,
iTcMc_DriveEL2521,
iTcMc_DrivePumpEtcIO,
iTcMc_DriveEL2535_1Coil,
iTcMc_DriveEL2535_2Coil,
iTcMc_DriveEL7201,
iTcMc_DriveEL7037,
iTcMc_DriveEL7047,
iTcMc_DriveEM8908,
iTcMc_DriveAx5000_B110INC,
iTcMc_Drive_TestOnly:=1000
);
END_TYPE

```

iTcMc_Drive_Customized: Der Stellwert für den Antrieb wird nicht für die Ausgabe auf einer bestimmten Hardware aufbereitet. Diese Aufbereitung ist durch die PLC-Applikation selbst vorzunehmen.

iTcMc_DriveAx2000_B110A: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX2000 Stellgerät an einem Absolut-Multiturnencoder-Motor am EtherCAT Feldbus aufbereitet.

iTcMc_DriveAx2000_B110R: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX2000 Stellgerät an einem Resolver-Motor am EtherCAT Feldbus aufbereitet.

iTcMc_DriveAx2000_B200R: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX2000 Stellgerät an einem Resolver-Motor am Beckhoff II/O Feldbus aufbereitet.

iTcMc_DriveAx2000_B750A: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX2000 Stellgerät an einem Absolut-Multiturnencoder-Motor am Sercos Feldbus aufbereitet.

iTcMc_DriveAx2000_B900R: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX2000 Stellgerät an einem Resolver-Motor am Beckhoff RealTime Ethernet Feldbus aufbereitet.

iTcMc_DriveAx5000_B110A: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX5000 Stellgerät an einem Absolut-Multiturnencoder-Motor am EtherCAT Feldbus aufbereitet.

iTcMc_DriveAx5000_B110SR: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX5000 Stellgerät an einem Absolut-Singleturnencoder- oder Resolver-Motor am EtherCAT Feldbus aufbereitet.

iTcMc_DriveAx5000_B110INC: Der Stellwert für den Antrieb wird für die Ausgabe auf einem AX5000 Stellgerät an einem Inkremental-Encoder am EtherCAT Feldbus aufbereitet.

iTcMc_DriveEL2521: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Pulse Train Klemme KL2521 aufbereitet.

iTcMc_DriveEL2535_1Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf eine PWM-Endstufenklemme EL2535 aufbereitet.

iTcMc_DriveEL2535_2Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf eine PWM-Endstufenklemme EL2535 aufbereitet.

iTcMc_DriveEL4132: Der Stellwert für den Antrieb wird für die Ausgabe auf einer Analog-Ausgangsklemme ± 10 V EL4132 aufbereitet.

iTcMc_DriveEL4x22: In Vorbereitung.

iTcMc_DriveEL7031: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Schrittmotorendstufenklemme EL7031 aufbereitet.

iTcMc_DriveEL7037: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Schrittmotorendstufenklemme EL7037 aufbereitet.

iTcMc_DriveEL7041: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Schrittmotorendstufenklemme EL7041 aufbereitet.

iTcMc_DriveEL7047: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Schrittmotorendstufenklemme EL7047 aufbereitet.

iTcMc_DriveEL7201: Der Stellwert für den Antrieb wird für die Ausgabe auf einer Servo-Klemme EL7201 aufbereitet.

iTcMc_DriveEM8908: Reserviert für Branchenpaket.

iTcMc_DriveIx2512_1Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf einer Feldbus Box IP/IE2512 aufbereitet. Es gelten die Regeln für Ventile mit einer Spule.

iTcMc_DriveIx2512_2Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf einer Feldbus Box IP/IE2512 aufbereitet. Es gelten die Regeln für Ventile mit zwei Spulen.

iTcMc_DriveKL2521: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Pulse Train Klemme KL2521 aufbereitet.

iTcMc_DriveKL2531: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Schrittmotorendstufenklemme KL2531 aufbereitet.

iTcMc_DriveKL2532: Der Stellwert für den Antrieb wird für die Ausgabe auf eine DC-Motorendstufenklemme KL2532 aufbereitet.

iTcMc_DriveKL2535_1Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf eine PWM-Endstufenklemme KL2535 aufbereitet.

iTcMc_DriveKL2535_2Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf eine PWM-Endstufenklemme KL2535 aufbereitet.

iTcMc_DriveKL2541: Der Stellwert für den Antrieb wird für die Ausgabe auf eine Schrittmotorendstufenklemme KL2541 aufbereitet.

iTcMc_DriveKL2542: Der Stellwert für den Antrieb wird für die Ausgabe auf eine DC-Motorendstufenklemme KL2542 aufbereitet.

iTcMc_DriveKL2545_1Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf eine PWM-Endstufenklemme KL2545 aufbereitet.

iTcMc_DriveKL2545_2Coil: Der Stellwert für den Antrieb wird für die Ausgabe auf eine PWM-Endstufenklemme KL2545 aufbereitet.

iTcMc_DriveKL4032: Der Stellwert für den Antrieb wird für die Ausgabe auf einer Analog-Ausgangsklemme ± 10 V KL4032 aufbereitet.

iTcMc_DriveLowCostInverter: Der Stellwert für den Antrieb wird für die Ausgabe als digitale Ausgangssignale für einen Frequenzumrichter mit programmierten Festfrequenzen aufbereitet.

iTcMc_DriveLowCostStepper: Die inkrementelle Sollpositionsänderungen werden als digitale Ausgangssignale für einen direkt angesteuerten Schrittmotor erzeugt. Dies ist ein aus Gründen der Kompatibilität weiterhin unterstützter Code und ist gleichbedeutend mit **iTcMc_DriveLowCostStepperHS**.

iTcMc_DriveLowCostStepperHS: Die inkrementelle Sollpositionsänderungen werden als digitale Ausgangssignale für einen direkt angesteuerten Schrittmotor erzeugt. Es wird das Halbschritt-Verfahren angewendet.

iTcMc_DriveLowCostStepperFS: Die inkrementelle Sollpositionsänderungen werden als digitale Ausgangssignale für einen direkt angesteuerten Schrittmotor erzeugt. Es wird das Vollschritt-Verfahren angewendet.

iTcMc_DriveM2400_D1: Der Stellwert für den Antrieb wird für die Ausgabe auf dem ersten Kanal einer M2400 Box am Beckhoff II/O aufbereitet.

iTcMc_DriveM2400_D2: Der Stellwert für den Antrieb wird für die Ausgabe auf dem zweiten Kanal einer M2400 Box am Beckhoff II/O aufbereitet.

iTcMc_DriveM2400_D3: Der Stellwert für den Antrieb wird für die Ausgabe auf dem dritten Kanal einer M2400 Box am Beckhoff II/O aufbereitet.

iTcMc_DriveM2400_D4: Der Stellwert für den Antrieb wird für die Ausgabe auf dem vierten Kanal einer M2400 Box am Beckhoff II/O aufbereitet.

iTcMc_Drive_CoE_DS402: In Vorbereitung.

iTcMc_Drive_CoE_DS408: Der Stellwert für den Antrieb wird für die Ausgabe auf einem Proportionalventil am EtherCAT Feldbus aufbereitet.

iTcMc_DrivePumpEtclO: reserviert

iTcMc_Drive_TestOnly: reserviert für interne Tests; nicht benutzen.

3.3.6 E_TcMcEncoderType (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung von für die Istwerterfassung einer Achse verwendeter Hardware benutzt.

```

TYPE E_TcMcEncoderType : (
  (*
   The sequence below must not be changed!
   New types have to be added at the end.
   In case a type becomes obsolete it has to be replaced by a dummy
   to ensure the numerical meaning of the other codes.
  *)
  (*
   Die bestehende Reihenfolge darf nicht veraendert werden.
   Neue Typen muessen am Ende eingefuegt werden.
   Wenn ein Typ wegfallen sollte, muss er durch einen Dummy
   ersetzt werden, um die numerische Zuordnung zu garantieren.
  *)
  (* last modification: 17.01.2013 *)
  iTcMc_EncoderSim,
  iTcMc_EncoderDigIncrement,
  iTcMc_EncoderLowCostStepper,
  iTcMc_EncoderKL2521,
  iTcMc_EncoderKL3042,
  iTcMc_EncoderKL5001,
  iTcMc_EncoderKL5101,
  iTcMc_EncoderAx2000_B900R,
  iTcMc_EncoderDigCam,
  iTcMc_EncoderIx5009,
  iTcMc_EncoderM2510,
  iTcMc_EncoderKL3002,
  iTcMc_EncoderKL2531,
  iTcMc_EncoderKL5111,
  iTcMc_EncoderAbs32,
  iTcMc_EncoderM3120,
  iTcMc_EncoderKL2541,
  iTcMc_EncoderEL3102,
  iTcMc_EncoderEL3142,
  iTcMc_EncoderEL5001,
  iTcMc_EncoderEL5101,
  iTcMc_EncoderEL5111,
  iTcMc_EncoderKL3062,
  iTcMc_EncoderKL3162,
  iTcMc_EncoderAx2000_B200R,
  iTcMc_EncoderAx2000_B110R,
  iTcMc_EncoderEL3162,
  iTcMc_EncoderKL2542,
  iTcMc_EncoderKL2545,
  iTcMc_EncoderAx2000_B110A,
  iTcMc_EncoderAx5000_B110A,
  iTcMc_EncoderAx2000_B750A,
  iTcMc_EncoderCoE_DS406,
  iTcMc_EncoderCoE_DS402SR,

```

```

iTcMc_EncoderAx5000_B110SR,
iTcMc_EncoderCoE_DS402A,
iTcMc_EncoderEL2521,
iTcMc_EncoderAbs32Etc,
iTcMc_EncoderEL7201SR,
iTcMc_EncoderDigPulseCount,
iTcMc_EncoderEL3255,
iTcMc_EncoderEL7047,
iTcMc_DriveEM8908A,
iTcMc_DriveEM8908C,
iTcMc_EncoderCoE5001,
iTcMc_EncoderEL7201A,
iTcMc_DriveAx5000_B110INC,
iTcMc_Encoder_TestOnly:=1000
);
END_TYPE

```

iTcMc_EncoderAbs32: Die absolute Istposition wird aus dem 32-Bit-Wert einer allgemeinen Eingangselektronik erzeugt.

iTcMc_EncoderAbs32Etc: Die absolute Istposition wird aus dem 32-Bit-Wert einer allgemeinen EtherCAT Eingangselektronik erzeugt. Es wird keine Profil-Unterstützung vorausgesetzt.

iTcMc_EncoderAx2000_B110R: Die inkrementelle Istposition wird aus dem Zähler-Wert eines AX2000 Stellgeräts an einem Resolver-Motor am EtherCAT Feldbus ermittelt.

iTcMc_EncoderAx2000_B110A: Die absolute Istposition wird aus dem Zähler-Wert eines AX2000 Stellgeräts an einem Absolut-Multiturnencoder-Motor am EtherCAT Feldbus ermittelt.

iTcMc_EncoderAx2000_B200R: Die inkrementelle Istposition wird aus dem Zähler-Wert eines AX2000 Stellgeräts an einem Resolver-Motor am Beckhoff II/O Feldbus ermittelt.

iTcMc_EncoderAx2000_B750A: Die absolute Istposition wird aus dem Zähler-Wert eines AX2000 Stellgeräts an einem Absolut-Multiturnencoder-Motor am Sercos Feldbus ermittelt.

iTcMc_EncoderAx2000_B900R: Die inkrementelle Istposition wird aus dem Zähler-Wert eines AX2000 Stellgeräts an einem Resolver-Motor am Beckhoff RealTime Ethernet Feldbus ermittelt.

iTcMc_EncoderAx5000_B110A: Die absolute Istposition wird aus dem Zähler-Wert eines AX5000 Stellgeräts an einem Absolut-Multiturnencoder-Motor am EtherCAT Feldbus ermittelt.

iTcMc_EncoderAx5000_B110SR: Die inkrementelle Istposition wird aus dem Zähler-Wert eines AX5000 Stellgeräts an einem Singleturnencoder-Motor am EtherCAT Feldbus ermittelt.

iTcMc_EncoderAx5000_B110INC: Die inkrementelle Istposition wird aus dem Zähler-Wert eines AX5000 Stellgeräts an einem Inkremental-Encoder am EtherCAT Feldbus ermittelt.

iTcMc_EncoderDigCam: Das Positionsnocken-Byte wird aus vier digitalen Eingangsbits gebildet.

iTcMc_EncoderDigPulseCount: Zählt die Flanken (positive und negative) von Pulsen. Die Drehrichtung wird über die Ausgabe des Drives bestimmt. **☐ HINWEIS! Es kann in einem PLC-Zyklus nur ein Puls erkannt werden.**

iTcMc_EncoderDigIncrement: Der inkrementelle Istwert der Achse wird durch Auswertung von zwei digitalen Eingangsbits gebildet. Diese stellen die A- und B-Spur eines Inkremental-Encoders dar und werden entsprechend dem Prinzip des Quadratur-Decoders mit Vierfachauswertung ausgewertet. **☐ HINWEIS! Es darf pro PLC-Zyklus nur eines der Eingangsbits seinen Zustand ändern. Die maximale Geschwindigkeit beträgt also 1 Inkrement pro TCycle.**

iTcMc_EncoderEL2521: Die inkrementelle Istposition wird aus dem Impulzzähler einer Pulse Train Klemme EL2521 erzeugt.

iTcMc_EncoderEL3102: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme ±10V EL3102 erzeugt.

iTcMc_EncoderEL3142: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme 0..20 mA EL3142 erzeugt.

iTcMc_EncoderEL3162: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme 0..10V EL3162 erzeugt.

iTcMc_EncoderEL3255: In Vorbereitung.

iTcMc_EncoderEL5001: Die absolute Istposition wird aus dem Zähler-Wert einer SSI-Eingangsklemme EL5001 erzeugt.

iTcMc_EncoderEL5101: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Eingangsklemme EL5101 erzeugt.

iTcMc_EncoderEL5111: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Eingangsklemme EL5111 erzeugt.

iTcMc_EncoderEL7041: Die inkrementelle Istposition wird aus dem Impulzzähler (Motorpulse oder Encoder) einer Schrittmotor Klemme EL7041 erzeugt.

iTcMc_EncoderEL7201SR: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Servo-Klemme EL7201 erzeugt.

iTcMc_EncoderCoE_DS402A: In Vorbereitung.

iTcMc_EncoderCoE_DS402SR: In Vorbereitung.

iTcMc_EncoderCoE_DS406: Ein Encoder mit CoE_406 Unterstützung am EtherCAT Feldbus.

iTcMc_EncoderIx5009: Die absolute Istposition wird aus dem Zähler-Wert einer SSI-Feldbusbox IP/IE5009 erzeugt.

iTcMc_EncoderKL2521: Die inkrementelle Istposition wird aus dem Impulzzähler einer Pulse Train Klemme KL2521 erzeugt.

iTcMc_EncoderKL2531: Die inkrementelle Istposition wird aus dem Impulzzähler einer Schrittmotor Klemme KL2531 erzeugt.

iTcMc_EncoderKL2541: Die inkrementelle Istposition wird aus dem Impulzzähler (Motorpulse oder Encoder) einer Schrittmotor Klemme KL2541 erzeugt.

iTcMc_EncoderKL2542: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Eingangsklemme KL2542 erzeugt.

iTcMc_EncoderKL2545: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Eingangsklemme KL2545 erzeugt.

iTcMc_EncoderKL3002: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme $\pm 10V$ KL3002 erzeugt.

iTcMc_EncoderKL3042: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme 0..20 mA KL3042 erzeugt.

iTcMc_EncoderKL3062: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme 0..10V KL3062 erzeugt.

iTcMc_EncoderKL3162: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsklemme 0..10V KL3162 erzeugt.

iTcMc_EncoderKL5001: Die absolute Istposition wird aus dem Zähler-Wert einer SSI-Eingangsklemme KL5001 erzeugt.

iTcMc_EncoderKL5101: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Eingangsklemme KL5101 erzeugt.

iTcMc_EncoderKL5111: Die inkrementelle Istposition wird aus dem Zähler-Wert einer Eingangsklemme KL5111 erzeugt.

iTcMc_EncoderLowCostStepper: Die inkrementelle Istposition wird aus den Ausgangssignalen für einen digital angesteuerten Schrittmotor erzeugt.

iTcMc_EncoderM2510: Die absolute Istposition wird aus dem ADW-Wert einer Analog-Eingangsbox $\pm 10V$ M2510 erzeugt.

iTcMc_EncoderM3120: Die inkrementelle Istposition wird aus dem Zähler-Wert eines M3120 Lightbus Moduls erzeugt.

iTcMc_EncoderSim: Die virtuelle Istposition der Achse ist eine Kopie der Sollposition.



Achtung

iTcMc_EncoderSim

Dieser Typ darf an einer realen Maschine nur bei virtuellen Achsen eingestellt sein. Andernfalls wird die Achse unkontrollierte und nicht vorhersagbare Bewegungen ausführen

iTcMc_Encoder_TestOnly: reserviert für interne Tests; nicht benutzen.

3.3.7 E_TcMCFbState (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Laufzeitzustände von Achsen benutzt.

```
TYPE E_TcMCFbState :
(
  McState_Standstill := 0,
  McState_DiscreteMotion,
  McState_Continuousmotion,
  McState_Synchronizedmotion,
  McState_Stopping,
  McState_Errorstop,
  McState_Homing,
  McState_Disabled
);
END_TYPE
```

McState_Standstill: Die Achse hat keinen Fahrauftrag. Je nach Parametrierung erfolgt eine aktive Lageregelung, eine Repositionier-Überwachung, die Ausgabe eines Press-Stellwerts oder nichts davon.

McState_DiscreteMotion: Die Achse führt eine Bewegung mit einer definierten Zielposition und Geschwindigkeit aus.

McState_Continuousmotion: Die Achse führt eine Bewegung ohne definierte Zielposition aus. Lediglich die Geschwindigkeit ist vorgegeben.

McState_Synchronizedmotion: Die Achse führt eine Bewegung aus, die von der Bewegung einer anderen Achse abgeleitet ist.

McState_Stopping: Die Achse führt eine Stop-Operation aus.

McState_Errorstop: Die Achse wurde wegen eines Problems gestoppt. Sie ist derzeit nicht startbar und muss durch einen Reset wieder in einen startbereiten Zustand versetzt werden.

McState_Homing: Die Achse führt eine Referenzfahrt aus.

McState_Disabled: Die Reglerfreigabe der Achse ist FALSE.

3.3.8 E_TcMcHomingType (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Referenziermethode von Achsen benutzt.

```
TYPE E_TcMcHomingType : (
  iTcMc_HomingOnBlock,
  iTcMc_HomingOnIndex,
  iTcMc_HomingOnSync,
  iTcMc_HomingOnMultiSync,
  iTcMc_HomingOnExec
);
END_TYPE
```

iTcMc_HomingOnBlock: Die Achse wird mit ST_TcHydAxParam.fEnc_RefIndexVelo in der durch ST_TcHydAxParam.bEnc_RefIndexPositive festgelegten Richtung bewegt. Wird für eine Zeitdauer von 2 Sekunden keine Bewegung festgestellt gilt der Festanschlag (Block) als erreicht. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

iTcMc_HomingOnIndex: Die Achse wird mit ST_TcHydAxParam.fEnc_RefIndexVelo in der durch ST_TcHydAxParam.bEnc_RefIndexPositive festgelegten Richtung bewegt. Wird in ST_TcHydAxRtData.nDeCtrlDWord die Referenznocke (Bit 5, dwTcHydDcDwRefIndex) erkannt stoppt die Achse. Anschließend wird sie mit ST_TcHydAxParam.fEnc_RefSyncVelo in der durch ST_TcHydAxParam.bEnc_RefSyncPositive festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

iTcMc_HomingOnSync: Die Achse wird mit ST_TcHydAxParam.fEnc_RefIndexVelo in der durch ST_TcHydAxParam.bEnc_RefIndexPositive festgelegten Richtung bewegt. Wird in ST_TcHydAxRtData.nDeCtrlDWord die Referenznocke (Bit 5, dwTcHydDcDwRefIndex) erkannt stoppt die Achse. Anschließend wird sie mit ST_TcHydAxParam.fEnc_RefSyncVelo in der durch ST_TcHydAxParam.bEnc_RefSyncPositive festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Dann wird das Hardware-Latch des Encoders aktiviert und die Achse so lange weiter bewegt, bis das Latch gültig wird. Nach dem Stoppen der Achse wird der Istwert der Achse auf einen Wert gesetzt, der aus der Referenzposition und der seit dem Sync-Puls des Encoders zurückgelegten Strecke errechnet wird.

iTcMc_HomingOnExec: Der Istwert der Achse wird unmittelbar auf den Wert der Referenzposition gesetzt.

iTcMc_HomingOnMultiSync: Das Hardware-Latch des Encoders wird aktiviert. Die Achse wird mit ST_TcHydAxParam.fEnc_RefSyncVelo in der durch ST_TcHydAxParam.bEnc_RefIndexPositive festgelegten Richtung so lange bewegt, bis das Latch zweimal gültig geworden ist. Wird das Ende des Fahrwegs erkannt bevor zwei Sync-Pulse erkannt wurden wird der Vorgang in entgegengesetzter Richtung wiederholt. Führt auch dies nicht zum Erfolg wird die Referenzfahrt abgebrochen. Andernfalls wird aus dem Abstand der Sync-Pulse und dem fEnc_BaseDistance die aktuelle Istposition bestimmt.

3.3.9 E_TcMCPParameter (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Parameternummerierung benutzt.

```

TYPE E_TcMCPParameter :
(
(*
=====
N T I O N                               =                               A T T E
=====
= These Codes are also used to identify parameters in files.           =
= Any change of the meaning of any code here will make any file       =
= incompatible without notice and may even cause a crash of           =
= the control system!                                                 =
=====
= CONSEQUENCE: Only adding new codes is allowed!                       =
=====
= These codes are also used for ADS communication                       =
=====
*)
(*
=====
A C H T U N G                           =                               =
=====
= Diese Codes werden auch zur Kennzeichnung von Parametern             =
= in den Dateien verwendet. Eine Veraenderung der Codes wuerde        =
= die Dateien (nicht erkennbar) inkompatibel machen und koennte        =
= zum Systemabsturz fuehren!                                           =
=====
= ALSO: Es duerfen nur neue Codes dazugefuegt werden!                 =
=====
= Diese Codes werden ebenfalls fuer die ADS-Kommunikation benutzt      =
=====
*)
McPara_CommandedPosition:=1,
McPara_SWLimitPos,
McPara_SWLimitNeg,
McPara_EnableLimitPos,
McPara_EnableLimitNeg,
McPara_EnablePosLagMonitoring,

```

```

McPara_MaxPositionLag,
McPara_MaxVelocitySystem,
McPara_MaxVelocityAppl,
McPara_ActualVelocity,
McPara_CommandedVelocity,
McPara_MaxAccelerationSystem,
McPara_MaxAccelerationAppl,
McPara_MaxDecelerationSystem,
McPara_MaxDecelerationAppl,
McPara_MaxJerk,
(* ===== *)
McPara_BkPlcMc_ProfilType:=1000,
McPara_BkPlcMc_EnvCycletime,
McPara_BkPlcMc_AxName,
McPara_BkPlcMc_TimeBased,
McPara_BkPlcMc_JerkEnabled,
McPara_BkPlcMc_LogLevel,
McPara_BkPlcMc_CycleDivider,
McPara_BkPlcMc_ParamFileName,
McPara_BkPlcMc_EncoderType:=1100,
McPara_BkPlcMc_EncoderHomingType,
McPara_BkPlcMc_EncoderZeroShift,
McPara_BkPlcMc_EncoderIncWeighting,
McPara_BkPlcMc_EncoderIncInterpolation,
McPara_BkPlcMc_EncoderRefIndexVelo,
McPara_BkPlcMc_EncoderRefIndexPositive,
McPara_BkPlcMc_EncoderRefSyncVelo,
McPara_BkPlcMc_EncoderRefSyncPositive,
McPara_BkPlcMc_EncoderDefaultHomePosition,
McPara_BkPlcMc_EncoderReversed,
McPara_BkPlcMc_EncoderBaseDistance,
McPara_BkPlcMc_EncoderModuloBase,
McPara_BkPlcMc_EncoderEnableLatch,
McPara_BkPlcMc_EncoderLatchedPos,
McPara_BkPlcMc_EncoderRefShift,
McPara_BkPlcMc_EncoderRefFlag,
McPara_BkPlcMc_EncoderPotIRgToRl,
McPara_BkPlcMc_EncoderOverrunMask,
McPara_BkPlcMc_EncoderPositionMask,
McPara_BkPlcMc_EncoderZeroSwap,
McPara_BkPlcMc_ValveOverlapCompP:=1200,
McPara_BkPlcMc_ValveBendPointVelo,
McPara_BkPlcMc_ValveBendPointOutput,
McPara_BkPlcMc_ValveResponseTime,
McPara_BkPlcMc_ValveOverlapCompM,
McPara_BkPlcMc_CylinderArreaA:=1280,
McPara_BkPlcMc_CylinderArreaB,
McPara_BkPlcMc_DriveType:=1300,
McPara_BkPlcMc_AreaRatio,
McPara_BkPlcMc_DriveReversed,
McPara_BkPlcMc_DriveDefaultPowerOk
McPara_BkPlcMc_DriveAbsoluteOutput,
McPara_BkPlcMc_DriveIncWeighting,
McPara_BkPlcMc_DriveIncInterpolation,
McPara_BkPlcMc_StartRamp:=1400,
McPara_BkPlcMc_obsolete_1,
McPara_BkPlcMc_obsolete_2,
McPara_BkPlcMc_StopRamp:=1500,
McPara_BkPlcMc_EmergencyRamp,
McPara_BkPlcMc_BrakeOn,
McPara_BkPlcMc_BrakeOff,
McPara_BkPlcMc_BrakeSafety,
McPara_BkPlcMc_CreepSpeedP:=1600,
McPara_BkPlcMc_CreepDistanceP,
McPara_BkPlcMc_BrakeDistanceP,
McPara_BkPlcMc_BrakeDeadTimeP,
McPara_BkPlcMc_CreepSpeedM,
McPara_BkPlcMc_CreepDistanceM,
McPara_BkPlcMc_BrakeDistanceM,
McPara_BkPlcMc_BrakeDeadTimeM,
McPara_BkPlcMc_AsymetricalTargeting,
McPara_BkPlcMc_LagAmp:=1700,
McPara_BkPlcMc_LagAmpAdaptLimit,
McPara_BkPlcMc_LagAmpAdaptFactor,
McPara_BkPlcMc_ZeroCompensation,
McPara_BkPlcMc_TargetClamping,
McPara_BkPlcMc_ReposDistance,
McPara_BkPlcMc_AutoBrakeDistance,
McPara_BkPlcMc_EnableControlLoopOnFault,
McPara_BkPlcMc_LagAmpDx,

```

```

McPara_BkPlcMc_LagAmpTi,
McPara_BkPlcMc_LagAmpWuLimit,
McPara_BkPlcMc_LagAmpOutLimit,
McPara_BkPlcMc_VeloAmp,
McPara_BkPlcMc_VeloAmpDx,
McPara_BkPlcMc_VeloAmpTi,
McPara_BkPlcMc_VeloAmpWuLimit,
McPara_BkPlcMc_VeloAmpOutLimit,
McPara_BkPlcMc_FeedForward,
McPara_BkPlcMc_MonPositionRange:=1800,
McPara_BkPlcMc_MonTargetRange,
McPara_BkPlcMc_MonTargetFilter,
McPara_BkPlcMc_MonPositionLagFilter,
McPara_BkPlcMc_MonDynamicLagLimit,
McPara_BkPlcMc_MonPehEnable,
McPara_BkPlcMc_MonPehTimeout,
McPara_BkPlcMc_DigInputReversed,
McPara_PFW_EnableLimitPos:=1898,
McPara_PFW_EnableLimitNeg:=1899,
McPara_BkPlcMc_JogVeloFast:=1900,
McPara_BkPlcMc_JogVeloSlow,
McPara_BkPlcMc_CustomerData:=2000,
McPara_BkPlcMc_AutoId_EnaEoT:=3000,
McPara_BkPlcMc_AutoId_EnaOvl,
McPara_BkPlcMc_AutoId_EnaZadj,
McPara_BkPlcMc_AutoId_EnaAratio,
McPara_BkPlcMc_AutoId_EnaLinTab,
McPara_BkPlcMc_AutoId_EoT_N:=3100,
McPara_BkPlcMc_AutoId_EoT_P,
McPara_BkPlcMc_AutoId_EoI_N,
McPara_BkPlcMc_AutoId_EoI_P,
McPara_BkPlcMc_AutoId_EoTlim_N,
McPara_BkPlcMc_AutoId_EoTlim_P,
McPara_BkPlcMc_AutoId_DecFactor,
McPara_BkPlcMc_AutoId_EoVlim_N,
McPara_BkPlcMc_AutoId_EoVlim_P,
McPara_BkPlcMc_AutoId_LastIdent_N,
McPara_BkPlcMc_AutoId_LastIdent_P,
McPara_BkPlcMc_AutoId_TblCount:=3150,
McPara_BkPlcMc_AutoId_TblLowEnd,
McPara_BkPlcMc_AutoId_TblHighEnd,
McPara_BkPlcMc_AutoId_TblRamp,
McPara_BkPlcMc_AutoId_TblSettling,
McPara_BkPlcMc_AutoId_TblRecovery,
McPara_BkPlcMc_AutoId_TblMinCycle,
McPara_BkPlcMc_AutoId_LinTblAvailable,
McPara_BkPlcMc_AutoId_TblValveType,
McPara_BkPlcMc_AutoId_LinTab_1:=3200,
McPara_BkPlcMc_AutoId_LinTab_2:=3400,
(* ----- *)
McRtData_BkPlcMc_ActualPosition:=10000,
McRtData_BkPlcMc_ActualAcceleration,
McRtData_BkPlcMc_PosError,
McRtData_BkPlcMc_DistanceToTarget,
McRtData_BkPlcMc_ActPressure,
McRtData_BkPlcMc_ActPressureA,
McRtData_BkPlcMc_ActPressureB,
McRtData_BkPlcMc_ActForce,
McRtData_BkPlcMc_ValvePressure,
McRtData_BkPlcMc_SupplyPressure,
McRtData_BkPlcMc_SetPosition,
McRtData_BkPlcMc_SetVelocity,
McRtData_BkPlcMc_SetAcceleration,
McRtData_BkPlcMc_SetPressure,
McRtData_BkPlcMc_SetOverride,
McRtData_BkPlcMc_LatchPosition,
McRtData_BkPlcMc_CtrlOutLag,
McRtData_BkPlcMc_CtrlOutClamping,
McRtData_BkPlcMc_CtrlOutOverlapComp,
McRtData_BkPlcMc_TargetPosition,
McRtData_BkPlcMc_NSDW:=11000,
McRtData_BkPlcMc_DCDW,
McRtData_BkPlcMc_ErrCode,
McRtData_BkPlcMc_FbState,
McRtData_BkPlcMc_CurStep,
McRtData_BkPlcMc_ParamsUnsave,
McRtData_BkPlcMc_RawPosition,
McRtData_BkPlcMc_ActPosCams,
McRtData_BkPlcMc_ReloadParams,
McRtData_BkPlcMc_EncoderMinPos,

```

```

McRtData_BkPlcMc_EncoderMaxPos,
McRtData_BkPlcMc_BufferedEntries,
(* ----- *)
(**)
McPara_BkPlcMc_
(**)
McPara_BkPlcMc_FileMarkComplete:=32767 (* Ax.Params.bLinTabAvailable AutoIdent: .. / AutoI-
dent: .. *)
);
END_TYPE

```

McPara_ActualVelocity: Die Ist-Geschwindigkeit der Achse.

McPara_BkPlcMc_AreaRatio: Das richtungsabhängige Geschwindigkeitsverhältnis.

McPara_BkPlcMc_AsymmetricalTargeting: Die Freigabe für die asymmetrische Zielfahrt.

McPara_BkPlcMc_AutoID_EnaAratio: Automatische Identifikation: Ermittlung des richtungsbezogenen Geschwindigkeits-Verhältnis.

McPara_BkPlcMc_AutoID_DecFactor: Automatische Identifikation: Faktor für die Gewichtung der Verzögerung.

McPara_BkPlcMc_AutoID_EnaEol_N: Automatische Identifikation: Ermittelte negativer Hart-Anschlag des Zylinders in Inkrementen.

McPara_BkPlcMc_AutoID_EnaEol_P: Automatische Identifikation: Ermittelte positiver Hart-Anschlag des Zylinders in Inkrementen.

McPara_BkPlcMc_AutoID_EnaEoT: Automatische Identifikation: Ermittlung der Hart-Anschläge des Zylinders.

McPara_BkPlcMc_AutoID_EnaLinTab: Automatische Identifikation: Ermittlung der Kennlinie.

McPara_BkPlcMc_AutoID_EnaOvl: Automatische Identifikation: Ermittlung der Ventil-Überdeckung.

McPara_BkPlcMc_AutoID_EnaZadj: Automatische Identifikation: Ermittlung des Offsets.

McPara_BkPlcMc_AutoID_EoTlim_N: Automatische Identifikation: Ermittelte negativer Hart-Anschlag des Zylinders.

McPara_BkPlcMc_AutoID_EoTlim_P: Automatische Identifikation: Ermittelte positiver Hart-Anschlag des Zylinders.

McPara_BkPlcMc_AutoID_EoT_N: Automatische Identifikation: Hart-Anschlag des Zylinders in negativer Richtung.

McPara_BkPlcMc_AutoID_EoT_P: Automatische Identifikation: Hart-Anschlag des Zylinders in positiver Richtung.

McPara_BkPlcMc_AutoID_EoVlim_N: Automatische Identifikation: Geschwindigkeits-Begrenzung der Kennlinien-Ermittlung in negativer Richtung.

McPara_BkPlcMc_AutoID_EoVlim_P: Automatische Identifikation: Geschwindigkeits-Begrenzung der Kennlinien-Ermittlung in positiver Richtung.

McPara_BkPlcMc_AutoID_LastIdent_N: Automatische Identifikation: Der Ausgabewert der letzten erfolgreichen Messung in negativer Richtung.

McPara_BkPlcMc_AutoID_LastIdent_P: Automatische Identifikation: Der Ausgabewert der letzten erfolgreichen Messung in positiver Richtung.

McPara_BkPlcMc_AutoID_LinTab_1: Automatische Identifikation: Punkte der Kennlinie, bezogene Geschwindigkeit.

McPara_BkPlcMc_AutoID_LinTab_2: Automatische Identifikation: Punkte der Kennlinie, bezogene Ausgabe.

McPara_BkPlcMc_AutoID_LinTbIAvailable: Dieses Signal wird am Ende einer erfolgreichen Kennlinien-Vermessung auf TRUE gesetzt.

McPara_BkPlcMc_AutoID_MinCycle: Automatische Identifikation: Minimale Mess-Strecke.

McPara_BkPlcMc_AutoID_TblCount: Automatische Identifikation: Die Anzahl der geforderten Tabellenpunkte. Da der Nullpunkt mitgezählt wird aber nur einmal vorhanden ist muss dieser Parameter immer ungerade sein. Werte von 3 bis 99 werden akzeptiert. Allerdings ist ein Wert kleiner als 11 nicht empfehlenswert.

McPara_BkPlcMc_AutoID_TblHighEnd: Automatische Identifikation: Oberes Ende der vorgesehenen Mess-Strecke.

McPara_BkPlcMc_AutoID_TblLowEnd: Automatische Identifikation: Unteres Ende der vorgesehenen Mess-Strecke.

McPara_BkPlcMc_AutoID_TblRamp: Automatische Identifikation: Rampe für das Aufbauen der Mess-Ausgabe. Die angegebene Zeit bezieht sich auf einen Wechsel von Null auf Vollaussteuerung. Kleinere Ausgabe-Änderungen werden in einem proportionalen Zeitanteil durchgeführt.

McPara_BkPlcMc_AutoID_TblRecovery: Automatische Kennlinien-Identifikation: Wartezeit bei einem Richtungswechsel.

McPara_BkPlcMc_AutoID_TblSettling: Automatische Identifikation: Wartezeit zwischen Aufbau der Mess-Ausgabe und Beginn der Messung.

McPara_BkPlcMc_AutoID_TblValveType: Automatische Identifikation: Der erwartete Typ der Kennlinie.

McPara_BkPlcMc_BrakeDeadTimeM: Die Bremsstotzeit in negativer Richtung.

McPara_BkPlcMc_BrakeDeadTimeP: Die Bremsstotzeit in positiver Richtung.

McPara_BkPlcMc_Auto_BrakeDistance: Die Freigabe für die automatische Berechnung der Bremsstrecke.

McPara_BkPlcMc_BrakeDistanceM: Bei asymmetrischer Zielfahrt: Die Bremsstrecke in negativer Richtung.

McPara_BkPlcMc_BrakeDistanceP: Die Bremsstrecke in positiver Richtung. Bei symmetrischer Zielfahrt: Die Bremsstrecke in negativer Richtung.

McPara_BkPlcMc_BrakeOff: Eine Verzögerung zwischen dem aktiven Fahren der Achse und dem Signal zum Aktivieren einer Bremse.

McPara_BkPlcMc_BrakeOn: Eine Verzögerung zwischen dem Signal zum Lösen einer Bremse und dem aktiven Fahren der Achse.

McPara_BkPlcMc_BrakeSafety: Eine Verzögerung zwischen dem aktiven Fahren der Achse in eine Richtung und einem aktiven Fahren in entgegengesetzter Richtung.

McPara_BkPlcMc_CreepDistanceM: Bei asymmetrischer Zielfahrt: Die Schleichstrecke in negativer Richtung.

McPara_BkPlcMc_CreepDistanceP: Die Schleichstrecke in positiver Richtung. Bei symmetrischer Zielfahrt: Die Schleichstrecke in negativer Richtung.

McPara_BkPlcMc_CreepSpeedM: Bei asymmetrischer Zielfahrt: Die Schleichgeschwindigkeit in negativer Richtung.

McPara_BkPlcMc_CreepSpeedP: Die Schleichgeschwindigkeit in positiver Richtung. Bei symmetrischer Zielfahrt: Die Schleichgeschwindigkeit in negativer Richtung.

McPara_BkPlcMc_CustomerData: Ein Feld mit Parametern, die eine Applikation frei verwenden kann. Diese Parameter werden mit den Achsparametern gespeichert und geladen.

McPara_BkPlcMc_CycleDevider: reserviert, nicht implementiert.

McPara_BkPlcMc_CylinderArreaA: Die Zylinderfläche der in positiver Richtung drückenden Seite des Zylinders.

McPara_BkPlcMc_CylinderArreaB: Die Zylinderfläche der in negativer Richtung drückenden Seite des Zylinders.

McPara_BkPlcMc_DigInputsReversed: Freigabe für die Invertierung der Eingangssignale einer Achse mit digitalen Positionssensoren.

McPara_BkPlcMc_DriveAbsoluteOutput: Freigabe für die Absolutwertbildung bei der Ausgabe-Anpassung.

McPara_BkPlcMc_DriveDefaultPowerOk: Drive-Power wird als verfügbar angenommen, kein Hardware Signal erforderlich.

McPara_BkPlcMc_DriveIncInterpolation: Die Zwischenteilung der Ausgabe-Anpassung.

McPara_BkPlcMc_DriveIncWeighting: Die Gewichtung der Ausgabe-Anpassung.

McPara_BkPlcMc_DriveReversed: Freigabe für die invertierte Ausgabe-Anpassung.

McPara_BkPlcMc_DriveType: Der Typ der Antriebs-Anpassung.

McPara_BkPlcMc_EmergencyRamp: Bei einem Nothalt: Die Zeit zum Abbremsen von Maximalgeschwindigkeit zum Stillstand.

McPara_BkPlcMc_EnableControlLoopOnFaults: Die Freigabe für die Lageregelung bei Achsfehlern.

McPara_BkPlcMc_EncoderBaseDistance: reserviert für abstandskodierte Encoder.

McPara_BkPlcMc_EncoderDefaultHomePosition: Achsen mit inkrementellem Wegmess-System: Ein Default-Wert für die Referenz-Fahrt.

McPara_BkPlcMc_EncoderEnableLatch: Die Freigabe für die Latchfunktion einer Encoder-Hardware.

McPara_BkPlcMc_EncoderHomingType: Achsen mit inkrementellem Wegmess-System: Die Default-Methode der Referenz-Fahrt.

McPara_BkPlcMc_EncoderIncInterpolation: Die Inkrement-Zwischenteilung der Encoder-Auswertung.

McPara_BkPlcMc_EncoderIncWeighting: Die Inkrement-Gewichtung der Encoder-Auswertung.

McPara_BkPlcMc_EncoderModuloBase: reserviert, nicht implementiert.

McPara_BkPlcMc_EncoderLatchedPosition: Die bei einem Homing gelatchte Position.

McPara_BkPlcMc_EncoderOverrunMask: Eine Maske zum Feststellen eines Encoder-Überlaufs.

McPara_BkPlcMc_EncoderPositionMask: Eine Maske zum Isolieren der gültigen Bits innerhalb der gemappten Variablen.

McPara_BkPlcMc_EncoderPotiRgToRI: Bei Potentiometer-Encodern: Das Verhältnis von Potentiometer-Gesamtwiderstand zum Lastwiderstand (Eingangswiderstand der Klemme).

McPara_BkPlcMc_EncoderRefFlag: Der Referenzier-Status der Achse.

McPara_BkPlcMc_EncoderRefIndexPositive: Achsen mit inkrementellem Wegmess-System: Die Referenz-Fahrt sucht den Index (Nocken) in positiver Richtung.

McPara_BkPlcMc_EncoderRefIndexVelo: Achsen mit inkrementellem Wegmess-System: Die R

McPara_BkPlcMc_EncoderRefShift: Achsen mit inkrementellem Wegmess-System: Die Nullpunktverschiebung der Encoder-Auswertung.

Referenz-Fahrt sucht den Index (Nocken) mit dieser Geschwindigkeit.

McPara_BkPlcMc_EncoderRefSyncPositive: Achsen mit inkrementellem Wegmess-System: Die Referenz-Fahrt sucht das Referenzier-Signal in positiver Richtung.

McPara_BkPlcMc_EncoderRefSyncVelo: Achsen mit inkrementellem Wegmess-System: Die Referenz-Fahrt sucht das Referenzier-Signal mit dieser Geschwindigkeit.

McPara_BkPlcMc_EncoderReversed: Freigabe für die invertierte Encoder-Auswertung.

McPara_BkPlcMc_EncoderType: Der Typ der Encoder-Auswertung.

McPara_BkPlcMc_EncoderZeroShift: Achsen mit absolutem Wegmess-System: Die Nullpunktverschiebung der Encoder-Auswertung.

McPara_BkPlcMc_EncoderZeroSwap: Die blockweise Verschiebung des Zählbereichs der Encoder-Auswertung.

McPara_BkPlcMc_EnvCycleTime: Die Zykluszeit der Task, in der die Kern-Bausteine (Encoder, Sollwertgenerator etc.) der Achse aufgerufen werden.

McPara_BkPlcMc_FeedForward: Die Vorsteuergewichtung der Achse.

McPara_BkPlcMc_JerkEnabled: Das Steuerwort für die Ruckbegrenzung.

McPara_BkPlcMc_JogVeloFast: Ein Default-Wert für eine schnelle Jog-Geschwindigkeit.

McPara_BkPlcMc_JogVeloSlow: Ein Default-Wert für eine langsame Jog-Geschwindigkeit.

McPara_BkPlcMc_LagAmp: Der Verstärkungsfaktor für den proportionalen Anteil im Lageregler.

McPara_BkPlcMc_LagAmpAdaptFactor: reserviert.

McPara_BkPlcMc_LagAmpAdaptLimit: reserviert.

McPara_BkPlcMc_LagAmpDx: Der Schwellwert für den integrierenden Anteil im Lageregler.

McPara_BkPlcMc_LagAmpTi: Die Zeitkonstante für den integrierenden Anteil im Lageregler.

McPara_BkPlcMc_LagAmpOutLimit: Die Ausgabe-Begrenzung für den Lageregler.

McPara_BkPlcMc_LagAmpWuLimit: Die Begrenzung (wind up limit) für den integrierenden Anteil im Lageregler.

McPara_BkPlcMc_LogLevel: Der Schwellwert für die Meldungsaufzeichnung.

McPara_BkPlcMc_MonDynamicLagLimit: Die Toleranz für die dynamische Schleppabstandsüberwachung.

McPara_BkPlcMc_MonPehEnable: Die Freigabe für die Überwachung der Fertigmeldung im Ziel.

McPara_BkPlcMc_MonPehTimeout: Die Filterzeit für die Überwachung der Fertigmeldung im Ziel.

McPara_BkPlcMc_MonPositionLagFilter: Die Filterzeit der Schleppabstandsüberwachung.

McPara_BkPlcMc_MonPositionRange: Die Toleranz für das Positionsfenster.

McPara_BkPlcMc_MonTargetFilter: Die Filterzeit für das Zielfenster.

McPara_BkPlcMc_MonTargetRange: Die Toleranz für das Zielfenster.

McPara_BkPlcMc_obsolete_XYZ: Platzhalter für nicht mehr unterstützte Parameter. Diese Parameter-Codes dürfen nicht für neue Parameter wiederverwendet werden. Um dies sicher zu stellen werden solche Zahlenwerte mit Namen dieser Form belegt.

McPara_BkPlcMc_ParamFileName: Name der Parameter-Datei.

McPara_BkPlcMc_ProfilType: Der Typ der Sollwert-Generierung.

McPara_BkPlcMc_ReposDistance: Der Schwellwert für die automatische Repositionierung.

McPara_BkPlcMc_StartRamp: Nur für bestimmte Sollwertgeneratoren: Die Beschleunigungsrampe.

McPara_BkPlcMc_StopRamp: Nur für bestimmte Sollwertgeneratoren: Die Bremsrampe.

McPara_BkPlcMc_TargetClamping: Der Default-Ausgabewert für die Klemm-Funktion.

McPara_BkPlcMc_TimeBased: Die Umschaltung der Sollwert-Generierung: Timebased oder Displacementbased.

McPara_BkPlcMc_ValveBendPointOutput: Die Ventilausgabe für die Kompensation des Kennlinienknicks.

McPara_BkPlcMc_ValveBendPointVelo: Die Geschwindigkeit für die Kompensation des Kennlinienknicks.

McPara_BkPlcMc_ValveOverlapCompM: Die Kompensation der Ventilüberdeckung für die negative Richtung.

McPara_BkPlcMc_ValveOverlapCompP: Die Kompensation der Ventilüberdeckung für die positive Richtung.

McPara_BkPlcMc_ValveResponseTime: Die Kompensation der Ventilreaktionszeit.

McPara_BkPlcMc_VeloAmp: Der Verstärkungsfaktor für den proportionalen Anteil im Geschwindigkeitsregler.

McPara_BkPlcMc_VeloAmpDx: Der Schwellwert für den integrierenden Anteil im Geschwindigkeitsregler.

McPara_BkPlcMc_VeloAmpTi: Die Zeitkonstante für den integrierenden Anteil im Geschwindigkeitsregler.

McPara_BkPlcMc_VeloAmpOutLimit: Die Ausgabe-Begrenzung für den Geschwindigkeitsregler.

McPara_BkPlcMc_VeloWuLimit: Die Begrenzung (wind up limit) für den integrierenden Anteil im Geschwindigkeitsregler.

McPara_BkPlcMc_ZeroCompensation: Die Offset-Kompensation der Ausgabe.

McPara_CommandedPosition: Die letzte kommandierte Zielposition der Achse.

McPara_CommandedVelocity: Die letzte kommandierte Geschwindigkeit der Achse.

McPara_EnableLimitNeg: Die Freigabe für den Software-Endschalter in negativer Richtung.

McPara_EnableLimitPos: Die Freigabe für den Software-Endschalter in positiver Richtung.

McPara_MaxAccelerationAppl: Die von der Applikation maximal kommandierbare Beschleunigung.

McPara_MaxAccelerationSystem: Die vom System gesetzte Obergrenze für die von der Applikation maximal kommandierbare Beschleunigung.

McPara_MaxDecelerationAppl: Die von der Applikation maximal kommandierbare Verzögerung.

McPara_MaxDecelerationSystem: Die vom System gesetzte Obergrenze für die von der Applikation maximal kommandierbare Verzögerung.

McPara_MaxJerk: Die vom System gesetzte Obergrenze für den von der Applikation maximal kommandierbaren Ruck.

McPara_MaxPositionLag: Der Schwellwert für die Schleppabstandsüberwachung.

McPara_MaxVelocityAppl: Die von der Applikation maximal kommandierbare Geschwindigkeit.

McPara_MaxVelocitySystem: Die vom System gesetzte Obergrenze für die von der Applikation maximal kommandierbare Geschwindigkeit.

McPara_PFW_Xyz: Diese Parameter sind für eine Branchenlösung reserviert.

McPara_SWLimitNeg: Der Software-Endschalter in negativer Richtung.

McPara_SWLimitPos: Der Software-Endschalter in positiver Richtung.

McRtData_BkPlcMc_ActForce: Die Ist-Kraft.

McRtData_BkPlcMc_AxName: Der textuelle Name der Achse.

McRtData_BkPlcMc_ActPosCams: Bei Achsen mit digitalen Positions-Sensoren: Die Sensor-Signale.

McRtData_BkPlcMc_ActPressure: Die Ist-Druckdifferenz am Ventil.

McRtData_BkPlcMc_ActPressureA: Der Ist-Druck in der A-Kammer des Zylinders.

McRtData_BkPlcMc_ActPressureB: Der Ist-Druck in der B-Kammer des Zylinders.

McRtData_BkPlcMc_ActualAcceleration: Die Ist-Beschleunigung.

McRtData_BkPlcMc_ActualPosition: Die Ist-Position.

McRtData_BkPlcMc_BufferedEntries: Für Achsen mit einem Kommando-Puffer: Die Anzahl der gepufferten Kommandos.

McRtData_BkPlcMc_CtrlOutOverlapComp: Der aktuell ausgegebene Anteil der Überdeckungskompensation.

McRtData_BkPlcMc_CtrlOutClamping: Der aktuelle Wert der Klemm-Ausgabe.

McRtData_BkPlcMc_CtrlOutLag: Die aktuelle Ausgabe des Lagereglers.

McRtData_BkPlcMc_CurStep: Der aktuelle (interne) Arbeitsschritt der Achse. Siehe auch E_TcMcCurrentStep.

McRtData_BkPlcMc_DCDW: Das Steuerwort der Achse mit (u.a.) den Freigaben. **HINWEIS! Es besteht kein Zusammenhang mit dem Steuerwort eines externen Geräts.**

McRtData_BkPlcMc_DistanceToTarget: Die Reststrecke zum Ziel.

McRtData_BkPlcMc_EncoderMaxPos: reserviert.

McRtData_BkPlcMc_EncoderMinPos: reserviert.

McRtData_BkPlcMc_ErrCode: Der aktuelle Fehler-Code der Achse.

McRtData_BkPlcMc_FbState: Der aktuelle (durch PLCopen definierte) Arbeitsschritt der Achse. Siehe auch E_TcMCFbState.

McRtData_BkPlcMc_FileMarkComplete: In einer Parameter-Datei: Die logische Ende-Kennung.

McRtData_BkPlcMc_LatchPosition: Die (verrechnete) Referenzposition. Dies ist die Position, an der bei einer Homing-Operation die Istposition abschließend gesetzt wurde.

McRtData_BkPlcMc_NSDW: Das Zustandswort der Achse mit (u.a.) den Betriebszuständen. **HINWEIS! Es besteht kein Zusammenhang mit dem Statuswort eines externen Geräts.**

McRtData_BkPlcMc_ParamsUnsave: Ein TRUE hier signalisiert, dass ein Parameter signifikant geändert wurde, die Parameter-Datei aber noch nicht wieder geschrieben wurde **HINWEIS! . Dieses Signal kann nicht durch die Bibliothek gegeben werden wenn der Parameter direkt (ohne Verwendung der Schreib-Bausteine) verändert wurde.**

McRtData_BkPlcMc_PosError: Der Schleppabstand.

McRtData_BkPlcMc_SetAcceleration: Der aktuelle Beschleunigungs-Sollwert.

McRtData_BkPlcMc_SetOverride: Der aktuell geltende Override-Wert.

McRtData_BkPlcMc_SetPosition: Der aktuelle Positions-Sollwert.

McRtData_BkPlcMc_SetPressure: Der Sollwert für Druck- oder Kraftregler.

McRtData_BkPlcMc_SetVelocity: Der aktuelle Geschwindigkeits-Sollwert.

McRtData_BkPlcMc_SupplyPressure: Der Istwert des Versorgungsdrucks.

McRtData_BkPlcMc_RawPosition: Die nicht durch eine Nullpunkt-Verschiebung manipulierte Ist-Position.

McRtData_BkPlcMc_ReloadParams: Bei Parameter-Änderung durch die Laufzeit: Eine Aufforderung an den PlcMcManager, die Parameter neu auszulesen.

McRtData_BkPlcMc_TargetPos: Die letzte kommandierte Zielposition der Achse. **HINWEIS! Diese Position wird nicht durch ein Stop-Kommando verändert.**

McRtData_BkPlcMc_ValvePressure: Der Druckabfall am Ventil.

3.3.10 E_TcMcProfileType (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Regeln für die Stellwertgenerierung einer Achse verwendet.

```

TYPE E_TcMcProfileType :
(
  (*
    The sequence below must not be changed!
    New types have to be added at the end.
    In case a type becomes obsolete it has to be replaced by a dummy
    to ensure the numerical meaning of the other codes.
  *)
  (*
    Die bestehende Reihenfolge darf nicht veraendert werden.
    Neue Typen muessen am Ende eingefuegt werden.
    Wenn ein Typ wegfallen sollte, muss er durch einen Dummy
    ersetzt werden, um die numerische Zuordnung zu garantieren.
  *)
  iTcMc_ProfileConstAcc,
  iTcMc_ProfileTimePosCtrl,
  iTcMc_ProfileCosine,
  iTcMc_ProfileCtrlBased,
  iTcMc_ProfileTimeRamp,
  iTcMc_ProfileJerkBased,
  iTcMc_ProfileBufferedJerk,
  iTcMc_ProfileSwitchedVelo,
  iTcMc_Profile_TestOnly:=100
);
END_TYPE

```

iTcMc_ProfileConstAcc: Nur aus Kompatibilitätsgründen vorhanden, ist durch **iTcMc_ProfileCtrlBased** abgelöst.

iTcMc_ProfileTimePosCtrl: Nur aus Kompatibilitätsgründen vorhanden, wird nicht mehr unterstützt.

iTcMc_ProfileCosine: Nur aus Kompatibilitätsgründen vorhanden, wird nicht mehr unterstützt.

iTcMc_ProfileCtrlBased: Der Stellwert für den Antrieb wird mit abschnittsweise konstanten Beschleunigungen und Verzögerungen erzeugt. Als steuernde Größe wirken die Zeit (Beschleunigung, Geschwindigkeitswechsel, Stopp) und der Weg (Positionieren).

ⓘ HINWEIS! Dieser Generatortyp kann optional auch rein zeitgesteuert mit ständig geschlossenem Lageregler arbeiten.

iTcMc_ProfileTimeRamp: Der Stellwert für den Antrieb wird mit zeitgesteuerten Rampen für Beschleunigungen und Verzögerungen erzeugt. Als steuernde Größe wirken die Zeit (Beschleunigung, Geschwindigkeitswechsel, Stopp) und der Weg (Abbremsen, anhalten).

ⓘ HINWEIS! Dieser Generatortyp ist für Achsen vorgesehen, die an Stelle eines Encoders lediglich über digitale Nocken verfügen.

iTcMc_ProfileJerkBased: Der Stellwert für den Antrieb wird mit abschnittsweise konstanten Beschleunigungen und Verzögerungen erzeugt. Dabei wird die Verzögerung zum Ziel hin mit begrenztem Ruck abgebaut. Optional kann auch der Beschleunigungsaufbau mit begrenztem Ruck erfolgen. Als steuernde Größe wirken die Zeit (Beschleunigung, Geschwindigkeitswechsel, Stopp) und der Weg (Positionieren).

ⓘ HINWEIS! Einige Funktionen werden von diesem Generatortyp nicht oder unvollständig unterstützt.

ⓘ HINWEIS! Dieser Generatortyp kann optional auch rein zeitgesteuert mit ständig geschlossenem Lageregler arbeiten.

iTcMc_ProfileBufferedJerk: reserviert.

iTcMc_ProfileSwitchedVelo: reserviert für Branchenpaket.

iTcMc_Profile_TestOnly: Dieser Typ ist nur für den internen Test von noch nicht freigegebenen Baustein-Prototypen vorgesehen. Er kann nicht über den PlcMcManager eingestellt werden.

3.3.11 E_TcMcPressureReadingMode (ab V3.0)

Die Konstanten in dieser Auflistung werden an Bausteine für die Erfassung von Kraft- oder Druckistwerten [▶ 14](#) übergeben. Sie legen fest, welcher Istwert in der ST_TcHydAxRtData [▶ 94](#) Struktur der Achse das Ergebnis der Auswertung zu aktualisieren ist.

```
TYPE E_TcMcPressureReadingMode :
(
  iTcHydPressureReadingDefault,
  iTcHydPressureReadingActPressure,
  iTcHydPressureReadingActPressureA,
  iTcHydPressureReadingActPressureB,
  iTcHydPressureReadingActForce,
  iTcHydPressureReadingSupplyPressure,
  iTcHydPressureReadingValvePressure
);
END_TYPE
```

iTcHydPressureReadingDefault: Die Zielvariable ist vom verwendeten Baustein abhängig.

iTcHydPressureReadingActPressure: Zielvariable ist **fActPressure**. Einige Bausteine aktualisieren automatisch **fActPressureA** und **fActPressureB**.

iTcHydPressureReadingActPressureA: Zielvariable ist **fActPressureA**.

iTcHydPressureReadingActPressureB: Zielvariable ist **fActPressureA**.

iTcHydPressureReadingActForce: Zielvariable ist **fActForce**. Einige Bausteine aktualisieren automatisch **fActPressure**, **fActPressureA** und **fActPressureB**.

iTcHydPressureReadingSupplyPressure: Zielvariable ist **fSupplyPressure**.

iTcHydPressureReadingValvePressure: Zielvariable ist **fValvePressure**.

3.3.12 MC_BufferMode_BkPlcMc (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Steuerung des Blendings nach PLC Open verwendet.

```
TYPE MC_BufferMode_BkPlcMc :
(
  Aborting_BkPlcMc := 0,
  Buffered_BkPlcMc,
  BlendingLow_BkPlcMc,
  BlendingPrevious_BkPlcMc,
  BlendingNext_BkPlcMc,
  BlendingHigh_BkPlcMc
);
END_TYPE
```

Aborting_BkPlcMc: Der Default Fall: Das neue Kommando wird sofort Wirksam und bricht ein bereits wirksam gewordenenes Kommando ab. Der das abgebrochene Kommando überwachende Baustein wird mit **CommandAborted** reagieren.

Buffered_BkPlcMc: Für Achsen mit Kommando-Puffer: Dieses Kommando wird automatisch gestartet, wenn alle vorherigen Kommandos vollständig abgearbeitet wurden.

BlendingLow_BkPlcMc: Für Achsen mit Kommando-Puffer: Dieses Kommando wird ohne Zwischenhalt an das vorherigen Kommando angeschlossen. Dabei wird der Übergangspunkt wenn möglich mit der niedrigeren Geschwindigkeit der beteiligten Kommandos passiert.

BlendingPrevious_BkPlcMc: Für Achsen mit Kommando-Puffer: Dieses Kommando wird ohne Zwischenhalt an das vorherigen Kommando angeschlossen. Dabei wird der Übergangspunkt wenn möglich mit der kommandierten Geschwindigkeit des vorherigen Kommandos passiert.

BlendingNext_BkPlcMc: Für Achsen mit Kommando-Puffer: Dieses Kommando wird ohne Zwischenhalt an das vorherigen Kommando angeschlossen. Dabei wird der Übergangspunkt wenn möglich mit der kommandierten Geschwindigkeit des neuen Kommandos passiert.

BlendingHigh_BkPlcMc: Für Achsen mit Kommando-Puffer: Dieses Kommando wird ohne Zwischenhalt an das vorherigen Kommando angeschlossen. Dabei wird der Übergangspunkt wenn möglich mit der höheren Geschwindigkeit der beteiligten Kommandos passiert.

3.3.13 MC_CAM_ID_BkPlcMc (ab V3.0)

(internal use only) .

```

TYPE MC_CAM_ID_BkPlcMc:
STRUCT
    stCamRef:      MC_CAM_REF_BkPlcMc;
    bValidated:    BOOL:=FALSE;
    bPeriodic:     BOOL:=FALSE;
    bMasterAbs:    BOOL:=FALSE;
    bSlaveAbs:     BOOL:=FALSE;
    bIsChanged:    BOOL:=TRUE;
END_STRUCT
END_TYPE
    
```

stCamRef: Eine Kopie der [MC_CAM_REF_BkPlcMc \[► 85\]](#) Struktur.

bValidated: Hier wird diese Struktur als gültig gekennzeichnet, wenn sie durch einen Baustein vom Typ [MC_CamTableSelect_BkPlcMc \[► 40\]](#) initialisiert wurde.

bPeriodic: Reserviert.

bMasterAbs: Hier wird festgelegt, ob die Daten der Master-Spalte absolut sind oder sich auch die Masterposition zum Zeitpunkt der Kopplung beziehen.

bSlaveAbs: Hier wird festgelegt, ob die Daten der Slave-Spalte absolut sind oder sich auch die Slaveposition zum Zeitpunkt der Kopplung beziehen.

bIsChanged: Reserviert.

3.3.14 MC_CAM_REF_BkPlcMc (ab V3.0)

(internal use only) .

```

TYPE MC_CAM_REF_BkPlcMc:
STRUCT
    pTable:        POINTER TO LREAL:=0;
    nFirstIdx:     UDINT:=1;
    nLastIdx:      UDINT:=1;
    bEquiDistant: BOOL:=FALSE;
END_STRUCT
END_TYPE
    
```

pTable: Die Adresse der Kurventabelle.

nFirstIdx: Der Index der ersten Tabellenzeile.

nLastIdx: Der Index der letzten Tabellenzeile.

bEquiDistant: Reserviert.

3.3.15 MC_Direction_BkPlcMc (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Bewegungsrichtung von Achsen benutzt.

```

TYPE MC_Direction_BkPlcMc:
(
MC_Positive_Direction_BkPlcMc := 1,
MC_Shortest_Way_BkPlcMc,
MC_Negative_Direction_BkPlcMc,
MC_Current_Direction_BkPlcMc,
MC_SwitchPositive_Direction_BkPlcMc,
MC_SwitchNegative_Direction_BkPlcMc
);
END_TYPE
    
```

MC_Positive_Direction_BkPlcMc: Die Bewegung erfolgt in Richtung steigender Positionswerte.

MC_Shortest_Way_BkPlcMc: Die Bewegungsrichtung wird so gewählt, dass sich eine möglichst kurze Weglänge ergibt.

MC_Negative_Direction_BkPlcMc: Die Bewegung erfolgt in Richtung sinkender Positionswerte.

MC_Current_Direction_BkPlcMc: Die Bewegung erfolgt in gleicher Richtung wie die zuletzt ausgeführte Bewegung.

MC_SwitchPositive_Direction_BkPlcMc: nicht unterstützt.

MC_SwitchNegative_Direction_BkPlcMc: nicht unterstützt.

3.3.16 MC_HomingMode_BkPlcMc (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Modi beim Referenzieren von Achsen benutzt.

```

TYPE MC_HomingMode_BkPlcMc:
(
  MC_DefaultHomingMode_BkPlcMc,
  MC_AbsSwitch_BkPlcMc,
  MC_LimitSwitch_BkPlcMc,
  MC_RefPulse_BkPlcMc,
  MC_Direct_BkPlcMc,
  MC_Absolute_BkPlcMc,
  MC_Block_BkPlcMc,
  MC_FlyingSwitch_BkPlcMc,
  MC_FlyingRefPulse_BkPlcMc
);
END_TYPE

```

MC_DefaultHomingMode_BkPlcMc: Es wird die in den Achsparametern festgelegte Referenziermethode verwendet.

MC_AbsSwitch_BkPlcMc: Es wird die Methode iTcMc_HomingOnIndex verwendet.

MC_LimitSwitch_BkPlcMc: nicht unterstützt.

MC_RefPulse_BkPlcMc: Es wird die Methode iTcMc_HomingOnSync verwendet.

MC_Direct_BkPlcMc: Es wird die Methode iTcMc_HomingOnExec verwendet.

MC_Absolute_BkPlcMc: nicht unterstützt.

MC_Block_BkPlcMc: Es wird die Methode iTcMc_HomingOnBlock verwendet.

MC_FlyingSwitch_BkPlcMc: nicht unterstützt.

MC_FlyingRefPulse_BkPlcMc: nicht unterstützt.

3.3.17 MC_StartMode_BkPlcMc (ab V3.0)

Die Konstanten in dieser Auflistung werden zur Kennzeichnung der Modi beim Starten von Achsen benutzt.

```

TYPE MC_StartMode_BkPlcMc:
(
  MC_StartMode_Absolute:=1,
  MC_StartMode_Relative,
  MC_StartMode_RampIn
);
END_TYPE

```

MC_StartMode_Absolute: Die von MC_CamIn_BkPlcMc Bausteinen ermittelte Slave-Sollposition wird als Absolutwert betrachtet.

MC_StartMode_Relative: Die von MC_CamIn_BkPlcMc Bausteinen ermittelte Slave-Sollposition wird als Entfernung vom Ort der Ankopplung betrachtet.

MC_StartMode_RampIn: Nicht unterstützt.

3.3.18 OUTPUT_REF_BkPlcMc (ab V3.0)

Eine Struktur dieses Typs wird an Bausteine der Typen [MC_ReadDigitalOutput_BkPlcMc\(\)](#) [► 25], [MC_WriteDigitalOutput_BkPlcMc\(\)](#) [► 35] und [MC_DigitalCamSwitch_BkPlcMc\(\)](#) [► 41] übergeben.

```
TYPE OUTPUT_REF_BkPlcMc:
STRUCT
    OutputBits: UDINT:=0;
END_STRUCT
END_TYPE
```

OutputBits: Die über diese Struktur adressierten Ausgänge.

3.3.19 ST_FunctionGeneratorFD_BkPlcMc (ab V3.0.31)

In dieser Struktur werden Parameter für die Definition der Ausgangssignale eines Funktionsgenerators zusammengefasst. Eine Struktur dieses Typs wird an [MC_FunctionGeneratorFD_BkPlcMc](#) [► 147]() Bausteine übergeben.

```
TYPE ST_FunctionGeneratorFD_BkPlcMc :
STRUCT
    Sin_Amplitude:    LREAL:=0.0;
    Sin_Phase:       LREAL:=0.0;
    Sin_Offset:      LREAL:=0.0;

    Cos_Amplitude:   LREAL:=0.0;
    Cos_Phase:       LREAL:=0.0;
    Cos_Offset:      LREAL:=0.0;

    Rect_Amplitude:  LREAL:=0.0;
    Rect_Phase:      LREAL:=0.0;
    Rect_Ratio:      LREAL:=0.5;
    Rect_Offset:     LREAL:=0.0;

    Saw_Amplitude:   LREAL:=0.0;
    Saw_Phase:       LREAL:=0.0;
    Saw_Ratio:       LREAL:=0.5;
    Saw_Offset:      LREAL:=0.0;

END_STRUCT
END_TYPE
```

Sin_Amplitude, Cos_Amplitude, Rect_Amplitude, Saw_Amplitude: Der Scheitelwert der Signale.

Sin_Phase, Cos_Phase, Rect_Phase, Saw_Phase: Die Phasenverschiebung der Signale.

Sin_Offset, Cos_Offset, Rect_Offset, Saw_Offset: Der Nullpunktversatz der Signale.

Rect_Ratio, Saw_Ratio: Das Tastverhältnis des Rechteck- bzw. des Sägezahnsignals.

3.3.20 ST_FunctionGeneratorTB_BkPlcMc (ab V3.0.31)

In dieser Struktur werden Parameter für die Zeitbasis eines oder mehrerer Funktionsgeneratoren zusammengefasst. Eine Struktur dieses Typs wird an [MC_FunctionGeneratorTB_BkPlcMc](#) [► 148](), [MC_FunctionGeneratorFD_BkPlcMc](#) [► 87]() und [MC_FunctionGeneratorSetFrq_BkPlcMc](#) [► 147]() Bausteine übergeben.

```
TYPE ST_FunctionGeneratorTB_BkPlcMc :
STRUCT
    Frequency:       LREAL:=0.000001;
    Freeze:          BOOL:=FALSE;

    CycleCount:     DINT:=0;
    CurrentTime:    LREAL:=0.0;
    CurrentRatio:   LREAL:=0.0;

END_STRUCT
END_TYPE
```

Frequency: Die Arbeitsfrequenz der von einem MC FunctionGeneratorTB BkPlcMc [► 148]() Baustein erzeugten Zeitbasis in Herz.

Freeze: Wenn diese Variable auf TRUE steht wertet ein MC FunctionGeneratorTB BkPlcMc [► 148]() Baustein die Struktur nicht aus.

CycleCount: Die Anzahl der vollständig erzeugten Signalabläufe.

CurrentTime: Die seit Beginn des aktuell erzeugten Signalablaufs vergangene Zeit.

CurrentRatio: Der normierte Fortschritt seit Beginn des aktuell erzeugten Signalablaufs.

3.3.21 ST_TcMcAutoIdent (ab V3.0.4)

In dieser Struktur werden die Parameter für einen MC AxUtiAutoIdent BkPlcMc [► 181] Baustein abgelegt. Näheres über die Bedeutung der einzelnen Elemente ist dort beschrieben.

```

TYPE ST_TcMcAutoIdent :
(* last modification: 15.05.2015 *)
STRUCT
  EndOfTravel_Negativ:      LREAL:=0.0;
  EndOfTravel_Positiv:     LREAL:=0.0;
  EndOfTravel_NegativLimit: LREAL:=0.0;
  EndOfTravel_PositivLimit: LREAL:=0.0;
  DecelerationFactor:      LREAL:=1.0;
  EndOfVelocity_NegativLimit: LREAL:=0.0;
  EndOfVelocity_PositivLimit: LREAL:=0.0;
  EndOfTravel_LastIdent_P:  LREAL:=0.0;
  EndOfTravel_LastIdent_M:  LREAL:=0.0;
  ValveCharacteristicLowEnd: LREAL:=0.0;
  ValveCharacteristicHighEnd: LREAL:=0.0;
  ValveCharacteristicRamp:  LREAL:=0.0;
  ValveCharacteristicSettling: LREAL:=0.0; (* starting with V3.0.32 *)
  ValveCharacteristicRecovery: LREAL:=0.0;
  ValveCharacteristicMinCycle: LREAL:=0.0;

  ValveCharacteristicTable: ARRAY[1..100,1..2] OF LREAL;

  EndOfIncrements_Negativ:  DINT:=0;
  EndOfIncrements_Positiv:  DINT:=0;

  ValveCharacteristicType:  INT:=0; (* starting with V3.0.33 *)
  ValveCharacteristicTblCount: INT:=0;

  EnableEndOfTravel:        BOOL:=FALSE;
  EnableOverlap:            BOOL:=FALSE;
  EnableZeroAdjust:         BOOL:=FALSE;
  EnableArreaRatio:         BOOL:=FALSE;
  EndOfTravel_PositivDone:  BOOL:=FALSE;
  EndOfTravel_NegativDone:  BOOL:=FALSE;
  EnableValveCharacteristic: BOOL:=FALSE;
  EnableNoUturn:           BOOL:=FALSE;
END_STRUCT
END_TYPE

```

3.3.22 ST_TcHydAxParam (ab V3.0)

Diese Struktur enthält sämtliche Parameter der Achse. Unter Setup [► 241] (teilweise in Vorbereitung) werden geeignete Vorgehensweisen zur Achsinbetriebnahme vorgestellt.

ⓘ HINWEIS! Die Reihenfolge der Parameter ist nicht garantiert.

```

TYPE ST_TcHydAxParam :
(* last modification: 29.03.2011 *)
STRUCT
  (* =====
  this section isn't saved / dieser Bereich wird nicht gesichert
  ===== *)
  sParamFileName:      STRING(255) := 'DefAxParmFile.dat';
  (* =====
  from this point all parameters are saved /
  von hier an werden alle Parameter gesichert
  =====

```



```

===== *)
fAcc:          LREAL := 2000.0;
fAreaRatio:    LREAL := 1.0;
fBrakeDeadTimeM: LREAL := 0.0;
fBrakeDeadTimeP: LREAL := 0.0;
fBrakeDistanceM: LREAL := 0.1;
fBrakeDistanceP: LREAL := 0.1;
fBrakeOffDelay: LREAL := 0.0;
fBrakeOnDelay:  LREAL := 0.0;
fBrakeSafetyDelay: LREAL := 0.0;
fCreepDistanceM: LREAL := 1.0;
fCreepDistanceP: LREAL := 1.0;
fCreepSpeedM:   LREAL := 80.0;
fCreepSpeedP:   LREAL := 80.0;
fCustomerData:  ARRAY [1..iTcHydfCustDataMaxIdx] OF LREAL;
fCycletime:     LREAL := 0.010;
fCylinder_ArreaA: LREAL := 1.0;
fCylinder_ArreaB: LREAL := 1.0;
fCylinder_Mass:  LREAL := 1.0;
fCylinder_Stroke: LREAL := 1.0;
fDec:           LREAL := 2000.0;
fDrive_IncInterpolation: LREAL := 1.0;
fDrive_IncWeighting:    LREAL := 0.001;
fEmergencyRamp:        LREAL := 0.1;
fEnc_BaseDistance:    LREAL := 0.001;
fEnc_DefaultHomePosition: LREAL := 0.0;
fEnc_IncInterpolation: LREAL := 1.0;
fEnc_IncWeighting:    LREAL := 0.001;
fEnc_ModuloBase:      LREAL := 0.001;
fEnc_PotiRgToRl:      LREAL := 0.0;
fEnc_RefIndexVelo:    LREAL := 0.1;
fEnc_RefSyncVelo:     LREAL := 0.1;
fEnc_ZeroShift:       LREAL := 0.0;
fJogVeloFast:         LREAL := 100.0;
fJogVeloSlow:         LREAL := 25.0;
fFeedForward:         LREAL := 1.0;
fLagAmp:              LREAL := 0.05;
fLagAmpDx:            LREAL := 0.0;
fLagAmpTi:            LREAL := 0.0;
fLagAmpOutL:          LREAL := 0.0;
fLagAmpWuL:           LREAL := 0.0;
fMaxAcc:              LREAL := 500.0;
fMaxDec:              LREAL := 500.0;
fMaxDynamicLag:       LREAL := 0.0;
fMaxJerk:             LREAL := 1000.0;
fMaxLag:              LREAL := 0.0;
fMaxLagFilter:        LREAL := 0.0;
fMaxVelo:             LREAL := 500.0;
fMonPositionRange:    LREAL := 1.0;
fMonTargetFilter:     LREAL := 1.0;
fMonTargetRange:     LREAL := 1.0;
fPEH_Timeout:         LREAL := 0.0;
fRefVelo:             LREAL := 500.0;
fReposDistance:       LREAL := 0.0;
fSoftEndMax:          LREAL := 10000.0;
fSoftEndMin:          LREAL := 0.0;
fStartAccDistance:    LREAL := 1.0;
fStartRamp:           LREAL := 1.0;
fStopRamp:            LREAL := 1.0;
fTargetClamping:      LREAL := 0.0;
fVeloAmp:             LREAL := 0.0;
fVeloAmpDx:           LREAL := 0.0;
fVeloAmpTi:           LREAL := 0.0;
fVeloAmpOutL:         LREAL := 0.0;
fVeloAmpWuL:          LREAL := 0.0;
fValve_BendPointOutput: LREAL := 0.0;
fValve_BendPointVelo: LREAL := 0.0;
fValve_OverlapCompM:  LREAL := 0.0;
fValve_OverlapCompP:  LREAL := 0.0;
fValve_ResponseTime:  LREAL := 0.0;
fZeroCompensation:    LREAL := 0.0;

nEnc_OverrunMask:    DWORD := 0;
nEnc_PositionMask:   DWORD := 0;
nEnc_ZeroSwap:       DINT  := 0;
nDigInReversed:     DINT  := 0;

nCycleDivider:      INT   := 1;
nDrive_Type:        E_TcMcDriveType:=iTcMc_Drive_Customized;
nEnc_HomingType:    E_TcMcHomingType:=iTcMc_HomingOnBlock;

```

```

nEnc_Type:          E_TcMcEncoderType:=iTcMc_EncoderSim;

nJerkEnabled:      WORD := 16#0101;
nProfileType:      E_TcMcProfileType:=iTcMc_ProfileCtrlBased;

bAsymmetricalTargeting:  BOOL := FALSE;
bDrive_AbsoluteOutput:   BOOL := FALSE;
bDrive_DefaultPowerOk:  BOOL := FALSE;
bDrive_Reversed:        BOOL := FALSE;
bEnableAutoBrakeDistance:  BOOL := FALSE;
bEnableControlLoopOnFault:  BOOL := FALSE;
bEnc_RefIndexPositive:   BOOL := FALSE;
bEnc_RefSyncPositive:   BOOL := FALSE;
bEnc_Reversed:          BOOL := FALSE;
bMaxLagEna:            BOOL := FALSE;
bPEH_Enable:           BOOL := FALSE;
bPosCtrlAccEna:        BOOL := FALSE;
bSoftEndMaxEna:        BOOL := FALSE;
bSoftEndMinEna:        BOOL := FALSE;
bTimeBased:            BOOL := FALSE;
bLinTabAvailable:      BOOL := FALSE;
(*-----*)
END_STRUCT
END_TYPE

```

bAsymmetricalTargeting: Ab V3.0.8: Wenn dieser Parameter auf TRUE steht werden richtungsabhängige Parameter bei der Zielannäherung und der Überdeckungskompensation wirksam.

bDrive_AbsoluteOutput: Wenn dieser Parameter auf TRUE steht werden Stellwerte richtungsunabhängig immer positiv ausgegeben.

bDrive_DefaultPowerOk: Ist dieser Parameter gesetzt wird die PowerOk-Rückmeldung in der [ST TcPlcDeviceInput \[► 98\]](#) Struktur der Achse ignoriert.

bDrive_Reversed: Ist dieser Parameter gesetzt wird der Stellwert negiert ausgegeben.

bEnableAutoBrakeDistance: In Vorbereitung: Wenn dieser Parameter auf TRUE steht werden **fCreepDistanceM** und **fCreepDistanceP** automatisch aus **fCreepSpeedM** bzw. **fCreepSpeedP** und **fLagAmp** errechnet.

bEnableControlLoopOnFault: In Vorbereitung: Wenn dieser Parameter auf TRUE steht wird der Stillstands-Lageregler der Achse auch im Fehlerfall aktiv. **Voraussetzung:** Seine Parameter sind dafür geeignet und die Achse ist in einem dafür geeigneten Zustand.

bEnc_RefIndexPositive: Ist dieser Parameter gesetzt wird bei einer [Referenzfahrt \[► 260\]](#) bei der Suche des Referenzindex (Nocken) ein positiver, andernfalls ein negativer Stellwert ausgegeben.

bEnc_RefSyncPositive: Ist dieser Parameter gesetzt wird bei einer [Referenzfahrt \[► 260\]](#) bei der Suche des Referenzpulses (Sync-Puls, Null-Impuls) ein positiver, andernfalls ein negativer Stellwert ausgegeben.

bEnc_Reversed: Ist dieser Parameter gesetzt wird der Positionswert negiert ausgewertet.

bLinTabAvailable: Ein TRUE hier bedeutet, dass bei der Initialisierung eine Linearisierungstabelle per Zeiger angebunden wurde, die eine erfolgreich ermittelte Kennlinie enthält.

bMaxLagEna: Dieser Parameter aktiviert die [Schlepp-Überwachung \[► 282\]](#).

bPEH_Enable: Mit diesem Parameter wird die [PEH-Überwachung \[► 282\]](#) aktiviert.

bPosCtrlAccEna: obsolet, wird in Kürze entfernt.

bSoftEndMaxEna: Dieser Parameter aktiviert den oberen [Software-Endschalter \[► 281\]](#).

bSoftEndMinEna: Dieser Parameter aktiviert den unteren [Software-Endschalter \[► 281\]](#).

bTimeBased: Wenn dieser Parameter auf TRUE steht werden Profilberechnungen zeitgesteuert ausgeführt. Der Lageregler ist ständig aktiv.

fAcc: Die absolute Beschleunigungs-Begrenzung der Achse.

fAreaRatio: Hier kann die Richtungsabhängigkeit der Geschwindigkeit kompensiert werden.

fBrakeDistance: Bis zu V3.0.7: In dieser nicht richtungsabhängigen Entfernung zum Ziel wird die aktive profilgesteuerte Stellwertgenerierung eingestellt und optional ein Stillstandslageregler oder ein anderer im Ziel geltender Mechanismus aktiviert.

fBrakeDistanceM: Ab V3.0.8: Bei **bAsymmetricalTargeting** = TRUE wird in dieser negativen Entfernung zum Ziel die aktive profilgesteuerte Stellwertgenerierung eingestellt und optional ein Stillstandslageregler oder ein anderer im Ziel geltender Mechanismus aktiviert.

fBrakeDistanceP: Ab V3.0.8: In dieser nicht richtungsabhängigen oder (bei **bAsymmetricalTargeting** = TRUE) positiven Entfernung zum Ziel wird die aktive profilgesteuerte Stellwertgenerierung eingestellt und optional ein Stillstandslageregler oder ein anderer im Ziel geltender Mechanismus aktiviert.

fBrakeDeadTime: Bis zu V3.0.7:

fBrakeDeadTimeM: Ab V3.0.8:

fBrakeDeadTimeP: Ab V3.0.8:

fBrakeOffDelay: Wenn dieser Parameter auf einen Wert größer 0 gesetzt wird hält der Stellwertgenerator zwischen der steigenden Flanke an ST_TcPlcDeviceOutput [▶ 100].bBrakeOff und dem Beginn der Beschleunigungsphase eine Wartezeit ein.

fBrakeOnDelay: Wenn dieser Parameter auf einen Wert größer 0 gesetzt wird hält der Stellwertgenerator zwischen dem Ende der aktiven Profilgenerierung und der fallenden Flanke an ST_TcPlcDeviceOutput [▶ 100].bBrakeOff eine Wartezeit ein.

fBrakeSafetyDelay: Wenn dieser Parameter auf einen Wert größer 0 gesetzt wird hält der Stellwertgenerator der fallenden Flanke an ST_TcPlcDeviceOutput [▶ 100].bBrakeOff am Ende einer aktiven Profilgenerierung und der steigenden Flanke des nächsten Fahrauftrags eine Wartezeit ein.

fCreepSpeed: Bis zu V3.0.7: Diese Geschwindigkeit wird nicht richtungsabhängig für die letzte Phase der profilgesteuerten Stellwertgenerierung verwendet.

fCreepSpeedM: Ab V3.0.8: Diese Geschwindigkeit wird bei **bAsymmetricalTargeting** = TRUE und negativer Bewegungsrichtung für die letzte Phase der profilgesteuerten Stellwertgenerierung verwendet.

fCreepSpeedP: Ab V3.0.8: Diese Geschwindigkeit wird nicht richtungsabhängig oder (bei **bAsymmetricalTargeting** = TRUE) bei positiver Bewegungsrichtung für die letzte Phase der profilgesteuerten Stellwertgenerierung verwendet.

fCreepDistance: Bis zu V3.0.7: Ab dieser nicht richtungsabhängigen Entfernung zum Ziel wird für die letzte Phase der profilgesteuerten Stellwertgenerierung **fCreepSpeed** als Stellwert verwendet.

fCreepDistanceM: Ab V3.0.8: Bei **bAsymmetricalTargeting** = TRUE wird ab dieser negativen Entfernung zum Ziel für die letzte Phase der profilgesteuerten Stellwertgenerierung **fCreepSpeedM** als Stellwert verwendet.

fCreepDistanceP: Ab V3.0.8: Ab dieser nicht richtungsabhängigen oder (bei **bAsymmetricalTargeting** = TRUE) positiven Entfernung zum Ziel wird für die letzte Phase der profilgesteuerten Stellwertgenerierung **fCreepSpeedP** als Stellwert verwendet.

fCustomerData: Hier stehen 20 LREAL Parameter zur freien Verwendung durch die Applikation zur Verfügung. Sie werden zusammen mit den anderen Achsparametern geladen und gespeichert. Bausteine der Bibliothek machen keinen selbständigen Gebrauch von diesen Parametern, können aber durch die Art des Aufrufs von der Applikation hierzu veranlasst werden.

fCycletime: Die Zykluszeit der PLC-Task, von der die Bibliotheksbausteine aufgerufen werden. Dieser Wert wird durch einen MC_AxUtiStandardInit_BkPlcMc [▶ 171]() Baustein automatisch ermittelt und darf durch die Applikation verwendet, aber **nicht verändert** werden.

fCylinder_ArreaA: Die aktive Fläche des Zylinders, die bei einer Bewegung in positiver Richtung mit Druck beaufschlagt ist.

fCylinder_ArreaB: Die aktive Fläche des Zylinders, die bei einer Bewegung in negativer Richtung mit Druck beaufschlagt ist.

fCylinder_Mass: reserviert.

fCylinder_Stroke: reserviert.

fDec: Die absolute Verzögerungs-Begrenzung der Achse.

fDrive_IncInterpolation: Dieser Parameter wird bei einigen Ausgabegeräten für die interne Umrechnung des Geschwindigkeits-Stellwerts verwendet.

fDrive_IncWeighting: Dieser Parameter wird bei einigen Ausgabegeräten für die interne Umrechnung des Geschwindigkeits-Stellwerts verwendet.

fEmergencyRamp: Dieser Parameter legt die Zeit fest die benötigt wird um von **fRefVelo** zum Stillstand zu verzögern. Er wird von verschiedenen Stellwertgeneratoren zur Reaktion auf nicht geplante Notstop-Anforderungen (wegfallende Reglerfreigabe, Fehlerfall, Bausteinaufruf) benutzt.

fEnc_BaseDistance: Dieser Parameter wird bei der Auswertung von Encodern mit abstandskodierten Nullmarken verwendet.

fEnc_DefaultHomePosition: Hier kann eine Position hinterlegt werden, die als Referenzposition an einen [MC_Home BkPlcMc \[► 51\]](#)() Baustein übergeben werden kann. Wird eine [Referenzfahrt \[► 260\]](#) durch den PlcMcManager ausgelöst wird der hier abgelegte Wert so verwendet. Soll dies auch bei einer von der PLC-Applikation ausgelösten Referenzfahrt der Fall sein ist dieser Parameter beim Aufruf des verwendeten Bausteins zu übergeben.

fEnc_IncInterpolation: Dieser Parameter legt die [Auflösung \[► 259\]](#) der Istpositionsermittlung der Achse fest.

fEnc_IncWeighting: Dieser Parameter legt die [Auflösung \[► 259\]](#) der Istpositionsermittlung der Achse fest.

fEnc_PotiRgToRI: Wird von einigen Bausteinen zur Linearisierung von einfachen durch den Eingangswiderstand der interface-Elektronik belasteten Potentiometer-Weggebern verwendet.

fEnc_RefIndexVelo: Dieser Parameter legt den Stellwert fest, der bei einer [Referenzfahrt \[► 260\]](#) während der Suche des Referenzindex (Nocken) ausgegeben wird.

fEnc_RefSyncVelo: Dieser Parameter legt den Stellwert fest, der bei einer [Referenzfahrt \[► 260\]](#) während der Suche des Referenzpulses (Sync-Puls, Null-Impuls) ausgegeben wird.

fEnc_ZeroShift: Dieser Parameter verschiebt den [Nullpunkt \[► 260\]](#) der Istpositionsermittlung der Achse.

fJogVeloFast: Sollgeschwindigkeit für die schnelle Handfahrt.

fJogVeloSlow: Sollgeschwindigkeit für die langsame Handfahrt.

fLagAmp: Die Kp-Verstärkung des Stillstands-Lagereglers.

fLagAmpDx: In Vorbereitung: Das Ansprechfenster des Stillstands-Lagereglers.

fLagAmpTi: In Vorbereitung: Die Integrationszeit des Stillstands-Lagereglers.

fLagAmpOutL: In Vorbereitung: Die Ausgabebegrenzung Stillstands-Lagereglers.

fLagAmpWuL: In Vorbereitung: Die Begrenzung des I-Anteils Stillstands-Lagereglers.

fMaxAcc: Die für die Bausteine geltende Beschleunigungs-Begrenzung der Achse. Dieser Wert ist auf **fAcc** begrenzt.

fMaxDec: Die für die Bausteine geltende Verzögerungs-Begrenzung der Achse. Dieser Wert ist auf **fDec** begrenzt.

fMaxDynamicLag: Dieser Parameter legt einen der Grenzwerte für die [Schlepp-Überwachung \[► 282\]](#) fest.

fMaxJerk: Die für die Bausteine geltende Ruckbegrenzung der Achse. Dieser Wert wird verwendet wenn als Profiltyp **iTcMc_ProfileJerkBased** eingestellt ist.

fMaxLag: Dieser Parameter legt einen der Grenzwerte für die [Schlepp-Überwachung \[► 282\]](#) fest.

fMaxLagFilter: Dieser Parameter legt einen der Grenzwerte für die [Schlepp-Überwachung \[► 282\]](#) fest.

fMaxVelo: Die maximale durch Bausteine verwendbare Geschwindigkeit. Wird von einem Baustein ein höherer Wert verwendet wird dieser in der Regel ohne eine Fehlermeldung entsprechend begrenzt. [□](#)

HINWEIS! Dieser Parameter wird auf fRefVelo begrenzt.

fMonPositionRange: Dieser Parameter wird für die Zielfenster-Überwachung [\[▶ 281\]](#) verwendet.

fMonTargetFilter: Dieser Parameter wird für die Zielfenster-Überwachung [\[▶ 281\]](#) verwendet.

fMonTargetRange: Dieser Parameter wird für die Zielfenster-Überwachung [\[▶ 281\]](#) verwendet.

fPEH_Timeout: Dieser Parameter legt den Grenzwert für die PEH-Überwachung [\[▶ 282\]](#) fest.

fRefVelo: Dieser Parameter legt die maximale absolute Geschwindigkeit der Achse fest.

fReposDistance: Wenn dieser Parameter größer als 0 ist und die Achse ein einmal angefahrenes Ziel um mehr als diese Strecke verlassen hat wird automatisch eine erneute Positionierung auf das Ziel veranlasst.

fSoftEndMax: Der obere (positive) Software-Endschalter [\[▶ 281\]](#).

fSoftEndMin: Der untere (negative) Software-Endschalter [\[▶ 281\]](#).

fStartAccDistance: *obsolet*, wird in Kürze entfernt.

fStartRamp: Dieser Parameter legt die Zeit fest die bei Profiltyp **iTcMc_ProfileTimeRamp** benötigt wird um auf **fRefVelo** zu beschleunigen.

fStopRamp: Dieser Parameter legt die Zeit fest die benötigt wird um von **fRefVelo** zum Stillstand zu verzögern. Er wird bei Profiltyp **iTcMc_ProfileTimeRamp** für die Zielfahrt verwendet, aber auch von verschiedenen Stellwertgeneratoren zur Reaktion auf nicht geplante Stop-Anforderungen (wegfallende Vorschubfreigabe, Fehlerfall, Bausteinaufruf) benutzt.

fTargetClamping: Ist dieser Parameter auf einen Wert größer als Null gesetzt wird beim Erreichen eines Fahrziels dieser Stellwert vorzeichenrichtig ausgegeben. Eine Lageregelung wird unterdrückt.

fValve_BendPointOutput: Bei Ventilen mit Kennlinienknick kann dieser Parameter für eine einfache Linearisierung genutzt werden.

fValve_BendPointVelo: Bei Ventilen mit Kennlinienknick kann dieser Parameter für eine einfache Linearisierung genutzt werden.

fValve_OverlapComp: Bis zu V3.0.7: Kompensation einer nicht richtungsabhängigen Ventilüberdeckung.

fValve_OverlapCompM: Ab V3.0.8: Kompensation einer (bei **bAsymmetricalTargeting** = TRUE) für die negative Richtung verwendete Ventilüberdeckung.

fValve_OverlapCompP: Ab V3.0.8: Kompensation einer nicht richtungsabhängigen oder (bei **bAsymmetricalTargeting** = TRUE) für die positive Richtung verwendete Ventilüberdeckung.

fValve_ResponseTime: Mit diesem Parameter kann eine Totzeitkompensation des Stellglieds vorgenommen werden.

fVeloAmp: In Vorbereitung: Die Kp-Verstärkung des unterlagerten Geschwindigkeitsreglers.

fVeloAmpDx: In Vorbereitung: Das Ansprechfenster des unterlagerten Geschwindigkeitsreglers.

fVeloAmpTi: In Vorbereitung: Die Integrationszeit des unterlagerten Geschwindigkeitsreglers.

fVeloAmpOutL: In Vorbereitung: Die Ausgabebegrenzung des unterlagerten Geschwindigkeitsreglers.

fVeloAmpWuL: In Vorbereitung: Die Begrenzung des I-Anteils des unterlagerten Geschwindigkeitsreglers.

fZeroCompensation: Hier kann ein Analog-Offset der Geschwindigkeitsausgabe kompensiert werden.

nCycleDivider: reserviert.

nDrive_Type: Hier wird der Antriebstyp [\[▶ 67\]](#) festgelegt.

nEnc_Type: Hier wird der Encodertyp [\[▶ 70\]](#) festgelegt.

nEnc_HomingType: Hier wird die Referenziermethode festgelegt, die ein [MC Home BkPlcMc \[► 51\]](#)() Baustein verwendet wenn [MC_DefaultHomingMode BkPlcMc \[► 86\]](#) als **HomingMode** übergeben wird.

nEnc_ZeroSwap: reserviert.

nJerkEnabled: Diese Bitmaske legt fest, an welchen Übergängen im Profil eine Ruckbegrenzung zu erfolgen hat. Dieser Wert wird verwendet wenn als Profiltyp **iTcMc_ProfileJerkBased** eingestellt ist.

nProfileType: Hier wird der [Stellwertgenerator \[► 83\]](#) festgelegt.

sParamFileName: Unter diesem Dateinamen werden die Achsparameter als DAT-Datei gespeichert.

Hinweise zur Achsinbetriebnahme finden Sie unter [Setup \[► 241\]](#).

3.3.23 ST_TcHydAxRtData (ab V3.0)

Die Variablen in dieser Struktur geben den Laufzeit-Zustand der Achse wieder.

[i] HINWEIS! Die Reihenfolge der Daten ist nicht garantiert.

```

TYPE ST_TcHydAxRtData :
(* last modification: 30.03.2015 *)
STRUCT
(*-----*)
fActForce:           LREAL := 0.0;
fActiveOverlap:     LREAL := 0.0;
fActPos:             LREAL := 0.0;
fActPosDelta:       LREAL := 0.0;
fActPosOffset:      LREAL := 0.0;
fActPressure:        LREAL := 0.0;
fActPressureA:       LREAL := 0.0;
fActPressureB:       LREAL := 0.0;
fActVelo:            LREAL := 0.0;
fBrakeOffTimer:      LREAL := 0.0;
fBrakeOnTimer:       LREAL := 0.0;
fBrakeSafetyTimer:   LREAL := 0.0;
fClampingOutput:     LREAL := 0.0;
fDestAcc:            LREAL := 0.0;
fDestCreepDistanceM: LREAL := 0.0;
fDestCreepDistanceP: LREAL := 0.0;
fDestCreepSpeedM:    LREAL := 0.0;
fDestCreepSpeedP:    LREAL := 0.0;
fDestDec:            LREAL := 0.0;
fDestJerk:           LREAL := 0.0;
fDestPos:            LREAL := 0.0;
fDestRampEnd:        LREAL := 0.0;
fDestSpeed:          LREAL := 0.0;
fDistanceToTarget:   LREAL := 0.0;
fEnc_RefShift:       LREAL := 0.0;
fEnc_ZeroSwap:       LREAL := 0.0;
fGearActive:         LREAL := 0.0;
fGearSetting:        LREAL := 0.0;
fLagCtrlOutput:      LREAL := 0.0;
fLatchedPos:         LREAL := 0.0;
fOilRequirred_A:     LREAL := 0.0;
fOilRequirred_B:     LREAL := 0.0;
fOilUsed_A:          LREAL := 0.0;
fOilUsed_B:          LREAL := 0.0;
fOutput:             LREAL := 0.0;
fOverride:           LREAL := 1.0;
fParamAccTime:       LREAL := 0.0;
fPosError:           LREAL := 0.0;
fSetAcc:             LREAL := 0.0;
fSetPos:             LREAL := 0.0;
fSetPressure:        LREAL := 0.0;
fSetSpeed:           LREAL := 0.0;
fSetSpeedOld:        LREAL := 0.0;
fSetVelo:            LREAL := 0.0;
fStartPos:           LREAL := 0.0;
fStartRamp:          LREAL := 0.0;
fStartRampAnchor:    LREAL := 0.0;
fSupplyPressure:     LREAL := 0.0;
fTargetPos:          LREAL := 0.0;
fTimerPEH:           LREAL := 0.0;

```

```

fTimerTPM:          LREAL := 0.0;
fValvePressure:    LREAL := 0.0;
fVeloError:        LREAL := 0.0;
(*-----*)
nAxisState:        DWORD := 0;
nCalibrationState: DWORD := 0;
nDeCtrlDWord:     DWORD := 0;
nErrorCode:        DWORD := 0;
nStateDWord:      DWORD := 0;
udiAmpErrorCode:   UDINT;
(*-----*)
iCurrentStep:      E_TcMcCurrentStep;
wEncErrMask:       WORD:=0;
wEncErrMaskInv:    WORD:=0;
nDrvWcCount:       INT:=0;
(**)
nEncWcCount:       INT:=0;
nDrvDeviceState:   UINT:=0;
nEncDeviceState:   INT:=0;
(*-----*)
bActPosCams:       BYTE := 0;
bBrakeOff:         BOOL := FALSE;
bBrakeOffInverted: BOOL := FALSE;
bControllable:     BOOL := FALSE;
bCountedCycles:    BYTE := 1;
bCycleCounter:     BYTE := 0;
bDriveResponse:    BOOL := FALSE;
bEncDoLatch:       BOOL := FALSE;
(**)
bEncoderResponse:  BOOL := FALSE;
bEncLatchValid:   BOOL := FALSE;
bLocked_Estop:     BOOL := FALSE;
bParamsUnsave:    BOOL := FALSE;
bReloadParams:     BOOL := FALSE;
bTargeting:        BOOL := FALSE;
bUnalignedOverlap: BOOL := FALSE;
bActPosOffsetEnable: BOOL := FALSE; (* starting with 09.03.2015 *)
(**)
bDriveStartup:     BOOL := FALSE;
bEncAlignRefShift: BOOL := FALSE;
bDrvWcsError:      BOOL := FALSE;
bEncWcsError:      BOOL := FALSE;
bFirstWcs:         BOOL := FALSE;
bChangeCount:      BYTE := 0;
bStartAutoIdent:   BOOL := FALSE;
bParamFileComplete: BOOL := FALSE;
(*-----*)
pMasterRtData:     POINTER TO BYTE;
pMasterParam:      POINTER TO BYTE;
(*-----*)
udiSercDeviceID:   UDINT := 0;
uiSercBoxAddr:     UINT := 0;
uiSercPort:        UINT := 0;
(*-----*)
stPosCtrlr:        stbkplcinternal_cplxctrl;
stVeloCtrlr:       stbkplcinternal_cplxctrl;
(*-----*)
sTopBlockName:     STRING(83) := '';
(*-----*)
END_STRUCT
END_TYPE

```

bActPosOffsetEnable: Durch ein TRUE in dieser Variablen wird die Istwert-Beeinflussung aktiviert. Siehe auch unter **fActPosOffset**.

bBrakeOff: Das Steuersignal für eine externe Bremse. Eine Ausgabe-Variable der Profilingeneratoren.

bBrakeOffInverted: Das invertierte **bBrakeOff** Signal.

bChangeCount: Dieser Wert wird bei jeder Parameteränderung erhöht.

bEncAlignRefShift: reserviert.

bEncDoLatch: Mit diesem Signal kommunizieren der [MC Home BkPlcMc \[► 51\]](#) und der [MC_AxRtEncoder_BkPlcMc \[► 123\]](#) Baustein der Achse während einer Referenzfahrt.

bEncLatchValid: Mit diesem Signal kommunizieren der MC Home BkPlcMc [► 51] und der MC AxRtEncoder BkPlcMc [► 123] Baustein der Achse während einer Referenzfahrt.

bLocked_Estop: Durch ein TRUE in dieser Variablen werden die Stellwertgeneratoren daran gehindert, den Zustand iTcHydStateEmergencyBreak / McState_Errorstop zu verlassen, obwohl die Ausgaben an den Antrieb auf 0 abgebaut werden. Verwendet von MC EmergencyStop BkPlcMc [► 44] und MC ImmediateStop BkPlcMc [► 53].

bParamFileComplete:

bParamsUnsave: Die Bausteine MC WriteParameter BkPlcMc [► 36] und MC WriteBoolParameter BkPlcMc [► 34] setzen dieses Flag wenn sie den Wert eines Parameters verändern. Ein MC AxParamSave BkPlcMc [► 193] Baustein löscht das Flag beim erfolgreichen Speichern der Parameter. Im Online Modus des PlcMcManagers [► 239] wird dieses Flag für die Status-Anzeige genutzt.

bStartAutolident:

bUnalignedOverlap:

fActForce: Istkraft des Zylinders. Dieser Wert wird üblicherweise durch einen Erfassungsbaustein-Baustein für die Erfassung von Kraft- oder Druckistwerten [► 14] ermittelt.

fActiveOverlap: Die aktuelle Ausgabe der Überdeckungs-Kompensation. Eine Ausgabe-Variable der Profilgeneratoren.

fActPos: Die aktuelle Istposition der Achse. Dieser Wert wird üblicherweise durch einen Encoder-Baustein ermittelt.

fActPosOffset: Der Offset für die Istwert-Beeinflussung. Wenn **bActPosOffsetEnable** auf TRUE steht wird dieser Offset in die **fActPos** addiert. Wenn sich **fActPosOffset** ändert wird dadurch die **fActVelo** nicht beeinflusst.

Durch **bActPosOffsetEnable:=TRUE** wird **fActPosOffset** unmittelbar und ohne Rampe wirksam.

! **HINWEIS! Ist die Istwert-Beeinflussung während eines Homings aktiv wird fActPosOffset beim Setzen der Ist-Position berücksichtigt.**

Beispiel: Bei einer Referenzposition von 100.0 und einem Offset von 1.0 wird die Istposition am Ort des Null-Impulses auf 101.0 gesetzt. Wenn anschließend die Beeinflussung deaktiviert oder auf 0.0 gesetzt wird zeigt die Istposition am Ort des Null-Impulses den Wert 100.0, wie sie es auch beim Homing ohne Beeinflussung gezeigt hätte.

! **HINWEIS! Diese Funktion ist nur für die folgenden Encoder-Typen realisiert:**

iTcMc_EncoderCoE_DS406, iTcMc_EncoderEL3255, iTcMc_EncoderSim, iTcMc_EncoderEL5101, iTcMc_EncoderKL5101, iTcMc_EncoderKL5111, iTcMc_EncoderEL5001, iTcMc_EncoderKL5001, iTcMc_EncoderKL3002, iTcMc_EncoderEL3102, iTcMc_EncoderKL3042, iTcMc_EncoderKL3062, iTcMc_EncoderEL3142, iTcMc_EncoderEM8908_A, iTcMc_EncoderEL3162, iTcMc_EncoderKL3162.

! **HINWEIS! Ist für ein zu einem dieser Typen kompatibles E/A-Gerät einer der aufgeführten Typen eingestellt wird die beschriebene Funktion ebenfalls realisiert.**

bActPosCams: Die aktuellen Positionsnocken der Achse. Dieser Wert wird nur verwendet, wenn als Encoder-Typ iTcMc_EncoderDigCam eingestellt ist.

fActPosDelta: Die Änderung der Istposition relativ zum vorangehenden Zyklus.

fActPressure: Istdruck im Zylinder. Dieser Wert wird üblicherweise durch einen Erfassungsbaustein-Baustein für die Erfassung von Kraft- oder Druckistwerten [► 14] ermittelt.

fActPressureA: Istdruck auf der A-Seite des Zylinders. Dieser Wert wird üblicherweise durch einen Erfassungsbaustein-Baustein für die Erfassung von Kraft- oder Druckistwerten [► 14] ermittelt.

fActPressureB: Istdruck auf der B-Seite des Zylinders. Dieser Wert wird üblicherweise durch einen Erfassungsbaustein-Baustein für die Erfassung von Kraft- oder Druckistwerten [► 14] ermittelt.

- fActVelo:** Die aktuelle Istgeschwindigkeit der Achse. Dieser Wert wird üblicherweise durch einen Encoder-Baustein ermittelt.
- nAxisState:** Der Bewegungszustand der Achse.
- fClampingOutput:** Eine Ausgabe-Variable der Profilgeneratoren.
- fDestAcc:** Die vom aktuellen oder dem zuletzt abgearbeiteten Bewegungsauftrag vorgegebene Beschleunigung.
- fDestCreepDistance:** Bis V3.0.7: Die Schleichstrecke.
- fDestCreepSpeed:** Bis V3.0.7: Die Schleichgeschwindigkeit.
- fDestCreepDistanceP:** Ab V3.0.8: Die Schleichstrecke in positiver Richtung.
- fDestCreepSpeedP:** Ab V3.0.8: Die Schleichgeschwindigkeit in positiver Richtung.
- fDestCreepDistanceM:** Ab V3.0.8: Die Schleichstrecke in negativer Richtung.
- fDestCreepSpeedM:** Ab V3.0.8: Die Schleichgeschwindigkeit in negativer Richtung.
- fDestDec:** Die vom aktuellen oder dem zuletzt abgearbeiteten Bewegungsauftrag vorgegebene Verzögerung.
- fDestJerk:** Der vom aktuellen oder dem zuletzt abgearbeiteten Bewegungsauftrag vorgegebene Ruck.
- fDestPos:** Die aktuell wirksame Zielposition.
- fDestSpeed:** Die vom aktuellen oder dem zuletzt abgearbeiteten Bewegungsauftrag vorgegebene Geschwindigkeit.
- fDistanceToTarget:** Der aktuelle Restweg der Achse. Dieser Wert wird üblicherweise durch einen Generator-Baustein ermittelt.
- fEnc_RefShift:** Der Offset zwischen dem umgerechneten (eventuell intern verlängerten) Zählerstand einer Inkrementalencoder-Eingangsklemme und der Istposition der Achse. Dieser Offset wird durch eine Referenzfahrt z.B. mit einem [MC Home BkPlcMc \[► 51\]](#) Baustein ermittelt oder mit einem [MC SetPosition BkPlcMc \[► 32\]](#) Baustein manipuliert.
- fLatchedPos:** Die Position (unter Berücksichtigung von geltenden Offsets) an der ein Homing erfolgte oder die Komponenten der Istwerterfassung (Encoder, E/A-Elektronik) eingeschaltet wurden.
- fLagCtrlOutput:** Die normierte Ausgabe des Lagereglers. Eine Ausgabe-Variable der Profilgeneratoren.
- fOilRequired_A:** Der unter Berücksichtigung der Sollgeschwindigkeit berechnete Ölbedarf der A-Seite.
- fOilRequired_B:** Der unter Berücksichtigung der Sollgeschwindigkeit berechnete Ölbedarf der B-Seite.
- fOilUsed_A:** Der unter Berücksichtigung der Istgeschwindigkeit berechnete Ölverbrauch der A-Seite.
- fOilUsed_B:** Der unter Berücksichtigung der Istgeschwindigkeit berechnete Ölverbrauch der B-Seite.
- fOutput:** Die auszugebende Stellgröße. Über diese Variable kommunizieren der [MC AxRtFinish BkPlcMc \[► 164\]](#) und der [MC AxRtDrive BkPlcMc \[► 114\]](#) Baustein.
- fOverride:** Der aktuelle Achs-Geschwindigkeitsoverride.
- fPosError:** Der aktuelle Lagefehler der Achse.
- fSetPos:** Der aktuelle Lagesollwert der Achse.
- fSetSpeed:** Die normierte Sollgeschwindigkeit der Achse. Eine Ausgabe-Variable der Profilgeneratoren.
- fSetAcc:** Der aktuelle Beschleunigungsstellwert. Eine Ausgabe-Variable der Profilgeneratoren.
- fSetPressure:** Hier ist der Sollwert einer optionalen Druck- oder Kraftregelung abzulegen.
- fStartPos:** Die Startposition des aktuellen oder des zuletzt abgearbeiteten Bewegungsauftrag.

fSupplyPressure: Versorgungsdruck. Dieser Wert wird üblicherweise durch einen Erfassungsbaustein-Baustein für die Erfassung von Kraft- oder Druckistwerten [► 14] ermittelt.

fTargetPos: Die vom aktuellen oder dem zuletzt abgearbeiteten Bewegungsauftrag vorgegebene Zielposition.

fValvePressure: Druckabfall am Ventil. Dieser Wert wird üblicherweise durch einen Erfassungsbaustein-Baustein für die Erfassung von Kraft- oder Druckistwerten [► 14] ermittelt.

iCurrentStep: Der interne Zustand der Stellwertgeneratoren. Werte aus E_TcMcCurrentStep [► 65].

nCalibrationState: Der aktuelle Zustand einer Referenzfahrt.

nDeCtrlDWord: Die Kontroll-Signale [► 223] der Achse.

nErrorCode: Der aktuelle ErrorCode [► 224] der Achse.

nStateDWord: Die Status-Signale [► 223] der Achse.

sTopBlockName: Die meisten von der Applikation direkt aufgerufenen Bausteine der Bibliothek tragen hier eine Debug-Kennung ein.

wEncErrMask:

wEncErrMaskInv:

Alle anderen Elemente dieser Struktur sind für eine interne Verwendung reserviert. Sie sind nicht garantiert und dürfen nicht von der Applikation verwendet oder verändert werden.

3.3.24 ST_TcMcAuxDataLabels (ab V3.0)

In dieser Struktur werden die Beschriftungstexte der kundenspezifischen Achsparameter abgelegt. Eine Struktur dieses Typs kann durch einen MC_AxUtiStandardInit_BkPlcMc [► 171] Baustein durch einen Pointer in der Axis_Ref_BkPlcMc [► 62] Struktur mit der Achse verbunden werden.

```

TYPE ST_TcMcAuxDataLabels :
STRUCT
    stLabel:          ARRAY [1..20] OF STRING(20);
END_STRUCT
END_TYPE

```

stLabel: Die Beschriftungstexte.

3.3.25 ST_TcPlcDeviceInput (ab V3.0)

Diese Struktur enthält die Eingangsabbild-Variablen einer Achse.

```

TYPE ST_TcPlcDeviceInput :
STRUCT
    uiCount:          UINT:=0;
    uiLatch:          UINT:=0;
    usiStatus:        USINT:=0;

    uiPZDL_RegDaten:  UINT:=0;
    uiPZDH:           UINT:=0;
    usiRegStatus:     USINT:=0;

    udiCount:         UDINT:=0;
    uiStatus:         UINT:=0;

    bTerminalState:   BYTE:=0;
    uiTerminalData:   WORD:=0;
    uiTerminalState2: WORD:=0;

    bDigInA:          BOOL:=FALSE;
    bDigInB:          BOOL:=FALSE;

    bDigCamMM:        BOOL:=FALSE;
    bDigCamM:         BOOL:=FALSE;

```

```

bDigCamP:      BOOL:=FALSE;
bDigCamPP:     BOOL:=FALSE;

DriveError:    UDINT:=0;
ActualPos:     ARRAY [0..1] OF UINT:=0;
DriveState:    ARRAY [0..3] OF BYTE:=0;

S_iReserve:    INT:=0;
S_DiReserve:   ARRAY [1..9] OF DINT:=0;

CiA_Reserve:   ARRAY [1..8] OF UINT:=0;

bPowerOk:     BOOL:=FALSE;
bEnAck:       BOOL:=FALSE;

wDriveDevState: WORD:=0;
wDriveWcState: BYTE:=0;
wEncDevState:  WORD:=0;
wEncWcState:   BYTE:=0;
uiDriveBoxState: UINT:=0;
uiEncBoxState: UINT:=0;

sEncAdsAddr:   ST_TcPlcAdsAddr;
nEncAdsChannel: BYTE:=0;
sDrvAdsAddr:   ST_TcPlcAdsAddr;
nDrvAdsChannel: BYTE:=0;

nReserve:     ARRAY [1..20] OF BYTE;
END_STRUCT
END_TYPE

```

uiCount: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderEL3102, iTcMc_EncoderEL3142, iTcMc_EncoderEL5101, iTcMc_EncoderKL2521, iTcMc_EncoderKL2531, iTcMc_EncoderKL2541, iTcMc_EncoderKL3002, iTcMc_EncoderKL3042, iTcMc_EncoderKL3062, iTcMc_EncoderKL3162, iTcMc_EncoderKL5101, iTcMc_EncoderKL5111, iTcMc_EncoderM2510, iTcMc_EncoderM3120, iTcMc_DriveKL2531, iTcMc_DriveKL2541.

uiLatch: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderEL5101, iTcMc_EncoderKL5101, iTcMc_EncoderKL5111.

usiStatus: Verwendet für Gerätezustandsinformation. Benutzt bei iTcMc_EncoderEL5101, iTcMc_EncoderKL3002, iTcMc_EncoderKL3042, iTcMc_EncoderKL3062, iTcMc_EncoderKL3162, iTcMc_EncoderKL5101, iTcMc_EncoderKL5111, iTcMc_EncoderM3120.

uiPZDL_RegDaten: Verwendet für Positionserfassung und Parameter-Kommunikation. Benutzt bei iTcMc_EncoderKL5001.

uiPZDH: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderKL5001.

usiRegStatus: Verwendet für Gerätezustandsinformation. Benutzt bei iTcMc_EncoderEL5001, iTcMc_EncoderKL5001.

udiCount: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderEL5001.

uiStatus: Verwendet für Gerätezustandsinformation. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110.

bTerminalState: Verwendet für Parameter-Kommunikation. Benutzt bei iTcMc_EncoderKL2521, iTcMc_EncoderKL2531, iTcMc_EncoderKL2541, iTcMc_DriveEL4132, iTcMc_DriveKL2521, iTcMc_DriveKL2531, iTcMc_DriveKL2541, iTcMc_DriveKL4032.

uiTerminalData: reserviert.

uiTerminalState2: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderKL2541.

bDigInA: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderDigIncrement.

bDigInB: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderDigIncrement.

bDigCamMM: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderDigCam.

bDigCamM: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderDigCam.

bDigCamP: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderDigCam.

bDigCamPP: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderDigCam.

DriveError: Verwendet für Gerätezustandsinformation. Benutzt bei iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.

ActualPos: Verwendet für Positionserfassung. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.

DriveState: Verwendet für Gerätezustandsinformation. Benutzt bei iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.

nReserve: reserviert.

S_iReserve: reserviert.

S_DiReserve: reserviert.

CiA_Reserve: reserviert.

bPowerOk: Optional verwendet für die Überwachung eines Netz-Schützes. Benutzt bei iTcMc_DriveAx2000_B110, iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.

bEnAck: reserviert.

wDriveDevState: reserviert.

wDriveWcState: Verwendet für die Überwachung der Verbindung zum Steller. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110.

wEncDevState: reserviert.

wEncWcState: Verwendet für die Überwachung der Verbindung zum Encoder. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110, iTcMc_EncoderEL3102, iTcMc_EncoderEL3142, iTcMc_EncoderEL5001, iTcMc_EncoderEL5101.

uiDriveBoxState: Verwendet für die Überwachung der Verbindung zum Steller. Benutzt bei iTcMc_DriveAx2000_B200, iTcMc_DriveAx2000_B900.

uiEncBoxState: Verwendet für die Überwachung der Verbindung zum Encoder. Benutzt bei iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.

sEncAdsAddr: Verwendet für Parameter-Kommunikation. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110, iTcMc_EncoderEL3102, iTcMc_EncoderEL3142, iTcMc_EncoderEL5001, iTcMc_EncoderEL5101.

nEncAdsChannel: Verwendet für Parameter-Kommunikation. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110.

sDrvAdsAddr: Verwendet für Parameter-Kommunikation. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110.

nDrvAdsChannel: Verwendet für Parameter-Kommunikation. Benutzt bei iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110.

nReserve: reserviert.

3.3.26 ST_TcPlcDeviceOutput (ab V3.0)

Diese Struktur enthält die Ausgangsabbild-Variablen einer Achse.

```
TYPE ST_TcPlcDeviceOutput :
STRUCT
```

```
    nDacOut:          INT:=0;
    bDigOutAp:        BOOL:=FALSE;
    bDigOutAn:        BOOL:=FALSE;
    bDigOutBp:        BOOL:=FALSE;
    bDigOutBn:        BOOL:=FALSE;
```

```

    uiCount:          UINT:=0;
    uiDacOutA:        UINT:=0;
    uiDacOutB:        UINT:=0;
    bMovePos:         BOOL:=FALSE;
    bMoveNeg:         BOOL:=FALSE;
    bBrakeOff:        BOOL:=FALSE;
    bBrakeOffInverted:BOOL:=FALSE;
    DriveCtrl:        ARRAY [0..3] OF BYTE:=0;
    NominalVelo:      DINT:=0;
    uiDriveCtrl:      UINT:=0;
    S_iReserve:       ARRAY [1..2] OF INT:=0;
    S_DiReserve:      ARRAY [1..7] OF DINT:=0;
    CiA_Reserve:      ARRAY [1..7] OF UINT:=0;
    bPowerOn:         BOOL:=FALSE;
    bEnable:          BOOL:=FALSE;
    bEnablePos:       BOOL:=FALSE;
    bEnableNeg:       BOOL:=FALSE;
    nResetState:      BYTE:=0;
    usiCtrl:          USINT:=0;
    uiTerminalData:   WORD:=0;
    bTerminalCtrl:    BYTE:=0;
    uiTerminalCtrl2:  WORD:=0;
    nReserve:         ARRAY [1..20] OF BYTE;
END_STRUCT
END_TYPE

```

nDacOut: Verwendet für Stellwertausgabe oder Parameter-Kommunikation. Benutzt bei **iTcMc_EncoderKL2531, iTcMc_EncoderKL2541, iTcMc_DriveEL4132, iTcMc_DriveKL2521, iTcMc_DriveKL2531, iTcMc_DriveKL2541, iTcMc_DriveKL4032, iTcMc_DriveM2400_Dn.**

bDigOutAp: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_DriveLowCostStepper.**

bDigOutAn: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_DriveLowCostStepper.**

bDigOutBp: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_DriveLowCostStepper.**

bDigOutBn: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_DriveLowCostStepper.**

uiCount: reserviert.

uiDacOutA: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_EncoderIx2512_1Coil, iTcMc_EncoderIx2512_2Coil.**

uiDacOutB: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_EncoderIx2512_2Coil.**

bMovePos: reserviert.

bMoveNeg: reserviert.

bBrakeOff: reserviert.

bBrakeOffInverted: reserviert.

DriveCtrl: Verwendet für Gerätesteuersignale. Benutzt bei **iTcMc_EncoderAx2000_B200, iTcMc_DriveAx2000_B200, iTcMc_EncoderAx2000_B900, iTcMc_DriveAx2000_B900.**

NominalVelo: Verwendet für Stellwertausgabe. Benutzt bei **iTcMc_DriveAx2000_B110, iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.**

uiDriveCtrl: Verwendet für Gerätesteuersignale. Benutzt bei **iTcMc_EncoderAx2000_B110, iTcMc_DriveAx2000_B110.**

S_iReserve: reserviert.

S_DiReserve: reserviert.

CiA_Reserve: reserviert.

bPowerOn: Optional verwendet für die Steuerung eines Netz-Schützes. Benutzt bei **iTcMc_DriveAx2000_B110, iTcMc_EncoderAx2000_B200, iTcMc_EncoderAx2000_B900.**

bEnable: reserviert.

bEnablePos: reserviert.

bEnableNeg: reserviert.

nResetState: reserviert.

usiCtrl: Verwendet für Gerätesteuersignale oder Parameter-Kommunikation. Benutzt bei **iTcMc_EncoderEL5101**, **iTcMc_EncoderKL3002**, **iTcMc_EncoderKL3042**, **iTcMc_EncoderKL3062**, **iTcMc_EncoderKL3162**, **iTcMc_EncoderKL5101**, **iTcMc_EncoderKL5111**, **iTcMc_EncoderM3120**.

uiTerminalData: Verwendet für Parameter-Kommunikation. Benutzt bei **iTcMc_EncoderKL2521**, **iTcMc_EncoderKL5001**, **iTcMc_EncoderKL5101**, **iTcMc_EncoderKL5111**, **iTcMc_DriveEL4132**, **iTcMc_DriveKL2521**, **iTcMc_DriveKL4032**.

bTerminalCtrl: Verwendet für Parameter-Kommunikation. Benutzt bei **iTcMc_EncoderKL2521**, **iTcMc_EncoderKL2531**, **iTcMc_EncoderKL2541**, **iTcMc_DriveEL4132**, **iTcMc_DriveKL2521**, **iTcMc_DriveKL2531**, **iTcMc_DriveKL2541**, **iTcMc_DriveKL4032**.

uiTerminalCtrl2: Verwendet für Gerätesteuersignale. Benutzt bei **iTcMc_EncoderKL2541**, **iTcMc_DriveKL2531**, **iTcMc_DriveKL2541**.

nReserve: reserviert.

3.3.27 ST_TcPlcMcLogBuffer (ab V3.0)

Eine Variable mit dieser Struktur bildet den LogBuffer der Library. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base \[► 207\]](#).



Hinweis

Die Daten in dieser Struktur dürfen nicht durch die Applikation verändert werden.

```
TYPE ST_TcMcLogBuffer:
STRUCT
  ReadIdx:      INT:=0;
  WriteIdx:     INT:=0;
  MessageArr:   ARRAY [0..19] OF ST_TcPlcMcLogEntry;
END_STRUCT
END_TYPE
```

[ST_TcPlcMcLogEntry \[► 102\]](#)

ReadIdx: Der Leseindex des Buffers.

WriteIdx: Der Schreibindex des Buffers.

MessageArr: Die aktuell gespeicherten Meldungen.

3.3.28 ST_TcPlcMcLogEntry (ab V3.0)

Eine Variable mit dieser Struktur enthält eine Meldung des LogBuffer der Library. Sie wird als Bestandteil in [ST_TcPlcMcLogBuffer \[► 102\]](#) verwendet. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base \[► 207\]](#).



Hinweis

Die Daten in dieser Struktur dürfen nicht durch die Applikation verändert werden.

```
TYPE ST_TcPlcMcLogEntry:
STRUCT
  TimeLow:      UDINT:=0;
  TimeHigh:     UDINT:=0;
  LogLevel:     DWORD:=0;
  Source:       DWORD:=0;
  Msg:          STRING(255);
  ArgType:      INT:=0;
```

```

diArg:      DINT:=0;
lrArg:      LREAL:=0;
sArg:       STRING(255);
END_STRUCT
END_TYPE

```

TimeLow, TimeHigh: Der Timestamp der Meldung. Hier wird der Zeitpunkt festgehalten, zu dem die Meldung erzeugt wurde.

LogLevel: Eine Kennzeichnung der Dringlichkeit der Meldung. Hier sollten nur Werte aus einem festgelegten [Zahlenvorrat \[► 232\]](#) erscheinen.

Source: Eine Kennzeichnung der Quelle der Meldung. Hier sollten nur Werte aus einem festgelegten [Zahlenvorrat \[► 232\]](#) erscheinen.

Msg: Der Meldungstext mit einem optionalen Platzhalter für einen variablen Bestandteil.

ArgType: Der Typ des optionalen Bestandteils.

diArg: Wenn ein optionaler Bestandteil vom Typ DINT verwendet wird befindet sich sein Wert hier.

lrArg: Wenn ein optionaler Bestandteil vom Typ LREAL verwendet wird befindet sich sein Wert hier.

sArg: Wenn ein optionaler Bestandteil vom Typ STRING verwendet wird befindet sich sein Wert hier.

3.3.29 ST_TcPlcRegDataItem (ab V3.0.7)

Diese Struktur enthält einen Parameter für eine KL-Klemme. Ein ARRAY von Elementen dieses Typs bildet den Typ [ST_TcPlcRegDataTable \[► 103\]](#).

```

TYPE ST_TcPlcRegDataItem :
STRUCT
  Access:      INT:=0;
  Select:      INT:=-1;
  RegData:     WORD:=0;
END_STRUCT
END_TYPE

```

Access: Hier wird die Art der auszuführenden Operation kodiert. Details sind unter [MC_AxUtiUpdateRegDriveTerm_BkPlcMc \[► 198\]](#) oder [MC_AxUtiUpdateRegEncTerm_BkPlcMc \[► 200\]](#) zu finden.

Select: Die Adresse des Registers in der Klemme.

RegData: Der für die auszuführende Operation zu verwendende Parameter.

3.3.30 ST_TcPlcRegDataTable (ab V3.0.7)

Diese Struktur enthält einen Parametersatz für eine KL-Klemme. Eine solche Tabelle wird von [MC_AxUtiUpdateRegDriveTerm_BkPlcMc \[► 198\]](#) oder [MC_AxUtiUpdateRegEncTerm_BkPlcMc \[► 200\]](#) Bausteinen verarbeitet.

```

TYPE ST_TcPlcRegDataTable :
STRUCT
  RegDataItem:  ARRAY [1..64] OF ST_TcPlcRegDataItem;
END_STRUCT
END_TYPE

```

3.3.31 TRACK_REF_BkPlcMc (ab V3.0)

```

TYPE TRACK_REF_BkPlcMc:
STRUCT
  Track:        ARRAY [ciBkPlcMc_TrackRef_MinIdx..ciBkPlcMc_TrackRef_MaxIdx] OF TRACK_REFTYPE;
PE_BkPlcMc;
END_STRUCT
END_TYPE

TYPE TRACK_REFTYPE_BkPlcMc:

```

```

STRUCT
  OnCompensation: LREAL;
  OffCompensation: LREAL;
  Hysteresis:      LREAL;
END_STRUCT
END_TYPE

```

OnCompensation: Die zu kompensierende Einschalttzeit in Sekunden.

OffCompensation: Die zu kompensierende Ausschalttzeit in Sekunden.

Hysteresis: Um diese Entfernung muss sich die Achse vom Schaltpunkt entfernt haben bevor ein erneutes Erreichen des Schaltpunkts ausgewertet wird.

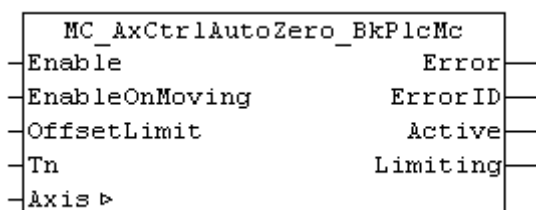
Wenn als Totzeitkompensation ein positiver Wert angegeben wird, erfolgt die Signalgabe verzögert. Ein negativer Wert verursacht eine voreilende Signalgabe.

ⓘ HINWEIS! Die Zeit kann nicht genau eingehalten werden, wenn sich die steuernde Größe mit schwankender Rate ändert. Ist diese steuernde Größe eine Achsen Istposition, muss dazu die Achsen Istgeschwindigkeit konstant sein.

3.4 System

3.4.1 Controller

3.4.1.1 MC_AxCtrlAutoZero_BkPlcMc (ab V3.0)



Der Funktionsbaustein führt einen automatischen Nullpunktgleich aus.

```

VAR_INPUT
  Enable:          BOOL:=FALSE;
  EnableOnMoving: BOOL:=FALSE;
  OffsetLimit:    LREAL:=0.0;
  Tn:             LREAL:=0.0;
  Threshold:      LREAL:=0.1;
  Filter:         LREAL:=0.1;
END_VAR
VAR_INOUT
  Axis:           Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Error:          BOOL;
  ErrorID:        UDINT;
  Active:         BOOL;
  Limiting:       BOOL;
  Done:          BOOL;
END_VAR

```

Enable: Dieser Eingang kontrolliert die Aktivität des Reglers.

EnableOnMoving: Dieser Eingang kontrolliert die Aktivität des Reglers.

Tn: Die Nachstellzeit des Reglers.

Threshold, Filter: Parameter für das **Done** Signal.

OffsetLimit: Der Wert in fZeroCompensation wird auf diesen Wert begrenzt.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Active: Hier wird signalisiert, dass der Baustein den Wert von fZeroCompensation in ST_TcHydAxParam [► 88] aktiv verstellt.

Limiting: Hier wird signalisiert, dass der Wert von fZeroCompensation in ST_TcHydAxParam [► 88] die von OffsetLimit festgelegte Grenze erreicht hat.

Done: Hier wird ein Einpendeln des Offsetabgleichs signalisiert.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Solange **Enable** auf FALSE steht wird der Baustein nicht aktiv und **Done** ist FALSE. Wird **Enable** auf TRUE gesetzt entscheidet der Baustein selbständig, ob er aktiv oder inaktiv ist. Dabei werden bei einer Achse im Zustand iTcHydStateIdle die Werte von fPosError und fActVelo berücksichtigt. Ist zusätzlich **EnableOnMoving** auf TRUE gesetzt wird der Baustein auch dann aktiv, wenn bei einer Achse in einem ungestörten und aktiven Zustand die Werte von fSetSpeed und fActVelo dies erlauben.

Im aktiven Zustand wird der Wert von fZeroCompensation in ST_TcHydAxParam [► 88] so nachgeführt, dass die Achse eine zumindest geringe Bewegung in die erwünschte Richtung zeigt. Dabei legt **Tn** die Zeit fest, in der bei einem bleibenden hohen Fehler eine Vollaussteuerung erreicht würde. Ist der Baustein inaktiv sind **Error** und **Active** FALSE, **ErrorID** ist 0 und der Baustein führt den Wert von fZeroCompensation in ST_TcHydAxParam [► 88] nicht nach.


Im aktiven Zustand wird der Wert von fZeroCompensation mit einem gespeicherten Wert verglichen. Solange diese Werte sich um weniger als **Threshold** unterscheiden läuft eine Zeit, andernfalls wird sie auf Null gesetzt und der gespeicherte Wert mit fZeroCompensation aktualisiert. Erreicht die Zeit die in **Filter** eingestellte Grenze wird **Done** signalisiert.

ⓘ HINWEIS! Die Begrenzung auf den von OffsetLimit festgelegten Wertebereich ist auch dann aktiv, wenn der Baustein nicht aktiv ist. Der Ausgang Limiting wird aktualisiert.

Der Wert **OffsetLimit** und ST_TcHydAxParam [► 88]. **fZeroCompensation** werden als Offsetspannung betrachtet. Somit entspricht der Wert 10.0 einer Vollaussteuerung. In der Regel ist für **OffsetLimit** je nach Einsatzfall ein Wert zwischen 0.1 und 1.0 sinnvoll.

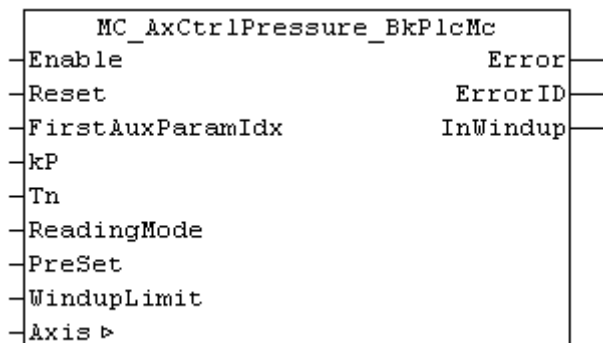
Integration des Bausteins in die Applikation

In der Aufruffreihenfolge der Bausteine einer Achse sollte ein MC_AxCtrlAutoZero_BkPlcMc Baustein unmittelbar vor dem MC_AxRtFinish_BkPlcMc [► 164] Baustein stehen. Wird an Stelle der Einzelbausteine ein MC_AxStandardBody_BkPlcMc [► 170] Baustein aufgerufen sollte der MC_AxCtrlAutoZero_BkPlcMc vor diesem Baustein aufgerufen werden.

 Achtung	<p>Gefährliche Achsbewegung</p> <p>Wenn beim Betrieb der Achse Situationen vorkommen, in denen die Achse eine Reglerfrei-gabe anstehen hat, aber nicht ihr normales Bewegungsverhalten zeigt ist es zwingend erforderlich, den MC_AxCtrlAutoZero_BkPlcMc Bausteins zu disablen. Denkbare Ursachen für eine solche Situation sind zum Beispiel das Anfahren eines Blocks mit oder ohne Übergang in eine Druckregelung oder ein Absenken oder Abschalten der Versorgung. Wird dies nicht beachtet kann der Wert von fZeroCompensation in <u>ST_TcHydAxParam</u> [► 88] in willkürlicher Richtung bis zur festgelegten Begrenzung laufen. Sobald die Achse zu einem späteren Zeitpunkt wieder reaktionsfähig wird ist eine unter Umständen gefährliche Bewegung unvermeidbar. Das Positionierverhalten wird anschließend stark beeinträchtigt sein. Wird der Baustein ohne EnableOnMoving aufgerufen kann er den verstellten Offset dann unter Umständen nicht mehr automatisch korrigieren. Die Achse wird dann außerhalb des Ziel-fensters stehen bleiben und niemals oder zumindest erst nach einer erheblichen Zeit die Bewegung als abgeschlossen melden.</p>
-------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Bei Kombination mit einem [MC_AxStandardBody_BkPlcMc](#) [▶ 170] Baustein werden alle Reaktionen des [MC_AxCtrlAutoZero_BkPlcMc](#) Bausteins um einen SPS Zyklus verzögert wirksam. In der Regel ist dies kein Problem. Sollte dieser Versatz störend sein müssen die Einzelbausteine für Encoder usw. verwendet und der [MC_AxCtrlAutoZero_BkPlcMc](#) Baustein unmittelbar vor dem [MC_AxRtFinish_BkPlcMc](#) [▶ 164] Baustein aufgerufen werden.

3.4.1.2 MC_AxCtrlPressure_BkPlcMc (ab V3.0)



Der Funktionsbaustein regelt den an einer Achse wirksamen Druck so, dass in dem durch **ReadingMode** ausgewählten Istwert ein gewünschter Vorgabewert aufgebaut und eingehalten wird.

Die Erfassung des Istdruck kann in den meisten Fällen mit Bausteinen vom Typ [MC_AxRtReadPressureSingle_BkPlcMc](#) [▶ 145] oder [MC_AxRtReadPressureDiff_BkPlcMc](#) [▶ 143] erfolgen.

```

VAR_INPUT
  Enable:      BOOL:=FALSE;
  Reset:       BOOL:=TRUE;
  FirstAuxParamIdx: INT:=0;
  kP:          LREAL:=0.0;
  Tn:          LREAL:=0.0;
  ReadingMode: E_TcMcPressureReadingMode:=iTcHydPressureReadingDefault;
  PreSet:      LREAL:=0.0;
  WindupLimit: LREAL:=0.0;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Error:       BOOL;
  ErrorID:     UDINT;
  InWindup:    UDINT;
END_VAR

```

Enable: Durch ein TRUE an diesem Eingang wird der Regler aktiviert.

Reset: Durch ein TRUE an diesem Eingang wird der Regler zurückgesetzt. Der Speicher des I-Anteils wird abgelöscht.

FirstAuxParamIdx: Hier kann ein Bereich in den [Axis_Ref_BkPlcMc](#) [▶ 62].[ST_TcHydAxParam](#) [▶ 88].[fCustomerData](#) als Parameter-Interface aktiviert werden.

InWindup: Dieser Ausgang wird TRUE wenn der I-Anteil durch **WindupLimit** begrenzt wird.

kP: Der Verstärkungsfaktor des P-Anteils.

Tn: Die Nachstellzeit des I-Anteils.

ReadingMode: Hier kann die zu regelnde Istgröße festgelegt werden. Als Defaultwert wird [Axis_Ref_BkPlcMc](#) [▶ 62].[ST_TcHydAxRtData](#) [▶ 94].[fActPressure](#) ausgewählt.

PreSet: Hier kann ein Voreinstellwert festgelegt werden, mit dem ein Initialwert für den I-Anteil des Reglers berechnet wird. Auf diesen Wert wird der I-Anteil bei Aktivierung vorgeladen.

WindupLimit: Hier kann ein Begrenzungswert für den I-Anteil festgelegt werden. Eine solche Begrenzung verhindert ein extremes Verhalten des I-Anteils in Situationen, in denen die Strecke nicht auf die Ausgaben des Reglers reagiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Durch ein TRUE an **Reset** wird der Baustein unabhängig von den anderen Steuersignalen in einen Ruhezustand versetzt. Dann ist sowohl der P- als auch der I-Anteil gelöscht. Durch **Enable** kann festgelegt werden, ob der Baustein den aktiven Zustand annimmt.

Der Eingang **ReadingMode** legt fest, welche Variable in der stAxRtData Struktur die zu regelnde Größe enthält.

- iTcHydPressureReadingDefault, iTcHydPressureReadingActPressure: fActPressure wird geregelt.
- iTcHydPressureReadingActForce: fActForce wird geregelt.
- jeder andere Wert deaktiviert den Regler.

ⓘ HINWEIS! Der Sollwert ist in fSetPressure in der stAxRtData Struktur der Achse vorzugeben.

Zunächst ermittelt der Baustein, ob er den aktiven Zustand annehmen oder beenden muss. Dazu wird das Signale **Enable** ausgewertet. Bei einer steigenden Flanke wird der I-Anteil mit **PreSet** initialisiert. Wenn der zu ST_TcHydAxRtData [► 94].fSetPressure passende Ausgabewert bekannt ist kann dies für ein schnelleres Erreichen des ausgeregelten Zustands genutzt werden. Anschließend wird mit **kP** ein P-Anteil und mit **Tn** ein I-Anteil berechnet. Die Summe dieser Regler-Anteile wird als Stellwert in ST_TcHydAxRtData [► 94].fSetSpeed ausgegeben. Da dieser Regler die Funktion eines Stellwertgenerators übernimmt löscht er ST_TcHydAxRtData [► 94].fLagCtrlOutput. Der nach dem Reglerbaustein zu platzierende MC_AxRtFinish_BkPlcMc [► 164] Baustein berücksichtigt die Reaktion dann automatisch.

Der Übergang in den inaktiven Zustand führt zu einem Löschen der Regleranteile.

Integration des Bausteins in die Applikation

Ein Baustein dieses Typs muss nach der Istwert- und Istdruck-Erfassung aufgerufen werden. Er übernimmt die volle Kontrolle über die Achse und ersetzt einen eventuell vorhandenen Baustein für die Stellwert-Generierung.

Es steht ein Programm-Beispiel [► 208] #15 zur Verfügung.

ⓘ HINWEIS! Ist sowohl ein Baustein für die Stellwert-Generierung als auch ein MC_AxCtrlPressure_BkPlcMc Baustein vorhanden sind diese Bausteine entweder alternativ aufzurufen oder der MC_AxCtrlPressure_BkPlcMc Baustein muss nach dem Stellwert-Baustein stehen, sodass er dessen Ausgaben überschreibt. Beides ist nicht mit jedem Generator-Typ zulässig.

ⓘ HINWEIS! Durch einen Wert größer als 0 in FirstAuxParamIdx kann der Baustein veranlasst werden drei aufeinander folgende Werte in den fCustomerData der Parameter-Struktur als Tn, kP und PreSet zu verwenden. Ist in Axis.pStAxAuxLabels die Adresse eines geeigneten ARRAY[..] OF STRING() eingetragen werden die Parameter automatisch mit einer Bezeichnung versehen.

Inbetriebnahme

Die vier Parameter **kP**, **Tn**, **PreSet** und **WindupLimit** lassen eine Anpassung des Reglers an eine Reihe unterschiedlicher Aufgaben zu.

**Achtung****Schwingungen in der Regelung**

Während der Inbetriebnahme kann es dazu kommen, dass die Achse mit dem vollen Systemdruck beaufschlagt wird oder dass gedämpfte oder ungedämpfte Schwingungen in einem weiten Frequenzbereich auftreten. Sollte dies für die Achse oder ihre Umgebung gefährlich sein sind Vorkehrungen zu treffen. In jedem Fall sollten Maßnahmen vorgesehen werden, die Regelungen schnell zu deaktivieren.

Zunächst sollten für **Tn** und **PreSet** der Wert 0.0 und für **WindupLimit** der Wert 1.0 eingetragen werden. Der Regler arbeitet jetzt als reiner P-Regler. Nachdem ein Block angefahren und der Regler aktiviert ist (Enable:=TRUE, Reset:=FALSE, **SetPressure**:=Sollwert) kann jetzt der maximal anwendbare Wert für **kP** festgestellt werden. Dazu ist der Wert schrittweise zu erhöhen, bis sich eine Schwingneigung zeigt. Durch wiederholtes Deaktivieren und Aktivieren sollte überprüft werden, ob der Regler tatsächlich stabil ist. In der Praxis wird der Wert zwischen etwa 0.1 und 0.5 liegen.

Als nächster Parameter ist **Tn** einzustellen. Dazu sollte zunächst ein relativ großer Wert wie 0.5 vorgegeben werden. Jetzt sollte der Istdruck mit großer Trägheit aber recht genau auf den Sollwert eingeregelt werden. Durch schrittweises Verkleinern wird jetzt die maximal mögliche Einstellung ermittelt. Auch dabei ist durch wiederholtes Deaktivieren und Aktivieren zu überprüfen, ob der Regler tatsächlich stabil ist. Zeigt sich eine Neigung zur gedämpften Schwingung beim Aktivieren ist **Tn** bereits zu niedrig eingestellt.

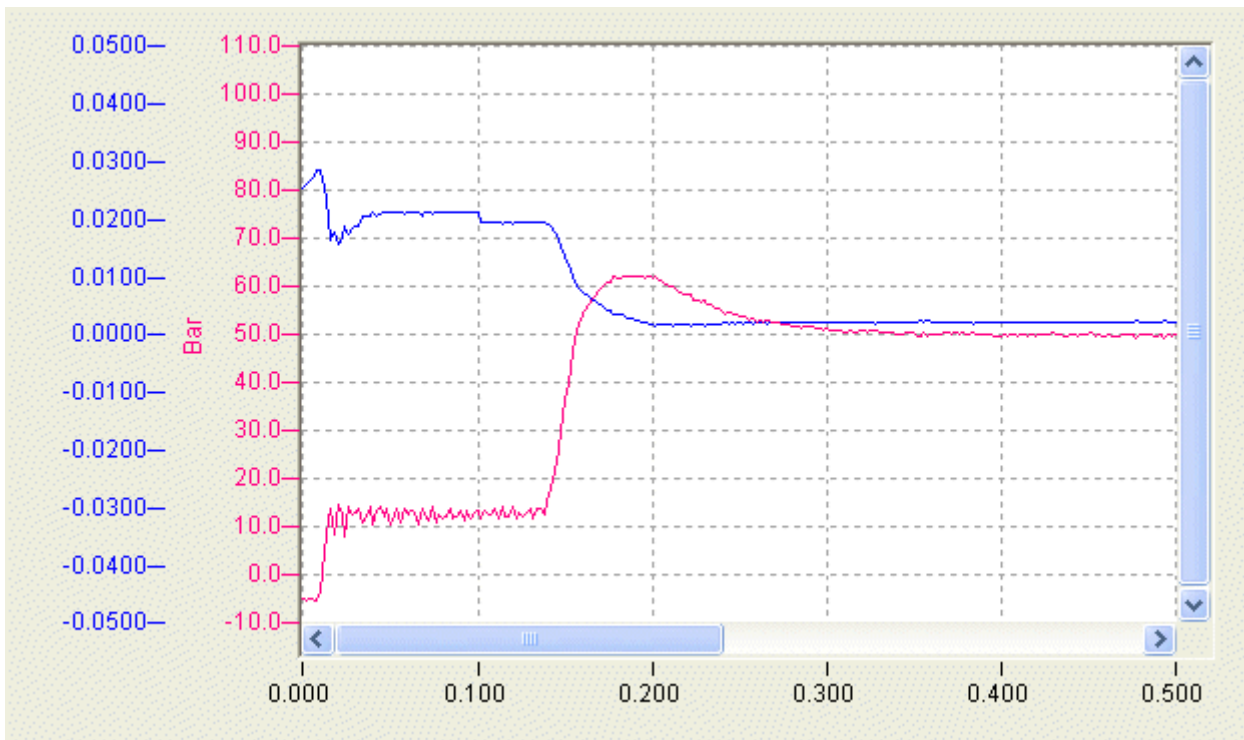
Die Einstellung von **WindupLimit** beeinflusst das Verhalten des Reglers nicht auf direktem Weg. Vielmehr dient dieser Parameter dazu, das Übergangsverhalten zu beeinflussen. Wenn der Regler den Druck unmittelbar aufbauen kann, weil die Achse keinen Weg zurücklegen muss sollte der Wert von **WindupLimit** so gewählt werden, dass der I-Anteil nicht größer als das drei- bis Vierfache des Wertes erreichen kann, der entsprechend der Ventilcharakteristik benötigt wird. Auf diese Weise kann das Einregeln des Drucks deutlich schneller erreicht werden. Hat die Achse noch einen Weg zurückzulegen wird ein zu niedriger Wert dieses Parameters die Bewegung der Achse zum Erreichen der Arbeitsposition bestimmen. Ist der Parameter zu niedrig gewählt wird die Achse sehr langsam oder kann sogar stehen bleiben. Andererseits wird ein zu hoher Wert dazu führen, dass die Achse die Arbeitsposition mit eher hoher Geschwindigkeit erreicht und der Druckanstieg steil ist. Der dabei auftretende Spitzendruck kann dann erhebliche Werte annehmen.

ⓘ HINWEIS! Es ist wenn irgend möglich zu vermeiden, einen Druckregler zu aktivieren, wenn sich die Achse nicht zumindest sehr nah an ihrer Arbeitsposition befindet.

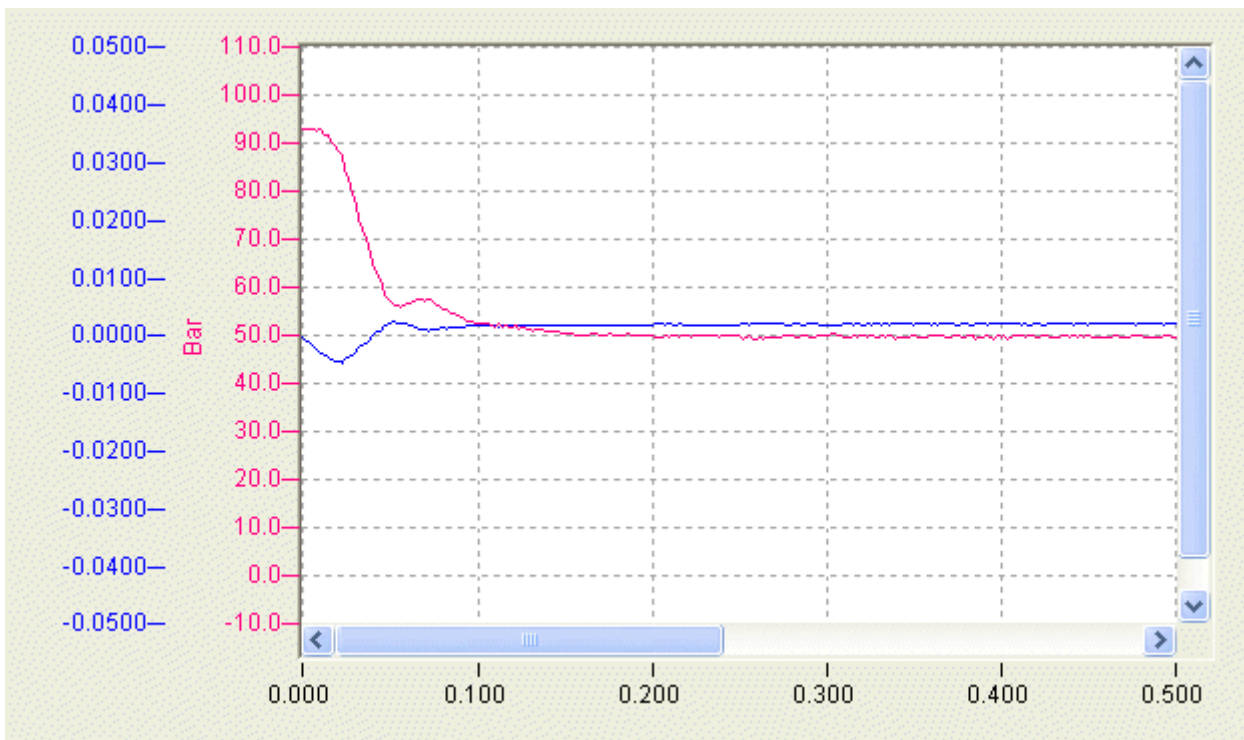
Der Wert für **PreSet** kann für zwei Vorgehensweisen eingesetzt werden. Soll der Druckregler den Stellwert eines anderen Bausteins kontinuierlich fortsetzen kann dessen Stellwert für die Berechnung von **PreSet** vorgegeben werden. So können die Stellwertsprünge beim Aktivieren des Reglers verringert oder völlig vermieden werden.

Ist der vom Regler zu erzeugende Stellwert bekannt kann als **PreSet** ein Wert vorgegeben werden, der in der Nähe dieses Wertes liegt. So kann die Zeit verringert werden, die der I-Anteil zum Aufbauen des Stellwerts benötigt. Da auch der P-Anteil wirksam ist sollte jedoch ein Wert eingestellt werden, der höher als der exakte Wert ist.

ⓘ HINWEIS! Letztlich ist bei der Einstellung dieser Parameter durch kleine Veränderungen und Beurteilung des Reglerverhaltens ein der Aufgabe angemessener Satz von Werten zu finden.

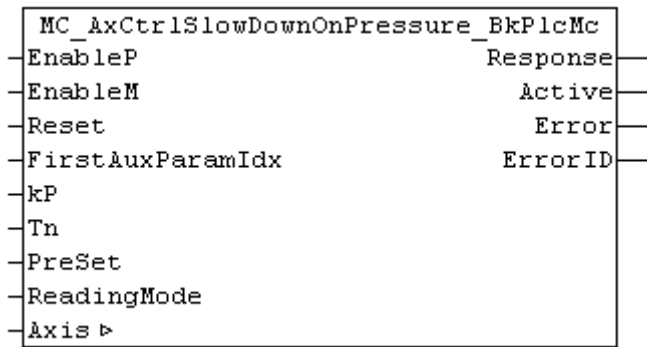


Beispiel für das Verhalten des Reglers, wenn die Achse zunächst einen Weg zurücklegen muss, bevor sie den geforderten Druck aufzubauen vermag.



Beispiel für das Verhalten des Reglers, wenn die Achse unmittelbar den geforderten Druck aufzubauen vermag.

3.4.1.3 MC_AxCtrlSlowDownOnPressure_BkPlcMc (ab V3.0)



Der Funktionsbaustein bremst eine Achse so ab, dass in dem durch **ReadingMode** ausgewählten Istwert ein gewünschter Vorgabewert nicht überschritten wird. Dabei gelten die Regeln der ablösenden Druckregelung.

Die Erfassung des Istdruck kann in den meisten Fällen mit Bausteinen vom Typ [MC_AxRtReadPressureSingle_BkPlcMc \[▶ 145\]](#) oder [MC_AxRtReadPressureDiff_BkPlcMc \[▶ 143\]](#) erfolgen.

```
VAR_INPUT
  EnableP:          BOOL:=FALSE;
  EnableM:          BOOL:=FALSE;
  Reset:            BOOL:=TRUE;
  FirstAuxParamIdx: INT:=0.0;
  kP:               LREAL:=0.0;
  Tn:               LREAL:=0.0;
  PreSet:           LREAL:=0.0;
  ReadingMode:      E_TcMcPressureReadingMode:=iTcHydPressureReadingDefault;
END_VAR
```

[E_TcMcPressureReadingMode \[▶ 84\]](#)

```
VAR_INOUT
  Axis:             Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Response:         LREAL;
  Active:           BOOL;
  Error:            BOOL;
  ErrorID:          UDINT;
END_VAR
```

EnableP: Durch ein TRUE an diesem Eingang wird dem Regler erlaubt, den Ausgabewert bei einer Bewegung in positiver Richtung zu beeinflussen.

EnableM: Durch ein TRUE an diesem Eingang wird dem Regler erlaubt, den Ausgabewert bei einer Bewegung in negativer Richtung zu beeinflussen.

Reset: Durch ein TRUE an diesem Eingang wird der Regler zurückgesetzt. Der Speicher des I-Anteils wird abgelöscht.

FirstAuxParamIdx: Hier kann ein Bereich in den [Axis_Ref_BkPlcMc \[▶ 62\].ST_TcHydAxParam \[▶ 88\].fCustomerData](#) als Parameter-Interface aktiviert werden.

kP: Der Verstärkungsfaktor des P-Anteils.

Tn: Die Nachstellzeit des I-Anteils.

PreSet: Hier kann ein Voreinstellwert festgelegt werden, mit dem ein Initialwert für den I-Anteil des Reglers berechnet wird. Auf diesen Wert wird der I-Anteil bei Aktivierung vorgeladen.

Response: Der Ausgabewert des Druckreglers.

ReadingMode: Hier kann die zu regelnde Istgröße festgelegt werden. Als Defaultwert wird [Axis_Ref_BkPlcMc \[▶ 62\].ST_TcHydAxRtData \[▶ 94\].fActPressure](#) ausgewählt.

Active: Ein TRUE an diesem Ausgang signalisiert, dass der Baustein einen Response erzeugt, um die Druckregelung zu übernehmen.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Durch ein TRUE an **Reset** wird der Baustein unabhängig von den anderen Steuersignalen in einen Ruhezustand versetzt. Dann ist **Active** FALSE und **Response** := 0.0, weil sowohl der P- als auch der I-Anteil gelöscht sind.

Der Eingang **ReadingMode** legt fest, welche Variable in der stAxRtData Struktur die zu regelnde Größe enthält.

- iTcHydPressureReadingDefault, iTcHydPressureReadingActPressure: fActPressure wird geregelt.
- iTcHydPressureReadingActForce: fActForce wird geregelt.
- jeder andere Wert deaktiviert den Regler.

ⓘ HINWEIS! Der Sollwert ist in fSetPressure in der stAxRtData Struktur der Achse vorzugeben.

Das Verhalten des Bausteins wird im aktiven Betrieb durch die Eingänge **EnableP** und **EnableM** festgelegt. Sie legen fest, ob der Baustein bei einer Bewegung in positiver oder negativer Richtung eingreifen soll. Es ist dabei zu beachten, dass der Baustein die Aufgabe hat, einer aktiven Fahrbewegung entgegen zu wirken. Wenn also eine Fahrbewegung in positiver Richtung ausgeführt wird, die einen festgelegten Druck nicht überschreiten soll ist **EnableP** zu setzen. Bei entgegengesetzter Fahrtrichtung gibt **EnableM** eine druckbegrenzende Reglerreaktion in positiver Richtung frei.

Zunächst ermittelt der Baustein, ob er den aktiven Zustand annehmen oder beenden muss. Dazu werden die Signale **EnableP**, **EnableM**, das Vorzeichen von ST_TcHydAxRtData [► 94].fSetSpeed und die Differenz zwischen **SetPressure** und dem ausgewählten Istwert ausgewertet.

Beim Übergang in den aktiven Zustand wird der I-Anteil mit **PreSet** initialisiert. Dabei wird er mit einem Startwert geladen, der in Kombination mit ST_TcHydAxRtData [► 94].fSetSpeed den Wert von **PreSet** ergibt. Wenn der zu **SetPressure** passende Ausgabewert bekannt ist kann dies für ein schnelleres Erreichen des ausgeregelten Zustands genutzt werden. In der Praxis sollte die Wahl dieses Parameters vom Verhalten der Regelstrecke abhängig gemacht werden. Diese ist vor allem durch die Nachgiebigkeit des eingepressten Objekts, aber auch durch die gewählte Geschwindigkeit beeinflusst. Ist der Anstieg im Vergleich zur verwendeten **Tn** eher langsam sollte als Vorgabe der aktuelle Stellwert aus ST_TcHydAxRtData [► 94].fSetSpeed verwendet werden. Reagiert der Istdruck mit einem schnellen Anstieg ist ein Wert zu empfehlen, der den Solldruck und die Druckverstärkung des Ventils berücksichtigt.

Anschließend wird mit **kP** ein P-Anteil und mit **Tn** ein I-Anteil berechnet. Die Summe dieser Regler-Anteile wird als **Response** ausgegeben und der Zustand des Reglers mit TRUE an **Active** signalisiert.

Der Übergang in den inaktiven Zustand führt zu einem Löschen der Regleranteile und wird mit FALSE an **Active** markiert.

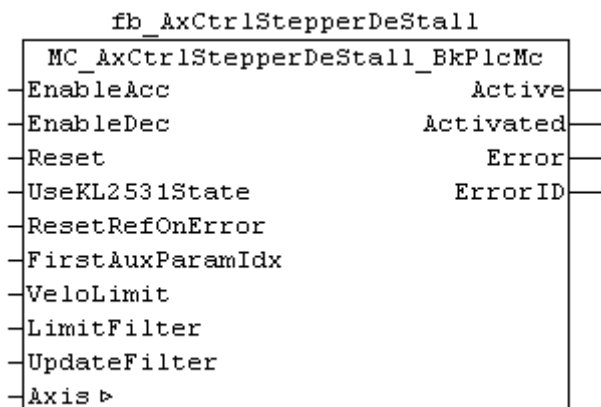
Integration des Bausteins in die Applikation

Ein Baustein dieses Typs muss nach der Istwert- und Istdruck-Erfassung und nach der Stellwert-Generierung aufgerufen werden. Werden Bausteine zur Geschwindigkeit- oder Positionsregelung aufgerufen sind diese ebenfalls vor dem Druckregler-Baustein zu platzieren oder die Reaktionen der Regler sind mit entsprechender Sorgfalt zu koordinieren.

Der Druckregler berechnet zwar eine Reaktion, trägt sie aber nicht in die ST_TcHydAxRtData [► 94] Struktur ein. Dies ist von der Applikation in Abhängigkeit von **Active** und unter eventueller Berücksichtigung von Signalen anderer Regler zu tun. In der Regel wird dabei **Response** auf die Variable ST_TcHydAxRtData [► 94].fLagCtrlOutput zugewiesen. Der nach dem Reglerbaustein zu platzierende MC_AxRtFinish_BkPlcMc [► 164] Baustein berücksichtigt die Reaktion dann automatisch.

HINWEIS! Durch einen Wert größer als 0 in `FirstAuxParamIdx` kann der Baustein veranlasst werden drei aufeinander folgende Werte in den `fCustomerData` der Parameter-Struktur als `Tn`, `kP` und `PreSet` zu verwenden. Ist in `Axis.pStAxAuxLabels` die Adresse eines geeigneten `ARRAY[..] OF STRING()` eingetragen werden die Parameter automatisch mit einer Bezeichnung versehen.

3.4.1.4 MC_AxCtrlStepperDeStall_BkPlcMc



Der Funktionsbaustein überwacht die Bewegung einer Schrittmotorachse, die mit einem Encoder betrieben wird.

HINWEIS! Der Einsatz eines echter Encoders (nicht Encoder-Emulation durch Impulzzählung einer Ausgabe-Klemme) ist für die Funktion dieses Bausteins unverzichtbar.

HINWEIS! Auch beim Einsatz eines solchen Bausteins kann es zum Stall (Drehmomentabriss) kommen. Es kann also auch dann nicht vorausgesetzt werden, dass die Geschwindigkeit konstant ist.

```

VAR_INPUT
  EnableAcc:      BOOL:=FALSE;
  EnableDec:      BOOL:=FALSE;
  Reset:          BOOL:=FALSE;
  UseKL2531State: BOOL:=FALSE;
  ResetRefOnError: BOOL:=FALSE;
  FirstAuxParamIdx: INT:=0;
  VeloLimit:     LREAL:=0.0;
  LimitFilter:   LREAL:=0.0;
  UpdateFilter:  LREAL:=0.0;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Active:        BOOL;
  Activated:     BOOL;
  Error:         BOOL;
  ErrorID:       UDINT;
END_VAR

```

EnableAcc, EnableDec: Diese Eingänge legen fest, ob die Überwachung auch in den Beschleunigungs- und Bremsphasen eingreifen darf.

Reset: Dieser Eingang kontrolliert die Aktivität des Reglers.

UseKL2531State: Wenn hier ein TRUE übergeben wird wertet der Baustein `ST_TcPlcDeviceInput` [► 98].bTerminalState aus.

ResetRefOnError: Wenn hier ein TRUE übergeben wird löscht der Baustein das Referenzflag der Achse.

FirstAuxParamIdx: Hier kann ein Bereich in den `Axis_Ref_BkPlcMc` [► 62].`ST_TcHydAxParam` [► 88].`fCustomerData` als Parameter-Interface aktiviert werden.

VeloLimit: Die Schwelle für die Geschwindigkeitsabweichung, ab der die Stall-Situation erkannt wird.

LimitFilter: Die Zeit, für die eine zu große Geschwindigkeitsabweichung ununterbrochen vorliegen muss, damit die Stall-Situation erkannt wird.

UpdateFilter: Die Zeitkonstante, mit der im Baustein der Geschwindigkeits-Stellwert an die Istgeschwindigkeit angepasst wird.

Active: Hier wird signalisiert, dass eine Stall-Situation erkannt ist.

Activated: Hier wird signalisiert, dass seit dem letzten Start einer aktiven Achsbewegung eine Stall-Situation erkannt wurde.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf überprüft der Baustein, ob er den Zustand wechseln muss. Er geht in den aktiven Zustand, wenn die interne Bewegungsphase dies nach den Regeln von **EnableAcc**, **EnableDec** zulässt und der Geschwindigkeitsfehler für mindestens **LimitFilter** ununterbrochen den Wert von **VeloLimit** überschreitet. Dabei erlaubt **EnableAcc** dem Baustein, während der Phasen mit konstanter oder im Betrag steigenden Phasen einzugreifen. **EnableDec** gibt die Aktivität des Bausteins für Phasen mit im Betrag sinkender oder konstanter Geschwindigkeit frei. Beim Wechsel in den aktiven Zustand werden **Active** und **Activated** gesetzt.

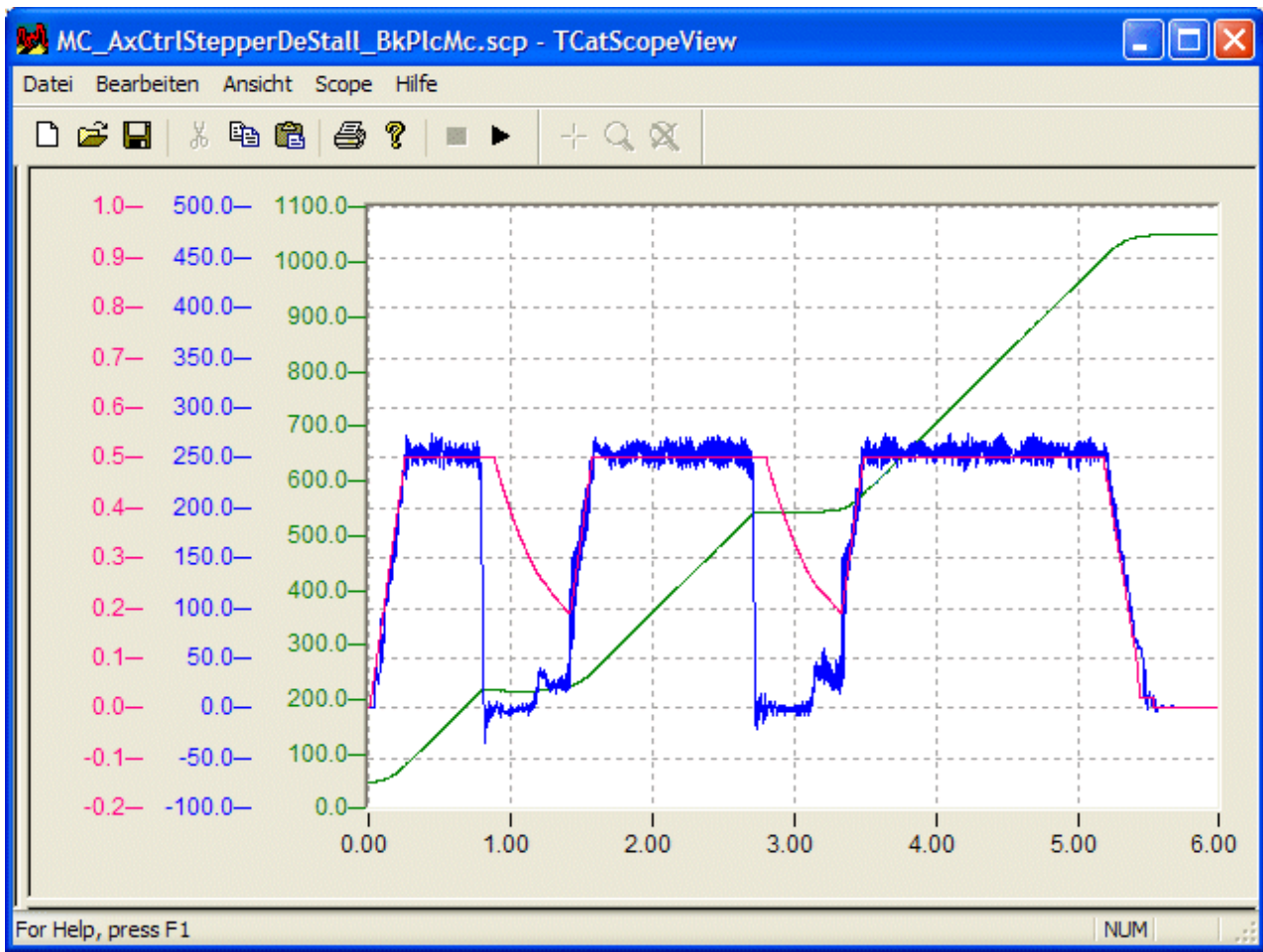
In den inaktiven Zustand wechselt der Baustein, wenn der Geschwindigkeitsfehler auf den halben Wert von **VeloLimit** abgebaut wurde. Beim Wechsel in den inaktiven Zustand wird **Active** gelöscht.

Im aktiven Zustand wird der Stellwert mit der Zeitkonstanten **UpdateFilter** an die Istgeschwindigkeit angeglichen. Ist die Zeitkonstante auf 0.0 gesetzt, wird die Istgeschwindigkeit direkt übernommen.

Im inaktiven Zustand wird **Activated** gelöscht, wenn die Achse den Ruhezustand verlässt und eine aktive Bewegung beginnt.

ⓘ HINWEIS! Da der Baustein die Differenz zwischen Soll- und Istgeschwindigkeit auswertet ist die korrekte Einstellung der Referenzgeschwindigkeit beim Einsatz des Bausteins wichtig. Eine zu ungenaue Einstellung dieses Parameters kann dazu führen, dass der Baustein ohne Notwendigkeit in die Bewegung eingreift.

Der nachstehende Scope View zeigt eine Positionierung, in deren Verlauf zwei mal auf ein Hindernis aufgefahren wurde. Dadurch kam die Achse jeweils vollständig zum Stillstand.



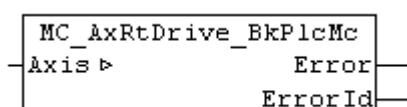
Integration des Bausteins in die Applikation

Ein Baustein dieses Typs muss nach der Istwert-Erfassung und Stellwert-Generierung aufgerufen werden. Der Baustein überlagert seine Reaktion mit der des Stellwert-Generators und trägt sie in die [ST_TcHydAxRtData \[▶ 94\]](#) Struktur ein. Der nach dem Reglerbaustein zu platzierende [MC_AxRtFinish_BkPlcMc \[▶ 164\]](#) Baustein berücksichtigt die Reaktion dann automatisch.

HINWEIS! Durch einen Wert größer als 0 in `FirstAuxParamIdx` kann der Baustein veranlasst werden drei aufeinander folgende Werte in den `fCustomerData` der Parameter-Struktur als `VeloLimit`, `LimitFilter` und `UpdateFilter` zu verwenden. Ist in `Axis.pStAxAuxLabels` die Adresse eines geeigneten `ARRAY[..] OF STRING()` eingetragen werden die Parameter automatisch mit einer Bezeichnung versehen.

3.4.2 Drive

3.4.2.1 MC_AxRtDrive_BkPlcMc (in V3.0)



Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einer Hardware-Baugruppe. Dazu wird in Abhängigkeit des in **Axis.ST_TcHydAxParam** [► 88] als `nDrive_Type` eingestellten Wertes ein Baustein aufgerufen, der die Besonderheiten der Hardware-Baugruppe berücksichtigt.

```
VAR_OUTPUT
    Error:      BOOL;
    ErrorID:    UDINT;
END_VAR
VAR_INOUT
    Axis:       Axis_Ref_BkPlcMc;
END_VAR
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn `nDrive_Type` in `pStAxParams` auf einen nicht zulässigen Wert eingestellt ist reagiert der Baustein mit **Error** und **ErrorID:=dwTcHydErrCdDriveType**. Wenn der Pointer `pStAxRtData` der Achse initialisiert ist wird sie in einen Störzustand versetzt werden.
- Wenn einer der spezifischen Unterbausteine ein Problem erkennt wird dieser (sofern möglich) die Achse in einen Störzustand versetzen. Dieser Fehler wird an den Ausgängen des `MC_AxRtDrive_BkPlcMc` wiederholt.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird der Stellwert der Achse entsprechend `nDrive_Type` [► 67] in **Axis.ST_TcHydAxParam** [► 88] verarbeitet.

Informationen über die erforderlichen Verknüpfung von E/A-Komponenten mit den Eingangs- und Ausgangsstrukturen der Achse finden Sie in der Knowledge Base [► 207] unter FAQ #7.

Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollte zur Vereinfachung ein Baustein des Typs MC_AxStandardBody_BkPlcMc [► 170] verwendet werden.

Für einen asynchronen Datenaustausch mit E/A-Gerät der KL-Reihe stehen die Bausteine MC_AxUtiReadRegDriveTerm_BkPlcMc [► 196] und MC_AxUtiWriteRegDriveTerm_BkPlcMc [► 204] zur Verfügung.

iTcMc_DriveAx2000_B110A

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines AX2000 Servostellers am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Absolut-Encoder ausgerüstet ist. Wird ein Motor mit einem Resolver betrieben ist **iTcMc_DriveAx2000_B110R** einzustellen.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Encoder-Bausteins und werden dort beschrieben. Siehe hierzu im Vergleich auch iTcMc_EncoderAX2000_B110A [► 124].

iTcMc_DriveAx2000_B110R

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe an einen AX2000 Servoverstärker am EtherCAT Feldbus.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Encoder-Bausteins und werden dort beschrieben. Siehe hierzu im Vergleich auch [iTcMc_EncoderAx2000_B110R \[▶ 125\]](#).

iTcMc_DriveAx2000_B200R, iTcMc_DriveAx2000_B900R

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe an einen AX2000 Servoverstärker am Beckhoff LightBus (B200) oder RtEthernet-Feldbus (B900).

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Encoder-Bausteins und werden dort beschrieben. Siehe hierzu im Vergleich auch [iTcMc_EncoderAx2000_B200R \[▶ 126\]](#).

iTcMc_DriveAx2000_B750A

Der Funktionsbaustein übernimmt (ab V3.0.26) die Aufbereitung des Stellwerts der Achse für die Ausgabe an einem AX2000 Servostellers am Sercos Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Absolut-Encoder ausgerüstet ist.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Encoder-Bausteins und werden dort beschrieben. Siehe hierzu im Vergleich auch [iTcMc_EncoderAX2000_B750A \[▶ 127\]](#).

Beachten Sie eine Reihe von Besonderheiten. Näheres hierzu finden Sie in der [Knowledge Base \[▶ 241\]](#).

iTcMc_DriveAx5000_B110A, iTcMc_DriveAx5000_B110SR

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe an einem AX5000 Servostellers am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Absolut-Encoder ausgerüstet ist. Wird ein Motor mit einem Resolver betrieben ist [iTcMc_EncoderAx5000_B110SR](#) einzustellen.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Encoder-Bausteins und werden dort beschrieben. Siehe hierzu im Vergleich auch [iTcMc_EncoderAX5000_B110A \[▶ 128\]](#).

Unter [iTcMc_EncoderAX5000_B110A \[▶ 128\]](#) finden Sie eine Liste mit erfolgreich getesteten kompatiblen Geräten.

Beachten Sie eine Reihe von Besonderheiten. Näheres hierzu finden Sie in der [Knowledge Base \[▶ 241\]](#).

iTcMc_DriveCoE_DS402

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines Servostellers mit CoE DS402 Profil am EtherCAT Feldbus.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Encoder-Bausteins und werden dort beschrieben. Siehe hierzu im Vergleich auch [iTcMc_EncoderCoE_DS402A \[▶ 129\]](#) und [iTcMc_EncoderCoE_DS402SR \[▶ 129\]](#).

Unter [iTcMc_EncoderCoE_DS402SR \[▶ 129\]](#) finden Sie eine Liste mit erfolgreich getesteten kompatiblen Geräten.

ⓘ HINWEIS! Es werden derzeit nur Antriebe mit Resolver oder Singleturn-Encodern unterstützt.

iTcMc_Drive_CoE_DS408

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe an ein Proportionalventil am EtherCAT Feldbus. Das Ventil muss dazu das CiA DS408 Profil unterstützen.

E/A-Variable	Interface.Variable	Verwendung
siehe Hinweis	ST_TcPlcDeviceInput.nDacOut	Ausgabe des Geschwindigkeits-Signals.
siehe Hinweis	ST_TcPlcDeviceOutput.uiDriveCtrl	Steuerung des Geräts.
siehe Hinweis	ST_TcPlcDeviceInput.uiStatus	Status des Geräts
WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung.
uiDriveBoxState	ST_TcPlcDeviceInput.InfoData.State	Überwachung des Onlinestatus
InfoData.AdsAddr	ST_TcPlcDeviceInput.sDrvAdsAddr	Automatische Identifikation.

ⓘ HINWEIS! Die Namen der mit dem Gerät ausgetauschten Prozessdaten werden durch die XML Datei des Herstellers festgelegt.

Die nachstehenden Index.SubIndex-Kombinationen müssen vom Ventil unterstützt werden.

Index	Subindex	Bedeutung
1000	0	Identifikation
1008	0	Gerätebezeichnung (optional)
1018	1	Hersteller-Kennung
1018	2	Gerätetyp

Die nachstehende Liste kompatibler Geräte ist naturgemäß unvollständig. Sie stellt keine Empfehlung dar sondern ist nur als allgemeiner Hinweis zu verstehen. Der problemlose Einsatz der genannten Geräte kann von Beckhoff Automation GmbH nicht garantiert werden. Ist ein Hersteller oder eines seiner Geräte nicht aufgeführt kann ein problemloser Betrieb durchaus gegeben sein, wird jedoch nicht zugesichert.

Hersteller	Typ	Beschreibung
Moog	D638Exxx	Proportionalventil
Parker	DxxFP /DxxFE /TDP /TPQ	Proportionalventil

x: Stellt einen Platzhalter für verschiedene Zeichen dar.

iTcMc_DriveI2512_1Coil

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einen PWM-Feldbusmodul IP2512.

E/A-Variable	Interface.Variable	Verwendung
Daten Aus	ST_TcPlcDeviceOutput.uiDacOutA	Ausgabe des PWM-Faktors.

iTcMc_DriveI2512_2Coil

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einen PWM-Feldbusmodul IP2512.

E/A-Variable	Interface.Variable	Verwendung
Daten Aus	ST_TcPlcDeviceOutput.uiDacOutA	Ausgabe des PWM-Faktors für Spule 1.
Daten Aus	ST_TcPlcDeviceOutput.uiDacOutB	Ausgabe des PWM-Faktors für Spule 2.

iTcMc_DriveEL4132

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einer ± 10 V Ausgabeklemme.

E/A-Variable	Interface.Variable	Verwendung
Output	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals.
InfoData.State	ST_TcPlcDeviceInput.uiDriveBoxState	Verbindungsüberwachung, Zustandsüberwachung.
WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung.

iTcMc_DriveEL7031

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einer Schrittmotorendstufenklemme EL7031.

E/A-Variable	Interface.Variable	Verwendung
STM Velocity.Velocity	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals.
STM Control.Control	ST_TcPlcDeviceOutput.uiDriveCtrl	Betrieb: Steuerung der Endstufe.
STM Status.Status	ST_TcPlcDeviceInput.uiStatus	Betrieb: Status der Endstufe.
WcState.WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung.
InfoData.State	ST_TcPlcDeviceInput.uiDriveBoxState	Verbindungsüberwachung, Zustandsüberwachung.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sDrvAdsAddr	Kommunikation.

iTcMc_DriveEL7041

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einer Schrittmotorendstufenklemme EL7041.

E/A-Variable	Interface.Variable	Verwendung
STM Velocity.Velocity	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals.
STM Control.Control	ST_TcPlcDeviceOutput.uiDriveCtrl	Betrieb: Steuerung der Endstufe.
STM Status.Status	ST_TcPlcDeviceInput.uiStatus	Betrieb: Status der Endstufe.
ENC Status.Counter Value	ST_TcPlcDeviceInput.uiCount	Betrieb: Einlesen der Ist-Position.
ENC Status.Latch Value	ST_TcPlcDeviceInput.uiLatch	Betrieb: Einlesen der Latch-Position.
ENC Status.Status	ST_TcPlcDeviceInput.uiTerminalState2	Betrieb: Status der Encoder-Schnittstelle.
ENC Control.Control	ST_TcPlcDeviceOutput.uiTerminalCtrl2	Betrieb: Steuerung der Encoder-Schnittstelle.
WcState.WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung.
InfoData.State	ST_TcPlcDeviceInput.uiDriveBoxState	Verbindungsüberwachung, Zustandsüberwachung.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sDrvAdsAddr	Kommunikation.

iTcMc_DriveKL2521

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf eine Pulsausgabeklemme KL2521.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Encoder-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_EncoderKL2521](#) [► 134].

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation [► 221]: Schnittstelle für gelesene Daten.
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Register-Kommunikation
Status	ST_TcPlcDeviceInput.bTerminalState	Register-Kommunikation
Daten Aus	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Register-Kommunikation: Schnittstelle für geschriebene Daten.

iTcMc_DriveKL2531

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf eine Schrittmotorendstufenklemme KL2531.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Encoder-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_EncoderKL2531](#) [► 135].

E/A-Variable	Interface.Variable	Verwendung
Velocity	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei Register-Kommunikation ▶ 221 : Schnittstelle für geschriebene Daten.
Position	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Ctrl	ST_TcPlcDeviceOutput.bTerminalCtrl	Steuerung der Endstufe, Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Status der Endstufe, Register-Kommunikation.
ExtStatus	ST_TcPlcDeviceInput.uiTerminalState2	Diagnose von Endstufe und Motor

iTcMc_DriveKL2532

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf eine DC-Motorendstufenklemme KL2532.

E/A-Variable	Interface.Variable	Verwendung
Daten ein	ST_TcPlcDeviceInput.uiTerminalData	Bei Register-Kommunikation ▶ 221 : Schnittstelle für gelesene Daten.
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Register-Kommunikation
Daten aus	ST_TcPlcDeviceOutput.nDacOut	Register-Kommunikation

iTcMc_DriveKL2535_1Coil, iTcMc_DriveKL2535_2Coil

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf eine PWM-Endstufenklemme KL2535.

E/A-Variable	Interface.Variable	Verwendung
Daten ein	ST_TcPlcDeviceInput.uiTerminalData	Register-Kommunikation ▶ 221
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Register-Kommunikation
Daten aus	ST_TcPlcDeviceOutput.nDacOut	Register-Kommunikation

iTcMc_DriveKL2541

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf eine Schrittmotorendstufenklemme KL2541.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Encoder-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_EncoderKL2541 ▶ 135](#).

E/A-Variable	Interface.Variable	Verwendung
Velocity	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei <u>Register-Kommunikation</u> [▶ 221]: Schnittstelle für geschriebene Daten.
Position	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Ctrl	ST_TcPlcDeviceOutput.bTerminalCtrl	Steuerung der Endstufe, Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Status der Endstufe, Register-Kommunikation.
ExtCtrl	ST_TcPlcDeviceOutput.uiTerminalCtrl2	Latch-Steuerung bei Homing mit dem Synchron-Impuls des Encoders
ExtStatus	ST_TcPlcDeviceInput.uiTerminalState2	Diagnose von Endstufe und Motor, Latch-Status bei Homing mit dem Synchron-Impuls des Encoders

iTcMc_DriveKL2542

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf eine DC-Motorendstufenklemme KL2542.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Encoder-Bausteins. Siehe hierzu im Vergleich auch iTcMc_EncoderKL2542 [▶ 136].

E/A-Variable	Interface.Variable	Verwendung
Daten aus	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei <u>Register-Kommunikation</u> [▶ 221]: Schnittstelle für geschriebene Daten.
Daten ein	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Steuerung der Endstufe, Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Status der Endstufe, Register-Kommunikation.

iTcMc_DriveKL4032

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einer ±10 V Ausgabeklemme.

E/A-Variable	Interface.Variable	Verwendung
Daten Aus	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei <u>Register-Kommunikation</u> [▶ 221] : Schnittstelle für geschriebene Daten.
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Register-Kommunikation
Status	ST_TcPlcDeviceInput.bTerminalState	Register-Kommunikation
Daten Ein	ST_TcPlcDeviceOutput.uiTerminalData	Register-Kommunikation: Schnittstelle für gelesene Daten.

iTcMc_DriveLowCostStepper

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf digitalen Ausgangsklemmen. Für die Emulation einer Istposition wird ein Pulszähler aktualisiert, der mit einem iTcMc_EncoderLowCostStepper [\[▶ 138\]](#) Encoder ausgewertet werden kann.

E/A-Variable	Interface.Variable	Verwendung
Ausgang	ST_TcPlcDeviceOutput.nDigOutAp	Nichtinvertierte Ansteuerung der A-Phase.
Ausgang	ST_TcPlcDeviceOutput.nDigOutAn	Invertierte Ansteuerung der A-Phase.
Ausgang	ST_TcPlcDeviceOutput.nDigOutBp	Nichtinvertierte Ansteuerung der B-Phase.
Ausgang	ST_TcPlcDeviceOutput.nDigOutBn	Invertierte Ansteuerung der B-Phase.

iTcMc_DriveLowCostInverter

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf digitalen Ausgangsklemmen für den Betrieb einer Polwendeschutz-Schaltung oder eines Frequenzumrichters mit Festfrequenzen. Beim Einsatz dieses Antriebstyps ist die Beachtung einiger Besonderheiten [\[▶ 274\]](#) erforderlich. Bei der Verknüpfung sind zwei Varianten zu unterscheiden:

Bremse, Freigabe, Richtung und Geschwindigkeitsstufe

Nach dem Aufruf des MC_AxRtFinish_BkPlcMc [\[▶ 164\]](#)- oder MC_AxStandardBody_BkPlcMc [\[▶ 170\]](#)-Bausteins der Achse stehen vier dekodierte Signale zur Verfügung. Um die hier benötigten Signale zu erzeugen sind nach dem Bausteinaufruf die nachstehenden Zusammenfassungen der richtungsspezifischen Signale erforderlich.

Beispiel:

stAxDeviceOut.bDigOutAp:=stAxDeviceOut.bDigOutAp OR stAxDeviceOut.bDigOutBp;

stAxDeviceOut.bDigOutAn:=stAxDeviceOut.bDigOutAn OR stAxDeviceOut.bDigOutBn;

ⓘ HINWEIS! Ab V3.0.11 kann auf dem Ventil-Reiter die Ausgabe eines Absolutwerts aktiviert werden. Die oben gezeigte Zusammenfassung der Signale für diesen Einsatzfall wird dann intern vorgenommen.

E/A-Variable	Interface.Variable	Verwendung
Ausgang	ST_TcPlcDeviceOutput.nDigOutAp	Anwahl der Festfrequenz für den Eilgang.
Ausgang	ST_TcPlcDeviceOutput.nDigOutAn	Anwahl der Festfrequenz für den Schleichgang.
Ausgang	ST_TcPlcDeviceOutput.bMovePos	Vorgabe der Bewegungsrichtung: Positiv.
Ausgang	ST_TcPlcDeviceOutput.bMoveNeg	Vorgabe der Bewegungsrichtung: Negativ.
Ausgang	ST_TcPlcDeviceOutput.bPowerOn	Freigabe der Leistungsstufe.
Ausgang	ST_TcPlcDeviceOutput.bBrakeOff	Ansteuerung der Bremse.
Eingang	ST_TcPlcDeviceInput.bPowerOk	Status des Umrichters: Betriebsbereitschaft.

Bremse, Freigabe und richtungskodierte Geschwindigkeitsstufe

E/A-Variable	Interface.Variable	Verwendung
Ausgang	ST_TcPlcDeviceOutput.nDigOutAp	Anwahl der Festfrequenz für den Eilgang in positiver Bewegungsrichtung.
Ausgang	ST_TcPlcDeviceOutput.nDigOutAn	Anwahl der Festfrequenz für den Schleichgang in positiver Bewegungsrichtung.
Ausgang	ST_TcPlcDeviceOutput.nDigOutBn	Anwahl der Festfrequenz für den Schleichgang in negativer Bewegungsrichtung.
Ausgang	ST_TcPlcDeviceOutput.nDigOutBp	Anwahl der Festfrequenz für den Eilgang in negativer Bewegungsrichtung.
Ausgang	ST_TcPlcDeviceOutput.bPowerOn	Freigabe der Leistungsstufe.
Ausgang	ST_TcPlcDeviceOutput.bBrakeOff	Ansteuerung der Bremse.
Eingang	ST_TcPlcDeviceInput.bPowerOk	Status des Umrichters: Betriebsbereitschaft.

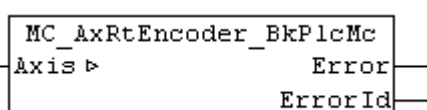
iTcMc_DriveM2400_Dn

Der Funktionsbaustein übernimmt die Aufbereitung des Stellwerts der Achse für die Ausgabe auf einem von vier Kanälen einer ±10 V M2400 Ausgabebox.

E/A-Variable	Interface.Variable	Verwendung
Daten Aus	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals.

3.4.3 Encoder

3.4.3.1 MC_AxRtEncoder_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Ermittlung der Istposition der Achse aus den Eingangsinformationen einer Hardware-Baugruppe. Dazu wird in Abhängigkeit des in **Axis.ST_TcHydAxParam** [► 88] als **nEnc_Type** eingestellten Wertes ein Baustein aufgerufen, der die Besonderheiten der Hardware-Baugruppe berücksichtigt.

```
VAR_OUTPUT
  Error:          BOOL;
  ErrorID:        UDINT;
END_VAR
VAR_INOUT
  Axis:           Axis_Ref_BkPlcMc;
END_VAR
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **nEnc_Type** in pStAxParams auf einen nicht zulässigen Wert eingestellt ist reagiert der Baustein mit **Error** und **ErrorID:=dwTcHydErrCdEncType**. Die Achse wird in einen Störzustand versetzt.
- Wenn einer der spezifischen Unterbausteine ein Problem erkennt wird dieser (sofern möglich) die Achse in einen Störzustand versetzen. Dieser Fehler wird an den Ausgängen des **MC_AxRtEncoder_BkPlcMc** wiederholt.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Istposition der Achse entsprechend **nEnc_Type** [► 70] in **Axis.ST_TcHydAxParam** [► 88] durch Aufruf eines entsprechenden typspezifischen Bausteins ermittelt.

Informationen über die erforderlichen Verknüpfung von E/A-Komponenten mit den Eingangs- und Ausgangsstrukturen der Achse finden Sie in der Knowledge Base unter FAQ #4 [► 211].

Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollte zur Vereinfachung ein Baustein des Typs MC_AxStandardBody_BkPlcMc [► 170] verwendet werden.

Für einen asynchronen Datenaustausch mit E/A-Gerät der KL-Reihe stehen die Bausteine MC_AxUtiReadRegEncTerm_BkPlcMc [► 197] und MC_AxUtiWriteRegEncTerm_BkPlcMc [► 205] zur Verfügung.

iTcMc_EncoderAx2000_B110A

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines AX2000 Servostellers am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Absolut-Encoder ausgerüstet ist. Wird ein Motor mit einem Resolver betrieben ist **iTcMc_EncoderAx2000_B110R** einzustellen.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch iTcMc_DriveAX2000_B110R [► 115].

E/A-Variable	Interface.Variable	Verwendung
Position actual value	ST_TcPlcDeviceInput.ActualPos[0..1]	Ermittlung der Ist-Position.
Status word	ST_TcPlcDeviceInput.uiStatus	Status des Geräts, der Encodernachbildung.
Control word	ST_TcPlcDeviceOutput.uiDriveCtrl	Steuerung des Geräts.
Velocity demand value	ST_TcPlcDeviceOutput.NominalVelocity	Ausgabe des Geschwindigkeits-Stellwerts.
WcState (siehe Hinweis)	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung für die Istwerterfassung.
WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung für den Antrieb.
InfoData.State	ST_TcPlcDeviceInput.uiDriveBoxState	Überwachung des Onlinestatus
InfoData.AdsAddr (siehe Hinweis)	ST_TcPlcDeviceInput.sEncAdsAddr	Parameterkommunikation.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sDrvAdsAddr	Steuerung des Echtzeitstatus, Parameterkommunikation.
Chn0 (siehe Hinweis)	ST_TcPlcDeviceInput.nEncAdsChannel	Parameterkommunikation.
Chn0	ST_TcPlcDeviceInput.nDrvAdsChannel	Steuerung des Echtzeitstatus, Parameterkommunikation.
Ausgang (auf einer DA-Klemme)	ST_TcPlcDeviceOutput.PowerOn	Optionale Ansteuerung des Netzschütz. Hierfür ist eine Digital-Ausgangsklemme vorzusehen.
Eingang (auf einer DE-Klemme)	ST_TcPlcDeviceInput.PowerOk	Optionale Auswertung des Netzschütz. Hierfür ist eine Digital-Eingangsklemme vorzusehen.

ⓘ HINWEIS! Um die Erstellung der E/A-Verknüpfung zu vereinfachen kann dann auf die Verknüpfung von `ST_TcPlcDeviceInput.sEncAdsAddr`, `ST_TcPlcDeviceInput.nEncAdsChannel` und `ST_TcPlcDeviceInput.wEncWcState` verzichtet werden, wenn wie üblich die Istwerterfassung über dasselbe Gerät erfolgt. Die Bausteine für Parameterkommunikation und Encoderauswertung verwenden dann die entsprechenden Variablen der Antriebs-Verknüpfung.

iTcMc_EncoderAx2000_B110R

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines AX2000 Servostellers am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Resolver ausgerüstet ist. Wird ein Motor mit einem Absolut-Encoder betrieben ist `iTcMc_EncoderAx2000_B110A` einzustellen.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch `iTcMc_DriveAX2000_B110R` [► 115].

E/A-Variable	Interface.Variable	Verwendung
Position actual value	ST_TcPlcDeviceInput.ActualPos[0..1]	Ermittlung der Ist-Position.
Status word	ST_TcPlcDeviceInput.uiStatus	Status des Geräts, der Encodernachbildung.
Control word	ST_TcPlcDeviceOutput.uiDriveCtrl	Steuerung des Geräts.
Velocity demand value	ST_TcPlcDeviceOutput.NominalVelocity	Ausgabe des Geschwindigkeits-Stellwerts.
WcState (siehe Hinweis)	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung für die Istwerterfassung.
WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung für den Antrieb.
uiDriveBoxState	ST_TcPlcDeviceInput.InfoData.State	Überwachung des Onlinestatus
InfoData.AdsAddr (siehe Hinweis)	ST_TcPlcDeviceInput.sEncAdsAddr	Parameterkommunikation.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sDrvAdsAddr	Parameterkommunikation.
Chn0 (siehe Hinweis)	ST_TcPlcDeviceInput.nEncAdsChannel	Parameterkommunikation.
Chn0	ST_TcPlcDeviceInput.nDrvAdsChannel	Parameterkommunikation.
Ausgang (auf einer DA-Klemme)	ST_TcPlcDeviceOutput.PowerOn	Optionale Ansteuerung des Netzschütz. Hierfür ist eine Digital-Ausgangsklemme vorzusehen.
Eingang (auf einer DE-Klemme)	ST_TcPlcDeviceInput.PowerOk	Optionale Auswertung des Netzschütz. Hierfür ist eine Digital-Eingangsklemme vorzusehen.

ⓘ HINWEIS! Um die Erstellung der E/A-Verknüpfung zu vereinfachen kann dann auf die Verknüpfung von `ST_TcPlcDeviceInput.sEncAdsAddr`, `ST_TcPlcDeviceInput.nEncAdsChannel` und `ST_TcPlcDeviceInput.wEncWcState` verzichtet werden, wenn wie üblich die Istwerterfassung über dasselbe Gerät erfolgt. Die Bausteine für Parameterkommunikation und Encoderauswertung verwenden dann die entsprechenden Variablen der Antriebs-Verknüpfung.

iTcMc_EncoderAx2000_B200R, iTcMc_EncoderAx2000_B900R

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines AX2000 Servostellers mit LightBus (B200) bzw. RealtimeEthernet (B900).

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveAX2000_B200R](#) [► 116].

E/A-Variable	Interface.Variable	Verwendung
ActualPos[0..1]	ST_TcPlcDeviceInput.ActualPos[0..1]	Ermittlung der Ist-Position.
DriveError	ST_TcPlcDeviceInput.DriveError	Status des Geräts.
DriveState[0..3]	ST_TcPlcDeviceInput.DriveState[0..3]	Status des Geräts.
BoxState	ST_TcPlcDeviceInput.uiDriveBoxState	Verbindungsüberwachung.
DriveCtrl0	ST_TcPlcDeviceOutput.DriveCtrl[0]	Steuerung des Geräts.
DriveCtrl1	ST_TcPlcDeviceOutput.DriveCtrl[1]	Steuerung des Geräts.
DriveCtrl2	ST_TcPlcDeviceOutput.DriveCtrl[2]	Steuerung des Geräts.
DriveCtrl3	ST_TcPlcDeviceOutput.DriveCtrl[3]	Steuerung des Geräts.
NominalVelo	ST_TcPlcDeviceOutput.NominalVelo	Ausgabe des Geschwindigkeits-Stellwerts.
Ausgang (auf einer DA-Klemme)	ST_TcPlcDeviceOutput.PowerOn	Optionale Ansteuerung des Netzschütz. Hierfür ist eine Digital-Ausgangsklemme vorzusehen.
Eingang (auf einer DE-Klemme)	ST_TcPlcDeviceInput.PowerOk	Optionale Auswertung des Netzschütz. Hierfür ist eine Digital-Eingangsklemme vorzusehen.

iTcMc_EncoderAx2000_B750A

Der Funktionsbaustein übernimmt (ab V3.0.26) die Auswertung der Istwerte eines AX2000 Servostellers am Sercos Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Absolut-Encoder ausgerüstet ist.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch iTcMc_DriveAX2000_B750A [▶ 116].

E/A-Variable	Interface.Variable	Verwendung
Antriebs-Statuswort	ST_TcPlcDeviceInput.uiStatus	Status des Geräts.
Lage-Istwert Geber 1	ST_TcPlcDeviceInput.udiCount	Ermittlung der Ist-Position.
Master-Steuerwort	ST_TcPlcDeviceOutput.uiDriveCtrl	Steuerung des Geräts.
Geschwindigkeits-Sollwert	ST_TcPlcDeviceOutput.NominalVelo	Ausgabe des Geschwindigkeits-Stellwerts.
SystemState (vom Sercos Master)	ST_TcPlcDeviceInput.uiDriveBoxState	Überwachung der Sercos-Phase. Hinweis: Diese Variable wird vom Sercos Master (z.B. FC7501) bereitgestellt.
Ausgang (auf einer DA-Klemme)	ST_TcPlcDeviceOutput.PowerOn	Optionale Ansteuerung des Netzschütz. Hierfür ist eine Digital-Ausgangsklemme vorzusehen.
Eingang (auf einer DE-Klemme)	ST_TcPlcDeviceInput.PowerOk	Optionale Auswertung des Netzschütz. Hierfür ist eine Digital-Eingangsklemme vorzusehen.

Beachten Sie eine Reihe von Besonderheiten. Näheres hierzu finden Sie in der [Knowledge Base](#) [▶ 207].

iTcMc_EncoderAx5000_B110A, iTcMc_EncoderAx5000_B110SR

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines AX5000 Servostellers am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Absolut-Encoder ausgerüstet ist. Wird ein Motor mit einem Resolver betrieben ist iTcMc_EncoderAx5000_B110SR einzustellen.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch iTcMc_DriveAX5000_B110A [► 116].

E/A-Variable	Interface.Variable	Verwendung
Position feedback 1 value	ST_TcPlcDeviceInput.udiCount	Ermittlung der Ist-Position.
Drive status word	ST_TcPlcDeviceInput.uiStatus	Status des Geräts.
Master control word	ST_TcPlcDeviceOutput.uiDriveCtrl	Steuerung des Geräts.
Velocity command value	ST_TcPlcDeviceOutput.NominalVelocity	Ausgabe des Geschwindigkeits-Stellwerts.
WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung für den Antrieb.
WcState (siehe Hinweis)	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung für die Istwerterfassung.
InfoData.State	ST_TcPlcDeviceInput.uiDriveBoxState	Überwachung des Onlinestatus
InfoData.AdsAddr	ST_TcPlcDeviceInput.sDrvAdsAddr	Steuerung des Echtzeitstatus, Parameterkommunikation.
InfoData.AdsAddr (siehe Hinweis)	ST_TcPlcDeviceInput.sEncAdsAddr	Parameterkommunikation.
Chn0 (siehe Hinweis 2)	ST_TcPlcDeviceInput.nDrvAdsChannel	für Einfachgeräte oder den ersten Antrieb eines Doppelgeräts: Steuerung des Echtzeitstatus, Parameterkommunikation.
Chn0 (siehe Hinweise 1,2)	ST_TcPlcDeviceInput.nEncAdsChannel	für Einfachgeräte oder den ersten Antrieb eines Doppelgeräts: Parameterkommunikation.
Chn1 (siehe Hinweis 2)	ST_TcPlcDeviceInput.nDrvAdsChannel	nur für den zweiten Antrieb eines Doppelgeräts: Steuerung des Echtzeitstatus, Parameterkommunikation.
Chn1 (siehe Hinweise 1,2)	ST_TcPlcDeviceInput.nEncAdsChannel	nur für den zweiten Antrieb eines Doppelgeräts: Parameterkommunikation.
Ausgang (auf einer DA-Klemme)	ST_TcPlcDeviceOutput.PowerOn	Optionale Ansteuerung des Netzschütz. Hierfür ist eine Digital-Ausgangsklemme vorzusehen.
Eingang (auf einer DE-Klemme)	ST_TcPlcDeviceInput.PowerOk	Optionale Auswertung des Netzschütz. Hierfür ist eine Digital-Eingangsklemme vorzusehen.

Die nachstehende Liste kompatibler Geräte ist naturgemäß unvollständig. Sie stellt keine Empfehlung dar sondern ist nur als allgemeiner Hinweis zu verstehen. Der problemlose Einsatz der genannten Geräte kann von Beckhoff Automation GmbH nicht garantiert werden. Ist ein Hersteller oder eines seiner Geräte nicht aufgeführt kann ein problemloser Betrieb durchaus gegeben sein, wird jedoch nicht zugesichert.

Hersteller	Typ	Beschreibung
Baumüller	b-maxx	Servoregler mit Singleturn-Absolutencoder

HINWEIS! 1 Um die Erstellung der E/A-Verknüpfung zu vereinfachen kann dann auf die Verknüpfung von `ST_TcPlcDeviceInput.sEncAdsAddr`, `ST_TcPlcDeviceInput.nEncAdsChannel` und `ST_TcPlcDeviceInput.wEncWcState` verzichtet werden, wenn wie üblich die Istwerterfassung über dasselbe Gerät erfolgt. Die Bausteine für Parameterkommunikation und Encoderauswertung verwenden dann die entsprechenden Variablen der Antriebs-Verknüpfung.

HINWEIS! 2 Die Variablen `Chn0` und `Chn2` sind für die Unterscheidung der Kanäle eines Doppelgeräts erforderlich. Für den ersten Antrieb des Geräts ist `Chn0` zu verbinden, für den zweiten entsprechend `Chn1`. Bei Einfachgeräten ist wie für den ersten Kanal eines Doppelgeräts zu verfahren.

HINWEIS! Beachten Sie eine Reihe von Besonderheiten. Näheres hierzu finden Sie in der Knowledge Base.

iTcMc_EncoderCoE_DS402A

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines Servostellers mit CoE DS402 Profil am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Multiturn-Absolutencoder ausgerüstet ist.

HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveCoE_DS402](#) [▶ 117].

E/A-Variable	Interface.Variable	Verwendung
siehe Hinweis	<code>ST_TcPlcDeviceInput.udiCount</code>	Ermittlung der Ist-Position.
	<code>ST_TcPlcDeviceInput.uiStatus</code>	
	<code>ST_TcPlcDeviceOutput.uiDriveCtrl</code>	
	<code>ST_TcPlcDeviceOutput.NominalVelocity</code>	
<code>WcState</code>	<code>ST_TcPlcDeviceInput.wDriveWcState</code>	Verbindungsüberwachung.
<code>InfoData.State</code>	<code>ST_TcPlcDeviceInput.uiDriveBoxState</code>	Überwachung des Onlinestatus
<code>InfoData.AdsAddr</code>	<code>ST_TcPlcDeviceInput.sEncAdsAddr</code>	Automatische Identifikation.

HINWEIS! Die Namen der mit dem Gerät ausgetauschten Prozessdaten werden durch die XML Datei des Herstellers festgelegt.

Eine Liste mit kompatiblen Geräten finden Sie weiter unten.

iTcMc_EncoderCoE_DS402SR

Der Funktionsbaustein übernimmt die Auswertung der Istwerte eines Servostellers mit CoE DS402 Profil am EtherCAT Feldbus. Dabei wird vorausgesetzt, dass der angeschlossene Motor mit einem Resolver oder Singleturn-Absolutencoder ausgerüstet ist.

ⓘ HINWEIS! Der TwinCAT System Manager wird beim manuellen Einfügen oder bei der automatischen Erkennung eines Antriebsstellers vorschlagen, eine NC-Achse im Projekt einzufügen und mit diesem Steller zu verbinden. Soll dieser Steller mit der Hydraulik Bibliothek kontrolliert werden ist dieser Vorschlag unbedingt abzulehnen.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch iTcMc_DriveCoE_DS402 [▶ 117].

E/A-Variable	Interface.Variable	Verwendung
siehe Hinweis	ST_TcPlcDeviceInput.udiCount	Ermittlung der Ist-Position.
	ST_TcPlcDeviceInput.uiStatus	
	ST_TcPlcDeviceOutput.uiDriveCtrl	
	ST_TcPlcDeviceOutput.NominalVelocity	
WcState	ST_TcPlcDeviceInput.wDriveWcState	Verbindungsüberwachung.
InfoData.State	ST_TcPlcDeviceInput.uiDriveBoxState	Überwachung des Onlinestatus
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Automatische Identifikation.

ⓘ HINWEIS! Die Namen der mit dem Gerät ausgetauschten Prozessdaten werden durch die XML Datei des Herstellers festgelegt.

Die nachstehenden Index.SubIndex-Kombinationen müssen vom Encoder unterstützt werden.

Index	Subindex	Bedeutung
1000	0	Identifikation
1008	0	Gerätebezeichnung (optional)
1018	1	Hersteller-Kennung
1018	2	Gerätetyp
6080	0	Maximal-Drehzahl in RPM (optional, wenn dieses Objekt nicht unterstützt wird ist die Bezugsgeschwindigkeit manuell einzugeben).
608F	1	Anzahl Encoder-Inkremete pro Motorumdrehung.
6090	1	Für die Stellwertausgabe verwendete Anzahl der Inkremete pro Motorumdrehung.

Die nachstehende Liste kompatibler Geräte ist naturgemäß unvollständig. Sie stellt keine Empfehlung dar sondern ist nur als allgemeiner Hinweis zu verstehen. Der problemlose Einsatz der genannten Geräte kann von Beckhoff Automation GmbH nicht garantiert werden. Ist ein Hersteller oder eines seiner Geräte nicht aufgeführt kann ein problemloser Betrieb durchaus gegeben sein, wird jedoch nicht zugesichert.

Hersteller	Typ	Beschreibung
LTi DRIVES GmbH		Servoregler mit Singleturn-Absolutencoder

iTcMc_EncoderCoE_DS406

Der Funktionsbaustein übernimmt die Auswertung von Encodern mit direktem EtherCAT-Anschluss. Der Encoder muss dazu das CiA DS406 Profil unterstützen.

E/A-Variable	Interface.Variable	Verwendung
siehe Hinweis	ST_TcPlcDeviceInput.udiCount	Ermittlung der Ist-Position.
siehe Hinweise	ST_TcPlcDeviceInput.wEncDevState	Überwachung des Gerätestatus.
WcState	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung.
InfoData.State	ST_TcPlcDeviceInput.uiEncBoxState	Überwachung des Onlinestatus.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Automatische Identifikation.

HINWEIS! Die Namen der mit dem Gerät ausgetauschten Prozessdaten werden durch die XML Datei des Herstellers festgelegt.

HINWEIS! Die Überwachung des Gerätestatus ist nicht für alle Geräte aller Hersteller garantiert. Bei einigen Geräten wird ein 8-Bit Status bereitgestellt. Eine solche Information ist auf den unteren 8 Bit des wEncDevState Elements abzubilden.

Die nachstehenden Index.SubIndex-Kombinationen müssen vom Encoder unterstützt werden.

Index	Subindex	Bedeutung
1000	0	Identifikation
1008	0	Gerätebezeichnung (optional)
1018	1	Hersteller-Kennung
1018	2	Gerätetyp
6001	0	Rotatorische Encoder: Inkremente pro Umdrehung (obligatorisch)
6002	0	Rotatorische Encoder: Gesamter Zählbereich (Option A, alternativ: Index 6502) Lineare Encoder: Gesamter Zählbereich (obligatorisch)
6005	1	Lineare Encoder: Auflösung (Option A, alternativ: Index 6501)
6501	0	Lineare Encoder: Auflösung (Option B, alternativ: Index 6005)
6502	0	Rotatorische Encoder: Anzahl der gezählten Umdrehungen (Option B, alternativ: Index 6002)
650A	2	Lineare Encoder: Untere Grenze des vorgesehenen Arbeitsfeldes (Option)
650B	3	Lineare Encoder: Obere Grenze des vorgesehenen Arbeitsfeldes (Option)

Die nachstehende Liste kompatibler Geräte ist naturgemäß unvollständig. Sie stellt keine Empfehlung dar sondern ist nur als allgemeiner Hinweis zu verstehen. Der problemlose Einsatz der genannten Geräte kann von Beckhoff Automation GmbH nicht garantiert werden. Ist ein Hersteller oder eines seiner Geräte nicht aufgeführt kann ein problemloser Betrieb durchaus gegeben sein, wird jedoch nicht zugesichert.

Abhängig von der Unterstützung der aufgelisteten Objekte können bestimmte Parameter automatisch ermittelt werden. Dies gilt für den Zählbereich bzw. die Überlauf-Erkennung und (für lineare Encoder) die Auflösung. Werden die betreffenden Objekte nicht oder nicht in einer unterstützten Kombination bereitgestellt ist dies nicht möglich. In einem solchen Fall kann ein Betrieb möglich sein. Allerdings sind dann die Parameter manuell bei der Inbetriebnahme einzustellen.

Hersteller	Typ	Beschreibung
Fritz Kübler GmbH	58x8	Multiturn-Absolutencoder.
IVO GmbH & Co. KG	GXMMW_H	Multiturn-Absolutencoder.
MTS	Temposonics R	Linear-Absolutencoder.
TR Electronic GmbH	LMP	Linear-Absolutencoder.
TWK-Electronic GmbH	CRKxx12R12C1xx	Multiturn-Absolutencoder.

iTcMc_EncoderDigCam

Der Funktionsbaustein übernimmt die Auswertung von vier digitalen Eingängen als Positionsnocken.

E/A-Variable	Interface.Variable	Verwendung
Eingang	ST_TcPlcDeviceInput.bDigCamPP	Ermittlung der Ist-Position: Positive Zielnocke.
Eingang	ST_TcPlcDeviceInput.bDigCamP	Ermittlung der Ist-Position: Positive Bremsnocke.
Eingang	ST_TcPlcDeviceInput.bDigCamM	Ermittlung der Ist-Position: Negative Bremsnocke.
Eingang	ST_TcPlcDeviceInput.bDigCamMM	Ermittlung der Ist-Position: Negative Zielnocke.

iTcMc_EncoderDigIncrement

Der Funktionsbaustein übernimmt die Auswertung von zwei digitalen Eingängen für die Emulation einer Inkremental-Encoderauswertung.

E/A-Variable	Interface.Variable	Verwendung
Eingang	ST_TcPlcDeviceInput.bDigInA	Ermittlung der Ist-Position.
Eingang	ST_TcPlcDeviceInput.bDigInB	Ermittlung der Ist-Position.

iTcMc_EncoderEL3102

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme EL3102.

E/A-Variable	Interface.Variable	Verwendung
Value	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Adress-Informationen für Parameter-Kommunikation via CoE.
InfoData.State	ST_TcPlcDeviceInput.uiEncBoxState	Verbindungsüberwachung, Zustandsüberwachung.
WcState	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung.

iTcMc_EncoderEL3142

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme EL3142. Das Mapping erfolgt wie bei der schnittstellenkompatiblen EL3102.

iTcMc_EncoderEL3162

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme EL3162. Das Mapping erfolgt wie bei der schnittstellenkompatiblen EL3102.

iTcMc_EncoderEL3255

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme EL3255.

E/A-Variable	Interface.Variable	Verwendung
AI Standard Channel x.Value	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
AI Standard Channel x.Status	ST_TcPlcDeviceInput.wEncDevState	Auswertung des Fehlersignals des Encoders.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Adress-Informationen für Parameter-Kommunikation via CoE.
InfoData.State	ST_TcPlcDeviceInput.uiEncBoxState	Verbindungsüberwachung, Zustandsüberwachung.
WcState	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung.

HINWEIS! Die Klemme unterstützt bis zu fünf Encoder. Die Variablen **InfoData.AdsAddr**, **InfoData.State** und **WcState** sind durch Mehrfach-Mapping auf alle beteiligten Achsen zu verteilen.

iTcMc_EncoderEL5001

Der Funktionsbaustein übernimmt die Auswertung der Daten einer SSI-Encoderklemme EL5001.

E/A-Variable	Interface.Variable	Verwendung
Value	ST_TcPlcDeviceInput.udiCount	Einlesen der Ist-Position.
Status	ST_TcPlcDeviceOutput.usiRegStatus	Auswertung des Fehlersignals des Encoders.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Adress-Informationen für Parameter-Kommunikation via CoE.
InfoData.State	ST_TcPlcDeviceInput.uiEncBoxState	Verbindungsüberwachung, Zustandsüberwachung.
WcState	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung.

iTcMc_EncoderEL5101

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Inkremental-Encoderklemme EL5101.

E/A-Variable	Interface.Variable	Verwendung
Value	ST_TcPlcDeviceInput.uiCount	Betrieb: Einlesen der Ist-Position.
Latch	ST_TcPlcDeviceInput.uiLatch	Beim Homing unter Verwendung des Synchron-Impulses des Encoders.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Steuerung der Latch-Funktion usw.
Status	ST_TcPlcDeviceInput.usiStatus	Status des Encoders, der Latchfunktion.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Adress-Informationen für Parameter-Kommunikation via CoE.
InfoData.State	ST_TcPlcDeviceInput.uiEncBoxState	Verbindungsüberwachung, Zustandsüberwachung.
WcState	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung.

iTcMc_EncoderEL5111

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Inkremental-Encoderklemme EL5111.

E/A-Variable	Interface.Variable	Verwendung
Value	ST_TcPlcDeviceInput.uiCount	Betrieb: Einlesen der Ist-Position.
Latch	ST_TcPlcDeviceInput.uiLatch	Beim Homing unter Verwendung des Synchron-Impulses des Encoders.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Steuerung der Latch-Funktion usw.
Status	ST_TcPlcDeviceInput.usiStatus	Status des Encoders, der Latchfunktion.
InfoData.AdsAddr	ST_TcPlcDeviceInput.sEncAdsAddr	Adress-Informationen für Parameter-Kommunikation via CoE.
InfoData.State	ST_TcPlcDeviceInput.uiEncBoxState	Verbindungsüberwachung, Zustandsüberwachung.
WcState	ST_TcPlcDeviceInput.wEncWcState	Verbindungsüberwachung.

iTcMc_EncoderEL7041

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Schrittmotor-Ausgangsklemme EL7041.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveEL7041](#) [► 119].

iTcMc_EncoderIx5009

Der Funktionsbaustein übernimmt die Auswertung der Daten einer SSI-Encoderbox IP5009.

E/A-Variable	Interface.Variable	Verwendung
PZDL_RegDaten	ST_TcPlcDeviceInput.uiPZDL_RegDaten	Betrieb: Einlesen der Ist-Position. Bei <u>Register-Kommunikation</u> [► 221]: Schnittstelle für gelesene Daten.
PZDH	ST_TcPlcDeviceInput.uiPZDH	Einlesen der Ist-Position.
RegStatus	ST_TcPlcDeviceInput.usiRegStatus	Diverse Status-Informationen.

iTcMc_EncoderKL2521

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Pulsausgabeklemme KL2521. Es werden die ausgegebenen Pulse gezählt und für eine Encoder-Emulation benutzt.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveKL2521](#) [► 119].

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei <u>Register-Kommunikation</u> [▶ 221] : Schnittstelle für gelesene Daten.
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Register-Kommunikation
Status	ST_TcPlcDeviceInput.bTerminalState	Register-Kommunikation
Daten Aus	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Register-Kommunikation: Schnittstelle für geschriebene Daten.

iTcMc_EncoderKL2531

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Pulsausgabeklemme KL2531. Es werden die ausgegebenen Pulse gezählt und für eine Encoder-Emulation benutzt.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveKL2531 \[▶ 119\]](#).

E/A-Variable	Interface.Variable	Verwendung
Velocity	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei <u>Register-Kommunikation</u> [▶ 221] : Schnittstelle für geschriebene Daten.
Position	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Ctrl	ST_TcPlcDeviceOutput.bTerminalCtrl	Steuerung der Endstufe, Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Status der Endstufe, Register-Kommunikation.
ExtStatus	ST_TcPlcDeviceInput.uiTerminalState2	Diagnose von Endstufe und Motor

iTcMc_EncoderKL2541

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Pulsausgabeklemme KL2541. Es werden die ausgegebenen Pulse gezählt und für eine Encoder-Emulation benutzt.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveKL2541 \[▶ 120\]](#).

E/A-Variable	Interface.Variable	Verwendung
Velocity	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei Register-Kommunikation [► 221] : Schnittstelle für geschriebene Daten.
Position	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Ctrl	ST_TcPlcDeviceOutput.bTerminalCtrl	Steuerung der Endstufe, Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Status der Endstufe, Register-Kommunikation.
ExtCtrl	ST_TcPlcDeviceOutput.uiTerminalCtrl2	Latch-Steuerung bei Homing mit dem Synchron-Impuls des Encoders
ExtStatus	ST_TcPlcDeviceInput.uiTerminalState2	Diagnose von Endstufe und Motor, Latch-Status bei Homing mit dem Synchron-Impuls des Encoders

iTcMc_EncoderKL2542

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Motorendstufenklemme KL2542.

Dieses E/A-Gerät gehört zu einer Gruppe von Geräten, die sowohl für die Stellwert-Ausgabe als auch für die Istwert-Ermittlung benutzt werden. Die benötigten Mapping-Definitionen insbesondere zur Parameter-Kommunikation überschneiden sich mit denen des Drive-Bausteins. Siehe hierzu im Vergleich auch [iTcMc_DriveKL2542 \[► 121\]](#).

E/A-Variable	Interface.Variable	Verwendung
Daten aus	ST_TcPlcDeviceOutput.nDacOut	Betrieb: Ausgabe des Geschwindigkeits-Signals. Bei Register-Kommunikation [► 221] : Schnittstelle für geschriebene Daten.
Daten ein	ST_TcPlcDeviceInput.uiTerminalData	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Kontrolle	ST_TcPlcDeviceOutput.bTerminalCtrl	Steuerung der Endstufe, Register-Kommunikation.
Status	ST_TcPlcDeviceInput.bTerminalState	Status der Endstufe, Register-Kommunikation.

iTcMc_EncoderKL3002

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme KL3002.

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Register-Kommunikation [► 221]
Status	ST_TcPlcDeviceInput.usiStatus	Register-Kommunikation.

iTcMc_EncoderKL3042

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme KL3042.

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Register-Kommunikation [▶ 221]
Status	ST_TcPlcDeviceInput.usiStatus	Register-Kommunikation.

iTcMc_EncoderKL3062

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme KL3062.

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Register-Kommunikation [▶ 221]
Status	ST_TcPlcDeviceInput.usiStatus	Register-Kommunikation.

iTcMc_EncoderKL3162

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsklemme KL3162.

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Register-Kommunikation [▶ 221]
Status	ST_TcPlcDeviceInput.usiStatus	Register-Kommunikation.

iTcMc_EncoderKL5001

Der Funktionsbaustein übernimmt die Auswertung der Daten einer SSI-Encoderklemme KL5001.

E/A-Variable	Interface.Variable	Verwendung
PZDL_RegDaten	ST_TcPlcDeviceInput.uiPZDL_Reg Daten	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation [▶ 221]: Schnittstelle für gelesene Daten.
PZDH	ST_TcPlcDeviceInput.uiPZDH	Einlesen der Ist-Position.
RegStatus	ST_TcPlcDeviceInput.usiRegStatus	Diverse Status-Informationen.
RegDaten	ST_TcPlcDeviceOutput.bTerminal Data	Register-Kommunikation.

iTcMc_EncoderKL5101

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Inkremental-Encoderklemme KL5101.

E/A-Variable	Interface.Variable	Verwendung
Counter	ST_TcPlcDeviceInput.uiCount	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation : Schnittstelle für gelesene Daten.
Latch	ST_TcPlcDeviceInput.uiLatch	Beim Homing unter Verwendung des Synchron-Impulses des Encoders.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Steuerung der Latch-Funktion usw., Register-Kommunikation [▶ 221]
Status	ST_TcPlcDeviceInput.usiStatus	Diverse Status-Informationen.
RegDaten	ST_TcPlcDeviceOutput.bTerminal Data	Register-Kommunikation.

iTcMc_EncoderKL5111

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Inkremental-Encoderklemme KL5111.

E/A-Variable	Interface.Variable	Verwendung
Counter	ST_TcPlcDeviceInput.uiCount	Betrieb: Einlesen der Ist-Position. Bei Register-Kommunikation: Schnittstelle für gelesene Daten.
Latch	ST_TcPlcDeviceInput.uiLatch	Beim Homing unter Verwendung des Synchron-Impulses des Encoders.
Ctrl	ST_TcPlcDeviceOutput.usiCtrl	Steuerung der Latch-Funktion usw., <u>Register-Kommunikation</u> ▶ 221
Status	ST_TcPlcDeviceInput.usiStatus	Diverse Status-Informationen.
RegDaten	ST_TcPlcDeviceOutput.bTerminal Data	Register-Kommunikation.

iTcMc_EncoderLowCostStepper

Wenn als nDrive_Type der Wert iTcMc_DriveLowCostStepper [▶ 122](#) eingetragen ist werden in ST_TcPlcDeviceOutput.uiCount die ausgegebenen Halbschritte gezählt. Daraus wird die Istposition errechnet. Ein Mapping ist für den Encoder nicht erforderlich.

ⓘ HINWEIS! Dieser Encoder-Typ ist nur in Kombination mit einem iTcMc_DriveLowCostStepper Antrieb nutzbar.

iTcMc_EncoderM2510

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Analog-Eingangsbox M2510.

E/A-Variable	Interface.Variable	Verwendung
Daten Ein	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.

iTcMc_EncoderM3120

Der Funktionsbaustein übernimmt die Auswertung der Daten einer Inkremental-Encoderbox M3120.

E/A-Variable	Interface.Variable	Verwendung
Value_N	ST_TcPlcDeviceInput.uiCount	Einlesen der Ist-Position.
State_N	ST_TcPlcDeviceInput.usiStatus	Diverse Status-Informationen.
Ctrl_N	ST_TcPlcDeviceOutput.usiCtrl	Steuerung der Latch-Funktion usw.

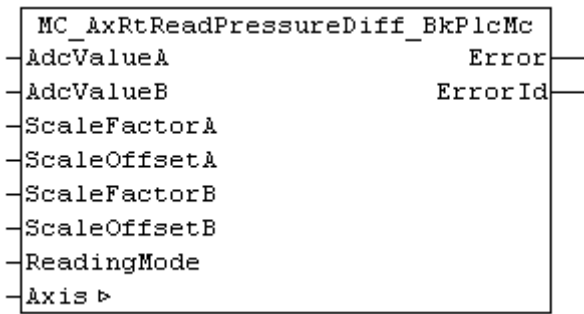
iTcMc_EncoderSim

Ein Simulationsencoder errechnet die Istposition durch Integration der Sollgeschwindigkeit. Es ist kein Mapping erforderlich.

Sehen Sie dazu auch

 Knowledge Base [▶ 207](#)

3.4.3.2 MC_AxRtReadForceDiff_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Ermittlung der Istkraft der Achse aus den Eingangsdaten von zwei Analog-Eingangsklemmen. Dabei wird der Istdruck der A- und B-Seite unter Berücksichtigung der Flächen und der Gleitreibung in die an der Last wirksam werdende Kraft umgerechnet.

HINWEIS! Steht nur ein Eingangssignal zur Verfügung sollte ein Baustein des Typs

MC_AxRtReadForceSingle_BkPlcMc [▶ 141] verwendet werden. Soll der Istdruck ermittelt werden ist ein Baustein vom Typ **MC_AxRtReadPressureDiff_BkPlcMc** [▶ 143] einzusetzen.

```
VAR_INPUT
  AdcValueA:      INT:=0;
  AdcValueB:      INT:=0;
  ScaleFactorA:   LREAL:=0.0;
  ScaleOffsetA:   LREAL:=0.0;
  ScaleFactorB:   LREAL:=0.0;
  ScaleOffsetB:   LREAL:=0.0;
  SlippingOffset: LREAL:=0.0;
  ReadingMode:    E_TcMcPressureReadingMode:=iTcHydPressureReadingDefault;
END_VAR
```

E_TcMcPressureReadingMode [▶ 84]

```
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Error:         BOOL;
  ErrorID:       UDINT;
END_VAR
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

AdcValueA, AdcValueB: Hier sind die Eingangsdaten von den Analog-Klemmen zu übergeben.

ScaleFactorA, ScaleFactorB: Dieser Wert stellt die Gewichtung dar. Er legt fest, welche Druckerhöhung einer Stufe des AD-Wandlers entspricht.

ScaleOffsetA, ScaleOffsetB: Dieser Offset dient dazu, den Nullpunkt der Druckskala zu korrigieren.

SlippingOffset: Wenn der Baustein für die Berechnung der wirksam werdenden Kraft genutzt wird kann hier die für die Überwindung der Gleitreibung benötigte Kraft eingetragen werden.

ReadingMode: Hier kann die zu ermittelnde Istgröße festgelegt werden. Als Defaultziel wird **Axis_Ref_BkPlcMc** [▶ 62].**ST_TcHydAxRtData** [▶ 94].**fActPressure** ausgewählt.

Axis: Hier ist die Adresse einer Variablen vom Typ **Axis_Ref_BkPlcMc** [▶ 62] zu übergeben.

Verhalten des Bausteins

Der Baustein ermittelt durch Auswertung der Variablen **AdcValueA** und **AdcValueB** den Istdruck bzw. die Istkraft der Achse. Das Ergebnis wird in **ST_TcHydAxRtData** [▶ 94].**fActPressure** eingetragen.

HINWEIS! Die Speicherung von einer Achse zugeordneten Parametern kann zum Beispiel in den `ST_TcHydAxParam.fCustomerData[...]` erfolgen. So ist sichergestellt, dass die Daten zusammen mit den Standard-Parametern der Achse geladen, gespeichert und gesichert und bei Bedarf auch exportiert und importiert werden.

Ermittlung eines Differenz-Istdrucks

Für die Inbetriebnahme sind drei Vorgehensweisen üblich.

Inbetriebnahme-Variante A (bevorzugt für $\pm 10V$)

Hierbei ist keine Bewegung der Achse erforderlich. Die erreichbare Genauigkeit ist bei hochwertigen Drucksensoren in den meisten Fällen ausreichend.

- Als **ScaleFactorA** und **ScaleFactorB** ist der Bemessungsdruck der Drucksensoren dividiert durch **AdcValueA_{MAX}** bzw. **AdcValueB_{MAX}** einzutragen.
- Wird der Baustein zur Ermittlung des Istdrucks verwendet sind die Parameter **ScaleArreaA** und **ScaleArreaB** auf den Wert 1.0 einzustellen. Andernfalls sind diese Parameter für eine Istkraft in N (= Newton) in mm² anzugeben.

Inbetriebnahme-Variante B

Hierbei ist es erforderlich, das in beide Richtungen ein Block mit vollem Systemdruck angefahren werden kann. Dabei ist eine echte Bewegung der Achse nicht erforderlich. Also kann das Anfahren der Anschläge dadurch nachgebildet werden, dass die Achse durch Einlegen von provisorischen Begrenzungen nur einen Teil ihres Fahrweges zurücklegen kann oder sogar mechanisch vollständig fixiert wird.

- Es sind alle Bausteine zu deaktivieren, die auf den Wert von `ST_TcHydAxRtData [► 94].fActPressure` reagieren.
- Mit niedriger Geschwindigkeit wird zunächst der untere (in Richtung sinkender Istposition gelegene) Block angefahren. Dabei wird für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Jetzt sollte auf der A-Seite der Systemdruck und auf der B-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln.
- Mit niedriger Geschwindigkeit wird anschließend der obere (in Richtung steigender Istposition gelegene) Block angefahren. Dabei wird erneut für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Auch jetzt sind die Drücke festzustellen.
- Die einzutragenden Parameter errechnen sich dann wie folgt:

$$\text{ScaleFactorA} := (\text{DruckA}_{\text{MAX}} - \text{DruckA}_{\text{MIN}}) / (\text{AdcValueA}_{\text{MAX}} - \text{AdcValueA}_{\text{MIN}});$$

$$\text{ScaleFactorB} := (\text{DruckB}_{\text{MAX}} - \text{DruckB}_{\text{MIN}}) / (\text{AdcValueB}_{\text{MAX}} - \text{AdcValueB}_{\text{MIN}});$$

$$\text{ScaleOffsetA} := \text{DruckA}_{\text{MIN}} - \text{ScaleFactorA} * \text{AdcValueA}_{\text{MIN}}$$

$$\text{ScaleOffsetB} := \text{DruckB}_{\text{MIN}} - \text{ScaleFactorB} * \text{AdcValueB}_{\text{MIN}}$$

Inbetriebnahme-Variante C

Ersatzweise kann eine Inbetriebnahme der Erfassung ohne Ansteuerung der Achse erfolgen. Dabei wird jedoch eine weitaus geringere Genauigkeit erreicht.

- Zunächst ist die Achse druckfrei zu machen. Dazu ist der Kompressor still zu setzen und der Druckspeicher zu entlasten.
- Es ist sicher zu stellen, dass die Achse keinen Druck aufbaut. Dazu ist eine von außen mit Kräften (Gravitation usw.) beaufschlagte Achse mechanisch abzustützen. Das Ventil ist mehrfach manuell oder elektrisch in beide Richtungen zu öffnen.
- Jetzt wird für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Es sollte sowohl auf der A- als auch auf der B-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln. Die so gefundenen Werte sind in den oben genannten Gleichungen als **MIN** Werte einzusetzen.

- Aus den Datenblattangaben der Drucksensoren ist der Druck für die obere Grenze des elektrischen Signals (10 Volt, 20 mA) zu entnehmen. Als **AdcValueA** und **AdcValueB** ist der obere Grenzwert für die gewandelte elektrische Größe anzunehmen. Diese Werte sind als **MAX** Werte in die oben genannten Gleichungen einzusetzen.
- Die einzutragenden Parameter errechnen sich dann wie oben beschrieben.

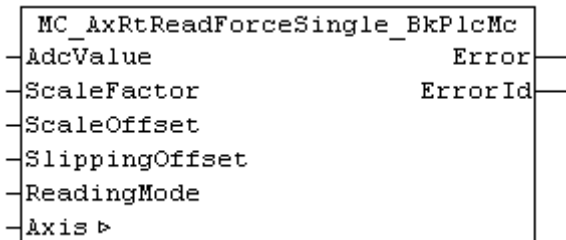
Ermittlung einer wirksam werden Kraft

Für die Ermittlung einer wirksam werdenden Kraft ist zunächst wie oben beschrieben der Istdruck zu ermitteln. Durch Eintragen der wirksamen Flächen unter **ScaleArreaA** und **ScaleArreaB** wird der Baustein veranlasst, die beidseitigen Drücke unter Berücksichtigung der Flächen in Kräfte umzurechnen.

Sehen Sie dazu auch

ST_TcHydAxParam (ab V3.0) [▶ 88]

3.4.3.3 MC_AxRtReadForceSingle_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Ermittlung der Istkraft der Achse aus den Eingangsdaten einer Analog-Eingangsklemme. Dabei wird der Istdruck der A- oder B-Seite unter Berücksichtigung der Fläche und der Gleitreibung in die an der Last wirksam werdende Kraft umgerechnet.



Hinweis

Steht nur ein Eingangssignal zur Verfügung sollte ein Baustein des Typs MC_AxRtReadForceDiff_BkPlcMc [▶ 139] verwendet werden. Soll der Istdruck ermittelt werden ist ein Baustein vom Typ MC_AxRtReadPressureDiff_BkPlcMc [▶ 143] einzusetzen.

```

VAR_INPUT
  AdcValueA:      INT:=0;
  AdcValueB:      INT:=0;
  ScaleFactorA:   LREAL:=0.0;
  ScaleOffsetA:   LREAL:=0.0;
  ScaleFactorB:   LREAL:=0.0;
  ScaleOffsetB:   LREAL:=0.0;
  SlippingOffset: LREAL:=0.0;
  ReadingMode:    E_TcMcPressureReadingMode:=iTcHydPressureReadingDefault;
END_VAR
VAR_INOUT
  Axis:           Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Error:          BOOL;
  ErrorID:        UDINT;
END_VAR
  
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

AdcValueA, AdcValueB: Hier sind die Eingangsdaten von den Analog-Klemmen zu übergeben.

ScaleFactorA, ScaleFactorB: Dieser Wert stellt die Gewichtung dar. Er legt fest, welche Druckerhöhung einer Stufe des AD-Wandlers entspricht.

ScaleOffsetA, ScaleOffsetB: Dieser Offset dient dazu, den Nullpunkt der Druckskala zu korrigieren.

SlippingOffset: Wenn der Baustein für die Berechnung der wirksam werdenden Kraft genutzt wird kann hier die für die Überwindung der Gleitreibung benötigte Kraft eingetragen werden.

ReadingMode: Hier kann die zu ermittelnde Istgröße festgelegt werden. Als Defaultziel wird Axis_Ref_BkPlcMc [▶ 62].ST_TcHydAxRtData [▶ 94].**fActPressure** ausgewählt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

Verhalten des Bausteins

Der Baustein ermittelt durch Auswertung der Variablen **AdcValueA** den Istdruck bzw. die Istkraft der Achse. Das Ergebnis wird in ST_TcHydAxRtData [▶ 94].**fActPressure** eingetragen.

Die Speicherung von einer Achse zugeordneten Parametern kann zum Beispiel in den ST_TcHydAxParam [▶ 88].**fCustomerData**[...] erfolgen. So ist sicher gestellt, dass die Daten zusammen mit den Standard-Parametern der Achse geladen, gespeichert und gesichert und bei Bedarf auch exportiert und importiert werden.

Ermittlung eines Differenz-Istdrucks

Wird der Baustein zur Ermittlung des Istdrucks benutzt sind die Parameter **ScaleArreaA** und **ScaleArreaB** auf den Wert 1.0 und **SlippingOffset** auf den Wert 0.0 einzustellen.

Inbetriebnahme-Variante A

Hierbei ist keine Bewegung der Achse erforderlich. Die erreichbare Genauigkeit ist bei hochwertigen Drucksensoren in den meisten Fällen ausreichend.

- Als **ScaleFactorA** ist der Bemessungsdruck der Drucksensoren dividiert durch **AdcValueA_{MAX}** einzutragen.

Inbetriebnahme-Variante B

Hierbei ist es erforderlich, das in beide Richtungen ein Block mit vollem Systemdruck angefahren werden kann. Dabei ist eine echte Bewegung der Achse nicht erforderlich. Also kann das Anfahren der Anschläge dadurch nachgebildet werden, dass die Achse durch Einlegen von provisorischen Begrenzungen nur einen Teil ihres Fahrweges zurücklegen kann oder sogar mechanisch vollständig fixiert wird.

- Es sind alle Bausteine zu deaktivieren, die auf den Wert von ST_TcHydAxRtData [▶ 94].**fActPressure** reagieren.
- Mit niedriger Geschwindigkeit wird zunächst der untere (in Richtung sinkender Istposition gelegene) Block angefahren. Dabei wird für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Jetzt sollte auf der A-Seite der Systemdruck und auf der B-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln.
- Mit niedriger Geschwindigkeit wird anschließend der obere (in Richtung steigender Istposition gelegene) Block angefahren. Dabei wird erneut für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Auch jetzt sind die Drücke festzustellen.
- Die einzutragenden Parameter errechnen sich dann wie folgt:

$$\mathbf{ScaleFactorA} := (\text{DruckA}_{\text{MAX}} - \text{DruckA}_{\text{MIN}}) / (\mathbf{AdcValueA}_{\text{MAX}} - \mathbf{AdcValueA}_{\text{MIN}});$$

$$\mathbf{ScaleFactorB} := (\text{DruckB}_{\text{MAX}} - \text{DruckB}_{\text{MIN}}) / (\mathbf{AdcValueB}_{\text{MAX}} - \mathbf{AdcValueB}_{\text{MIN}});$$

$$\mathbf{ScaleOffsetA} := \text{DruckA}_{\text{MIN}} - \mathbf{ScaleFactorA} * \mathbf{AdcValueA}_{\text{MIN}}$$

$$\mathbf{ScaleOffsetB} := \text{DruckB}_{\text{MIN}} - \mathbf{ScaleFactorB} * \mathbf{AdcValueB}_{\text{MIN}}$$

Inbetriebnahme-Variante C

Ersatzweise kann eine Inbetriebnahme der Erfassung ohne Ansteuerung der Achse erfolgen. Dabei wird jedoch eine weitaus geringere Genauigkeit erreicht.

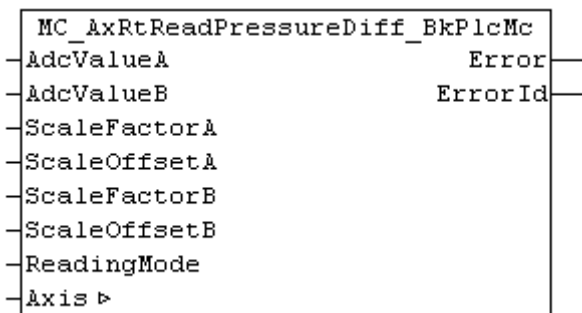
- Zunächst ist die Achse druckfrei zu machen. Dazu ist der Kompressor still zu setzen und der Druckspeicher zu entlasten.

- Es ist sicher zu stellen, dass die Achse keinen Druck aufbaut. Dazu ist eine von außen mit Kräften (Gravitation usw.) beaufschlagte Achse mechanisch abzustützen. Das Ventil ist mehrfach manuell oder elektrisch in beide Richtungen zu öffnen.
- Jetzt wird für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Es sollte sowohl auf der A- als auch auf der B-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln. Die so gefundenen Werte sind in den oben genannten Gleichungen als **MIN** Werte einzusetzen.
- Aus den Datenblattangaben der Drucksensoren ist der Druck für die obere Grenze des elektrischen Signals (10 Volt, 20 mA) zu entnehmen. Als **AdcValueA** und **AdcValueB** ist der obere Grenzwert für die gewandelte elektrische Größe anzunehmen. Diese Werte sind als **MAX** Werte in die oben genannten Gleichungen einzusetzen.
- Die einzutragenden Parameter errechnen sich dann wie oben beschrieben.

Ermittlung einer wirksam werden Kraft

Für die Ermittlung einer wirksam werdenden Kraft ist zunächst wie oben beschrieben der Istdruck zu ermitteln. Durch Eintragen der wirksamen Fläche unter **ScaleArreaA** wird der Baustein veranlasst, den einseitigen Druck unter Berücksichtigung der Fläche in eine Kraft umzurechnen.

3.4.3.4 MC_AxRtReadPressureDiff_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Ermittlung des Istdrucks der Achse aus den Eingangsdaten von zwei Analog-Eingangsklemmen.



Hinweis

Steht nur ein Eingangssignal zur Verfügung sollte ein Baustein des Typs MC_AxRtReadPressureSingle_BkPlcMc [► 145] verwendet werden. Soll nicht der Druck sondern die Kraft ermittelt werden ist ein Baustein vom Typ MC_AxRtReadForceDiff_BkPlcMc [► 139] einzusetzen.

```

VAR_INPUT
  AdcValueA:      INT:=0;
  AdcValueB:      INT:=0;
  ScaleFactorA:   LREAL:=0.0;
  ScaleOffsetA:   LREAL:=0.0;
  ScaleFactorB:   LREAL:=0.0;
  ScaleOffsetB:   LREAL:=0.0;
  ReadingMode:    E_TcMcPressureReadingMode:=iTcHydPressureReadingDefault;
END_VAR
VAR_INOUT
  Axis:           Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Error:          BOOL;
  ErrorID:        UDINT;
END_VAR
    
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

AdcValueA, AdcValueB: Hier sind die Eingangsdaten von den Analog-Klemmen zu übergeben.

ScaleFactorA, ScaleFactorB: Dieser Wert stellt die Gewichtung dar. Er legt fest, welche Druckerhöhung einer Stufe des AD-Wandlers entspricht.

ScaleOffsetA, ScaleOffsetB: Dieser Offset dient dazu, den Nullpunkt der Druckskala zu korrigieren.

ReadingMode: Mit diesem Parameter wird festgelegt, wo das Ergebnis der Auswertung abzulegen ist.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei kann ein Problem erkannt und gemeldet werden:

- Ist der Pointer pStAxRtData in Axis_Ref_BkPlcMc [► 62] nicht initialisiert reagiert der Baustein mit **Error** und **ErrorID:=dwTcHydErrCdPtrMcPlc**. Die Achse kann in diesem Fall nicht in einen Störszustand versetzt werden.

Wenn diese Überprüfung ohne Problem durchgeführt werden konnte wird der Istdruck der Achse durch Auswertung der Variablen **AdcValueA** und **AdcValueB** ermittelt. Das Ergebnis wird in ST_TcHydAxRtData [► 94].fActPressure eingetragen.

Die Speicherung von einer Achse zugeordneten Parametern kann zum Beispiel in den ST_TcHydAxParam [► 88].fCustomerData[...] erfolgen. So ist sichergestellt, dass die Daten zusammen mit den Standard-Parametern der Achse geladen, gespeichert und gesichert und bei Bedarf auch exportiert und importiert werden.

Inbetriebnahme-Variante A

Hierbei ist keine Bewegung der Achse erforderlich. Die erreichbare Genauigkeit ist bei hochwertigen Drucksensoren in den meisten Fällen ausreichend.

- Als **ScaleFactorA** und **ScaleFactorB** ist der Bemessungsdruck der Drucksensoren dividiert durch **AdcValueA_{MAX}** bzw. **AdcValueB_{MAX}** einzutragen.

Inbetriebnahme-Variante B

Hierbei ist keine Bewegung der Achse erforderlich. Die erreichbare Genauigkeit ist bei hochwertigen Drucksensoren in den meisten Fällen ausreichend.

- Als **ScaleFactorA** und **ScaleFactorB** ist der Bemessungsdruck der Drucksensoren dividiert durch **AdcValueA_{MAX}** bzw. **AdcValueB_{MAX}** einzutragen.

Inbetriebnahme-Variante C

Hierbei ist es erforderlich, das in beide Richtungen ein Block mit vollem Systemdruck angefahren werden kann. Dabei ist eine echte Bewegung der Achse nicht erforderlich. Also kann das Anfahren der Anschläge dadurch nachgebildet werden, dass die Achse durch Einlegen von provisorischen Begrenzungen nur einen Teil ihres Fahrweges zurücklegen kann oder sogar mechanisch vollständig fixiert wird.

- Es sind alle Bausteine zu deaktivieren, die auf den Wert von ST_TcHydAxRtData [► 94].fActPressure reagieren.
- Mit niedriger Geschwindigkeit wird zunächst der untere (in Richtung sinkender Istposition gelegene) Block angefahren. Dabei wird für **AdcValueA** und **AdcValueB** der Wert zu ermittelt und festgehalten. Jetzt sollte auf der B-Seite der Systemdruck und auf der A-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln.
- Mit niedriger Geschwindigkeit wird anschließend der obere (in Richtung steigender Istposition gelegene) Block angefahren. Dabei wird erneut für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Auch jetzt sind die Drücke festzustellen.
- Die einzutragenden Parameter errechnen sich dann wie folgt:

$$\mathbf{ScaleFactorA} := (\text{DruckA}_{\text{MAX}} - \text{DruckA}_{\text{MIN}}) / (\mathbf{AdcValueA}_{\text{MAX}} - \mathbf{AdcValueA}_{\text{MIN}});$$

$$\mathbf{ScaleFactorB} := (\text{DruckB}_{\text{MAX}} - \text{DruckB}_{\text{MIN}}) / (\mathbf{AdcValueB}_{\text{MAX}} - \mathbf{AdcValueB}_{\text{MIN}});$$

$$\text{ScaleOffsetA} := \text{DruckA}_{\text{MIN}} - \text{ScaleFactorA} * \text{AdcValueA}_{\text{MIN}}$$

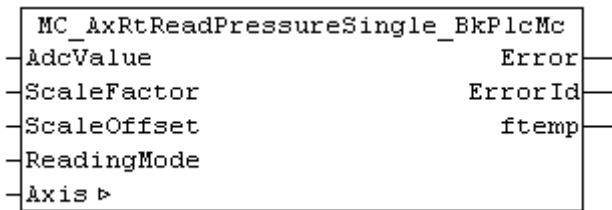
$$\text{ScaleOffsetB} := \text{DruckB}_{\text{MIN}} - \text{ScaleFactorB} * \text{AdcValueB}_{\text{MIN}}$$

Inbetriebnahme-Variante D

Ersatzweise kann eine Inbetriebnahme der Erfassung ohne Ansteuerung der Achse erfolgen. Dabei wird jedoch eine weitaus geringere Genauigkeit erreicht.

- Zunächst ist die Achse druckfrei zu machen. Dazu ist der Kompressor still zu setzen und der Druckspeicher zu entlasten.
- Es ist sicher zu stellen, dass die Achse keinen Druck aufbaut. Dazu ist eine von außen mit Kräften (Gravitation usw.) beaufschlagte Achse mechanisch abzustützen. Das Ventil ist mehrfach manuell oder elektrisch in beide Richtungen zu öffnen.
- Jetzt wird für **AdcValueA** und **AdcValueB** der Wert ermittelt und festgehalten. Es sollte sowohl auf der A- als auch auf der B-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln. Die so gefundenen Werte sind in den oben genannten Gleichungen als **MIN** Werte einzusetzen.
- Aus den Datenblattangaben der Drucksensoren ist der Druck für die obere Grenze des elektrischen Signals (10 Volt, 20 mA) zu entnehmen. Als **AdcValueA** und **AdcValueB** ist der obere Grenzwert für die gewandelte elektrische Größe anzunehmen. Diese Werte sind als **MAX** Werte in die oben genannten Gleichungen einzusetzen.
- Die einzutragenden Parameter errechnen sich dann wie oben beschrieben.

3.4.3.5 MC_AxRtReadPressureSingle_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Ermittlung des Istdrucks der Achse aus den Eingangsdaten einer Analog-Eingangsklemme.



Hinweis

Stehen zwei Eingangssignale für die A- und B-Seite zur Verfügung sollte ein Baustein des Typs [MC_AxRtReadPressureDiff_BkPlcMc](#) [► 143] verwendet werden.

```

VAR_INPUT
  AdcValue:      INT:=0;
  ScaleFactor:   LREAL:=0.0;
  ScaleOffset:   LREAL:=0.0;
  ReadingMode:   E_TcMcPressureReadingMode:=iTcHydPressureReadingDefault;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Error:         BOOL;
  ErrorID:       UDINT;
END_VAR
    
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

AdcValue: Hier sind die Eingangsdaten von der Analog-Klemme zu übergeben.

ScaleFactor: Dieser Wert stellt die Gewichtung dar. Er legt fest, welche Druckerhöhung einer Stufe des AD-Wandlers entspricht.

ScaleOffset: Dieser Offset dient dazu, den Nullpunkt der Druckskala zu korrigieren.

ReadingMode: Hier kann die zu ermittelnde Istgröße festgelegt werden. Als Defaultwert wird [Axis_Ref_BkPlcMc \[► 62\].ST_TcHydAxRtData \[► 94\].fActPressure](#) ausgewählt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei kann ein Problem erkannt und gemeldet werden:

- Ist der Pointer `pStAxRtData` in [Axis_Ref_BkPlcMc \[► 62\]](#) nicht initialisiert reagiert der Baustein mit **Error** und **ErrorID:=dwTcHydErrCdPtrMcPlc**. Die Achse kann in diesem Fall nicht in einen Störzustand versetzt werden.

Wenn diese Überprüfung ohne Problem durchgeführt werden konnte wird der Istdruck der Achse durch Auswertung der Variablen **AdcValue** ermittelt. Das Ergebnis wird in [ST_TcHydAxRtData \[► 94\].fActPressure](#) eingetragen.



Hinweis

Die Speicherung von einer Achse zugeordneten Parametern kann zum Beispiel in den [ST_TcHydAxParam \[► 88\].fCustomerData\[...\]](#) erfolgen. So ist sichergestellt, dass die Daten zusammen mit den Standard-Parametern der Achse geladen, gespeichert und gesichert und bei Bedarf auch exportiert und importiert werden.

Inbetriebnahme-Variante A

Hierbei ist es erforderlich, das in beide Richtungen ein Block mit vollem Systemdruck angefahren werden kann. Dabei ist eine echte Bewegung der Achse nicht erforderlich. Also kann das Anfahren der Anschläge dadurch nachgebildet werden, dass die Achse durch Einlegen von provisorischen Begrenzungen nur einen Teil ihres Fahrweges zurücklegen kann oder sogar mechanisch vollständig fixiert wird.

- Es sind alle Bausteine zu deaktivieren, die auf den Wert von [ST_TcHydAxRtData \[► 94\].fActPressure](#) reagieren.
- Mit niedriger Geschwindigkeit wird zunächst der untere (in Richtung sinkender Istposition gelegene) Block angefahren. Dabei wird für **AdcValue** der Wert zu ermittelt und festgehalten. Jetzt sollte auf der B-Seite der Systemdruck und auf der A-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall sind die A- und B-seitigen Drücke durch Messung zu ermitteln.
- Mit niedriger Geschwindigkeit wird anschließend der obere (in Richtung steigender Istposition gelegene) Block angefahren. Dabei wird erneut für **AdcValue** der Wert zu ermittelt und festgehalten. Auch jetzt sind die Drücke festzustellen.
- Die einzutragenden Parameter errechnen sich dann wie folgt:

$$\text{ScaleFactor} := (\text{Druck}_{\text{MAX}} - \text{Druck}_{\text{MIN}}) / (\text{AdcValue}_{\text{MAX}} - \text{AdcValue}_{\text{MIN}});$$

$$\text{ScaleOffset} := \text{Druck}_{\text{MIN}} - \text{ScaleFactor} * \text{AdcValue}_{\text{MIN}}$$

Inbetriebnahme-Variante B

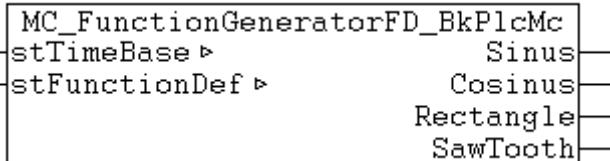
Ersatzweise kann eine Inbetriebnahme der Erfassung ohne Ansteuerung der Achse erfolgen. Dabei wird jedoch eine weitaus geringere Genauigkeit erreicht.

- Zunächst ist die Achse druckfrei zu machen. Dazu ist der Kompressor still zu setzen und der Druckspeicher zu entlasten.
- Es ist sicher zu stellen, dass die Achse keinen Druck aufbaut. Dazu ist eine von außen mit Kräften (Gravitation usw.) beaufschlagte Achse mechanisch abzustützen. Das Ventil ist mehrfach manuell oder elektrisch in beide Richtungen zu öffnen.
- Jetzt wird für **AdcValue** der Wert ermittelt und festgehalten. Es sollte sowohl auf der A- als auch auf der B-Seite der Tankdruck und somit der Umgebungsdruck anstehen. Ist dies aus irgendwelchen Gründen nicht der Fall ist der A-seitige Druck durch Messung zu ermitteln. Die so gefundenen Werte sind in den oben genannten Gleichungen als **MIN** Werte einzusetzen.

- Aus den Datenblattangaben der Drucksensoren ist der Druck für die obere Grenze des elektrischen Signals (10 Volt, 20 mA) zu entnehmen. Als **AdcValue** ist der obere Grenzwert für die gewandelte elektrische Größe anzunehmen. Diese Werte sind als **MAX** Werte in die oben genannten Gleichungen einzusetzen.
- Die einzutragenden Parameter errechnen sich dann wie oben beschrieben.

3.4.4 FunctionGenerator

3.4.4.1 MC_FunctionGeneratorFD_BkPlcMc (ab V3.0.31)



Der Funktionsbaustein berechnet die Signale eines Funktionsgenerators.

```

VAR_OUTPUT
  Sinus:          LREAL;
  Cosinus:        LREAL;
  Rectangle:      LREAL;
  SawTooth:       LREAL;
END_VAR
VAR_INOUT
  stTimeBase:     ST_FunctionGeneratorTB_BkPlcMc;
  stFunctionDef:  ST_FunctionGeneratorFD_BkPlcMc;
END_VAR
    
```

Sinus, Cosinus, Rectangle, SawTooth: Die Ausgangssignale des Funktionsgenerators.

stTimeBase: Eine Struktur mit den Parametern der Zeitbasis dieses Funktionsgenerators.

stFunctionDef: Eine Struktur mit den Definitionen der Ausgangssignale eines Funktionsgenerators.

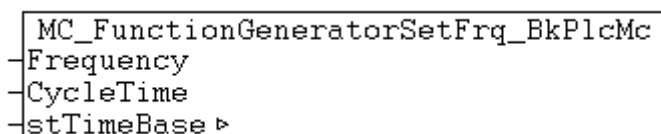
Verhalten des Bausteins

Aus **stTimeBase.CurrentRatio** und den Parametern in [stFunctionDef \[▶ 87\]](#) werden die Ausgangssignale ermittelt.

Die Zeitbasis in **stTimeBase** ist mit einem [MC_FunctionGeneratorTB_BkPlcMc \[▶ 148\]](#)(**)** Baustein zu aktualisieren.

Zur Veränderung der Arbeitsfrequenz sollte ein [MC_FunctionGeneratorSetFrq_BkPlcMc \[▶ 147\]](#)(**)** Baustein verwendet werden.

3.4.4.2 MC_FunctionGeneratorSetFrq_BkPlcMc (ab V3.0.31)



Der Funktionsbaustein aktualisiert die Arbeitsfrequenz einer Zeitbasis für einen oder mehrere Funktionsgeneratoren [\[▶ 147\]](#).

```

VAR_INPUT
  Frequency:    LREAL;
  CycleTime:    LREAL;
END_VAR
VAR_INOUT
  stTimeBase:   ST_FunctionGeneratorTB_BkPlcMc;
END_VAR

```

Frequency: Die zu verwendende Arbeitsfrequenz.

CycleTime: Die Zykluszeit der aufrufenden Task.

stTimeBase: Eine Struktur mit den Parametern der Zeitbasis eines oder mehrerer [Funktionsgeneratoren](#) [► 87].

Verhalten des Bausteins

Der Baustein setzt **stTimeBase.Frequency** auf den übergebenen Wert. Dabei wird **stTimeBase.CurrentTime** bei Bedarf angepasst.

Der Baustein verhindert mit Hilfe von **stTimeBase Freeze** eine Kollision mit [MC_FunctionGeneratorTB_BkPlcMc](#) [► 148]() Bausteinen. Somit kann er auch aus einer anderen Task aufgerufen werden.

3.4.4.3 MC_FunctionGeneratorTB_BkPlcMc (ab V3.0.31)

```

MC_FunctionGeneratorTB_BkPlcMc
- CycleTime
- stTimeBase ▶

```

Der Funktionsbaustein aktualisiert eine Zeitbasis für einen oder mehrere [Funktionsgeneratoren](#) [► 147].

```

VAR_INPUT
  CycleTime:    LREAL;
END_VAR
VAR_INOUT
  stTimeBase:   ST_FunctionGeneratorTB_BkPlcMc;
END_VAR

```

CycleTime: Die Zykluszeit der aufrufenden Task.

stTimeBase: Eine Struktur mit den Parametern der Zeitbasis eines oder mehrerer [Funktionsgeneratoren](#) [► 87].

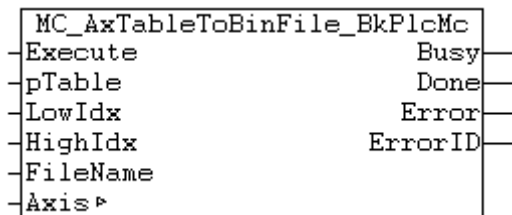
Verhalten des Bausteins

Wenn **stTimeBase.Freeze** nicht gesetzt ist wird **stTimeBase.CurrentTime** mit **CycleTime** aktualisiert und **stTimeBase.CurrentRatio** ermittelt. Dabei wird **stTimeBase.Frequency** berücksichtigt.

Zur Veränderung der Arbeitsfrequenz sollte ein [MC_FunctionGeneratorSetFrq_BkPlcMc](#) [► 147]() Baustein verwendet werden.

3.4.5 TableFunctions

3.4.5.1 MC_AxTableToBinFile_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt den Inhalt einer Tabelle in eine binäre Datei.

```

VAR_INPUT
  Execute:    BOOL:=FALSE;
  pTable:    POINTER TO LREAL:=0;
  LowIdx:    INT:=0;
  HighIdx:   INT:=0;
  FileName:  STRING(255):='';
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Busy:      BOOL;
  Done:      BOOL;
  Error:     BOOL;
  ErrorID:   UDINT;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

pTable: Hier ist die Adresse eines ARRAY[nFirstIdx..nLastIdx,1..2] zu übergeben.

LowIdx: Hier ist der untere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

HighIdx: Hier ist der obere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

FileName: Hier kann ein Dateiname vorgegeben werden.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Referenzfahrt signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein die übergebenen Parameter. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **LowIdx** negativ ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **pTable=0** ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **LowIdx** und **HighIdx** eine Tabelle mit weniger als fünf Zeilen beschreiben wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.

Sind diese Überprüfungen ohne Probleme durchgeführt wird der Schreibvorgang gestartet. Für die Dauer des Vorgangs ist **Busy** auf TRUE. Dabei kann es zu einigen weiteren Problemen kommen, die durch verschiedene Fehlercodes signalisiert werden. Ein erfolgreiches Schreiben der Datei wird mit **Done** signalisiert.

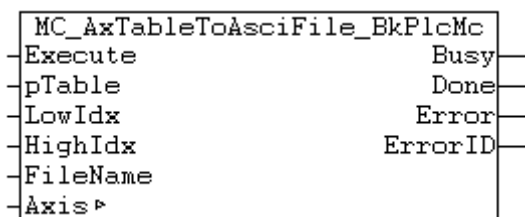
Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Vorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende des Vorgangs (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

Wird ein **FileName** vorgegeben muss dieser vollständig (falls nötig inklusive Angabe des Laufwerks und des Pfades, immer inklusive Dateityp) sein, da er vom Baustein ohne jede weitere Veränderung oder Ergänzung benutzt wird.

Wird kein **FileName** vorgegeben verwendet der Baustein den Pfad und den Dateinamen, der durch den [MC_AxUtiStandardInit_BkPlcMc \[► 171\]](#) Baustein festgelegt wurde. Zur Unterscheidung von der Parameterdatei mit Dateityp DAT wird hier der Typ TBL verwendet.

❗ HINWEIS! Der Inhalt der Datei ist nicht mit einem ASCII-Editor les- oder änderbar.

3.4.5.2 MC_AxTableToAsciiFile_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt den Inhalt einer Tabelle in eine Textdatei.

```

VAR_INPUT
  Execute:    BOOL:=FALSE;
  pTable:     POINTER TO LREAL:=0;
  LowIdx:     INT:=0;
  HighIdx:    INT:=0;
  FileName:   STRING(255):='';
END_VAR
VAR_INOUT
  Axis:       Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Busy:       BOOL;
  Done:       BOOL;
  Error:      BOOL;
  ErrorID:    UDINT;
END_VAR
  
```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

pTable: Hier ist die Adresse eines ARRAY[nFirstIdx..nLastIdx,1..2] zu übergeben.

LowIdx: Hier ist der untere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

HighIdx: Hier ist der obere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

FileName: Hier kann ein Dateiname vorgegeben werden.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Referenzfahrt signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein die übergebenen Parameter. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **LowIdx** negativ ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **pTable=0** ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **LowIdx** und **HighIdx** eine Tabelle mit weniger als fünf Zeilen beschreiben wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.

Sind diese Überprüfungen ohne Probleme durchgeführt wird der Schreibvorgang gestartet. Für die Dauer des Vorgangs ist **Busy** auf TRUE. Dabei kann es zu einigen weiteren Problemen kommen, die durch verschiedene Fehlercodes signalisiert werden. Ein erfolgreiches Schreiben der Datei wird mit **Done** signalisiert.

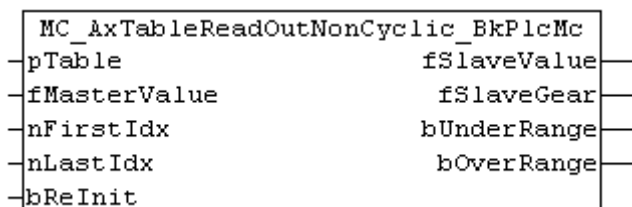
Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Vorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende des Vorgangs (**Error, ErrorID, Done**) werden für einen Zyklus gegeben.

Wird ein **FileName** vorgegeben muss dieser vollständig (falls nötig inklusive Angabe des Laufwerks und des Pfades, immer inklusive Dateityp) sein, da er vom Baustein ohne jede weitere Veränderung oder Ergänzung benutzt wird.

Wird kein **FileName** vorgegeben verwendet der Baustein den Pfad und den Dateinamen, der durch den [MC_AxUtiStandardInit_BkPlcMc \[► 171\]](#) Baustein festgelegt wurde. Zur Unterscheidung von der Parameterdatei mit Dateityp DAT wird hier der Typ TXT verwendet.

ⓘ HINWEIS! Der Inhalt der Datei ist mit einem ASCII-Editor les- oder änderbar. Veränderungen des Inhalts können ein korrektes Lesen oder die vorgesehene Verwendung unmöglich machen oder die Wirkung der Tabelle auf kaum nachvollziehbare Weise verändern. Manuelle Veränderungen sollten darum wenn überhaupt mit der gebotenen Vorsicht und nur von sachkundigen Personen vorgenommen werden.

3.4.5.3 MC_AxTableReadOutNonCyclic_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt mit Hilfe einer Tabelle die einem Master-Wert zugeordneten Slave-Werte.

ⓘ HINWEIS! Dieser Baustein ist eine Komponente von Kurvenscheiben oder ähnlichen nichtlinearen Kopplungen. Er wird in der Regel nicht direkt von einer Applikation aufgerufen.

```

VAR_INPUT
  pTable:      POINTER TO LREAL:=0;
  fMasterValue: LREAL:=0.0;
  nFirstIdx:   UDINT:=1;
  nLastIdx:   UDINT:=1;
  bReInit:    BOOL:=FALSE;
END_VAR
VAR_OUTPUT
  fSlaveValue: LREAL:=0.0;
  fSlaveGear:  LREAL:=0.0;
  bUnderRange: BOOL;
  bOverRange:  BOOL;
END_VAR
    
```

pTable: Hier ist die Adresse eines ARRAY[nFirstIdx..nLastIdx,1..2] zu übergeben.

**Achtung****Absturz der PLC-Applikation**

Eine nicht zutreffende Angabe an dieser Stelle führt zum **Absturz der PLC-Applikation** durch Auslösung von schweren Laufzeitfehlern (**Page Fault Exception**).

fMasterValue: Hier ist der Wert des Masters zu übergeben, für den die zugehörigen Slave-Werte ermittelt werden sollen.

nFirstIdx: Hier ist der untere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

Achtung: Eine nicht zutreffende Angabe an dieser Stelle führt zum **Absturz der PLC-Applikation** durch Auslösung von schweren Laufzeitfehlern (**Page Fault Exception**).

nLastIdx: Hier ist der obere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

**Achtung****Absturz der PLC-Applikation**

Eine nicht zutreffende Angabe an dieser Stelle führt zum **Absturz der PLC-Applikation** durch Auslösung von schweren Laufzeitfehlern (**Page Fault Exception**).

bReInit: Dieser Eingang signalisiert dem Baustein, dass die Suchprozedur am Anfang der Tabelle aufgesetzt werden soll.

fSlaveValue: Hier wird der zum **fMasterValue** gehörende Slave-Wert ausgegeben.

fSlaveGear: Hier wird die lokale Steigung der Slave-Werte an der vom Master-Wert festgelegten Stelle der Tabelle ausgegeben.

bUnderRange: Dieser Ausgang wird TRUE, Wenn der Master-Wert das untere Ende der Tabelle berührt oder unterschreitet.

bOverRange: Dieser Ausgang wird TRUE, Wenn der Master-Wert das obere Ende der Tabelle berührt oder überschreitet.

Verhalten des Bausteins

Der Baustein sucht innerhalb der übergebenen Tabelle nach einem Master-Wertepaar, das den übergebenen **fMasterValue** trifft oder umfasst. Innerhalb des gefundenen Intervalls wird eine lineare Zwischeninterpolation gerechnet. Das Ergebnis wird als **fSlaveValue** ausgegeben. Die bei dieser Berechnung ermittelte lokale Steigung wird als **fSlaveGear** ausgegeben.

Liegt der **fMasterValue** unterhalb des von der Tabelle beschriebenen Wertebereichs wird **bUnderRange** signalisiert. Als **fSlaveValue** wird der Wert ausgegeben, der dem untersten Tabellenpunkt zugeordnet ist. Als **fSlaveGear** wird 0.0 zurückgegeben.

Liegt der **fMasterValue** oberhalb des von der Tabelle beschriebenen Wertebereichs wird **bOverRange** signalisiert. Als **fSlaveValue** wird der Wert ausgegeben, der dem obersten Tabellenpunkt zugeordnet ist. Als **fSlaveGear** wird 0.0 zurückgegeben.

Der Rückgabewert **fSlaveGear** stellt das Verhältnis der ersten Ableitungen von **fMasterValue** und **fSlaveValue** dar. Stellt **fMasterValue** eine Position oder eine virtuelle Zeit dar ergibt die Multiplikation von Master-Fortschrittsgeschwindigkeit und **fSlaveGear** die Slave-Sollgeschwindigkeit. Dies kann für die Erzeugung einer Vorsteuergeschwindigkeit ausgenutzt werden. Dazu ist vorzugsweise ein [MC_AxRtSetExtGenValues_BkPlcMc](#) [▶ 169] Baustein zu verwenden.

3.4.5.4 MC_AxTableFromBinFile_BkPlcMc (ab V3.0)

MC_AxTableFromBinFile_BkPlcMc	
Execute	Busy
pTable	Done
LowIdx	Error
HighIdx	ErrorID
FileName	LastIdx
Axis	

Der Funktionsbaustein liest den Inhalt einer Tabelle aus einer binären Datei.

```

VAR_INPUT
  Execute:    BOOL:=FALSE;
  pTable:    POINTER TO LREAL:=0;
  LowIdx:    INT:=0;
  HighIdx:   INT:=0;
  FileName:  STRING(255) := '';
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Busy:      BOOL;
  Done:      BOOL;
  Error:     BOOL;
  ErrorID:   UDINT;
  LastIdx:   INT:=0;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Lesevorgang.

pTable: Hier ist die Adresse eines ARRAY[nFirstIdx..nLastIdx,1..2] zu übergeben.

LowIdx: Hier ist der untere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

HighIdx: Hier ist der obere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

FileName: Hier kann ein Dateiname vorgegeben werden.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Referenzfahrt signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

LastIdx: Hier wird der Index der letzten durch den Lesevorgang definierte Tabellenzeile signalisiert.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein die übergebenen Parameter. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **LowIdx** negativ ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **pTable=0** ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **LowIdx** und **HighIdx** eine Tabelle mit weniger als fünf Zeilen beschreiben wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.

Sind diese Überprüfungen ohne Probleme durchgeführt wird der Lesevorgang gestartet. Für die Dauer des Vorgangs ist **Busy** auf TRUE. Dabei kann es zu einigen weiteren Problemen kommen, die durch verschiedene Fehlercodes signalisiert werden. Ein erfolgreiches Lesen der Datei wird mit **Done** signalisiert.

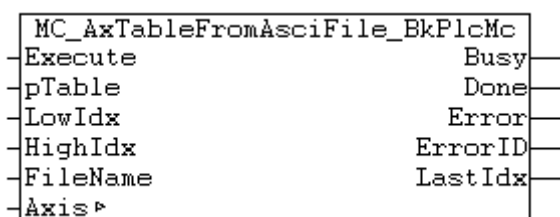
Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Vorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende des Vorgangs (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

Wird ein **FileName** vorgegeben muss dieser vollständig (falls nötig inklusive Angabe des Laufwerks und des Pfades, immer inklusive Dateityp) sein, da er vom Baustein ohne jede weitere Veränderung oder Ergänzung benutzt wird.

Wird kein **FileName** vorgegeben verwendet der Baustein den Pfad und den Dateinamen, der durch den [MC_AxUtiStandardInit_BkPlcMc \[► 171\]](#) Baustein festgelegt wurde. Zur Unterscheidung von der Parameterdatei mit Dateityp DAT wird hier der Typ TBL verwendet.

❗ HINWEIS! Der Inhalt der Datei ist nicht mit einem ASCII-Editor les- oder änderbar.

3.4.5.5 MC_AxTableFromAsciiFile_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest den Inhalt einer Tabelle aus einer Textdatei.

```

VAR_INPUT
  Execute:    BOOL:=FALSE;
  pTable:    POINTER TO LREAL:=0;
  LowIdx:    INT:=0;
  HighIdx:   INT:=0;
  FileName:  STRING(255):='';
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
VAR_OUTPUT
  Busy:      BOOL;
  Done:      BOOL;
  Error:     BOOL;
  ErrorID:   UDINT;
  LastIdx:   INT:=0;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Lesevorgang.

pTable: Hier ist die Adresse eines ARRAY[nFirstIdx..nLastIdx,1..2] zu übergeben.

LowIdx: Hier ist der untere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

HighIdx: Hier ist der obere Index des ARRAY zu übergeben, dessen Adresse als **pTable** übergeben wird.

FileName: Hier kann ein Dateiname vorgegeben werden.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Abarbeitung der Referenzfahrt signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

LastIdx: Hier wird der Index der letzten durch den Lesevorgang definierte Tabellenzeile signalisiert.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein die übergebenen Parameter. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn **LowIdx** negativ ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **pTable=0** ist wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.
- Wenn **LowIdx** und **HighIdx** eine Tabelle mit weniger als fünf Zeilen beschreiben wird mit **Error** und **ErrorID=dwTcHydErrCdTblEntryCount** reagiert.

Sind diese Überprüfungen ohne Probleme durchgeführt wird der Lesevorgang gestartet. Für die Dauer des Vorgangs ist **Busy** auf TRUE. Dabei kann es zu einigen weiteren Problemen kommen, die durch verschiedene Fehlercodes signalisiert werden. Ein erfolgreiches Lesen der Datei wird mit **Done** signalisiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Vorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende des Vorgangs (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

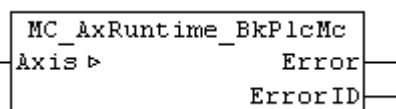
Wird ein **FileName** vorgegeben muss dieser vollständig (falls nötig inklusive Angabe des Laufwerks und des Pfades, immer inklusive Dateityp) sein, da er vom Baustein ohne jede weitere Veränderung oder Ergänzung benutzt wird.

Wird kein **FileName** vorgegeben verwendet der Baustein den Pfad und den Dateinamen, der durch den [MC_AxUtiStandardInit_BkPlcMc \[► 171\]](#) Baustein festgelegt wurde. Zur Unterscheidung von der Parameterdatei mit Dateityp DAT wird hier der Typ TXT verwendet.

ⓘ HINWEIS! Der Inhalt der Datei ist mit einem ASCII-Editor les- oder änderbar. Veränderungen des Inhalts können ein korrektes Lesen oder die vorgesehene Verwendung unmöglich machen oder die Wirkung der Tabelle auf kaum nachvollziehbare Weise verändern. Manuelle Veränderungen sollten darum wenn überhaupt mit der gebotenen Vorsicht und nur von sachkundigen Personen vorgenommen werden.

3.4.6 Generatoren

3.4.6.1 MC_AxRuntime_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Aufgabe des Stellwertgenerators. Dazu wird in Abhängigkeit des in [Axis.ST_TcHydAxParam \[► 88\]](#) als nProfileType eingestellten Wertes ein profilspezifischer Baustein aufgerufen.

```

VAR_OUTPUT
  Error:      BOOL;
  ErrorID:    UDINT;
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
  
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn einer der Pointer nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** oder **dwTcHydErrCdPtrMcPlc** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird die Stellwertgenerierung der Achse entsprechend nProfileType in **Axis.ST_TcHydAxParam** [► 88] durch Aufruf eines entsprechenden Bausteins durchgeführt.

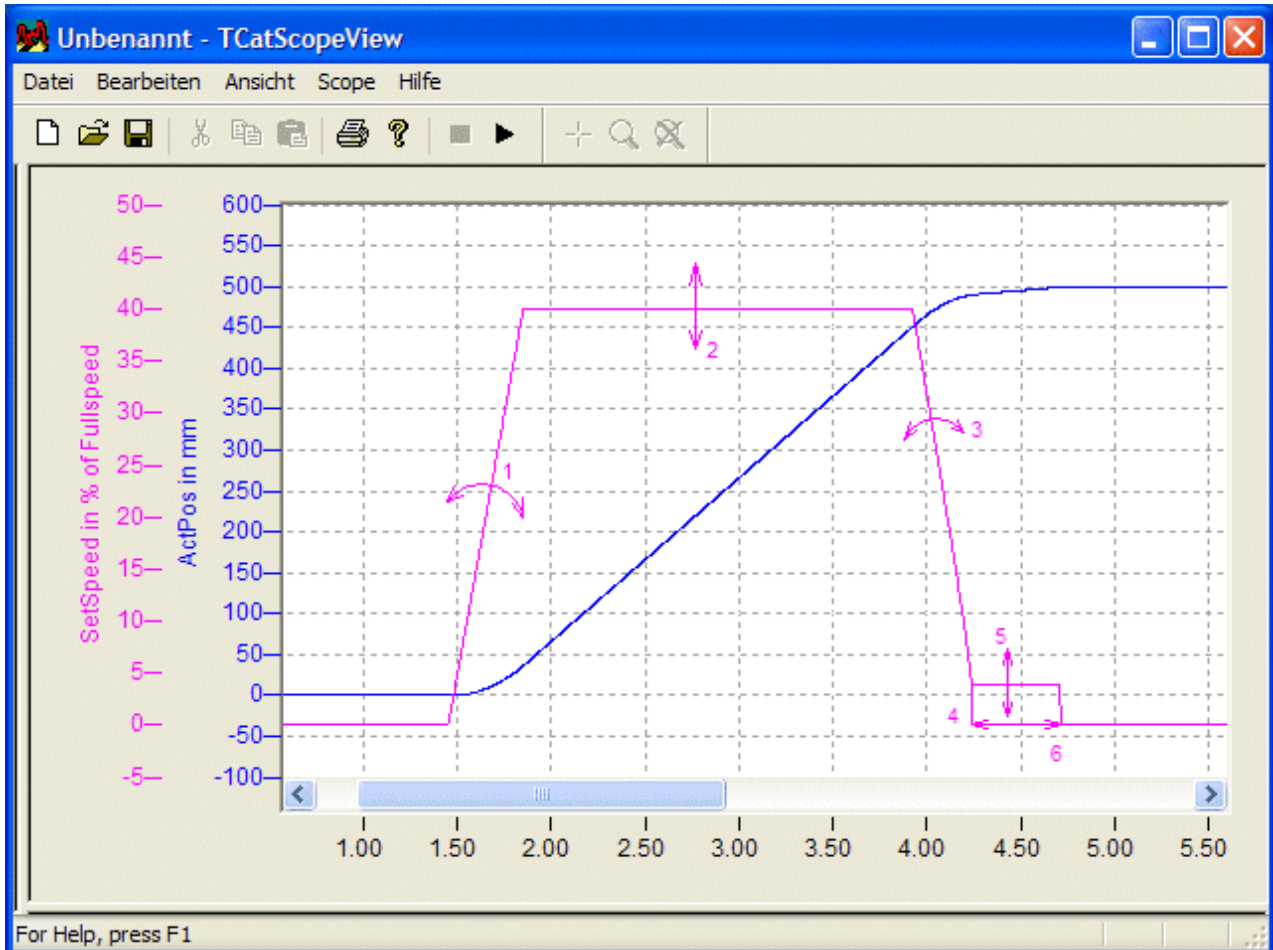
Derzeit stehen die folgenden Generatoren zur Verfügung:

nProfileType	Beschreibung
iTcMc_ProfileCtrlBased ▶ 158	<p>Standardprofil: Einstufige zeitgesteuerte Beschleunigung, weggesteuerte (Wurzelbildner) Bremsrampe, Zielanfahrt mit Schleichgeschwindigkeit, wählbares Verhalten im Ruhezustand.</p> <p>Das Starten einer fahrenden Achse (neues Ziel, neue Geschwindigkeit usw.) ist außer im Störzustand oder einem Zustand mit abhängiger Stellwertgenerierung immer möglich.</p> <p>Hinweis: Es kann zu einem Überfahren des neuen Ziels auch dann kommen, wenn sich die Achse zum Zeitpunkt des Startens noch vor der Zielposition befindet.</p> <p>Hinweis: Der Baustein kann so parametrierbar werden, dass er unter bestimmten durch seine Parameter definierten Bedingungen automatisch starten und in einen aktiven Fahrzustand übergehen wird.</p> <p>Hinweis: Dieser Generatortyp kann optional auch rein zeitgesteuert mit ständig geschlossenem Lageregler arbeiten.</p>
iTcMc_ProfileJerkBased	<p>Standardprofil: Ein- oder zweistufige zeitgesteuerte Beschleunigung durch optionale Ruckbegrenzung, weggesteuerte (Wurzelbildner) Bremsrampe, Zielanfahrt mit Ruckbegrenzung, wählbares Verhalten im Ruhezustand.</p> <p>Das Starten einer fahrenden Achse (neues Ziel, neue Geschwindigkeit usw.) ist außer im Störzustand oder einem Zustand mit abhängiger Stellwertgenerierung immer möglich.</p> <p>Hinweis: Es kann zu einem Überfahren des neuen Ziels auch dann kommen, wenn sich die Achse zum Zeitpunkt des Startens noch vor der Zielposition befindet.</p> <p>Hinweis: Der Baustein kann so parametrierbar werden, dass er unter bestimmten durch seine Parameter definierten Bedingungen automatisch starten und in einen aktiven Fahrzustand übergehen wird.</p> <p>Hinweis: Dieser Generatortyp kann optional auch rein zeitgesteuert mit ständig geschlossenem Lageregler arbeiten.</p> <p>Hinweis: Einige Funktionen werden von diesem Generatortyp nicht oder unvollständig unterstützt.</p>
iTcMc_ProfileTimePosCtrl	<p>Hinweis: Nur aus Kompatibilitätsgründen vorhanden, wird in Kürze nicht mehr unterstützt werden.</p> <p>Sonderprofil: Zweistufige (zunächst zeitgesteuert, dann weggesteuert mit Wurzelbildner) Beschleunigung, weggesteuerte (Wurzelbildner) Bremsrampe, Zielanfahrt mit Schleichgeschwindigkeit, wählbares Verhalten im Ruhezustand.</p> <p>Das Starten einer fahrenden Achse (neues Ziel, neue Geschwindigkeit usw.) ist nicht möglich.</p>
iTcMc_ProfileCosine	<p>Hinweis: Nur aus Kompatibilitätsgründen vorhanden, wird in Kürze nicht mehr unterstützt werden.</p> <p>Sonderprofil: Zweistufige (zunächst zeitgesteuert, dann weggesteuert mit Cosinusbildner) Beschleunigung, weggesteuerte (Cosinusbildner) Bremsrampe, Zielanfahrt mit Schleichgeschwindigkeit, wählbares Verhalten im Ruhezustand.</p> <p>Das Starten einer fahrenden Achse (neues Ziel, neue Geschwindigkeit usw.) ist nicht möglich.</p>
iTcMc_ProfileTimeRamp ▶ 160	<p>Sonderprofil: Einstufige zeitgesteuerte Beschleunigung, zeitgesteuerte Bremsrampe, Zielanfahrt mit Schleichgeschwindigkeit, eingeschränkt wählbares Verhalten im Ruhezustand. Der Generator arbeitet mit Positionsnocken anstelle eines Encoders.</p> <p>Das Starten einer fahrenden Achse (neues Ziel, neue Geschwindigkeit usw.) ist außer im Störzustand möglich.</p> <p>Hinweis: Dieser Generatortyp ist für Achsen vorgesehen, die an Stelle eines Encoders lediglich über digitale Nocken verfügen.</p>

Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollte zur Vereinfachung ein Baustein des Typs [MC_AxStandardBody_BkPlcMc](#) [► 170] verwendet werden.

iTcMc_ProfileCtrlBased

Es wird ein Profil mit einer zeitgesteuerten Beschleunigungsphase, einer weggesteuerten Bremsphase nach dem Prinzip des Wurzelbildners und eine Zielannäherung mit Schleichgeschwindigkeit erzeugt.



Die Pfeile am Stellwertprofil deuten die Gestaltungsmöglichkeiten durch Parameter des Bewegungsauftrags bzw. der Achse an. Zunächst wird mit einer zeitgesteuerten Rampenfunktionen "1" auf die geforderten Fahrgeschwindigkeit "2" beschleunigt. Dieser Stellwert wird aufrechterhalten, bis ein beim Start vorberechneter Wegpunkt erreicht wird. Ab hier wird mit einer weggesteuerten Rampe "3" so vom Fahrstellwert auf den Schleichstellwert "5" heruntergebremst, dass dieser Stellwert in einer festgelegten Entfernung "4" zum Ziel erreicht wird. Dieser Stellwert wird aufrechterhalten, bis die Entfernung sich dem Ziel bis auf eine Reststrecke "6" genähert hat. Jetzt wird in das Ruheverhalten umgeschaltet.

Aktive Parameter im Fahrprofil

Startrampe "1": Wirksam wird der kleinste von folgenden Werten: **fMaxAcc** und **fAcc** in [Axis.ST_TcHydAxParam](#) [► 88], **Acceleration** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute_BkPlcMc](#) [► 54]).

Fahrphase "2": Wirksam wird der kleinste von folgenden Werten: **fRefVelo** und **fMaxVelo** in [Axis.ST_TcHydAxParam](#) [► 88], **Velocity** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute_BkPlcMc](#) [► 54]).

Bremsrampe "3": Wirksam wird der kleinste von folgenden Werten: **fMaxDec** und **fDec** in [Axis.ST_TcHydAxParam](#) [► 88], **Deceleration** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute_BkPlcMc](#) [► 54]).

Schleichphase "4", "5": Wirksam werden die Werte **fCreepSpeed** und **fCreepDistance** in **Axis.ST_TcHydAxParam** [▶ 88].

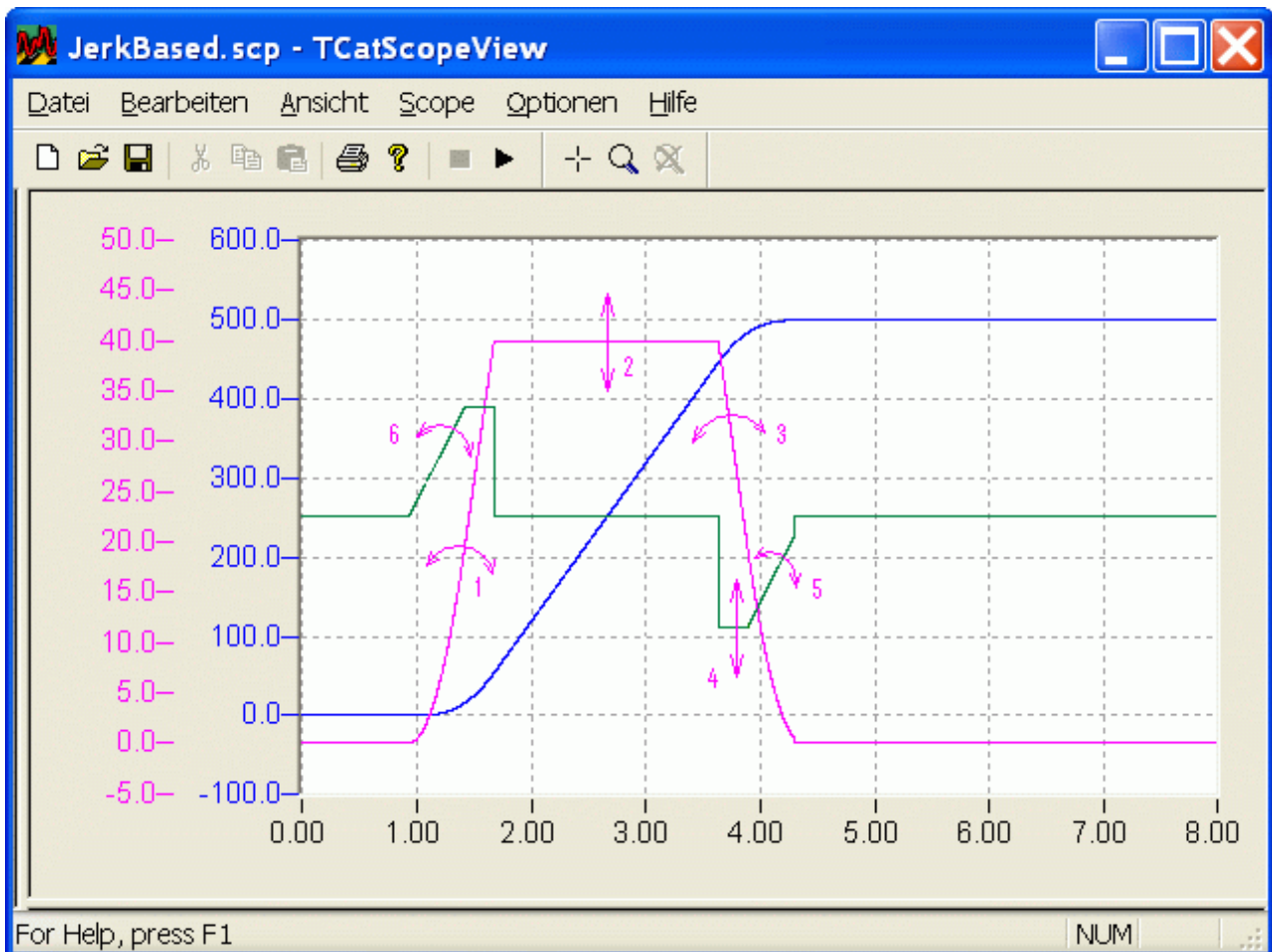
Übergang im Ziel "6": Wirksam wird **fBrakeDistance** und/oder **fBrakeDeadTime** in **Axis.ST_TcHydAxParam** [▶ 88].

Automatisches Starten der Achse

Wenn die Abweichung zwischen der Istposition und der aktuell geltenden Zielposition den Wert in **Axis.ST_TcHydAxParam** [▶ 88].**fReposDistance** überschreitet wird ein automatischer Start ausgelöst.

iTcMc_ProfileJerkBased

Es wird ein Profil mit einer zeitgesteuerten Beschleunigungsphase (mit optionaler Ruckbegrenzung), einer weggesteuerten Bremsrampe nach dem Prinzip des Wurzelbildners und einer Zielfahrt mit Ruckbegrenzung erzeugt.



Die Pfeile am Stellwertprofil deuten die Gestaltungsmöglichkeiten durch Parameter des Bewegungsauftrags bzw. der Achse an. Zunächst wird mit einer zeitgesteuerten Rampenfunktionen "1" auf die geforderten Fahrgeschwindigkeit "2" beschleunigt. Dabei kann die optionale Ruckbegrenzung "6" wirksam werden. Die Fahrgeschwindigkeit wird aufrechterhalten, bis ein beim Start vorberechneter Wegpunkt erreicht wird. Ab hier wird mit einer weggesteuerten Rampe "3" heruntergebremst bis die Entfernung sich dem Ziel bis auf die Reststrecke genähert hat. Dabei wird die Verzögerung "4" mit einem Begrenzten Ruck "5" zum Ziel hin abgebaut. Jetzt wird in das Ruheverhalten umgeschaltet.

Aktive Parameter im Fahrprofil

Startrampe "1": Wirksam wird der kleinste von folgenden Werten: **fMaxAcc** und **fAcc** in **Axis.ST_TcHydAxParam** [▶ 88], **Acceleration** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute BkPlcMc](#) [▶ 54]).

Fahrphase "2": Wirksam wird der kleinste von folgenden Werten: **fRefVelo** und **fMaxVelo** in **Axis.ST_TcHydAxParam** [▶ 88], **Velocity** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute BkPlcMc](#) [▶ 54]).

Bremsrampe "3", "4": Wirksam wird der kleinste von folgenden Werten: **fMaxDec** und **fDec** in **Axis.ST_TcHydAxParam** [▶ 88], **Deceleration** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute BkPlcMc](#) [▶ 54]).

Übergang im Ziel "5": Wirksam werden **fMaxJerk** in **Axis.ST_TcHydAxParam** [▶ 88] und **fJerk** des zum Achsstart verwendeten Bausteins (Beispiel: [MC_MoveAbsolute BkPlcMc](#) [▶ 54]) sowie **fBrakeDistance** und/oder **fBrakeDeadTime** in **Axis.ST_TcHydAxParam** [▶ 88].

iTcMc_ProfileTimePosCtrl

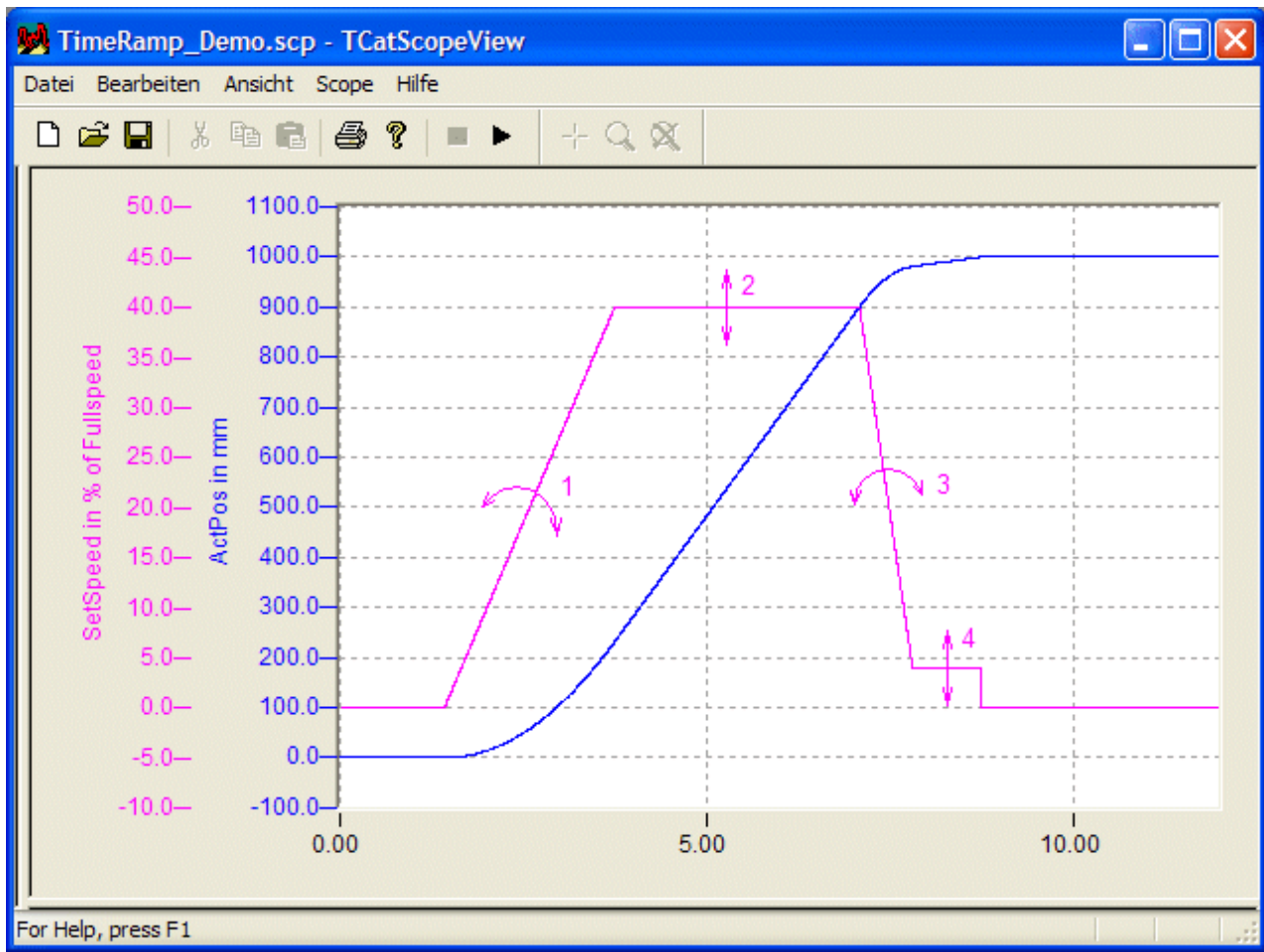
HINWEIS! Nur aus Kompatibilitätsgründen vorhanden, wird in Kürze nicht mehr unterstützt werden. Er sollte bei neuen Projekten nicht eingesetzt und bei der Überarbeitung von bestehenden Projekten möglichst ersetzt werden.

iTcMc_ProfileCosine

HINWEIS! Nur aus Kompatibilitätsgründen vorhanden, wird in Kürze nicht mehr unterstützt werden. Er sollte bei neuen Projekten nicht eingesetzt und bei der Überarbeitung von bestehenden Projekten möglichst ersetzt werden.

iTcMc_ProfileTimeRamp

Es wird ein Profil mit einer zeitgesteuerten Beschleunigungsphase, einer zeitgesteuerten Bremsphase und eine Zielannäherung mit Schleichgeschwindigkeit erzeugt.



Die Pfeile am Stellwertprofil deuten die Gestaltungsmöglichkeiten durch Parameter des Bewegungsauftrags bzw. der Achse an. Zunächst wird mit einer zeitgesteuerten Rampenfunktionen "1" auf die geforderten Fahrgeschwindigkeit "2" beschleunigt. Dieser Stellwert wird aufrechterhalten, bis der richtungsspezifische Zielfenster nocken erkannt wird. Ab hier wird mit einer zeitgesteuerten Rampe "3" vom Fahrstellwert auf den Schleichstellwert "5" heruntergebremst. Dieser Stellwert wird aufrechterhalten, bis der richtungsspezifische Zielnocken erkannt wird. Jetzt wird in das Ruherverhalten umgeschaltet.

Aktive Parameter im Fahrprofil

Startrampe "1": Wirksam wird **fStartRamp** in **Axis.ST_TcHydAxParam** [► 88].

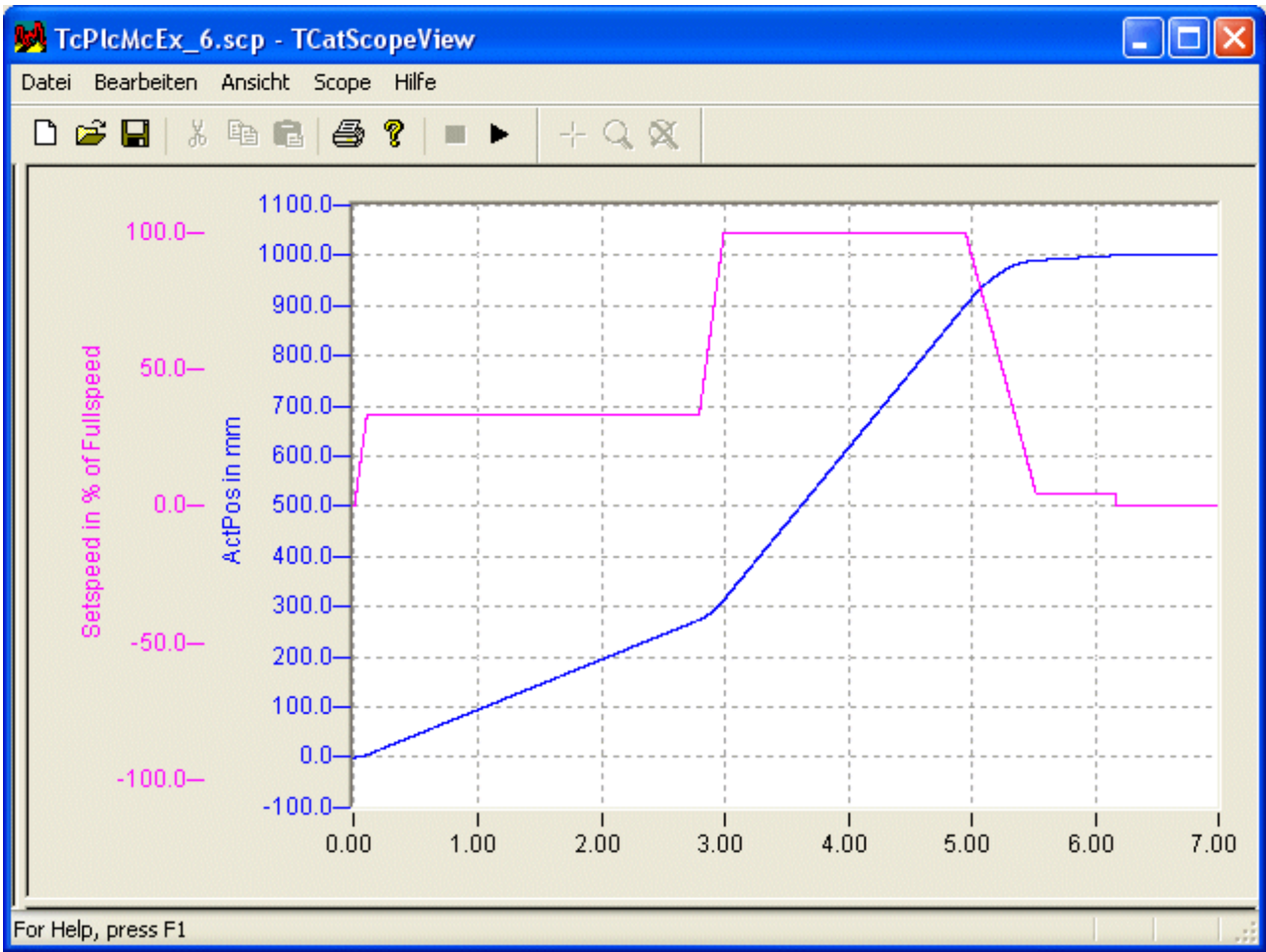
Fahrphase "2": Wirksam wird der kleinste von folgenden Werten: **fRefVelo** und **fMaxVelo** in **Axis.ST_TcHydAxParam** [► 88], **Velocity** des zum Achsstart verwendeten Bausteins (Beispiel: **MC_MoveAbsolute_BkPlcMc** [► 54]).

Bremsrampe "3": Wirksam wird **fStopRamp** in **Axis.ST_TcHydAxParam** [► 88].

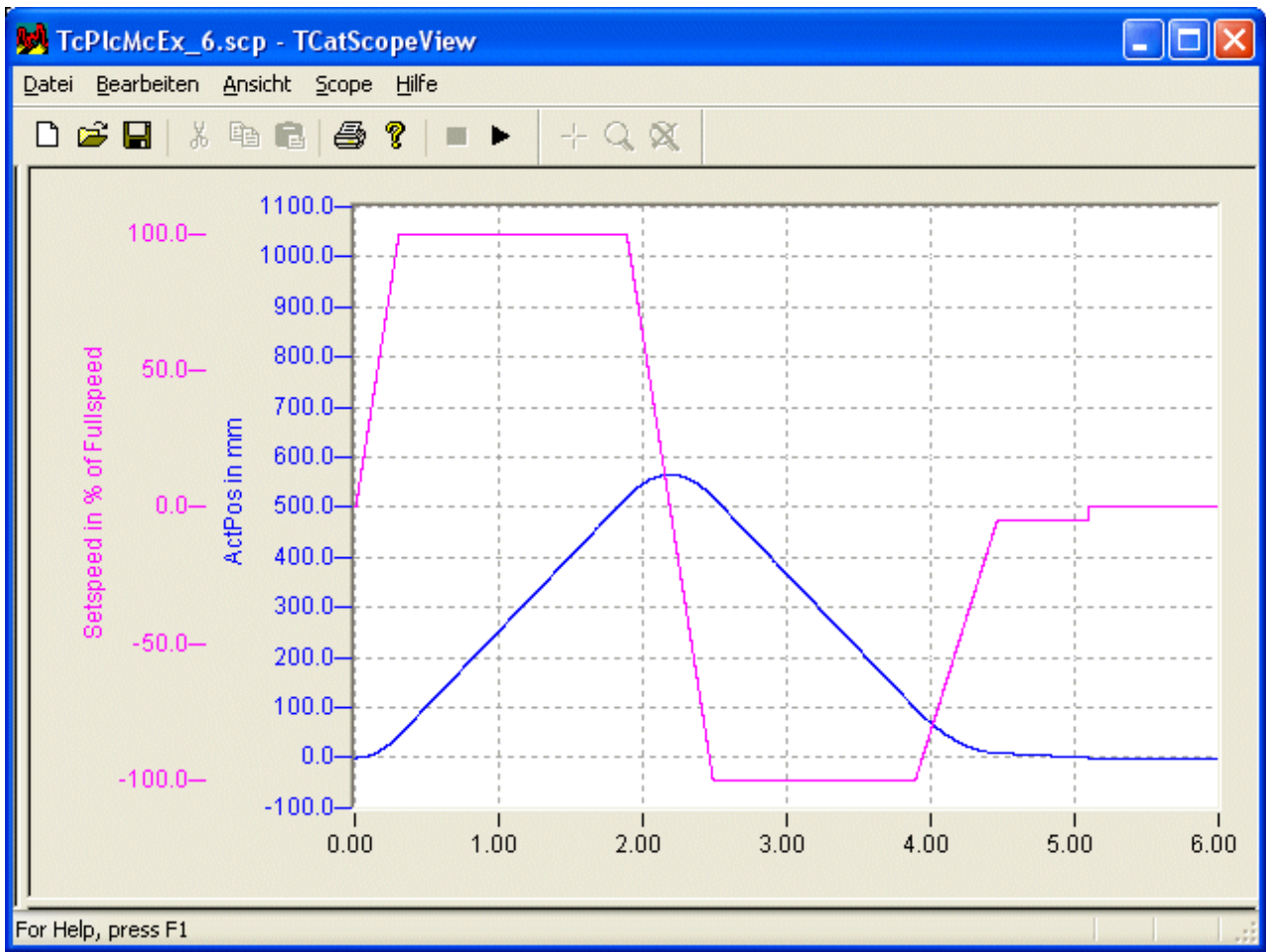
Schleichphase "4": Wirksam wird **fCreepSpeed** in **Axis.ST_TcHydAxParam** [► 88].

Verhalten des Bausteins beim Durchstarten während einer Bewegung

Wird während einer aktiven Bewegung ein weiterer Startbefehl gegeben ist sind zwei Fälle zu unterscheiden.



Dieses Profil entsteht beim Nachstarten in gleicher Richtung mit anderer (hier mit höherer) Geschwindigkeit.

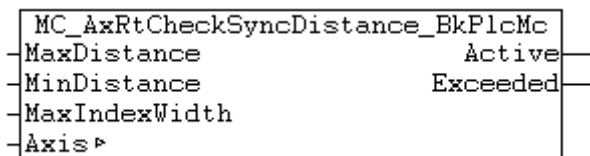


Dieses Profil entsteht beim Nachstarten in entgegengesetzter Richtung, hier mit gleicher Geschwindigkeit.

Dieser Profiltyp kann sinnvoll nur in Kombination mit dem Encodertyp [iTcMc EncoderDigCam \[132\]](#) verwendet werden. Siehe auch [Sonderfall digitale Positionsnocken \[263\]](#).

3.4.7 Runtime

3.4.7.1 MC_AxRtCheckSyncDistance_BkPlcMc (ab V3.0)



Der Funktionsbaustein überprüft, ob bei einer Referenzfahrt nach dem Verlassen des Nockens eine unzulässige Strecke zurückgelegt wird.

```

VAR_INPUT
  MaxDistance:    LREAL;
  MinDistance:    LREAL;
  MaxIndexWidth:  LREAL;
END_VAR
VAR_INOUT
  Axis:           Axis_Ref_BkPlcMc;
END_VAR
    
```

```

VAR_OUTPUT
  Active:          BOOL;
  Exceeded:       BOOL;
END_VAR

```

MaxDistance: Hier ist die maximal zulässige Strecke vorzugeben, die vom Referenznocken bis zum Erreichen des Nullimpulses zurückgelegt werden darf.

MinDistance: Hier ist die minimal zulässige Strecke vorzugeben, die vom Referenznocken bis zum Erreichen des Nullimpulses zurückgelegt werden muss.

MaxIndexWidth: Hier ist die minimal zulässige Strecke vorzugeben, die zum Verlassen des Referenznockens zurückgelegt werden muss. (ab V3.0.20)

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Exceeded: Hier wird signalisiert, dass die Achse nach dem Verlassen des Nockens einen Weg von mehr als **MaxDistance** zurückgelegt hat ohne dass der Nullimpuls des Gebers erkannt wurde.

Active: Hier wird signalisiert, dass die Achse den Nocken verlassen hat und den Nullimpuls des Gebers erwartet.

Verhalten des Bausteins

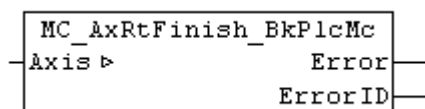
Der Baustein erkennt den Teil der Referenzfahrt, in dem die Achse den Nullimpuls des Gebers sucht und überwacht die dabei zurückgelegte Strecke. Dabei können zwei Probleme erkannt werden:

- Die Achse legt **MaxIndexWidth** zurück, ohne dass die fallenden Flanken des Referenznockens erkannt werden.
- Die Achse legt **MaxDistance** zurück, ohne dass ein Nullimpuls erkannt wird.
- Der Nullimpuls wird erkannt, bevor die Achse **MinDistance** zurückgelegt hat.

Erkannte Probleme werden mit **Exceeded** signalisiert. Soll dies zu einem Achsfehler führen muss von der Applikation ein solcher Zustandswechsel vorgegeben werden. Hierzu sollte ein [MC_AxRtGoErrorState_BkPlcMc \[► 167\]](#) Baustein und ein kodierter [Error Code \[► 224\]](#) verwendet werden.

ⓘ HINWEIS! Die Überwachung auf MinDistance und MaxDistance kann unterdrückt werden, indem der jeweilige Parameter auf 0.0 gestellt wird.

3.4.7.2 MC_AxRtFinish_BkPlcMc (ab V3.0)



Der Funktionsbaustein sorgt für die Anpassung des erzeugten Stellwerts an die Besonderheiten der Achse. Soll eine Kennlinienlinearisierung durchgeführt werden ist ein [MC_AxRtFinishLinear_BkPlcMc \[► 165\]](#) Baustein zu verwenden.

```

VAR_OUTPUT
  Error:          BOOL;
  ErrorID:       UDINT;
END_VAR
VAR_INOUT
  Axis:          Axis_Ref_BkPlcMc;
END_VAR

```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

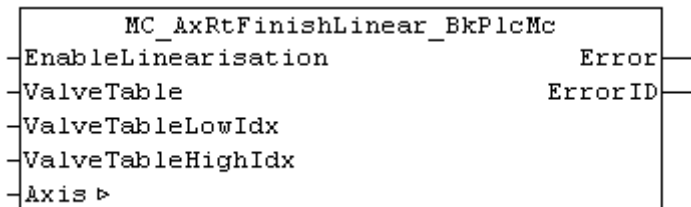
- Wenn einer der Pointer nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** oder **dwTcHydErrCdPtrMcPlc** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnte wird die Stellwertanpassung für die Achse entsprechend der Werte in **Axis.ST_TcHydAxParam** [► 88] durchgeführt.

- Der Vorschubstellwert und die Lageregelreaktion werden zum Ausgabestellwert zusammengefasst.
- Die Flächenkompensation wird berücksichtigt.
- Ein Kennlinienknick wird kompensiert.
- Die Überdeckungskompensation, der Klemmstellwert und die Offsetkompensation werden eingerechnet.

Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollte zur Vereinfachung ein Baustein des Typs **MC_AxStandardBody_BkPlcMc** [► 170] verwendet werden.

3.4.7.3 MC_AxRtFinishLinear_BkPlcMc (ab V3.0.16)



Der Funktionsbaustein sorgt für die Anpassung des erzeugten Stellwerts an die Besonderheiten der Achse unter Berücksichtigung einer Kennlinie.

```

VAR_INPUT
  EnableLinearisation:  BOOL;
  ValveTable:           POINTER TO LREAL:=0;
  ValveTableLowIdx:    INT:=0;
  ValveTableHighIdx:   INT:=0;
END_VAR
VAR_OUTPUT
  Error:                BOOL;
  ErrorID:              UDINT;
END_VAR
VAR_INOUT
  Axis:                 Axis_Ref_BkPlcMc;
END_VAR

```

EnableLinearisation: Ein TRUE an diesem Eingang aktiviert die Linearisierung.

ValveTable: Die Adresse der Linearisierungstabelle ist hier zu übergeben. Dabei sollte es sich möglichst um den ValveCharacteristicTable einer **ST_TcMcAutoIdent** [► 88] handeln, die mit der Achse verbunden ist.

ValveTableLowIdx: Der Index des ersten Punktes in der Linearisierungstabelle.

ValveTableHighIdx: Der Index des letzten Punktes in der Linearisierungstabelle. Dabei sollte es sich möglichst um den ValveCharacteristicTblCount einer **ST_TcMcAutoIdent** [► 88] handeln, die mit der Achse verbunden ist.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ **Axis_Ref_BkPlcMc** [► 62] zu übergeben.

Verhalten des Bausteins

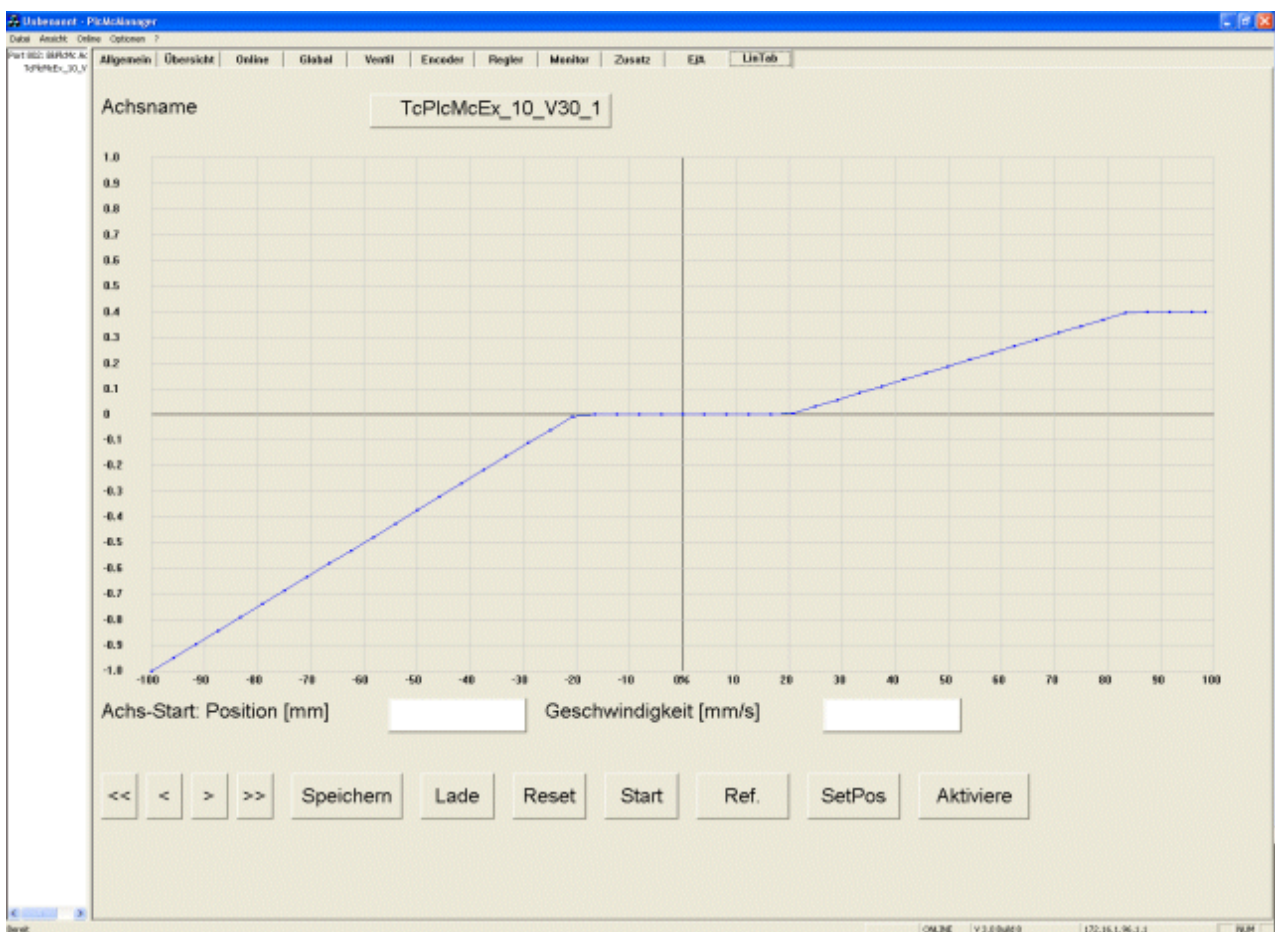
Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt werden:

- **EnableLinearisation** ist FALSE.
- Der Pointer **ValveTable** ist nicht initialisiert.
- **ValveTableLowIdx** ist kleiner als 0.
- **ValveTableHighIdx** ist kleiner oder gleich **ValveTableLowIdx** .

In diesen Fällen wird intern ein [MC_AxRtFinish_BkPlcMc](#) [▶ 164] Baustein aufgerufen und dessen Ausgänge durchgereicht. Andernfalls wird die Tabellenlinearisierung für die Achse durchgeführt. Dabei sind folgende Besonderheiten zu beachten:

- Der Parameter für die Kompensation der Richtungsabhängigkeit (Flächenverhältnis, Gravitation usw.) der Achsgeschwindigkeit ist unwirksam. Diese Kompensation ist in der Tabelle zu berücksichtigen.
- Die Parameter für die Kompensation eines Kennlinienknicks sind unwirksam. Diese Kompensation ist in der Tabelle zu berücksichtigen.
- Der Parameter für die Überdeckungskompensation ist unwirksam. Diese Kompensation ist in der Tabelle zu berücksichtigen.
- Eine Pressdruckausgabe oder eine Offsetkompensation können nicht durch eine Linearisierung realisiert werden. Die entsprechenden Parameter sind aktiv.

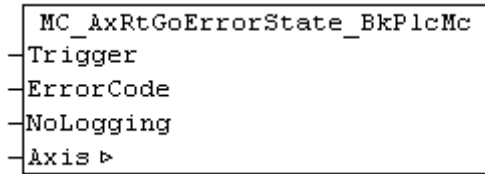
Beispiel: Anzeige einer Linearisierung im PlcMcManager:



Die Linearisierungstabellen können mit einem [MC_AxTableFromAsciiFile_BkPlcMc](#) [▶ 154] oder [MC_AxTableFromBinFile_BkPlcMc](#) [▶ 153] Baustein aus einer [Textdatei](#) [▶ 221] geladen werden.

Ein Programmbeispiel finden Sie in der [SampleList \[▶ 234\]](#) der [Knowledge Base \[▶ 207\]](#). Hier wird auch die automatische Ermittlung einer Kennlinie mit einem [MC_AxUtiAutoIdent_BkPlcMc \[▶ 181\]](#) Baustein demonstriert.

3.4.7.4 MC_AxRtGoErrorState_BkPlcMc (ab V3.0)



(not recommended) Der Funktionsbaustein versetzt die Achse in einen Störzustand.

```

VAR_INPUT
    Trigger:          BOOL;
    ErrorID:          UDINT;
    NoLogging:        BOOL;
END_VAR
VAR_INOUT
    Axis:             Axis_Ref_BkPlcMc;
END_VAR
    
```

Trigger: Eine steigende Flanke an diesem Eingang versetzt die Achse in einen Störzustand.

ErrorCode: Hier wird eine codierte Fehlerursache bereitgestellt.

NoLogging: Ein TRUE an diesem Eingang unterdrückt die Ausgabe einer Meldung.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[▶ 62\]](#) zu übergeben.

Verhalten des Bausteins

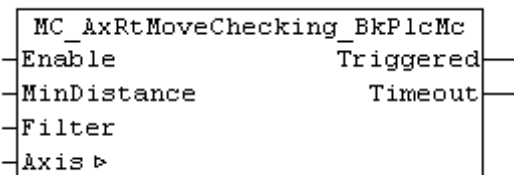
Mit einer steigenden Flanke am Eingang **Trigger** wird die Achse in einen Störzustand versetzt.

Voraussetzungen:

- Der Wert am Eingang **ErrorCode** ist ungleich 0.
- Die Achse befindet sich nicht bereits in einem Störzustand.

ⓘ HINWEIS! Wenn **NoLogging** auf **FALSE** steht (Default-Zustand) wird beim Übergang der Achse in den Störzustand eine Meldung erzeugt, die Angaben über die betroffene Achse und den **ErrorCode** enthält. Diese Default-Meldung sollte unbedingt durch eine aussagefähige Meldung der Applikation ersetzt werden. In diesem Fall ist die Default-Meldung mit **NoLogging** auf **TRUE** zu unterdrücken.

3.4.7.5 MC_AxRtMoveChecking_BkPlcMc (ab V3.0)



Der Funktionsbaustein überwacht die Reaktion einer Achse.

```

VAR_INPUT
  Enable:      BOOL;
  MinDistance: LREAL;
  Filter:      LREAL;
END_VAR
VAR_OUTPUT
  Triggered:   BOOL;
  Timeout:     BOOL;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Ein TRUE an diesem Eingang aktiviert die Überwachung.

MinDistance: Hier ist die geforderte Mindeststrecke zu übergeben.

Filter: Hier ist die Messzeit anzugeben.

Triggered: Dieser Ausgang signalisiert, dass die Achse in den Fehlerzustand versetzt wurde.

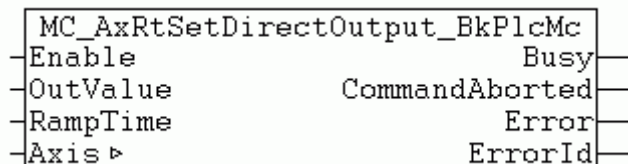
Timeout: Dieser Ausgang signalisiert, dass die Überwachung angesprochen hat.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Der Baustein überprüft kontinuierlich, ob die Achse innerhalb von **Filter** mindestens einen Weg von **MinDistance** in der zur geforderten Bewegung passenden Richtung zurückgelegt hat. Ist dies nicht der Fall wird **Timeout** signalisiert. Ist **Enable** auf TRUE wird in diesem Fall die Achse in den Stöorzustand `dwTcHydErrCdNotMoving = 16#435D = 17245` versetzt. Dies wird durch **Triggered** signalisiert.

3.4.7.6 MC_AxRtSetDirectOutput_BkPlcMc (ab V3.0)



Der Funktionsbaustein gibt unabhängig von einer Profildgenerierung einen Stellwert aus.

```

VAR_INPUT
  Enable:      BOOL;
  OutValue:    LREAL;
  RampTime:    LREAL;
END_VAR
VAR_OUTPUT
  Busy:        BOOL;
  CommandAborted: BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Enable: Ein TRUE an diesem Eingang aktiviert die Ausgabe.

OutValue: Hier ist der auszugebende Stellwert zu übergeben.

RampTime: Hier ist die Zeit anzugeben in der der Stellwert die Vollaussteuerung erreichen würde.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird ein Abbruch der Funktion signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

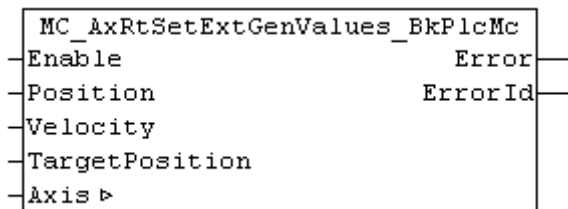
Verhalten des Bausteins

Eine steigende Flanke an **Enable** aktiviert die Funktion. Die Achse wird in die Zustände McState_Continousmotion [► 73] und iTcHydStateExtGenerated [► 65] versetzt und **Busy** wird **TRUE**. Der Stellwert der Achse wird mit OutValue aktualisiert. Dabei wird die Änderungsrate durch **RampTime** festgelegt.

Wird **Enable** auf **FALSE** gesetzt wird der Stellwert unter Verwendung von **RampTime** auf 0.0 gebracht und die Funktion beendet. Erst dann wird **BusyFALSE**.

Wenn die Kontrolle über die Achse durch einen anderen Baustein übernommen wird während der **MC_AxRtSetDirectOutput_BkPlcMc** Baustein aktiv ist stellt es seine Funktion ein und signalisiert **CommandAborted**.

3.4.7.7 MC_AxRtSetExtGenValues_BkPlcMc (ab V3.0)



Der Funktionsbaustein versorgt eine Achse mit Führungsgrößen, die nicht aus dem achseigenen Generator stammen.

```

VAR_INPUT
  Enable:          BOOL;
  Position:        LREAL:=0.0;
  Velocity:        LREAL:=0.0;
  TargetPosition: LREAL:=0.0;
END_VAR
VAR_OUTPUT
  Error:          BOOL;
  ErrorID:        UDINT;
END_VAR
VAR_INOUT
  Axis:           Axis_Ref_BkPlcMc;
END_VAR
  
```

Enable: Ein TRUE an diesem Eingang aktiviert die Übernahme der bereitgestellten Führungsgrößen.

Position: Zyklisch zu übergebender Wert für die Sollposition.

Velocity: Zyklisch zu übergebender Wert für die Sollgeschwindigkeit.

TargetPosition: Zyklisch zu übergebender Wert für die Zielposition der aktuellen Bewegung.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

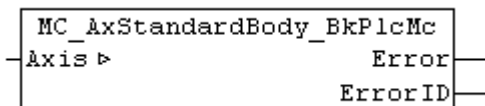
Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface. Erkennt er eine steigende Flanke an **Execute** versetzt er die Achse in den Zustand **McState_Synchronizedmotion** und **iTcHydStateExtGenerated**.

Wenn Execute TRUE ist werden die Werte von **Position**, **Velocity** und **TargetPosition** in die Laufzeitvariablen der Achse eingetragen. Dabei wird soweit möglich das Verhalten des Generatorbaustein bei einer vergleichbaren Bewegung nachgebildet.

Wird eine fallende Flanke an Execute erkannt versetzt der Baustein die Achse in den Zustand **McState_Standstill**. Ist die Achse zu diesem Zeitpunkt nicht im Stillstand wird sie über die fStopRamp eingestellte zeitgesteuerte Rampe angehalten.

HINWEIS! Der Generatorbaustein der Achse ist nach wie vor zyklisch aufzurufen. Er nimmt die Lageregelung vor und aktualisiert weitere interne Variablen.

3.4.7.8 MC_AxStandardBody_BkPlcMc (V3.0)



Der Funktionsbaustein ruft je einen Baustein der Typen [MC_AxRtEncoder_BkPlcMc \[► 123\]](#), [MC_AxRuntime_BkPlcMc \[► 155\]](#), [MC_AxRtFinish_BkPlcMc \[► 164\]](#) und [MC_AxRtDrive_BkPlcMc \[► 114\]](#) auf.

```

VAR_OUTPUT
  Error:      BOOL;
  ErrorID:    UDINT;
END_VAR
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
  
```

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

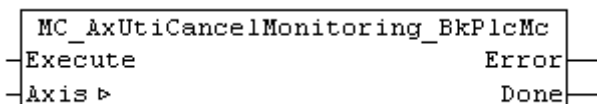
Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Die üblichen Bestandteile der Achse werden in Abhängigkeit der Werte in [ST_TcHydAxParam \[► 88\]](#) aufgerufen. Sollte einer der aufgerufenen Bausteine einen **Error** zurückmelden wird dieser mit seinem **ErrorID** an den Ausgängen dieses Bausteins wiedergegeben.

HINWEIS! Bei mehreren Problemen gilt eine Priorisierung entsprechend der Aufrufreihenfolge (Encoder, Generator, Finish, Drive).

3.4.7.9 MC_AxUtiCancelMonitoring_BkPlcMc (ab V3.0)



Die Überwachung einer Funktion wird abgebrochen.

HINWEIS! Viele Bausteine sind entsprechend den PLCopen Definitionen auch dann aktivierbar wenn für die Achse durch denselben oder einen anderen Baustein bereits eine Aktivität gestartet wurde. In den meisten Fällen kann dieses Verhalten ausgenutzt und der Abbruch der Überwachung ist nicht nötig.



Hinweis

Eine Reihe von Bausteinen benötigt die Möglichkeit zur Überwachung für die korrekte und vollständige Abarbeitung ihrer Funktion. Die Überwachung darf nur bei Bausteinen abgebrochen werden, bei denen diese Möglichkeit ausdrücklich genannt wird.

```
VAR_INOUT
  Axis:      Axis_Ref_BkPlcMc;
END_VAR
VAR_INPUT
  Execute:   BOOL;
END_VAR
VAR_OUTPUT
  Error:     BOOL;
  Done:      BOOL;
END_VAR
```

Execute: Eine steigende Flanke an diesem Eingang beendet die Überwachung.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

Done: Hier wird die erfolgreiche Durchführung des Abbruchs signalisiert.

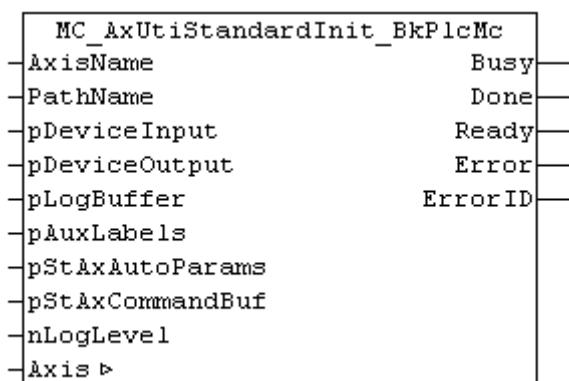
Verhalten des Bausteins

Bei einer steigenden Flanke an **Execute** manipuliert der Baustein das übergebene Interface **Axis** in einer Weise, dass andere Bausteine dies als den Start einer Funktion interpretieren. Sollte ein anderer Baustein die Abarbeitung einer von ihm gestarteten Funktion überwachen wird er dies einstellen und an seinem Ausgang das Signal **CommandAborted** setzen.

- ❏ **HINWEIS!** Derzeit sind keine Fehlerursachen bekannt. Der Ausgang **Error** ist immer **FALSE**.
- ❏ **HINWEIS!** Die Funktion des Bausteins benötigt zur Abarbeitung keine Zeit. Der Ausgang **Done** wird beim Aufruf des Bausteins unmittelbar dem Eingang **Execute** entsprechen.

Eine Liste der Bausteine, für die ein solcher Abbruch der Überwachung zulässig ist finden Sie in der Knowledge Base [► 207].

3.4.7.10 MC_AxUtiStandardInit_BkPlcMc (ab V3.0)



Der Funktionsbaustein übernimmt die Initialisierung und Überwachung der Bestandteile einer Achse.

```
VAR_INPUT
  AxisName:      STRING(255);
  PathName:      STRING(255);
  pDeviceInput:  POINTER TO ST_TcPlcDeviceInput:=0;
  pDeviceOutput: POINTER TO ST_TcPlcDeviceOutput:=0;
  pLogBuffer:    POINTER TO ST_TcPlcMcLogBuffer:=0;
  pStAxAuxLabels: POINTER TO ST_TcMcAuxDataLabels:=0;
  pStAxAutoParams: POINTER TO ST_TcMcAutoIdent;
```

```

pStAxCommandBuf:    POINTER TO ST_TcPlcCmdBuffer_BkPlcMc:=0;    (* ab/from V3.0.8 *)
nLogLevel:          DINT:=0;
END_VAR
VAR_OUTPUT
  Busy:             BOOL;
  Done:             BOOL;
  Ready:           BOOL;
  Error:           BOOL;
  ErrorID:         UDINT;
END_VAR
VAR_INOUT
  Axis:            Axis_Ref_BkPlcMc;
END_VAR

```

AxisName: Hier ist der textuelle Name der Achse (Beispiel: 'Achse_1') zu übergeben.

PathName: Hier ist der textuelle Pfadname (Beispiel: 'C:\TwinCAT\Projekt\') zu übergeben, unter dem die Parameter der Achse zu speichern sind.

pDeviceInput: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcDeviceInput \[► 98\]](#) zu übergeben.

pDeviceOutput: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcDeviceOutput \[► 100\]](#) zu übergeben.

pLogBuffer: Hier kann die Adresse einer Variablen vom Typ [ST_TcPlcMcLogBuffer \[► 102\]](#) übergeben werden.

pAuxLabels: Hier kann die Adresse einer [ST_TcMcAuxDataLabels \[► 98\]](#) Struktur mit Beschriftungstexten der kundenspezifischen Achsparameter übergeben werden.

pStAxAutoParams: Hier kann die Adresse einer Variablen vom Typ [ST_TcMcAutoIdent \[► 88\]](#) übergeben werden.

pStAxCommandBuf: Ab V3.0.8 ist bei diversen Bausteinen der durch die PLCopen definierte Eingang **BufferMode** vorhanden. Die damit steuerbare Funktionalität wird derzeit vorbereitet. In diesem Zusammenhang ist dieser Befehls-Puffer ergänzt worden.



Hinweis

Der Eingang **pStAxCommandBuf** darf derzeit nicht oder nur mit dem Wert 0 versorgt werden.

nLogLevel: Hier ist ein [codierter Wert \[► 232\]](#) zu übergeben, der den Schwellwert für die Aufzeichnung von Meldungen festlegt.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.


Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf untersucht der Baustein das übergebene Achsinterface und die übergebenen Pointer. Wird eine Veränderung erkannt signalisiert der Baustein in der übergebenen [Axis_Ref_BkPlcMc \[► 62\]](#) Struktur, dass die Achse reinitialisiert werden muss. Der von diesem Baustein verwendete [MC_AxParamLoad_BkPlcMc \[► 192\]](#) Baustein wird nun automatisch die Parameter der Achse aus der Datei laden. Ist **pAuxLabels** versorgt werden anschließend mit einem [MC_AxParamAuxLabelsLoad_BkPlcMc \[► 191\]](#) Baustein die Beschriftungstexte der kundenspezifischen Achsparameter geladen.

ⓘ HINWEIS! Die als **AxisName** und **PathName** übergebenen Strings dürfen keine Leer- oder Sonderzeichen enthalten, die sie für die Erzeugung eines Dateinamens ungeeignet machen. Der Dateiname wird durch aneinanderhängen der übergebenen Strings und Anfügen der Extension **'.dat'** erzeugt. Der Dateiname für die Beschriftungstexte der kundenspezifischen Achsparameter wird auf die gleiche Weise, jedoch mit der Extension **'.txt'** erzeugt.

HINWEIS! Die Parameter `pDeviceInput` und `pDeviceOutput` sind bei allen Achsen zu versorgen, die für die Positionserfassung eine E/A-Hardware verwenden. Bei Verwendung von virtuellen Achsen sind diese Parameter nicht oder mit 0 zu belegen.

 Hinweis	Der Eingang <code>pStAxCommandBuf</code> darf derzeit nicht oder nur mit dem Wert 0 versorgt werden.
-----------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------

3.4.8 Message Logging

3.4.8.1 MC_AxRtLogAxisEntry_BkPlcMc (ab V3.0)

```

MC_AxRtLogAxisEntry_BkPlcMc
- pBuffer
- LogLevel
- Source
- Message
- ArgType
- diArg
- lrArg
- sArg
- Axis
    
```

Der Funktionsbaustein trägt eine achsbezogene Meldung in den LogBuffer der Library ein. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base](#) [▶ 207].

HINWEIS! Alle achsbezogenen Bausteine der Bibliothek verwenden diesen Baustein zur Meldungsausgabe.

```

VAR_INOUT
  Axis:          POINTER TO Axis_Ref_BkPlcMc;
END_VAR
VAR_INPUT
  Execute:       BOOL:=FALSE;
  pBuffer:       POINTER TO ST_TcPlcMcLogBuffer;
  LogLevel:      DWORD:=0;
  Source:        DWORD:=0;
  Message:       STRING(255);
  ArgType:       INT:=0;
  diArg:         DINT:=0;
  lrArg:         LREAL:=0;
  sArg:          STRING(255);
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang aktiviert den Baustein.

pBuffer: Hier ist die Adresse einer Variablen vom Typ `ST_TcPlcMcLogBuffer` [▶ 102] zu übergeben.

LogLevel: Eine kodierte Angabe des Meldungstyps. Es ist ein `Logger Levels` [▶ 232] Wert aus den `Global Constants` [▶ 222] zu verwenden.

Source: Eine kodierte Angabe der Meldungsquelle. Es ist ein `Logger Sources` [▶ 232] Wert aus den `Global Constants` [▶ 222] zu verwenden.

Message: Der Text der Meldung.

ArgType: Der Typ des optionalen Argumentes.

diArg: Der Wert des optionalen Arguments, wenn es vom Typ DINT ist.

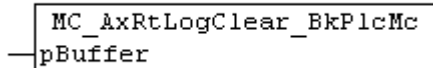
lrArg: Der Wert des optionalen Arguments, wenn es vom Typ LREAL ist.

sArg: Der Wert des optionalen Arguments, wenn es vom Typ STRING ist.

Verhalten des Bausteins

Der Baustein unterscheidet sich von [MC_AxRtLogEntry_BkPlcMc](#) [► 174] nur dadurch, dass er der Meldung den Achsnamen voranstellt.

3.4.8.2 MC_AxRtLogClear_BkPlcMc (ab V3.0)



Der Funktionsbaustein löscht einen LogBuffer der Library. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base](#) [► 207].

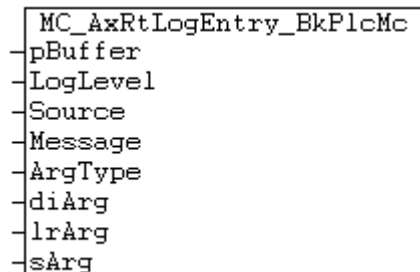
```
VAR_INOUT
  pBuffer:          POINTER TO ST_TcPlcMcLogBuffer;
END_VAR
```

pBuffer: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcMcLogBuffer](#) [► 102] zu übergeben.

Verhalten des Bausteins

Alle Einträge im LogBuffer werden gelöscht und initialisiert.

3.4.8.3 MC_AxRtLogEntry_BkPlcMc (ab V3.0)



Der Funktionsbaustein trägt eine Meldung in den LogBuffer der Library ein. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base](#) [► 207].

```
VAR_INPUT
  Execute:          BOOL:=FALSE;
  pBuffer:          POINTER TO ST_TcPlcMcLogBuffer;
  LogLevel:         DWORD:=0;
  Source:           DWORD:=0;
  Message:          STRING(255);
  ArgType:          INT:=0;
  diArg:            DINT:=0;
  lrArg:            LREAL:=0;
  sArg:             STRING(255);
END_VAR
```

Execute: Eine steigende Flanke an diesem Eingang aktiviert den Baustein.

pBuffer: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcMcLogBuffer](#) [► 102] zu übergeben.

LogLevel: Eine kodierte Angabe des Meldungstyps. Es ist ein [Logger Levels](#) [► 232] Wert aus den [Global Constants](#) [► 222] zu verwenden.

Source: Eine kodierte Angabe der Meldungsquelle. Es ist ein [Logger Sources \[► 232\]](#) Wert aus den [Global Constants \[► 222\]](#) zu verwenden.

Message: Der Text der Meldung.

ArgType: Der Typ des optionalen Argumentes.

diArg: Der Wert des optionalen Arguments, wenn es vom Typ DINT ist.

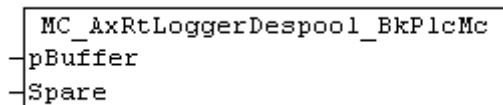
lrArg: Der Wert des optionalen Arguments, wenn es vom Typ LREAL ist.

sArg: Der Wert des optionalen Arguments, wenn es vom Typ STRING ist.

Verhalten des Bausteins

Wenn **pBuffer** korrekt versorgt wurde und auf einen [ST_TcPlcMcLogBuffer \[► 102\]](#) zeigt, der noch mindestens eine Meldung aufnehmen kann werden die übergebenen Meldungsdaten mit Informationen zur lokalen Zeit komplettiert und in den Meldungspuffer eingetragen.

3.4.8.4 MC_AxRtLoggerDespool_BkPlcMc (ab V3.0)



Der Funktionsbaustein stellt sicher, dass der LogBuffer der Library nicht überläuft. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base \[► 207\]](#).

```
VAR_INPUT
    Spare:      INT;
END_VAR
VAR_INOUT
    pBuffer:    POINTER TO ST_TcPlcMcLogBuffer;
END_VAR
```

Spare: Die sicher zu stellende Anzahl von freien Meldungen im LogBuffer.

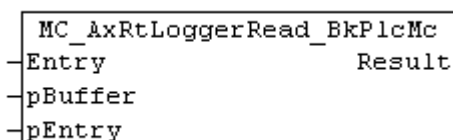
pBuffer: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcMcLogBuffer \[► 102\]](#) zu übergeben.

Verhalten des Bausteins

Bei jedem Aufruf entfernt der Baustein eine Meldung aus dem LogBuffer wenn die Anzahl der freien Meldungen kleiner ist als die in **Spare** übergebene Minimalanzahl. Soll die gesamte Historie in die Ereignisanzeige von Windows übernommen werden ist ein [MC_AxRtLoggerSpool_BkPlcMc \[► 176\]](#) Baustein zu verwenden.

ⓘ HINWEIS! Der Einsatz dieses Bausteins ist immer dann sinnvoll, wenn ein Massenspeicher mit eingeschränkter Beschreibbarkeit (typischerweise FLASH DISK) verwendet wird. Es wird zumindest eine eingeschränkte Historie von 10 bis 15 Meldungen ermöglicht.

3.4.8.5 MC_AxRtLoggerRead_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest eine Meldung aus dem LogBuffer der Library. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base \[► 207\]](#).

ⓘ HINWEIS! Dieser Baustein wird via ADS von Diagnose-Tools genutzt. Ein direkter Aufruf durch die SPS-Applikation macht in der Regel keinen Sinn.

```
VAR_INOUT
  Entry:      INT:=0;
  pBuffer:    POINTER TO ST_TcPlcMcLogBuffer;
  pEntry:     POINTER TO ST_TcPlcMcLogEntry;
END_VAR
VAR_OUTPUT
  Result:     DWORD:=0;
END_VAR
```

Entry: Die Nummer der zu lesenden Meldung,

pBuffer: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcMcLogBuffer \[► 102\]](#) zu übergeben.

pEntry: Hier ist als Ziel die Adresse einer Variablen vom Typ [ST_TcPlcMcLogEntry \[► 102\]](#) zu übergeben.

Result: Hier wird ein eine codierte Fehlerursache bereitgestellt.

Verhalten des Bausteins

Der Baustein überprüft bei jedem Aufruf die übergebenen Eingangswerte. Dabei können zwei Probleme erkannt werden:

- Ist **Entry** nicht im zulässigen Wertebereich 1..20 gibt der Baustein `dwTcHydAdsErrInvalidIdxOffset` als **Result** zurück.
- Sind **pBuffer** oder **pEntry** nicht definiert gibt der Baustein `dwTcHydAdsErrNotReady` als **Result** zurück.

Ist bei der Überprüfung kein Problem festgestellt worden kopiert der Baustein die von **Entry** selektierte Meldung aus dem LogBuffer **pBuffer** in die mit **pEntry** adressierte Meldungsstruktur. Dabei wird Entry als relative Altersangabe verstanden: Mit Entry:=1 wird die zuletzt eingetragene Meldung ausgewählt, mit Entry:=2 die nächst ältere usw. Steht die geforderte Meldung nicht zur Verfügung wird eine leere Meldung bereitgestellt.

3.4.8.6 MC_AxRtLoggerSpool_BkPlcMc (ab V3.0)

```
MC_AxRtLoggerSpool_BkPlcMc
-pBuffer
```

Der Funktionsbaustein übernimmt die Übertragung von Meldungen aus dem LogBuffer der Library in die Ereignisanzeige von Windows. Näheres über das Anlegen eines LogBuffers finden Sie unter FAQ #10 in der [Knowledge Base \[► 207\]](#).

```
VAR_INOUT
  pBuffer:    POINTER TO ST_TcPlcMcLogBuffer;
END_VAR
```

pBuffer: Hier ist die Adresse einer Variablen vom Typ [ST_TcPlcMcLogBuffer \[► 102\]](#) zu übergeben.

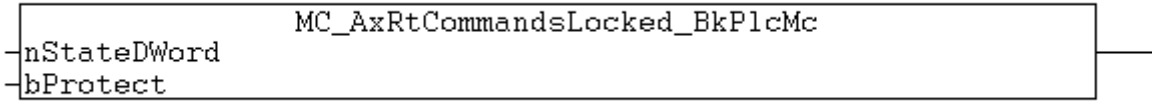
Verhalten des Bausteins

Bei jedem Aufruf entfernt der Baustein eine Meldung aus dem LogBuffer und überträgt sie in die Ereignisanzeige von Windows.

Wenn der eingesetzte Rechner einen Massenspeicher mit eingeschränkter Schreibbarkeit verwendet (typischerweise FLASH DISK) ist es nicht sinnvoll, die Ereignisanzeige von Windows zu nutzen. Um trotzdem eine Historie zu erzeugen kann ein [MC_AxRtLoggerDespool_BkPlcMc \[► 175\]](#) Baustein verwendet werden.

3.4.9 Utilitys

3.4.9.1 MC_AxRtCommandsLocked_BkPlcMc : DWORD



Die Funktion erleichtert das Setzen und Löschen einer Schutzfunktion im Status-Doppelwort einer Achse.

```
VAR_INPUT
    nStateDWord:  DWORD:=0;
    bProtect:     BOOL:=FALSE;
END_VAR
```

nStateDWord: Der aktuelle Zustand des Status-Doppelworts.

bProtect: Der gewünschte Zustand der Schutzfunktion.

Verhalten der Funktion

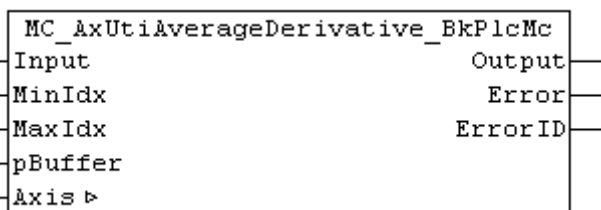
Die Funktion blendet abhängig von **bProtect** das Status-Bit der Schutzfunktion in das übergebene Status-Doppelwort ein.

HINWEIS! Das Ergebnis der Funktion muss von der Applikation auf das Status-Doppelwort der Achse zugewiesen werden.

Es steht ein <http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/pro/1599851275.pro> zur Verfügung.

3.4.9.2 Filters

3.4.9.2.1 MC_AxUtiAverageDerivative_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt die Ableitung eines Wertes durch numerische Differentiation über mehr als einen Zyklus.

```
VAR_INPUT
    Input:      LREAL:=0.0;
    MinIdx:    DINT:=0;
    MaxIdx:    DINT:=0;
    pBuffer:   POINTER TO LREAL:=0;
END_VAR
VAR_OUTPUT
    Output:    LREAL:=0.0;
    Error:     BOOL:=FALSE;
    ErrorID:   UDINT:=0;
END_VAR
VAR_INOUT
    Axis:      Axis_Ref_BkPlcMc;
END_VAR
```

Input: Der Roh-Wert der zu filternden Größe.

MinIdx: Der Index des ersten zu verwendenden Elements des Filterpuffers.

MaxIdx: Der Index des letzten zu verwendenden Elements des Filterpuffers.

pBuffer: Die Adresse des ersten Elements des Filterpuffers.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Output: Der gefilterte Wert.

Error: Dieser Ausgang signalisiert Probleme mit den übergebenen Parametern.

ErrorID: Im Fehlerfall wird hier eine kodierte Information über die Art des Problems gemeldet.

Verhalten des Bausteins

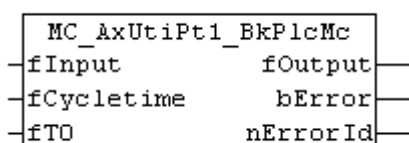
Bei jedem Aufruf überprüft der Baustein die Adresse des Filterpuffers **pBuffer** und die Indices der zu verwendenden Elemente **MinIdx** und **MaxIdx**. Sind die übergebenen Werte erkennbar unsinnig wird mit **Error** und einer kodierten Information in **ErrorID** reagiert. Andernfalls wird bei jedem Aufruf **Input** in den Filterpuffer eingetragen und der Mittelwert der Änderung über den im Puffer verfügbaren Wertevorrat gebildet und als **Output** zurückgegeben.

ⓘ HINWEIS! Der Wertevorrat für die Mittelwertbildung umfasst $(\text{MaxIdx} - \text{MinIdx} + 1)$ Werte. Die Messzeit ergibt sich durch Multiplikation dieser Anzahl mit der Zykluszeit.

ⓘ HINWEIS! Das Prinzip der gleitenden Mittelwertbildung führt zu einer Verzögerung in Höhe der halben Messzeit. Wird die gefilterte Größe in einem Regelkreis verwendet kann die dadurch verursachte frequenzabhängige Phasenverschiebung Einschränkungen bei der Parameterwahl verursachen.

Der Baustein hat keine Möglichkeit, die übergebenen Werte von **pBuffer**, **MinIdx** und **MaxIdx** vollständig zu überprüfen. Es ist besonders darauf zu achten, dass diese Werte gefahrlos verwendet werden können. Andernfalls kann es zu nicht vorhersagbarem Verhalten (Überschreiben von Speicher) oder zum Abbrechen des PLC-Betriebs kommen.

3.4.9.2.2 MC_AxUtiPT1_BkPlcMc (ab V3.0)



Der Funktionsbaustein berechnet einen Tiefpass 1. Ordnung.

```

VAR_INPUT
    fInput:          LREAL:=0.0;
    fCycletime:     LREAL:=0.001;
    fT0:            LREAL:=1.0;
END_VAR
VAR_OUTPUT
    fOutput:        LREAL;
    bError:         BOOL;
    nErrorId:       UDINT;
END_VAR

```

fInput: Der Roh-Wert der zu filternden Größe.

fCycletime: Die Zykluszeit der aufrufenden PLC-Task.

fT0: Die Filterzeitkonstante.

fOutput: Der gefilterte Wert.

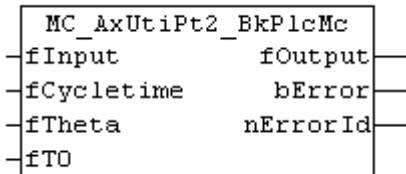
bError: Dieser Ausgang signalisiert Probleme mit den übergebenen Parametern.

nErrorId: Im Fehlerfall wird hier eine kodierte Information über die Art des Problems gemeldet.

Verhalten des Bausteins

Bei jedem Aufruf überprüft der Baustein die übergebenen Parameter. Sollte ein unzulässiger Wert erkannt werden reagiert der Baustein mit **bError** und einem entsprechenden Wert in **nErrorId**. Andernfalls werden die internen Variablen mit **fInput** aktualisiert und der gefilterte Wert als **fOutput** zurückgegeben.

3.4.9.2.3 MC_AxUtiPT2_BkPlcMc (ab V3.0)



Der Funktionsbaustein berechnet einen Tiefpass 2. Ordnung.

```

VAR_INPUT
    fInput:          LREAL:=0.0;
    fCycletime:      LREAL:=0.001;
    fTheta:          LREAL:=1.0;
    fT0:             LREAL:=1.0;
END_VAR
VAR_OUTPUT
    fOutput:         LREAL;
    bError:          BOOL;
    nErrorId:        UDINT;
END_VAR
  
```

fInput: Der Roh-Wert der zu filternden Größe.

fCycletime: Die Zykluszeit der aufrufenden PLC-Task.

fTheta: Die Dämpfung.

fT0: Die Filterzeitkonstante.

fOutput: Der gefilterte Wert.

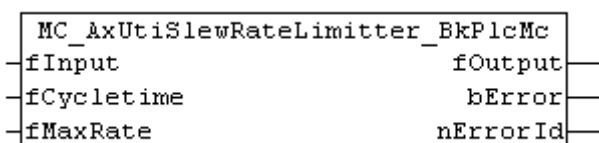
bError: Dieser Ausgang signalisiert Probleme mit den übergebenen Parametern.

nErrorId: Im Fehlerfall wird hier eine kodierte Information über die Art des Problems gemeldet.

Verhalten des Bausteins

Bei jedem Aufruf überprüft der Baustein die übergebenen Parameter. Sollte ein unzulässiger Wert erkannt werden reagiert der Baustein mit **bError** und einem entsprechenden Wert in **nErrorId**. Andernfalls werden die internen Variablen mit **fInput** aktualisiert und der gefilterte Wert als **fOutput** zurückgegeben.

3.4.9.2.4 MC_AxUtiSlewRateLimiter_BkPlcMc (ab V3.0)



Der Funktionsbaustein erzeugt eine anstiegsbegrenzte Rampe.

```

VAR_INPUT
  fInput:      LREAL:=0.0;
  fCycletime:  DINT:=0;
  fMaxRate:    DINT:=0;
END_VAR
VAR_OUTPUT
  fOutput:     LREAL:=0.0;
  bError:      BOOL:=FALSE;
  nErrorId:    UDINT:=0;
END_VAR

```

fInput: Der Roh-Wert der zu filternden Größe.

fCycletime: Die Zykluszeit der aufrufenden PLC Task in Sekunden.

fMaxRate: Der Betrag der am Ausgang maximal zulässigen Änderungsrate als Änderung pro Sekunde.

fOutput: Der gefilterte Wert.

bError: Dieser Ausgang signalisiert Probleme mit den übergebenen Parametern.

nErrorId: Im Fehlerfall wird hier eine kodierte Fehlerinformation ausgegeben.

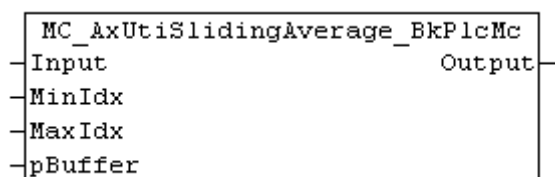
Verhalten des Bausteins

Bei jedem Aufruf überprüft der Baustein die übergebenen Werte für **fCycletime** und **fMaxRate**. Sind die Werte nicht korrekt wird mit **bError** und einer kodierten Information in **nErrorId** reagiert. Andernfalls wird bei jedem Aufruf die Differenz zwischen **Input** und **Output** ermittelt. Ist der Betrag dieser Differenz kleiner oder gleich **fMaxRate * fCycletime** wird der Wert von **Input** unmittelbar als **fOutput** übernommen. Andernfalls wird **fOutput** um **fMaxRate * fCycletime** geändert. Dabei wird das Vorzeichen automatisch gewählt.

HINWEIS! Der Wert für **fCycletime** muss ≥ 0.001 sein. Für **fMaxRate** sind negative Werte nicht zulässig.

In der Regel wird es sich bei **Input** um einen Wert handeln, der im Takt der Achsbausteine ermittelt und gefiltert wird. Hier kann für **fCycletime** [Axis_Ref_BkPlcMc \[► 62\].ST_TcHydAxParam \[► 88\].fCycletime](#) verwendet werden.

3.4.9.2.5 MC_AxUtiSlidingAverage_BkPlcMc (ab V3.0)



Der Funktionsbaustein ermittelt einen gleitenden Mittelwert.

```

VAR_INPUT
  Input:      LREAL:=0.0;
  MinIdx:     DINT:=0;
  MaxIdx:     DINT:=0;
  pBuffer:    POINTER TO LREAL:=0;
END_VAR
VAR_OUTPUT
  Output:     LREAL:=0.0;
END_VAR

```

Input: Der Roh-Wert der zu filternden Größe.

MinIdx: Der Index des ersten zu verwendenden Elements des Filterpuffers.


MaxIdx: Der Index des letzten zu verwendenden Elements des Filterpuffers.

pBuffer: Die Adresse des ersten Elements des Filterpuffers.

Output: Der gefilterte Wert.

Verhalten des Bausteins

Bei jedem Aufruf überprüft der Baustein die Adresse des Filterpuffers **pBuffer** und die Indices der zu verwendenden Elemente **MinIdx** und **MaxIdx**. Sind die übergebenen Werte erkennbar unsinnig wird **Input** als **Output** ausgegeben. Andernfalls wird bei jedem Aufruf **Input** in den Filterpuffer eingetragen und der Mittelwert über den im Puffer verfügbaren Wertevorrat gebildet und als **Output** zurückgegeben.

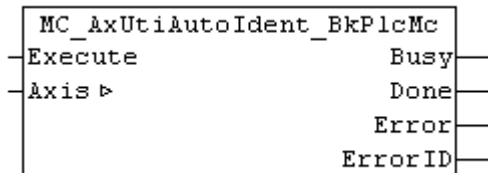
 Hinweis	Der Wertevorrat für die Mittelwertbildung umfasst (MaxIdx - MinIdx + 1) Werte. Die Filterzeit ergibt sich durch Multiplikation dieser Anzahl mit der Zykluszeit.
-----------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

HINWEIS! Das Prinzip der gleitenden Mittelwertbildung führt zu einer Verzögerung in Höhe der halben Filterzeit. Wird die gefilterte Größe in einem Regelkreis verwendet kann die dadurch verursachte frequenzabhängige Phasenverschiebung Einschränkungen bei der Parameterwahl verursachen.

HINWEIS! Der Baustein hat keine Möglichkeit, die übergebenen Werte von **pBuffer**, **MinIdx** und **MaxIdx** vollständig zu überprüfen. Es ist besonders darauf zu achten, dass diese Werte gefahrlos verwendet werden können. Andernfalls kann es zu nicht vorhersagbarem Verhalten (Überschreiben von Speicher) oder zum Abbrechen des PLC-Betriebs kommen.

3.4.9.3 Identification

3.4.9.3.1 MC_AxUtiAutoIdent_BkPlcMc (ab V3.0.28)



Der Funktionsbaustein ermittelt automatisch eine Reihe von Parametern der Achse.

```

VAR_INPUT
    Execute:    BOOL;
END_VAR
VAR_OUTPUT
    Busy:      BOOL;
    Done:     BOOL;
    Error:    BOOL;
    ErrorID:  UDINT;
END_VAR
VAR_INOUT
    Axis:      Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang löst die Identifikation aus.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird die erfolgreiche Durchführung der Identifikation signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Der Baustein überprüft, ob der Pointer `Axis_Ref_BkPlcMc` [▶ 62].`pStAxAutoParams` initialisiert wurde. Ist dies der Fall führt er bei einer steigenden Flanke am **Execute** Eingang eine Reihe von Initialisierungen aus und beginnt dann mit der Parameter-Identifikation. Die einzelnen Schritte der Identifikation werden durch die Werte in `ST_TcMcAutoIdent` [▶ 88] festgelegt.

EnableEndOfTravel: Wenn dieser boolesche Parameter gesetzt ist werden die mechanischen Fahrwegsgrenzen automatisch ermittelt. Hierzu wird zunächst sichergestellt, dass die Achse frei beweglich oder am positiven Block ist. Jetzt wird die Achse mit einer negativen Steuerspannung gefahren bis sie den Block erreicht hat. Anschließend wird die Achse mit positiver Steuerspannung betrieben bis der positive Block erkannt ist. Die Steuerspannung wird auf **EndOfTravel_NegativLimit** und **EndOfTravel_PositivLimit** begrenzt. Sollte die positive Fahrwegsgrenze kleiner sein als die negative werden die Werte getauscht und `Axis.stAxParams.bDrive_Reversed` invertiert.

EnableOverlap, EnableZeroAdjust: Wenn einer dieser booleschen Parameter gesetzt ist wird die Überdeckung bzw. die Offsetspannung des Ventils ermittelt. **HINWEIS! Diese Operation wird durch EndOfTravel_Negativ und EndOfTravel_Positiv beeinflusst.**

Zunächst wird die Achse in ein Positionsfenster bewegt, dass sich in der Mitte zwischen `pStAxAutoParams.EndOfTravel_Positiv` und `pStAxAutoParams.EndOfTravel_Negativ` befindet. Die Breite des Fensters ist 80% des durch diese Parameter festgelegten Bereichs. Dabei wird bei Bedarf die Ausgabepolarität des Drives invertiert. Jetzt wird die Ausgabespannung ermittelt, bei der die Achse beginnt, sich in positiver Richtung zu bewegen. Anschließend wird die entsprechende negative Spannung ermittelt. Durch Verrechnung dieser Parameter wird sowohl die Überdeckung als auch die Offsetspannung ermittelt. Die Art des Eintrags in die Achsparameter wird durch **EnableOverlap** und **EnableZeroAdjust** gesteuert.

EnableArreaRatio: Wenn dieser boolesche Parameter gesetzt ist wird das richtungsabhängige Geschwindigkeitsverhältnis ermittelt. Dazu wird die Achse zunächst in ein Positionsfenster bewegt, dass sich in der Mitte zwischen `pStAxAutoParams.EndOfTravel_Positiv` und `pStAxAutoParams.EndOfTravel_Negativ` befindet. Die Breite des Fensters ist 80% des durch diese Parameter festgelegten Bereichs. Anschließend wird in positiver und negativer Richtung für eine Sekunde mit einer Steuerspannung von 1 Volt gefahren. Die dabei ermittelten Geschwindigkeiten werden zur Ermittlung des Geschwindigkeitsverhältnisses dividiert.

EndOfTravel_Negativ: Wenn die Ermittlung der Fahrwegsgrenzen aktiviert ist wird dieser Wert vom Baustein ermittelt. Ist sie deaktiviert muss hier die Vorgabe durch die Applikation gemacht werden. **HINWEIS! Dieser Parameter beeinflusst die Ermittlung von Offsetspannung und Flächenverhältnis.**

EndOfTravel_Positiv: Wenn die Ermittlung der Fahrwegsgrenzen aktiviert ist wird dieser Wert vom Baustein ermittelt. Ist sie deaktiviert muss hier die Vorgabe durch die Applikation gemacht werden. **HINWEIS! Dieser Parameter beeinflusst die Ermittlung von Offsetspannung und Flächenverhältnis.**

EndOfIncrements_Negativ: Wenn die Ermittlung der Fahrwegsgrenzen aktiviert ist wird dieser Wert vom Baustein ermittelt. Er entspricht dann **EndOfTravel_Negativ**, ist aber der Encoder-Rohwert in Inkrementen.

EndOfIncrements_Positiv: Wenn die Ermittlung der Fahrwegsgrenzen aktiviert ist wird dieser Wert vom Baustein ermittelt. Er entspricht dann **EndOfTravel_Positiv**, ist aber der Encoder-Rohwert in Inkrementen.

EndOfTravel_NegativLimit: Dieser Parameter begrenzt negative Ausgangsspannungen.

EndOfTravel_PositivLimit: Dieser Parameter begrenzt positive Ausgangsspannungen.

EndOfTravel_PositivDone: Dieses Signal wird vom Baustein gesetzt, wenn die Ermittlung der Fahrwegsgrenzen deaktiviert ist oder die positive Fahrwegsgrenze ermittelt wurde.

EndOfTravel_NegativDone: Dieses Signal wird vom Baustein gesetzt, wenn die Ermittlung der Fahrwegsgrenzen deaktiviert ist oder die negative Fahrwegsgrenze ermittelt wurde.

DecelerationFactor: Nach dem Messhub wird die Achse für den nächsten Messhub zum Ende des Meßwegs bewegt. Dabei werden die mit diesem Faktor gewichteten regulären Achsparameter **fMaxAcc** und **fMaxDec** verwendet.

EnableValveCharacteristic: Wenn dieser boolesche Parameter gesetzt ist wird die Geschwindigkeitskennlinie automatisch ermittelt.

ValveCharacteristicTable: Dieses ARRAY[1..2,1..100] enthält die Wertepaare der Linearisierungstabelle. Dabei ist ValveCharacteristicTable[nnn,1] der normierte Geschwindigkeitswert und ValveCharacteristicTable[nnn,2] der normierte Ausgabewert. Innerhalb der Tabelle weisen die Wertepaare mit steigendem Index steigende Werte für Geschwindigkeitswert und Ausgabewert. Das erste Wertepaar beschreibt somit den schnellsten negativen und das letzte aktive Wertepaar den schnellsten positiven Punkt. Bei der automatischen Ermittlung wird die Steuerspannung auf **EndOfTravel_NegativLimit** und **EndOfTravel_PositivLimit** begrenzt. Die weiteren Punkte der Tabelle werden aus den letzten beiden Messpunkten durch Extrapolation ermittelt.

ValveCharacteristicType: reserviert.

ValveCharacteristicTblCount: Dieser Parameter legt die Anzahl der zu ermittelnden Wertepaare in **ValveCharacteristicTable** fest. Der Wert muss ungerade sein und zwischen 3 und 99 (einschließlich) liegen.

ValveCharacteristicLowEnd: Die untere Endposition des für die Kennlinienermittlung zugelassenen Bereichs.

ValveCharacteristicHighEnd: Die obere Endposition des für die Kennlinienermittlung zugelassenen Bereichs.

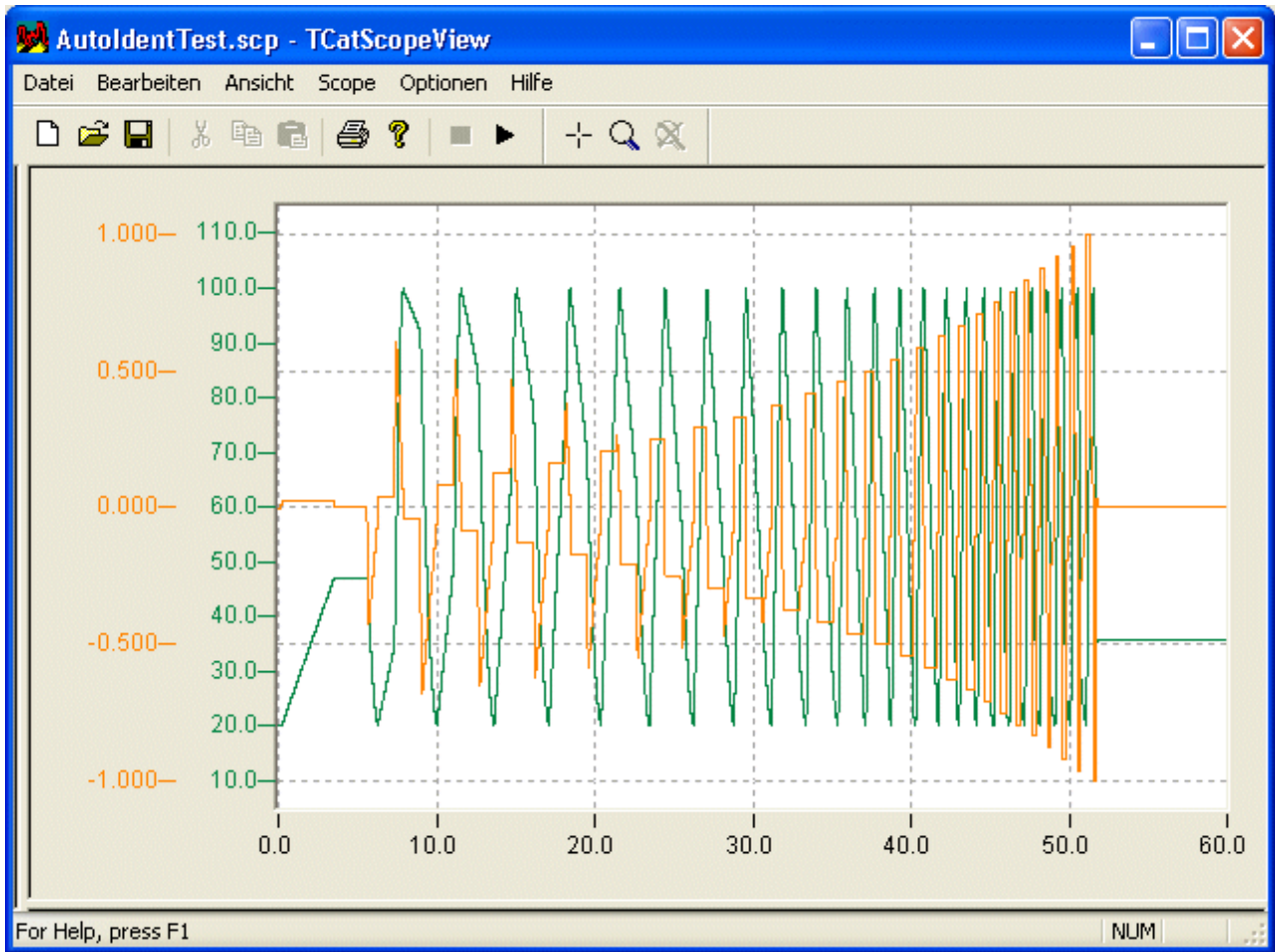
ValveCharacteristicRamp: Dieser Parameter legt die Rampe beim Aufbau der Messspannung für die Kennlinienermittlung fest. In der angegebenen Zeit wird die Spannung auf 10 Volt erhöht. Da die tatsächlichen Spannungen in der Regel geringer sind wird für den Aufbau eine entsprechend kleinere Zeit benötigt.

ValveCharacteristicSettling: Nachdem der Stellwert auf den Testpegel der Messung gerammt wurde kann der Start der Messung durch diesen Parameter verzögert werden.

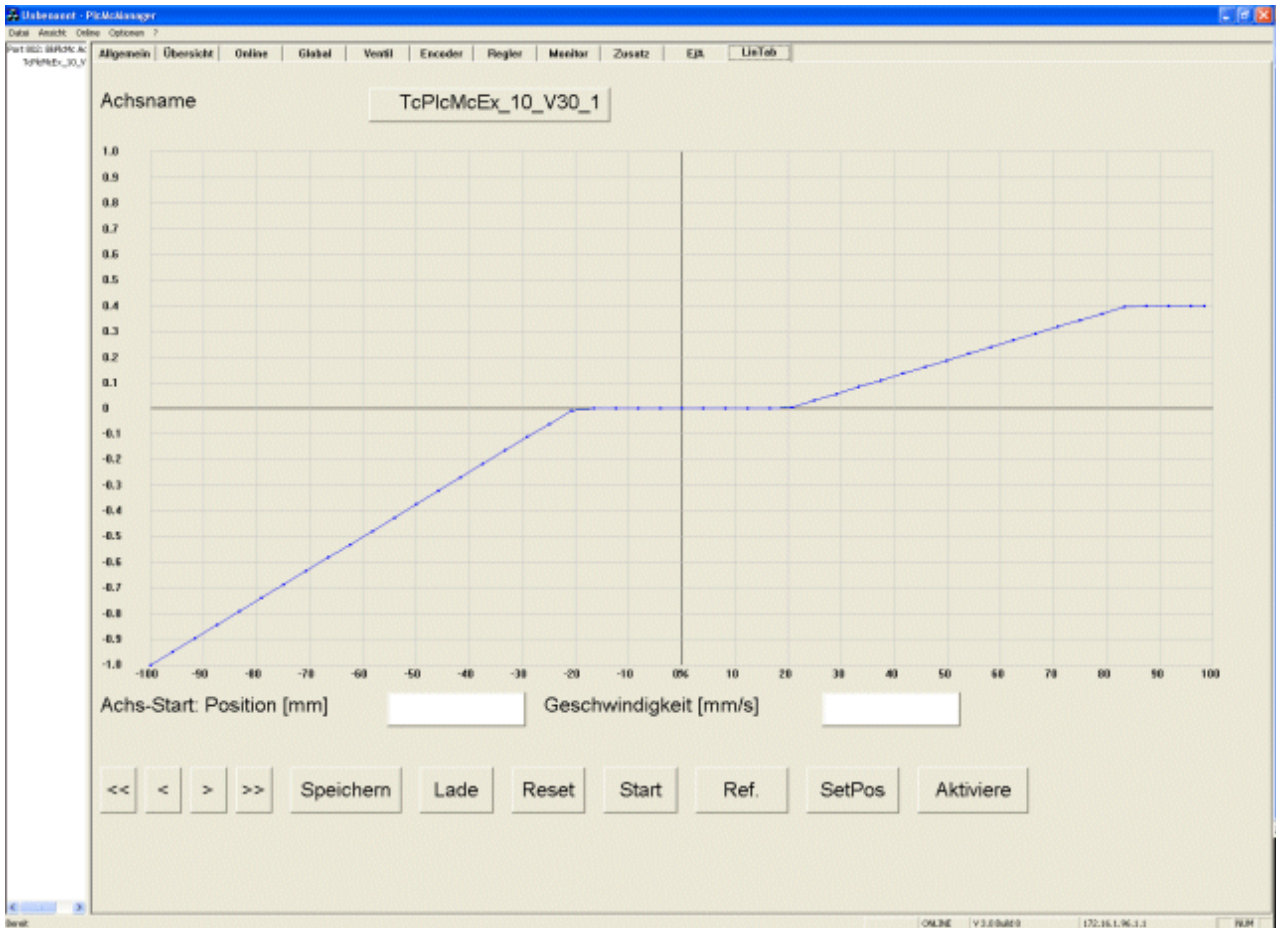
ValveCharacteristicRecovery: Dieser Parameter legt eine Verweilzeit fest, die vor der Messfahrt eingehalten wird. Dadurch erhält die Versorgung die Möglichkeit, einen von der vorherigen Messfahrt verursachten Druckabfall abzubauen. **HINWEIS! Die Achse wird während dieser Zeit nicht geregelt.**

ValveCharacteristicMinCycle: Die Messfahrt ist nur gültig wenn der Aufbau der Messspannung abgeschlossen ist, bevor die Achse sich der Mitte der durch **ValveCharacteristicHighEnd** und **ValveCharacteristicLowEnd** festgelegten Messstrecke auf weniger als der Hälfte dieses Wertes genähert hat. Andernfalls ist die effektive Messstrecke (ohne Rampen) kleiner als diese Strecke und diese Messung und alle weiteren in dieser Richtung werden durch einen unter Verwendung der Bezugsgeschwindigkeit der Achse errechneten Wert ersetzt.

Beispiel: Darstellung einer Kennlinienermittlung im TwinCAT ScopeView:



Beispiel: Anzeige einer Linearisierung im PlcMcManager:



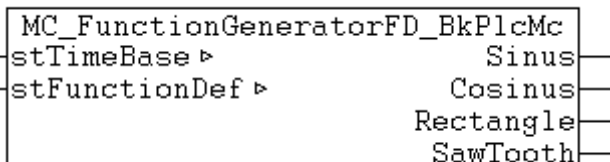
HINWEIS! Die so ermittelte Kennlinie kann mit einem MC_AxRtFinishLinear_BkPlcMc Baustein für die Linearisierung zur Laufzeit genutzt werden.

HINWEIS! Die Kennlinie wird in der Parameter-Datei der Achse gespeichert und beim Start des Systems automatisch gelesen.

HINWEIS! Es ist möglich, die Kennlinie mit einem MC_AxTableToAsciiFile_BkPlcMc Baustein in eine Textdatei oder mit einem MC_AxTableToBinFile_BkPlcMc in eine Binärdatei zu exportieren. Weiter ist es möglich, eine Kennlinie aus einer solchen Datei zu importieren.

3.4.9.4 Funktionsgenerator

3.4.9.4.1 MC_FunctionGeneratorFD_BkPlcMc (ab V3.0.31)



Der Funktionsbaustein berechnet die Signale eines Funktionsgenerators.

```

VAR_OUTPUT
Sinus:          LREAL;
Cosinus:        LREAL;
Rectangle:      LREAL;
SawTooth:       LREAL;
END_VAR
    
```

```

VAR_INOUT
  stTimeBase:    ST_FunctionGeneratorTB_BkPlcMc;
  stFunctionDef: ST_FunctionGeneratorFD_BkPlcMc;
END_VAR

```

Sinus, Cosinus, Rectangle, SawTooth: Die Ausgangssignale des Funktionsgenerators.

stTimeBase: Eine Struktur mit den Parametern der Zeitbasis dieses Funktionsgenerators.

stFunctionDef: Eine Struktur mit den Definitionen der Ausgangssignale eines Funktionsgenerators.

Verhalten des Bausteins

Aus **stTimeBase.CurrentRatio** und den Parametern in [stFunctionDef \[► 87\]](#) werden die Ausgangssignale ermittelt.

Die Zeitbasis in **stTimeBase** ist mit einem [MC_FunctionGeneratorTB_BkPlcMc \[► 148\]](#)() Baustein zu aktualisieren.

Zur Veränderung der Arbeitsfrequenz sollte ein [MC_FunctionGeneratorSetFrq_BkPlcMc \[► 147\]](#)() Baustein verwendet werden.

3.4.9.4.2 MC_FunctionGeneratorSetFrq_BkPlcMc (ab V3.0.31)

```

MC_FunctionGeneratorSetFrq_BkPlcMc
- Frequency
- CycleTime
- stTimeBase ►

```

Der Funktionsbaustein aktualisiert die Arbeitsfrequenz einer Zeitbasis für einen oder mehrere Funktionsgeneratoren [\[► 147\]](#).

```

VAR_INPUT
  Frequency:    LREAL;
  CycleTime:   LREAL;
END_VAR
VAR_INOUT
  stTimeBase:  ST_FunctionGeneratorTB_BkPlcMc;
END_VAR

```

Frequency: Die zu verwendende Arbeitsfrequenz.

CycleTime: Die Zykluszeit der aufrufenden Task.

stTimeBase: Eine Struktur mit den Parametern der Zeitbasis eines oder mehrerer Funktionsgeneratoren.

Verhalten des Bausteins

Der Baustein setzt **stTimeBase.Frequency** auf den übergebenen Wert. Dabei wird **stTimeBase.CurrentTime** bei Bedarf angepasst.

Der Baustein verhindert mit Hilfe von **stTimeBase.Freeze** eine Kollision mit [MC_FunctionGeneratorTB_BkPlcMc \[► 148\]](#)() Bausteinen. Somit kann er auch aus einer anderen Task aufgerufen werden.

3.4.9.4.3 MC_FunctionGeneratorTB_BkPlcMc (ab V3.0.31)

```

MC_FunctionGeneratorTB_BkPlcMc
- CycleTime
- stTimeBase ►

```

Der Funktionsbaustein aktualisiert eine Zeitbasis für einen oder mehrere Funktionsgeneratoren [► 147].

```
VAR_INPUT
  CycleTime:      LREAL;
END_VAR
VAR_INOUT
  stTimeBase:    ST_FunctionGeneratorTB_BkPlcMc;
END_VAR
```

CycleTime: Die Zykluszeit der aufrufenden Task.

stTimeBase: Eine Struktur mit den Parametern der Zeitbasis eines oder mehrerer Funktionsgeneratoren.

Verhalten des Bausteins

Wenn **stTimeBase.Freeze** nicht gesetzt ist wird **stTimeBase.CurrentTime** mit **CycleTime** aktualisiert und **stTimeBase.CurrentRatio** ermittelt. Dabei wird **stTimeBase.Frequency** berücksichtigt.

Zur Veränderung der Arbeitsfrequenz sollte ein MC FunctionGeneratorSetFrq_BkPlcMc [► 147]() Baustein verwendet werden.

3.5 Parameter


3.5.1 MC_AxAdsCommServer_BkPlcMc (ab V3.0)

```
MC_AxAdsCommServer_BkPlcMc
- nFirstAxisIndex
- nLastAxisIndex
- pAxItf
```

Der Funktionsbaustein stattet die Applikation mit den Fähigkeiten eines ADS-Servers aus. Er ruft bei Bedarf Bausteine vom Typ MC_AxAdsReadDecoder_BkPlcMc [► 189] und MC_AxAdsWriteDecoder_BkPlcMc [► 190] auf. Eine Auflistung der nutzbaren ADS-Codes [► 230] finden Sie in der Knowledge Base.

```
VAR_INPUT
  nFirstAxisIndex:  INT;
  nLastAxisIndex:  INT;
END_VAR
VAR_INOUT
  pAxItf:           POINTER TO Axis_Ref_BkPlcMc;
END_VAR
```

nFirstAxisIndex, nLastAxisIndex: Hier ist die Dimensionierung des Axis_Ref_BkPlcMc [► 62] Arrays anzugeben.

 <p>Achtung</p>	<p>Absturz der PLC-Applikation</p> <p>Eine nicht zutreffende Angabe an dieser Stelle schließt einen Teil der Achsen von der Kommunikation aus oder führt zum Absturz der PLC-Applikation durch Auslösung von schweren Laufzeitfehlern (Page Fault Exception)</p>
-----------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

pAxItf: Hier ist die Adresse einer Variablen oder eines Arrays von Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Durch zyklischen Aufruf dieses Bausteins in der PLC-Applikation erhält diese den Charakter eines ADS-Servers und beantwortet ADS-Read- und ADS-Write-Zugriffe wie jeder andere ADS Server. Dies schließt die Dekodierung einer IdxGroup/IdxOffset-Adressierung ein. Dazu werden bei Bedarf Bausteine vom Typ MC_AxAdsReadDecoder_BkPlcMc [► 189] und MC_AxAdsWriteDecoder_BkPlcMc [► 190] aufgerufen.

HINWEIS! Ist die PLC-Applikation bereits ein ADS-Server darf dieser Baustein nicht verwendet werden.

In diesem Fall sind die Bausteine vom Typ [MC_AxAdsReadDecoder_BkPlcMc](#) [▶ 189] und [MC_AxAdsWriteDecoder_BkPlcMc](#) [▶ 190] aus dem vorhandenen ADS Server-Baustein der Applikation aufzurufen.

3.5.2 MC_AxAdsPtrArrCommServer_BkPlcMc

```
MC_AxAdsPtrArrCommServer_BkPlcMc
nFirstAxisIndex
nLastAxisIndex
pAxItfArr
```

Der Funktionsbaustein stattet die Applikation mit den Fähigkeiten eines ADS-Servers aus. Er ruft bei Bedarf Bausteine vom Typ [MC_AxAdsReadDecoder_BkPlcMc](#) [▶ 189] und [MC_AxAdsWriteDecoder_BkPlcMc](#) [▶ 190] auf. Eine Auflistung der nutzbaren [ADS-Codes](#) [▶ 230] finden Sie in der Knowledge Base.

HINWEIS! Für die meisten Applikationen ist die Nutzung eines [MC_AxAdsCommServer_BkPlcMc](#) ausreichend und vorzuziehen. ([MC_AxAdsCommServer_BkPlcMc](#) [▶ 187])

```
VAR_INPUT
  nFirstAxisIndex: INT;
  nLastAxisIndex: INT;
END_VAR
VAR_INOUT
  pAxItfArr: POINTER TO DWORD;
END_VAR
```

nFirstAxisIndex, nLastAxisIndex: Hier ist die Dimensionierung des [Axis_Ref_BkPlcMc](#) [▶ 62] Arrays anzugeben.



Achtung

Absturz der PLC-Applikation

Eine nicht zutreffende Angabe an dieser Stelle schließt einen Teil der Achsen von der Kommunikation aus oder führt zum **Absturz der PLC-Applikation** durch Auslösung von schweren Laufzeitfehlern (**Page Fault Exception**).

pAxItfArr: Hier ist die Adresse einer Variablen vom Typ ARRAY [ncnstFirstAxId..ncnstLastAxId] OF POINTER TO [Axis_Ref_BkPlcMc](#) [▶ 62] zu übergeben.



Achtung

Absturz der PLC-Applikation

Eine nicht zutreffende Angabe an dieser Stelle führt unausweichlich zum Absturz der PLC-Applikation durch Auslösung von schweren Laufzeitfehlern (Page Fault Exception).

Verhalten des Bausteins

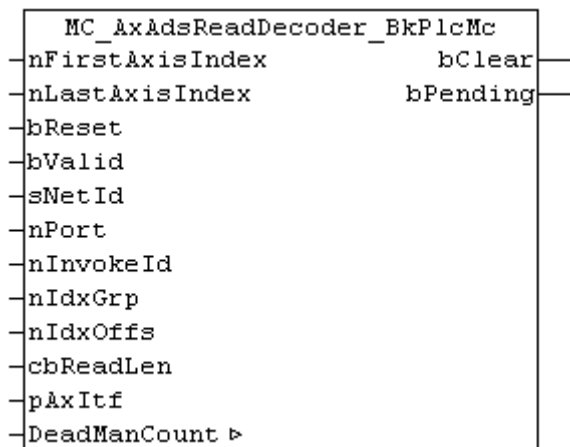
Durch zyklischen Aufruf dieses Bausteins in der PLC-Applikation erhält diese den Charakter eines ADS-Servers und beantwortet ADS-Read- und ADS-Write-Zugriffe wie jeder andere ADS Server. Dies schließt die Dekodierung einer IdxGroup/IdxOffset-Adressierung ein. Dazu werden bei Bedarf Bausteine vom Typ [MC_AxAdsReadDecoder_BkPlcMc](#) [▶ 189] und [MC_AxAdsWriteDecoder_BkPlcMc](#) [▶ 190] aufgerufen.

HINWEIS! Ist die PLC-Applikation bereits ein ADS-Server darf dieser Baustein nicht verwendet werden.

In diesem Fall sind die Bausteine vom Typ [MC_AxAdsReadDecoder_BkPlcMc](#) [▶ 189] und [MC_AxAdsWriteDecoder_BkPlcMc](#) [▶ 190] aus dem vorhandenen ADS Server-Baustein der Applikation aufzurufen.

Es steht ein [Programm-Beispiel \[► 208\]](#) #16 zur Verfügung.

3.5.3 MC_AxAdsReadDecoder_BkPlcMc (ab V3.0)



Der Funktionsbaustein dekodiert ADS-Read-Zugriffe. Eine Auflistung der nutzbaren [ADS-Codes \[► 230\]](#) finden Sie in der Knowledge Base.

```

VAR_INPUT
  nFirstAxisIndex:   INT;
  nLastAxisIndex:   INT;
  bReset:            BOOL;
  bValid:            BOOL;
  sNetId:            STRING(80);
  nPort:             UINT;
  nInvokeId:        UDINT;
  nIdxGroup:        UDINT;
  nIdxOffs:         UDINT;
  cbReadLen:        UDINT;
  pAxItf:           POINTER TO Axis_Ref_BkPlcMc:=0;
END_VAR
VAR_INOUT
  DeadManCount:     UDINT;
END_VAR
VAR_OUTPUT
  bClear:           BOOL;
  bPending:         BOOL;
END_VAR

```

nFirstAxisIndex, nLastAxisIndex: Hier ist die Dimensionierung des [Axis_Ref_BkPlcMc \[► 62\]](#) Arrays anzugeben.

VORSICHT! Eine nicht zutreffende Angabe an dieser Stelle schließt einen Teil der Achsen von der Kommunikation aus oder führt zum Absturz der PLC-Applikation durch Auslösung von schweren Laufzeitfehlern (Page Fault Exception).

bReset, bValid: Diese Signale dienen zur Koordinierung des Dekoders mit dem ADS-Server.

sNetId, nPort, nInvokeId: Diese Werte werden für die Erzeugung des ADS Response benötigt. Sie werden vom ADS-Indication-Baustein eines ADS-Servers geliefert.

nIdxGroup, nIdxOffs, cbReadLen: Diese Werte werden für die Dekodierung des Zugriffs benötigt. Sie werden vom ADS-Indication-Baustein eines ADS-Servers geliefert.

pAxItf: Hier ist die Adresse einer Variablen oder eines Arrays von Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

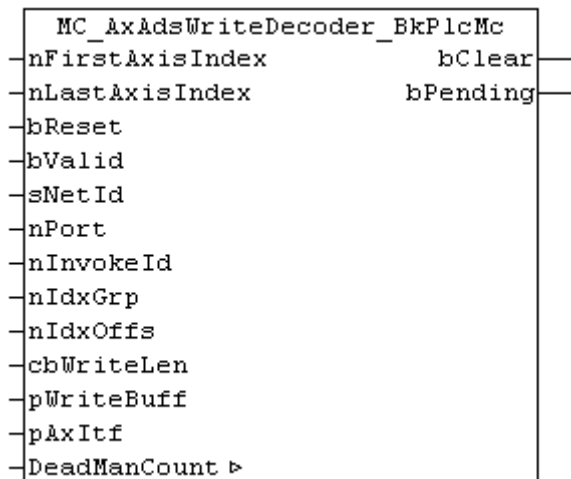
bClear: Hier wird signalisiert, dass ein mit bValid signalisierter ADS-Zugriff quittiert werden soll.

bPending: Hier wird signalisiert, dass ein mit bValid signalisierter ADS-Zugriff bearbeitet wird.

Verhalten des Bausteins

Signalisiert der Baustein bei anstehendem bValid-Signal weder bClear noch bPending hat er die Kombination aus nIdxGroup und nIdxOffs nicht dekodiert und es wurde kein Response erzeugt. In diesem Fall muss der ADS-Server (soweit vorhanden) andere Decoder aufrufen oder einen Response mit entsprechendem Errorcode erzeugen.

3.5.4 MC_AxAdsWriteDecoder_BkPlcMc (ab V3.0)



Der Funktionsbaustein dekodiert ADS-Write-Zugriffe. Eine Auflistung der nutzbaren [ADS-Codes](#) [► 230] finden Sie in der Knowledge Base.

```

VAR_INPUT
  nFirstAxisIndex:   INT;
  nLastAxisIndex:   INT;
  bReset:            BOOL;
  bValid:            BOOL;
  sNetId:            STRING(80);
  nPort:             UINT;
  nInvokeId:        UDINT;
  nIdxGroup:         UDINT;
  nIdxOffs:          UDINT;
  cbWriteLen:        UDINT;
  pWriteBuff:        DWORD;
  pAxItf:            POINTER TO Axis_Ref_BkPlcMc:=0;
END_VAR
VAR_INOUT
  DeadManCount:      UDINT;
END_VAR
VAR_OUTPUT
  bClear:            BOOL;
  bPending:          BOOL;
END_VAR

```

nFirstAxisIndex, nLastAxisIndex: Hier ist die Dimensionierung des [Axis_Ref_BkPlcMc](#) [► 62] Arrays anzugeben.

VORSICHT! Eine nicht zutreffende Angabe an dieser Stelle schließt einen Teil der Achsen von der Kommunikation aus oder führt zum Absturz der PLC-Applikation durch Auslösung von schweren Laufzeitfehlern (Page Fault Exception).

bReset, bValid: Diese Signale dienen zur Koordinierung des Decoders mit dem ADS-Server.

sNetId, nPort, nInvokeId: Diese Werte werden für die Erzeugung des ADS Response benötigt. Sie werden vom ADS-Indication-Baustein eines ADS-Servers geliefert.

nIdxGroup, nIdxOffs, cbWriteLen: Diese Werte werden für die Dekodierung des Zugriffs benötigt. Sie werden vom ADS-Indication-Baustein eines ADS-Servers geliefert.

pAxItf: Hier ist die Adresse einer Variablen oder eines Arrays von Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

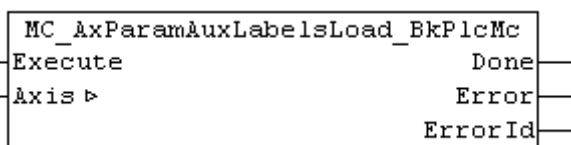
bClear: Hier wird signalisiert, dass ein mit bValid signalisierter ADS-Zugriff quittiert werden soll.

bPending: Hier wird signalisiert, dass ein mit bValid signalisierter ADS-Zugriff bearbeitet wird.

Verhalten des Bausteins

Signalisiert der Baustein bei anstehendem bValid-Signal weder bClear noch bPending hat er die Kombination aus nIdxGroup und nIdxOffs nicht dekodiert und es wurde kein Response erzeugt. In diesem Fall muss der ADS-Server (soweit vorhanden) andere Decoder aufrufen oder einen Response mit entsprechendem Errorcode erzeugen.

3.5.5 MC_AxParamAuxLabelsLoad_BkPlcMc (ab V3.0)



Der Funktionsbaustein lädt die Beschriftungstexte für die kundenspezifischen Achsparameter aus einer Datei. Diese Texte können mit einem einfachen Texteditor wie Microsoft Notepad erzeugt werden.



Hinweis

Die Datei muss strikt entsprechend der unten angegebenen Regeln aufgebaut sein. Andernfalls kann es zu erheblichen Problemen bis hin zum Systemabsturz kommen.

Dieser Baustein wird in der Regel nicht direkt durch die Applikation aufgerufen. Soweit möglich sollte ein Baustein des Typs MC_AxUtiStandardInit_BkPlcMc [► 171] verwendet werden, der einen Baustein des Typs **MC_AxParamAuxLabelsLoad_BkPlcMc** verwendet.

```

VAR_INPUT
  Execute:      BOOL;
END_VAR
VAR_OUTPUT
  Done:        BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet den Ladevorgang.

Done: Hier wird das erfolgreiche Laden der Parameter signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

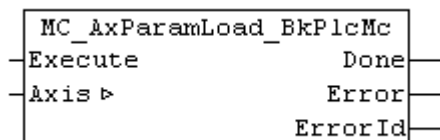
- Wenn einer der Pointer nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** oder **dwTcHydErrCdPtrMcPlc** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird der Ladevorgang initiiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Ladevorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

HINWEIS! Die Anzahl der Zeilen in der Datei muss mit der in den globalen Konstanten der Bibliothek als **iTcHydfCustDataMaxIdx** (derzeit: 20) festgelegten Anzahl übereinstimmen. Jede der Zeilen darf maximal 20 Zeichen (inklusive Leerzeichen, ohne Zeilenwechsel) enthalten.

3.5.6 MC_AxParamLoad_BkPlcMc (ab V3.0)



Der Funktionsbaustein lädt die Parameter einer Achse aus einer Datei. Für die kompatible Erzeugung der Parameterdatei ist ein Baustein vom Typ MC_AxParamSave_BkPlcMc [► 193] zu verwenden.

Dieser Baustein wird in der Regel nicht direkt durch die Applikation aufgerufen. Soweit möglich sollte ein Baustein des Typs MC_AxUtiStandardInit_BkPlcMc [► 171] verwendet werden, der einen Baustein des Typs **MC_AxParamLoad_BkPlcMc** verwendet.

```

VAR_INPUT
    Execute:      BOOL;
END_VAR
VAR_OUTPUT
    Done:         BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
END_VAR
VAR_INOUT
    Axis:         Axis_Ref_BkPlcMc;
END_VAR
  
```

Execute: Eine steigende Flanke an diesem Eingang startet den Ladevorgang.

Done: Hier wird das erfolgreiche Laden der Parameter signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

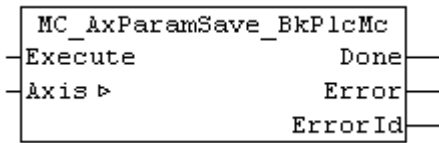
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Datei nicht zum Lesen geöffnet werden kann wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** oder **dwTcHydErrCdPtrMcPlc** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnte wird der Ladevorgang initiiert. Dabei wird die Version der Datei ermittelt und nicht durch die Datei festgelegte Parameter durch neutrale Default-Werte ersetzt. Sollte die Datei nicht mehr oder noch nicht verwendete Parameter enthalten werden diese ignoriert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Ladevorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

3.5.7 MC_AxParamSave_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt die Parameter einer Achse in eine Datei. Für das Einlesen der Datei ist ein Baustein vom Typ MC_AxParamLoad_BkPlcMc [► 192] zu verwenden.

```

VAR_INPUT
    Execute:          BOOL;
END_VAR
VAR_OUTPUT
    Done:            BOOL;
    Error:           BOOL;
    ErrorID:         UDINT;
END_VAR
VAR_INOUT
    Axis:            Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

Done: Hier wird das erfolgreiche Schreiben der Parameter signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

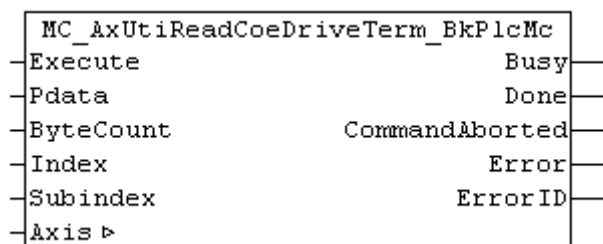
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Datei nicht zum Schreiben geöffnet werden kann wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** oder **dwTcHydErrCdPtrMcPlc** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnte wird der Schreibvorgang initiiert. Dabei wird die Version der abgespeicherten Parameter festgehalten.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

3.5.8 MC_AxUtiReadCoeDriveTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest den Inhalt eines Registers aus der EL-Klemme, die als Antriebsschnittstelle für die Achse dient.

```

VAR_INPUT
  Execute:      BOOL;
  Pdata:       POINTER TO BYTE:=0;
  ByteCount:   BYTE:=0;
  Index:       WORD:=0;
  Subindex:    BYTE:=0;
END_VAR
VAR_OUTPUT
  Busy:        BOOL;
  Done:        BOOL;
  CommandAborted: BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Lesevorgang.

Pdata: Hier ist die Adresse der Variablen anzugeben, in die der gelesene Wert ausgegeben werden soll.

ByteCount: Hier ist die Größe der Variablen in Bytes anzugeben.

Index, Subindex: Hier ist die Adressierung des Parameters in der Klemme anzugeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Laden des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

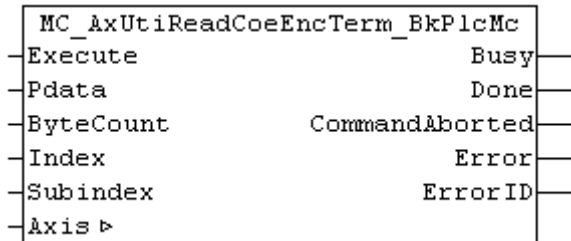
Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Index** oder **Subindex** ausserhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn **ByteCount** oder **Pdata** ausserhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn in den Achsparametern als nDrive_Type eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.
- Wenn es bei der ADS-Kommunikation mit der Klemme zu Problemen kommt wird der entsprechende ADS Error Code als **ErrorID** zurückgegeben und dies mit **Error** kenntlich gemacht. Dabei können unter anderen folgende Codes [► 224] auftreten:
 - 16#0006 = 6 = Die Portnummer der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0007 = 7 = Die AmsNetID der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0702 = 1794 = dwTcHydAdsErrInvalidIdxGroup = Die Klemme unterstützt nicht das CoE Protokoll.
 - 16#0703 = 1795 = dwTcHydAdsErrInvalidIdxOffset = Die Adresse in Index und Subindex ist in der Klemme nicht unterstützt.
 - 16#0745 = 1861 = dwTcHydAdsErrTimeout = Zeitüberschreitung.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Ladevorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Done**, **CommandAborted**, **Error**, **ErrorID**) werden für einen Zyklus gegeben.

3.5.9 MC_AxUtiReadCoeEncTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest den Inhalt eines Registers aus der EL-Klemme, die als Encoderschnittstelle für die Achse dient.

```

VAR_INPUT
  Execute:      BOOL;
  Pdata:        POINTER TO BYTE:=0;
  ByteCount:    BYTE:=0;
  Index:        WORD:=0;
  Subindex:     BYTE:=0;
END_VAR
VAR_OUTPUT
  Busy:         BOOL;
  Done:         BOOL;
  CommandAborted: BOOL;
  Error:        BOOL;
  ErrorID:      UDINT;
END_VAR
VAR_INOUT
  Axis:         Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Lesevorgang.

Pdata: Hier ist die Adresse der Variablen anzugeben, in die der gelesene Wert ausgegeben werden soll.

ByteCount: Hier ist die Größe der Variablen in Bytes anzugeben.

Index, Subindex: Hier ist die Adressierung des Parameters in der Klemme anzugeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Laden des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

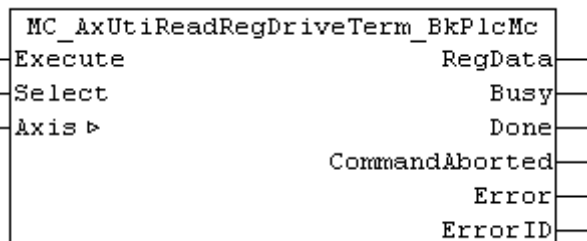
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Index** oder **Subindex** außerhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.

- Wenn **ByteCount** oder **Pdata** außerhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn in den Achsparametern als **nEncoder_Type** eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.
- Wenn es bei der ADS-Kommunikation mit der Klemme zu Problemen kommt wird der entsprechende ADS Error Code als **ErrorID** zurückgegeben und dies mit **Error** kenntlich gemacht. Dabei können unter anderen folgende **Codes** [► 224] auftreten:
 - 16#0006 = 6 = Die Portnummer der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0007 = 7 = Die AmsNetID der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0702 = 1794 = dwTcHydAdsErrInvalidIdxGroup = Die Klemme unterstützt nicht das CoE Protokoll.
 - 16#0703 = 1795 = dwTcHydAdsErrInvalidIdxOffset = Die Adresse in Index und Subindex ist in der Klemme nicht unterstützt.
 - 16#0745 = 1861 = dwTcHydAdsErrTimeout = Zeitüberschreitung.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Ladevorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Done**, **CommandAborted**, **Error**, **ErrorID**) werden für einen Zyklus gegeben.

3.5.10 MC_AxUtiReadRegDriveTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest den Inhalt eines Registers aus der KL-Klemme, die als Antriebsschnittstelle für die Achse dient.

```

VAR_INPUT
    Execute:      BOOL;
    Select:       INT;
END_VAR
VAR_OUTPUT
    RegData:      WORD;
    Busy:         BOOL;
    Done:         BOOL;
    CommandAborted: BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
END_VAR
VAR_INOUT
    Axis:         Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Lesevorgang.

Select: Hier ist die Registernummer zu übergeben.

RegData: Hier wird der gelesene Wert ausgegeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Laden des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ `Axis_Ref_BkPlcMc` [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

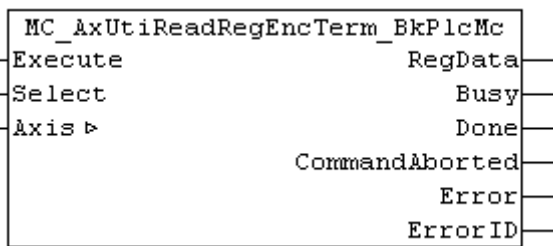
- Wenn einer der Pointer `ST_TcPlcDeviceInput` [► 98] und `ST_TcPlcDeviceOutput` [► 100] nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Select** außerhalb des zulässigen Bereichs von 0 bis 63 liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdTblIllegalIndex** reagiert.
- Wenn in den Achsparametern als `nDrive_Type` eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird der Lesevorgang initiiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Ladevorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**RegData**, **Done**, **CommandAborted**, **Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

HINWEIS! Die Drivetypes `iTcMc_DriveKL2521`, `iTcMc_DriveKL4032`, `iTcMc_DriveKL2531` und `iTcMc_DriveKL2541` unterstützen die Parameterkommunikation.

3.5.11 MC_AxUtiReadRegEncTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein liest den Inhalt eines Registers aus der KL-Klemme, die als Encoderschnittstelle für die Achse dient.

```

VAR_INPUT
    Execute:      BOOL;
    Select:       INT;
END_VAR
VAR_OUTPUT
    RegData:      WORD;
    Busy:         BOOL;
    Done:         BOOL;
    CommandAborted: BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
END_VAR
VAR_INOUT
    Axis:         Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet den Lesevorgang.

Select: Hier ist die Registernummer zu übergeben.

RegData: Hier wird der gelesene Wert ausgegeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Laden des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

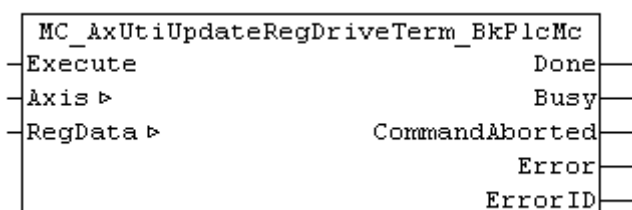
- Wenn einer der Pointer ST_TcPlcDeviceInput [► 98] und ST_TcPlcDeviceOutput [► 100] nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Select** ausserhalb des zulässigen Bereichs von 0 bis 63 liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdTblIllegalIndex** reagiert.
- Wenn in den Achsparametern als `nEncoder_Type` eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird der Lesevorgang initiiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Ladevorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**RegData**, **Done**, **CommandAborted**, **Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

HINWEIS! Die Drivetypes `iTcMc_EncoderKL3002`, `iTcMc_EncoderKL3042`, `iTcMc_EncoderKL3062`, `iTcMc_EncoderKL3162`, `iTcMc_EncoderKL5101`, `iTcMc_EncoderKL5111`, `iTcMc_EncoderKL2521`, `iTcMc_EncoderKL2531` und `iTcMc_EncoderKL2541` unterstützen die Parameterkommunikation.

3.5.12 MC_AxUtiUpdateRegDriveTerm_BkPlcMc (ab V3.0.7)



Der Funktionsbaustein schreibt einen Parametersatz in die Register einer KL-Klemme. Dazu benutzt er MC_AxUtiReadRegDriveTerm_BkPlcMc [► 196] und MC_AxUtiWriteRegDriveTerm_BkPlcMc [► 204] Bausteine.

```
VAR_INPUT
  Execute:      BOOL;
END_VAR
VAR_OUTPUT
  Done:        BOOL;
  Busy:        BOOL;
  CommandAborted: BOOL;
```

```

    Error:          BOOL;
    ErrorID:        UDINT;
END_VAR
VAR_INOUT
    Axis:          Axis_Ref_BkPlcMc;
    RegData:       ST_TcPlcRegDataTable;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

RegData: Hier ist die Adresse des Parametersatzes anzugeben, deren Inhalt in die Klemme geschrieben werden soll.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

Done: Hier wird das erfolgreiche Schreiben des Parameters signalisiert.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

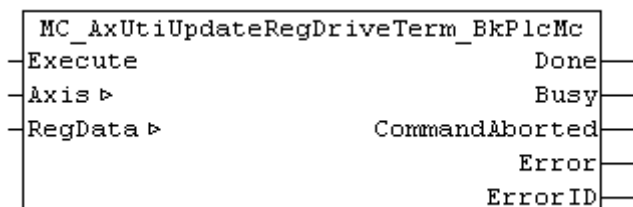
Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn einer der Pointer ST_TcPlcDeviceInput [► 98] und ST_TcPlcDeviceOutput [► 100] nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Select** außerhalb des zulässigen Bereichs von 0 bis 63 liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdTblIllegalIndex** reagiert.
- Wenn in den Achsparametern als `nDrive_Type` eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.
- Der Wert in ST_TcPlcRegDataTable [► 103].`RegDataItem[...].Access` legt fest, wie das Element behandelt wird.
 - 0: Element wird ignoriert.
 - 1: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem ungleichen Inhalt wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 2: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht größer ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 3: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht kleiner ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 4: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht größer oder gleich ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 5: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht kleiner oder gleich ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 10: Das durch **Select** adressierte Register wird mit **RegData** geschrieben.
 - Andere Werte werden derzeit ignoriert. Zukünftige Versionen der Bibliothek werden jedoch möglicherweise weitere Funktionen unterstützen. Ein leeres Element sollte daher immer mit 0 gekennzeichnet werden.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Done**, **CommandAborted**, **Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

3.5.13 MC_AxUtiUpdateRegEncTerm_BkPlcMc (ab V3.0.7)



Der Funktionsbaustein schreibt einen Parametersatz in die Register einer KL-Klemme. Dazu benutzt er [MC_AxUtiReadRegEncTerm_BkPlcMc \[► 197\]](#) und [MC_AxUtiWriteRegEncTerm_BkPlcMc \[► 205\]](#) Bausteine.

```

VAR_INPUT
  Execute:          BOOL;
END_VAR
VAR_OUTPUT
  Done:            BOOL;
  Busy:            BOOL;
  CommandAborted: BOOL;
  Error:           BOOL;
  ErrorID:        UDINT;
END_VAR
VAR_INOUT
  Axis:            Axis_Ref_BkPlcMc;
  RegData:        ST_TcPlcRegDataTable;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

RegData: Hier ist die Adresse des Parametersatzes anzugeben, deren Inhalt in die Klemme geschrieben werden soll.

Axis: Hier ist die Adresse einer Variablen vom Typ [Axis_Ref_BkPlcMc \[► 62\]](#) zu übergeben.

Done: Hier wird das erfolgreiche Schreiben des Parameters signalisiert.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Verhalten des Bausteins

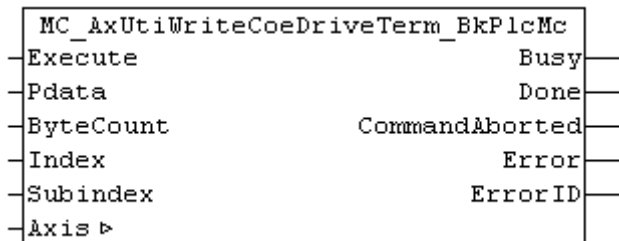
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn einer der Pointer [ST_TcPlcDeviceInput \[► 98\]](#) und [ST_TcPlcDeviceOutput \[► 100\]](#) nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Select** außerhalb des zulässigen Bereichs von 0 bis 63 liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdTblIllegalIndex** reagiert.
- Wenn in den Achsparametern als nDrive_Type eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

- Der Wert in `ST TcPlcRegDataTable [▶ 103].RegDataItem[...].Access` legt fest, wie das Element behandelt wird.
 - 0: Element wird ignoriert.
 - 1: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem ungleichen Inhalt wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 2: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht größer ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 3: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht kleiner ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 4: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht größer oder gleich ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 5: Das durch **Select** adressierte Register wird gelesen. Sein Inhalt wird mit **RegData** verglichen. Bei einem Inhalt der nicht kleiner oder gleich ist wird der Schreibvorgang mit **Error** und **ErrorID:=16#FFFFFFFF** abgebrochen.
 - 10: Das durch **Select** adressierte Register wird mit **RegData** geschrieben.
 - Andere Werte werden derzeit ignoriert. Zukünftige Versionen der Bibliothek werden jedoch möglicherweise weitere Funktionen unterstützen. Ein leeres Element sollte daher immer mit 0 gekennzeichnet werden.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt, wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Done**, **CommandAborted**, **Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

3.5.14 MC_AxUtiWriteCoeDriveTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt den Inhalt eines Registers der EL-Klemme, die als Antriebsschnittstelle für die Achse dient.

```

VAR_INPUT
    Execute:      BOOL;
    Pdata:        POINTER TO BYTE:=0;
    ByteCount:    BYTE:=0;
    Index:        WORD:=0;
    Subindex:     BYTE:=0;
END_VAR
VAR_OUTPUT
    Busy:         BOOL;
    Done:         BOOL;
    CommandAborted: BOOL;
    Error:        BOOL;
    ErrorID:      UDINT;
END_VAR
VAR_INOUT
    Axis:         Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

Pdata: Hier ist die Adresse der Variablen anzugeben, deren Inhalt in die Klemme geschrieben werden soll.

ByteCount: Hier ist die Größe der Variablen in Bytes anzugeben.

Index, Subindex: Hier ist die Adressierung des Parameters in der Klemme anzugeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Schreiben des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [► 62] zu übergeben.

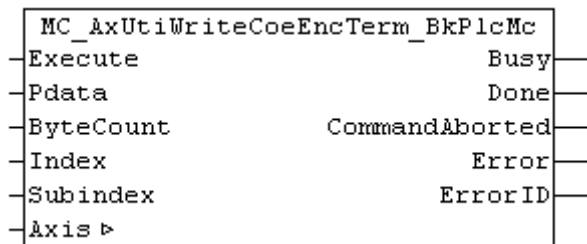
Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Index** oder **Subindex** außerhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn **ByteCount** oder **Pdata** außerhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn in den Achsparametern als nDrive_Type eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.
- Wenn es bei der ADS-Kommunikation mit der Klemme zu Problemen kommt wird der entsprechende ADS Error Code als **ErrorID** zurückgegeben und dies mit **Error** kenntlich gemacht. Dabei können unter anderen folgende Codes [► 224] auftreten:
 - 16#0006 = 6 = Die Portnummer der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0007 = 7 = Die AmsNetID der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0702 = 1794 = dwTcHydAdsErrInvalidIdxGroup = Die Klemme unterstützt nicht das CoE Protokoll.
 - 16#0703 = 1795 = dwTcHydAdsErrInvalidIdxOffset = Die Adresse in Index und Subindex ist in der Klemme nicht unterstützt.
 - 16#0745 = 1861 = dwTcHydAdsErrTimeout = Zeitüberschreitung.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Done, CommandAborted, Error, ErrorID**) werden für einen Zyklus gegeben.

3.5.15 MC_AxUtiWriteCoeEncTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt den Inhalt eines Registers der EL-Klemme, die als Encoderschnittstelle für die Achse dient.

```

VAR_INPUT
  Execute:      BOOL;
  Pdata:       POINTER TO BYTE:=0;
  ByteCount:   BYTE:=0;
  Index:       WORD:=0;
  Subindex:    BYTE:=0;
END_VAR
VAR_OUTPUT
  Busy:        BOOL;
  Done:        BOOL;
  CommandAborted: BOOL;
  Error:       BOOL;
  ErrorID:     UDINT;
END_VAR
VAR_INOUT
  Axis:        Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

Pdata: Hier ist die Adresse der Variablen anzugeben, deren Inhalt in die Klemme geschrieben werden soll.

ByteCount: Hier ist die Größe der Variablen in Bytes anzugeben.

Index, Subindex: Hier ist die Adressierung des Parameters in der Klemme anzugeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Schreiben des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

Verhalten des Bausteins

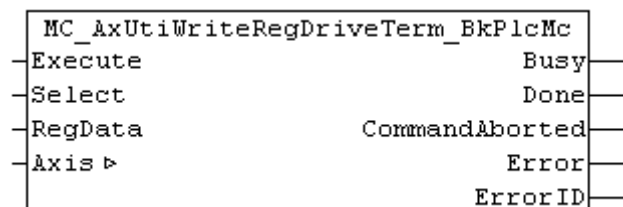
Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Index** oder **Subindex** außerhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn **ByteCount** oder **Pdata** außerhalb des zulässigen Bereichs liegen wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn in den Achsparametern als **nEncoder_Type** eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

- Wenn es bei der ADS-Kommunikation mit der Klemme zu Problemen kommt wird der entsprechende ADS Error Code als **ErrorID** zurückgegeben und dies mit **Error** kenntlich gemacht. Dabei können unter anderen folgende Codes [▶ 224] auftreten:
 - 16#0006 = 6 = Die Portnummer der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0007 = 7 = Die AmsNetID der verwendeten ADS-Adresse ist ungültig: Mapping des Info-Data Elements der Klemme überprüfen!
 - 16#0702 = 1794 = dwTcHydAdsErrInvalidIdxGroup = Die Klemme unterstützt nicht das CoE Protokoll.
 - 16#0703 = 1795 = dwTcHydAdsErrInvalidIdxOffset = Die Adresse in Index und Subindex ist in der Klemme nicht unterstützt.
 - 16#0745 = 1861 = dwTcHydAdsErrTimeout = Zeitüberschreitung.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**Done**, **CommandAborted**, **Error**, **ErrorID**) werden für einen Zyklus gegeben.

3.5.16 MC_AxUtiWriteRegDriveTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt den Inhalt eines Registers der KL-Klemme, die als Antriebsschnittstelle für die Achse dient.

```

VAR_INPUT
  Execute:      BOOL;
  Select:       INT;
  RegData:      WORD;
END_VAR
VAR_OUTPUT
  Busy:         BOOL;
  Done:         BOOL;
  CommandAborted: BOOL;
  Error:        BOOL;
  ErrorID:     UDINT;
END_VAR
VAR_INOUT
  Axis:         Axis_Ref_BkPlcMc;
END_VAR

```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

Select: Hier ist die Registernummer zu übergeben.

RegData: Hier ist der zu schreibende Wert zu übergeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Schreiben des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis_Ref_BkPlcMc [▶ 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

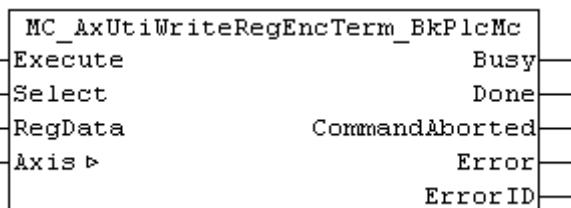
- Wenn einer der Pointer ST_TcPlcDeviceInput [▶ 98] und ST_TcPlcDeviceOutput [▶ 100] nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Select** außerhalb des zulässigen Bereichs von 0 bis 63 liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdTbllllegalIndex** reagiert.
- Wenn in den Achsparametern als nDrive_Type eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird der Schreibvorgang initiiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**RegData**, **Done**, **CommandAborted**, **Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

HINWEIS! Die Drivetypes **iTcMc_DriveKL2521**, **iTcMc_DriveKL4032**, **iTcMc_DriveKL2531** und **iTcMc_DriveKL2541** unterstützen die Parameterkommunikation.

3.5.17 MC_AxUtiWriteRegEncTerm_BkPlcMc (ab V3.0)



Der Funktionsbaustein schreibt den Inhalt eines Registers der KL-Klemme, die als Encoderschnittstelle für die Achse dient.

```

VAR_INPUT
  Execute:      BOOL;
  Select:       INT;
  RegData:      WORD;
END_VAR
VAR_OUTPUT
  Busy:         BOOL;
  Done:         BOOL;
  CommandAborted: BOOL;
  Error:        BOOL;
  ErrorID:      UDINT;
END_VAR
VAR_INOUT
  Axis:         Axis_Ref_BkPlcMc;
END_VAR
    
```

Execute: Eine steigende Flanke an diesem Eingang startet den Schreibvorgang.

Select: Hier ist die Registernummer zu übergeben.

RegData: Hier ist der zu schreibende Wert zu übergeben.

Busy: Hier wird signalisiert, dass ein Kommando abgearbeitet wird.

Done: Hier wird das erfolgreiche Schreiben des Parameters signalisiert.

CommandAborted: Hier wird ein Abbruch des Lesevorgangs signalisiert.

Error: Hier wird das Auftreten eines Fehlers signalisiert.

ErrorID: Hier wird eine codierte Fehlerursache bereitgestellt.

Axis: Hier ist die Adresse einer Variablen vom Typ Axis Ref BkPlcMc [► 62] zu übergeben.

Verhalten des Bausteins

Auf eine steigende Flanke an **Execute** hin untersucht der Baustein das übergebene Achsinterface. Dabei können eine Reihe von Problemen erkannt und gemeldet werden:

- Wenn einer der Pointer ST_TcPlcDeviceInput [► 98] und ST_TcPlcDeviceOutput [► 100] nicht initialisiert ist wird mit **Error** und **ErrorID:=dwTcHydErrCdPtrPlcMc** reagiert.
- Wenn die Achse für den Betrieb freigegeben ist wird mit **Error** und **ErrorID:=dwTcHydErrCdNotReady** reagiert.
- Wenn **Select** außerhalb des zulässigen Bereichs von 0 bis 63 liegt wird mit **Error** und **ErrorID:=dwTcHydErrCdTblIllegalIndex** reagiert.
- Wenn in den Achsparametern als nEncoder_Type eine E/A-Baugruppe eingestellt ist, die keine Parameterkommunikation unterstützt wird mit **Error** und **ErrorID:=dwTcHydErrCdNotCompatible** reagiert.

Wenn diese Überprüfungen ohne Problem durchgeführt werden konnten wird der Schreibvorgang initiiert.

Eine fallende Flanke an **Execute** löscht alle anstehenden Ausgangssignale. Wird **Execute** bereits bei noch aktivem Schreibvorgang auf FALSE gesetzt wird der eingeleitete Vorgang unbeeinflusst weiter bearbeitet. Die Signale am Ende der Operation (**RegData**, **Done**, **CommandAborted**, **Error**, **ErrorID**, **Done**) werden für einen Zyklus gegeben.

ⓘ HINWEIS! Die Drivetypes iTcMc_EncoderKL3002, iTcMc_EncoderKL3042, iTcMc_EncoderKL3062, iTcMc_EncoderKL3162, iTcMc_EncoderKL5101, iTcMc_EncoderKL5111, iTcMc_EncoderKL2521, iTcMc_EncoderKL2531 und iTcMc_EncoderKL2541 unterstützen die Parameterkommunikation.

4 Knowledge Base

Knowledge Base der SPS Bibliothek TcPlcHydraulics (ab V3.0)

Hier finden Sie eine Reihe von Antworten auf immer wiederkehrende Fragen.

Themengebiete


Name	Beschreibung
Global Constants [▶ 222]	Vordefinierte Fehlercodes, Masken zur Bitabfrage, ADS-Codes usw.
Setup [▶ 241]	Inbetriebnahme-Hinweise
SampleList [▶ 234]	Programm-Beispiele
Ideensammlung [▶ 208]	Tipps und Tricks
HMI-Tool [▶ 238]	Der PlcMcManager

Probleme beim Update der Bibliothek

Bei einem Update der Bibliothek kann es zu Problemen beim Übersetzen kommen. Die Ursache ist dann möglicherweise eine Änderung des Namens eines oder mehrerer Bausteine oder Datentypen. Diese Änderungen sind nicht immer vermeidbar und werden in der Regel aus einem der folgenden Gründe durchgeführt:

- Anpassung an die Regeln der PLC Open Motion Control Definitionen.
- Verfolgen der Weiterentwicklung der PLC Open Motion Control Definitionen.
- Weiterentwicklung der bereitgestellten Technologie.
- Anpassung an die verwendete Technologie, insbesondere Unterstützung weiterer E/A-Geräte.
- Vermeidung von Namenskollisionen und anderen Kompatibilitätsproblemen mit anderen Bibliotheken.


Ab V3.0 Build 22 verwendet die Bibliothek die TcEtherCAT.LIB zur Kommunikation via EtherCAT-Feldbus. In älteren TwinCAT-Umgebungen ist diese Bibliothek noch nicht verfügbar. Soll die Bibliothek TcPlcHydraulics in einer solchen Umgebung verwendet werden ist die mitgelieferte TcEtherCatDummy.LIB in das Projektverzeichnis kopiert und in TcEtherCAT.LIB umbenannt werden. Diese Bibliothek ist dann **VOR** der TcPlcHydraulics.LIB im Projekt einzufügen.

 Hinweis	Diese Vorgehensweise darf nicht in TwinCAT-Umgebungen angewendet werden, die EtherCAT unterstützen. Die mitgelieferte Datei darf NICHT dazu verwendet werden, eine vorhandene funktionstüchtige TcEtherCAT.LIB zu ersetzen.
-------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

HINWEIS! Es sind keine Funktionen verfügbar, die EtherCAT Technologien voraussetzen.

HINWEIS! Die in einem Projekt verwendete Version der Bibliothek sollte in das Projektverzeichnis kopiert und mit dem Projekt zusammen gesichert werden. Dadurch wird ein unbeabsichtigter Versionswechsel vermieden, der sonst durch ein in der Zwischenzeit durchgeführtes TwinCAT Update verursacht werden könnte. Ist ein Update der Bibliothek gewollt ist dann die neue Version gezielt in das Projektverzeichnis zu kopieren.

HINWEIS! Es wird dringend empfohlen, nach einem Update der Bibliothek eine probeweise Übersetzung des gesamten Projekts durchzuführen. Weiterhin sollte mit dem System Manager das Mapping aktualisiert werden. Sollte die unten stehende Tabelle eine Größenänderung einer der Strukturen anzeigen ist unbedingt die Adressvergabe zu überprüfen.

 Hinweis	Wenn ein Update der Bibliothek auf einen Stand erfolgt, der sich nicht nur in der dritten (Build), sondern in den ersten zwei (Major, Minor) Versionsangaben unterscheidet ist davon auszugehen, dass die vom System Manager erzeugten Mappings nicht mehr korrekt sind. Es ist zwingend erforderlich, die Verknüpfungen neu wirksam zu machen.
-------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Alter Name	Neuer Name	Grund der Änderung
ST_TcMcAxInterface	Axis_Ref_BkPlcMc	Anpassung an PLC Open Motion Control Definitionen.
ST_TcPlcMcCamId	MC_CAM_ID_BkPlcMc	Anpassung an PLC Open Motion Control Definitionen.
ST_TcPlcMcCamRef	MC_CAM_REF_BkPlcMc	Anpassung an PLC Open Motion Control Definitionen.
E_TcMCDirection	MC_Direction_BkPlcMc	Anpassung an PLC Open Motion Control Definitionen.
E_TcMCStartMode	MC_StartMode_BkPlcMc	Anpassung an PLC Open Motion Control Definitionen.
ST_TcPlcMcEncoderIn	---	entfällt, Aufgabe wird von ST_TcPlcDeviceInput übernommen
ST_TcPlcMcEncoderOut	---	entfällt, Aufgabe wird von ST_TcPlcDeviceOutput übernommen
ST_TcPlcMcDriveIn	---	entfällt, Aufgabe wird von ST_TcPlcDeviceInput übernommen
ST_TcPlcMcDriveOut	---	entfällt, Aufgabe wird von ST_TcPlcDeviceOutput übernommen
ST_TcPlcMcAx2000In	---	entfällt, Aufgabe wird von ST_TcPlcDeviceInput übernommen
ST_TcPlcMcAx2000Out	---	entfällt, Aufgabe wird von ST_TcPlcDeviceOutput übernommen
MC_AxUtiCancelMonitoring_BkPlcMc	---	entfällt, durch PLC Open Definitionen überflüssig geworden.

Größe der E/A-Strukturen in Bytes

Name	V 2.1.X	ab V3.0.0	ab V3.1.0 (geplant)
ST_TcPlcMcEncoderIn	16	-	-
ST_TcPlcMcEncoderOut	1	-	-
ST_TcPlcMcDriveIn	23	-	-
ST_TcPlcMcDriveOut	40	-	-
ST_TcPlcMcAx2000In	37	-	-
ST_TcPlcMcAx2000Out	26	-	-
ST_TcPlcDeviceInput [► 98]	-	143	?
ST_TcPlcDeviceOutput [► 100]	-	103	?

4.1 FAQs (ab V3.0)

Hier finden Sie eine Reihe von Antworten auf immer wieder auftauchende Fragen.

Name	Beschreibung
FAQ #1 > 209	Wie integriere ich eine oder mehrere Achsen in eine PLC-Applikation?
FAQ #2 > 210	Welche Daten müssen in der PLC-Applikation für die Achsen angelegt werden?
FAQ #3 > 210	Wie initialisiere ich die Daten und lade ich die Parameter einer Achsen beim Start der PLC?
FAQ #4 > 211	Wie wird die Istposition der Achsen ermittelt?
FAQ #5 > 214	Wie wird der Stellwert für eine Achse erzeugt?
FAQ #6 > 214	Wie wird der Stellwert einer Achse für die Ausgabe vorbereitet?
FAQ #7 > 214	Wie wird der Stellwert an die Achse ausgegeben?
FAQ #8 > 216	In welcher Reihenfolge sind die Bausteine einer Achse aufzurufen?
FAQ #9 > 216	Wie steuere ich eine Ventilendstufe (Onboard oder extern) an?
FAQ #10 > 216	Wie lege ich einen Meldungspuffer an?
FAQ #11 > 217	Wie breche ich die Überwachung einer Funktion ab?
FAQ #12 > 218	Wie überwache ich die Kommunikation mit einem E/A-Gerät?
FAQ #13 > 218	Wie versee ich die kundenspezifischen Achsparameter mit selbstgewählten Bezeichnungen?
FAQ #14 > 218	Wie steuere ich ein Stromventil an?
FAQ #15 > 218	Welche Variablen einer Achse sollten mit dem Scope aufgezeichnet werden?
FAQ #16 > 219	Welche Bedeutung hat die Variable nDebugTag in Axis_Ref_BkPlcMc?
FAQ #17 > 219	Welche Besonderheiten sind beim Einsatz von Sercos-Antrieben zu berücksichtigen?
FAQ #18 > 220	Wie wird ein Druck oder eine Kraft ermittelt?
FAQ #19 > 220	Welche Besonderheiten sind beim Einsatz von AX5000 Antrieben zu berücksichtigen?
FAQ #20 > 221	Wie wird eine Achse für die Nutzung des Blendings nach PLC Open vorbereitet?
FAQ #21 > 221	Wie kann auf Register einer Klemme zugegriffen werden an der ein Encoder oder ein Ventil einer Achse angeschlossen ist?
FAQ #22 > 221	Wie ist die ASCII Datei für eine Linearisierungstabelle aufgebaut?
FAQ #23 > 222	Wie können Kommandos des PlcMcManagers abgeblockt werden?
Setup > 241	Wie wird die Achse in Betrieb genommen und optimiert?

FAQ #1 Wie integriere ich eine oder mehrere Achsen in eine PLC-Applikation?

Die Vorgehensweise ist grundsätzlich anders als bei einer von der NC Task geführten Achse, weil hier alle Aufgaben der NC Task von der PLC mit übernommen werden. Für die meisten Themen stehen jedoch vorgefertigte Bausteine zur Verfügung, so dass sich der zusätzliche Programmieraufwand in Grenzen hält. Im Einzelnen sind die folgenden Punkte zu beachten:

- Daten der Achse in der PLC-Applikation ([FAQ #2 |> 210](#))
- Initialisierung und Laden der Achs-Parameter beim Start der PLC-Applikation ([FAQ #3 |> 210](#))
- Istwerterfassung ([FAQ #4 |> 211](#))
- Stellwert-Generierung ([FAQ #5 |> 214](#))
- Aufbereitung des Stellwerts für die Ausgabe ([FAQ #6 |> 214](#))
- Inbetriebnahme der Achse ([Setup |> 241](#))
- Inbetriebnahme einer Istdruckermittlung mit Bausteinen vom Typ [MC_AxRtReadPressureSingle_BkPlcMc |> 145](#) oder [MC_AxRtReadPressureDiff_BkPlcMc |> 143](#).
- Organisation des Bewegungsablaufs ([FAQ #7 |> 214](#))

ⓘ HINWEIS! Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollte zur Vereinfachung ein Baustein des Typs [MC_AxStandardBody_BkPlcMc](#) verwendet werden.

FAQ #2 Welche Daten müssen in der PLC-Applikation für die Achse angelegt werden?

Pro Achse muss je eine Variable der Typen [Axis_Ref_BkPlcMc](#) [▶ 62], [ST_TcPlcDeviceInput](#) [▶ 98] und [ST_TcPlcDeviceOutput](#) [▶ 100] angelegt werden. Für mehrere Achsen ist die Verwendung von Variablenfeldern unbedingt zu empfehlen. Je ein Beispiel für eine und fünf Achsen finden Sie in den Programm-Beispielen <http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599853451.zip> und <http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599855627.zip>.

Die in diesen Beispielen gezeigte Vorgehensweise unter Verwendung von [MC_AxUtiStandardInit_BkPlcMc](#) [▶ 171] Bausteinen sorgt für eine korrekte Initialisierung beim Start der PLC und veranlasst das Laden der Parameter der Achsen aus Dateien.

ⓘ HINWEIS! Für die Realisierung einer Meldungsaufzeichnung sind weitere Daten erforderlich. Siehe hierzu [FAQ #10](#) [▶ 216].

ⓘ HINWEIS! Um im PlcMcManager kundenspezifische Achsparameter mit selbstgewählten Bezeichnungen zu versehen sind weitere Daten erforderlich. Siehe hierzu [FAQ #13](#) [▶ 218]

ⓘ HINWEIS! Um das Blending nach PLC Open nutzen zu können sind weitere Daten erforderlich. Siehe hierzu [FAQ #20](#). [▶ 221]

FAQ #3 Wie initialisiere ich die Daten einer Achse?

Beim Start der PLC Applikation sind eine Reihe von Initialisierungen vorzunehmen. Dies geschieht sinnvollerweise in drei Stufen, die von einem [MC_AxUtiStandardInit_BkPlcMc](#) [▶ 171] Baustein bereitgestellt werden und nur in Sonderfällen durch die Applikation direkt realisiert werden sollten. Hier werden sie also nur aus Gründen der Vollständigkeit beschrieben.

1. Es sind eine Reihe von Pointern korrekt einzustellen, um die Bestandteile der Achsen zu verbinden. Diese Aufgabe sollte mit einem Baustein vom Typ [MC_AxUtiStandardInit_BkPlcMc](#) [▶ 171] gelöst werden, der eine Verschiebung oder Größenänderung im Speicher oder die Änderung eines Typcodes bei einem späteren Online Change erkennt und dann für eine Reinitialisierung der Pointer und ein Nachladen von Parametern sorgt.
2. Die Parameter der Achsen müssen eingestellt werden. Eine hart codierte Zuweisung der Werte durch die Applikation ist zwar technisch machbar, aber in der Regel nicht sinnvoll. Besser ist die Speicherung der Einstellungen in Dateien, die beim Systemstart unter Kontrolle der Applikation durch den [MC_AxUtiStandardInit_BkPlcMc](#) [▶ 171] Baustein geladen werden. Hinweise zur Inbetriebnahme finden Sie unter Setup.
3. Die Zykluszeit der Task ist in die Parameter der Achse zu übernehmen. Dies sollte am Ende der Parameterladeprozess erfolgen um diesen für die Funktion vieler Bausteine wichtigen Wert korrekt einzustellen. Ein [MC_AxUtiStandardInit_BkPlcMc](#) [▶ 171] Baustein übernimmt diese Aufgabe automatisch.

ⓘ HINWEIS! Wenn in der Applikation ein Baustein vom Typ [MC_AxAdsCommServer_BkPlcMc](#) genutzt wird, muss dieser Baustein in derselben Task aufgerufen werden, die auch die Zuweisung der Pointer vornimmt. Ist dies nicht möglich oder nachteilig muss der Aufruf des Bausteins für die Dauer der Zuweisungen gesperrt werden. Andernfalls kann es zum Absturz der PLC-Applikation durch Auslösung von schweren Laufzeitfehlern (Page Fault Exception) kommen.

ⓘ HINWEIS! Sämtliche hier aufgeführten Aktivitäten sollten durch einen [MC_AxUtiStandardInit_BkPlcMc](#) Baustein verwirklicht und koordiniert werden. Wenn die Variable `nInitState` im [Axis_Ref_BkPlcMc](#) der Achse den Wert 2 oder -2 angenommen hat ist die Initialisierung erfolgreich bzw. mit Fehler beendet. Bei einer erfolgreichen Initialisierung sind [MC_AxUtiStandardInit_BkPlcMc](#).Ready und `bParamsEnable` in [Axis_Ref_BkPlcMc](#) auf TRUE, andernfalls bleibt diese Variable FALSE.

ⓘ HINWEIS! Die zur Verfügung gestellten Beispielprogramme legen den Namen der Achse und den Namen (inklusive Pfad) der zugehörigen Parameterdatei fest. Diese Festlegungen sind unbedingt an die jeweilige Applikation anzupassen.

FAQ #4: Wie wird die Istposition der Achsen ermittelt?

Als Positionssensor kommt eine Reihe von Signalgebern in Frage, die nach diversen physikalischen Prinzipien eine positionabhängige elektrische Größe erzeugen. Diese Größe bestimmt den Typ der einzusetzenden E/A-Komponente. Die pro Achse anzulegenden Variablen der Typen [ST_TcPlcDeviceInput \[► 98\]](#) und [ST_TcPlcDeviceOutput \[► 100\]](#) enthalten Elemente, die mit den Istwert-, Zähl-, Latch-, Control- und Status-Variablen der E/A-Hardware zu verknüpfen sind.

Hier einige Beispiele:

E/A-Komponente	Signal	Encoder Type
AX2000 B110 mit Absolut-Encoder	EtherCAT	iTcMc_EncoderAx2000_B110A [▶ 124]
AX2000 B110 mit Resolver	EtherCAT	iTcMc_EncoderAx2000_B110R [▶ 115]
AX2000 B200 mit Resolver	EtherCAT	iTcMc_EncoderAx2000_B200R [▶ 116]
AX2000 B750 mit Absolut-Encoder	EtherCAT	iTcMc_EncoderAx2000_B750A [▶ 127]
AX2000 B900 mit Resolver	EtherCAT	iTcMc_EncoderAx2000_B900R [▶ 116]
AX5000 B110 mit Multiturn-Absolut-Encoder	EtherCAT	iTcMc_EncoderAX5000_B110A [▶ 128]
EtherCAT-Servoregler mit CoE DS402 Support und Multiturn-Encoder	EtherCAT	iTcMc_EncoderCoE_DS402A [▶ 129]
EtherCAT-Servoregler mit CoE DS402 Support und Resolver oder Singleturn-Encoder	EtherCAT	iTcMc_EncoderCoE_DS402SR [▶ 129]
EL3102	-10V .. 10V	iTcMc_EncoderEL3102 [▶ 132]
EL3142	0mA .. 20mA	iTcMc_EncoderEL3142 [▶ 132]
EL3162	0 .. 10V	iTcMc_EncoderEL3162 [▶ 132]
EL3255	Potentiometrischer Weggeber	iTcMc_EncoderEL3162 [▶ 132]
EL5001	SSI	iTcMc_EncoderEL5001 [▶ 133]
EL5101	A/B-Inkremente, RS422="TTL"	iTcMc_EncoderEL5101 [▶ 133]
EL7041	A/B-Inkremente, RS422="TTL"	iTcMc_EncoderEL7041 [▶ 134]
EtherCAT-Encoder mit CoE_DS406 Profil	EtherCAT	iTcMc_EncoderCoE_DS406 [▶ 130]
IE5009	SSI	iTcMc_EncoderIx5009 [▶ 134]
IP5009	SSI	iTcMc_EncoderIx5009 [▶ 134]
KL10xx	2 Bit, A/B-Inkremente	iTcMc_EncoderDigIncrement [▶ 132]
KL11xx	2 Bit, A/B-Inkremente	iTcMc_EncoderDigIncrement [▶ 132]
KL12xx	2 Bit, A/B-Inkremente	iTcMc_EncoderDigIncrement [▶ 132]
KL13xx	2 Bit, A/B-Inkremente	iTcMc_EncoderDigIncrement [▶ 132]
KL14xx	2 Bit, A/B-Inkremente	iTcMc_EncoderDigIncrement [▶ 132]
KL17xx	2 Bit, A/B-Inkremente	iTcMc_EncoderDigIncrement [▶ 132]
KL10xx	4 Bit, Positionsnocken	iTcMc_EncoderDigCam [▶ 132]
KL11xx	4 Bit, Positionsnocken	iTcMc_EncoderDigCam [▶ 132]
KL12xx	4 Bit, Positionsnocken	iTcMc_EncoderDigCam [▶ 132]
KL13xx	4 Bit, Positionsnocken	iTcMc_EncoderDigCam [▶ 132]
KL14xx	4 Bit, Positionsnocken	iTcMc_EncoderDigCam [▶ 132]
KL17xx	4 Bit, Positionsnocken	iTcMc_EncoderDigCam [▶ 132]

E/A-Komponente	Signal	Encoder Type
KL2521	Pulse Train	iTcMc_EncoderKL2521 > 134
KL2531	Schrittmotor, direkt (Encoder durch Pulszähler emuliert)	iTcMc_EncoderKL2531 > 135
KL2541	Schrittmotor, direkt (mit Encoder oder Encoder durch Pulszähler emuliert)	iTcMc_EncoderKL2541 > 135
KL2542	DC Motor, direkt mit Encoder	iTcMc_EncoderKL2542 > 136
KL3001	-10V .. 10V	iTcMc_EncoderKL3002 > 136
KL3002	-10V .. 10V	iTcMc_EncoderKL3002 > 136
KL3011	0mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3012	0mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3021	4mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3022	4mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3041	0mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3042	0mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3044	0mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3051	4mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3052	4mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3054	4mA .. 20mA	iTcMc_EncoderKL3042 > 136
KL3061	0V .. 10V	iTcMc_EncoderKL3062 > 137
KL3062	0V .. 10V	iTcMc_EncoderKL3062 > 137
KL3064	0V .. 10V	iTcMc_EncoderKL3062 > 137
KL3162	0V .. 10V	iTcMc_EncoderKL3162 > 137
KL5001	SSI	iTcMc_EncoderKL5001 > 137
KL5101	A/B-Inkremente, RS422="TTL"	iTcMc_EncoderKL5101 > 137
KL5111	A/B-Inkremente, RS422="HTL"	iTcMc_EncoderKL5111 > 138
M2510	-10V .. 10V	iTcMc_EncoderM2510 > 138
M3100	A/B-Inkremente, RS422="TTL"	iTcMc_EncoderM3120 > 138
M3120	A/B-Inkremente, RS422="TTL"	iTcMc_EncoderM3120 > 138

Wenn eine der hier genannten Komponenten eingesetzt wird kann in der Regel einer der bereitgestellten Encoder-Bausteine verwendet werden. Die Schnittstellen dieser Bausteine sind nicht garantiert und sollten somit nicht direkt von der Applikation aufgerufen werden. Besser ist es, den Encoder-Typ entsprechend der Konstanten in [E_TcMcEncoderType |> 70](#)] unter nEnc_Type in [ST_TcHydAxParam |> 88](#)] einzustellen und einen Baustein vom Typ [MC_AxRtEncoder_BkPlcMc |> 123](#)] zu verwenden. Dieser ruft dann den typrichtigen Unterbaustein automatisch auf.

Alle Encoder-Bausteine verwenden die Parameter fEnc_IncWeighting und fEnc_IncInterpolation als Inkrementbewertung. Bei absoluten Wegsensoren wird zusätzlich fEnc_ZeroShift als Nullpunktverschiebung verwendet. Inkrementelle Sensoren erfordern in der Regel eine Referenzfahrt mit einem [MC_Home_BkPlcMc |> 51](#)] Baustein, in deren Verlauf fEnc_RefShift in [ST_TcHydAxRtData |> 94](#)] ermittelt wird. Dieser Wert übernimmt dann die Funktion der Nullpunktverschiebung. Selbstverständlich kann die Nullpunktverschiebung in Sonderfällen auch mit einem [MC_SetPosition_BkPlcMc |> 32](#)] Baustein festgelegt werden. Der Referenziert-Status der Achse ist mit [MC_SetReferenceFlag_BkPlcMc |> 33](#)]() zu definieren.

Ist es aus technischen Gründen nicht möglich, die Istposition mit Bausteinen der Library zu ermitteln kann diese Aufgabe auch durch Applikationsbausteine ausgeführt und das Ergebnis in fActPos und bei Bedarf auch fActVelo in [ST_TcHydAxRtData |> 94](#)] eingetragen werden. Um die Durchgängigkeit zu erhalten sollte auch hier wenn möglich auf die Parameter fEnc_IncWeighting, fEnc_IncInterpolation und fEnc_ZeroShift bzw. fEnc_RefShift zurückgegriffen werden.

HINWEIS! Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollte zur Vereinfachung ein Baustein des Typs `MC_AxStandardBody_BkPlcMc` verwendet werden.

HINWEIS! Die Inbetriebnahme einer Istdruckermittlung mit Bausteinen vom Typ `MC_AxRtReadPressureSingle_BkPlcMc` oder `MC_AxRtReadPressureDiff_BkPlcMc` ist in der Dokumentation des Bausteins beschrieben.

FAQ #5: Wie wird der Stellwert für eine Achse erzeugt?

Durch die PLC-Applikation muss in jedem Zyklus für jede Achse ein Baustein vom Typ `MC_AxRuntime_BkPlcMc` [▶ 155] oder ersatzweise ein geeigneter Reglerbaustein (z.B. ein Druckregler) aufgerufen werden. Der Parameter `nProfileType` in `ST_TcHydAxParam` [▶ 88] legt fest, nach welchem Verfahren der Stellwert generiert wird. Dabei werden je nach Typ und in Abhängigkeit von weiteren Parametern der Achse und den Daten der Bewegung angepasste Geschwindigkeitsstellwerte errechnet. Diese Stellwerte sind jedoch auf den abstrakten Zahlenbereich ± 1.0 normiert und noch nicht für die direkte Ausgabe auf einer E/A-Hardware vorbereitet.

HINWEIS! Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollen, verwenden Sie zur Vereinfachung ein Baustein des Typs `MC_AxStandardBody_BkPlcMc`. [▶ 170]

FAQ #6: Wie wird der Stellwert einer Achse für die Ausgabe vorbereitet?

Nach dem Aufruf des `MC_AxRuntime_BkPlcMc` [▶ 155] Bausteins ist für jede Achse ein Baustein vom Typ `MC_AxRtFinish_BkPlcMc` [▶ 164] aufzurufen. Dieser Baustein fasst mehrere Geschwindigkeitskomponenten (Stellwert, Reglerausgabe, Offsetkompensation, Überdeckungskompensation) zusammen und berücksichtigt dabei eventuelle Knickpunkte der Vorsteuercharakteristik.

Für die Ausgabe auf einer E/A-Baugruppe ist in der Regel eine numerische Anpassung erforderlich. Zu diesem Zweck ist für jede Achse ein `MC_AxRtDrive_BkPlcMc` [▶ 114] Baustein aufzurufen. Der Wert in `nDrive_Type` in `ST_TcHydAxParam` [▶ 88] wählt den zu benutzenden hardwarespezifischen Unterbaustein aus.

Die pro Achse anzulegenden Variablen der Typen `ST_TcPlcDeviceInput` [▶ 98] und `ST_TcPlcDeviceOutput` [▶ 100] enthalten Elemente, die mit den Sollwert- und Steuer-Variablen der E/A-Hardware zu verknüpfen sind.

HINWEIS! Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollen, verwenden Sie zur Vereinfachung ein Baustein des Typs `MC_AxStandardBody_BkPlcMc`. [▶ 170]

FAQ #7: Wie wird der Stellwert an die Achse ausgegeben?

Als Steller kommen eine Reihe von Geräten und Einrichtungen in Frage, die abhängig von einer elektrischen Größe nach diversen physikalischen Prinzipien eine variable Geschwindigkeit dosieren. Diese Größe bestimmt den Typ der einzusetzenden E/A-Komponente. Die pro Achse anzulegenden Variablen der Typen `ST_TcPlcDeviceInput` [▶ 98] und `ST_TcPlcDeviceOutput` [▶ 100] enthalten Elemente, die mit den Variablen der E/A-Hardware zu verknüpfen sind.

Hier einige Beispiele:

E/A-Komponente	Signal	Drive Type
AX2000 B110 mit Absolut-Encoder	EtherCAT	iTcMc_DriveAX2000_B110A [► 115]
AX2000 B110 mit Resolver	EtherCAT	iTcMc_DriveAX2000_B110R [► 115]
AX2000 B200 mit Resolver	EtherCAT	iTcMc_DriveAX2000_B200R [► 116]
AX2000 B750 mit Absolut-Encoder	EtherCAT	iTcMc_DriveAx2000_B750A [► 116]
AX2000 B900 mit Resolver	EtherCAT	iTcMc_DriveAX2000_B900R [► 116]
AX5000 B110 mit Absolut-Encoder	EtherCAT	iTcMc_DriveAX5000_B110A [► 116]
EtherCAT-Servoregler mit CoE DS402 Support und Resolver, Singleturn- oder Multiturn-Encoder	EtherCAT	iTcMc_DriveCoE_DS402 [► 117]
EtherCAT-Ventil mit CoE_DS408 Profil	EtherCAT	iTcMc_Drive_CoE_DS408 [► 117]
EL2535	PWM	iTcMc_DriveEL2535
EL4031, EL4032, EL4034, EL4038 EL4131, EL4132, EL4134	-10V .. 10V	iTcMc_DriveEL4132 [► 118]
EL4011, EL4012, EL4014, EL4018, EL4112	0..20mA	iTcMc_DriveEL4x22
EL4021, EL4022, EL4024, EL4028, EL4122, EL4124	4..20mA	
EL7031	Schrittmotor, direkt	iTcMc_DriveEL7031 [► 118]
EL7041	Schrittmotor, direkt	iTcMc_DriveEL7041 [► 119]
IE2512	PWM	iTcMc_DriveIx2512_1Coil [► 118] iTcMc_DriveIx2512_2Coil [► 118]
IP2512	PWM	iTcMc_DriveIx2512_1Coil [► 118] iTcMc_DriveIx2512_2Coil [► 118]
KL20xx, KL21xx, KL22xx, KL24xx	5 Bit zum Betrieb eines Frequenzumrichters mit Festfrequenzen	iTcMc_DriveLowCostInverter [► 122]
KL20xx, KL21xx, KL22xx, KL24xx	4 Bit zum Betrieb eines spannungsgesteuerten Schrittmotors	iTcMc_DriveLowCostStepper [► 122]
KL2521	Pulse Train	iTcMc_DriveKL2521 [► 119]
KL2531	Schrittmotor, direkt	iTcMc_DriveKL2531 [► 119]
KL2532	DC Motor, direkt mit Encoder	iTcMc_DriveKL2532 [► 120]
KL2535	PWM	iTcMc_DriveKL2535_1Coil [► 120] iTcMc_DriveKL2535_2Coil [► 120]
KL2541	Schrittmotor, direkt	iTcMc_DriveKL2541 [► 120]
KL2542	DC Motor, direkt mit Encoder	iTcMc_DriveKL2542 [► 121]
KL4031	-10V .. 10V	iTcMc_DriveKL4032 [► 121]
KL4032	-10V .. 10V	iTcMc_DriveKL4032 [► 121]
KL4034	-10V .. 10V	iTcMc_DriveKL4032 [► 121]
M2400	-10V .. 10V	iTcMc_DriveM2400_D1 [► 123] , iTcMc_DriveM2400_D2 , iTcMc_DriveM2400_D3 , iTcMc_DriveM2400_D4

Wenn eine der hier genannten Komponenten eingesetzt wird kann in der Regel einer der bereitgestellten Drive-Bausteine verwendet werden. Diese Schnittstellen dieser Bausteine sind nicht garantiert und sollten somit nicht direkt von der Applikation aufgerufen werden. Besser ist es, den Drive-Type entsprechend der Konstanten in [E_TcMcDriveType](#) [▶ 67] unter `nDrive_Type` in [ST_TcHydAxParam](#) [▶ 88] einzustellen und einen Baustein vom Typ [MC_AxRtDrive_BkPlcMc](#) [▶ 114] zu verwenden.

ⓘ HINWEIS! Wenn nur die üblichen Bausteine (Encoder, Generator, Finish, Drive) für die Achse aufgerufen werden sollen, verwenden Sie zur Vereinfachung ein Baustein des Typs

[MC_AxStandardBody_BkPlcMc](#). [▶ 170]

FAQ #8: In welcher Reihenfolge sind die Bausteine einer Achse aufzurufen?

1. Obligatorisch: Alle Bausteine, die den Istzustand der Achse feststellen. Dazu gehören Bausteine der Typen [MC_AxRtEncoder_BkPlcMc](#) [▶ 123], [MC_AxRtReadPressureDiff_BkPlcMc](#) [▶ 143] oder [MC_AxRtReadPressureSingle_BkPlcMc](#) [▶ 145].
2. Üblich: Bausteine oder Befehle, die für eine Aktualisierung der Freigabe-Signale der Achse sorgen. Dies ist in der Regel ein Baustein vom Typ [MC_Power_BkPlcMc](#) [▶ 19]. Bei Achsen mit einem Inkremental-Encoder, der unter Verwendung eines Nocken referenziert wird kommt ein Funktionsaufruf [MC_AxRtSetReferencingCamSignal_BkPlcMc](#) dazu.
3. Optional: Bausteine, die aus einem Istzustand der Achse, einem E/A-Signal oder einem Signal der Applikation eine Entscheidung ableiten und einen Befehl auslösen. Zum Beispiel kann als Reaktion auf das Signal eines Näherungsschalters ein Achsstart ausgelöst werden oder in Abhängigkeit vom Druckanstieg eine Achsbewegung vor dem Erreichen der Zielposition gestoppt werden.
4. Obligatorisch: Stellwertgeneratoren wie Bausteine des Typs [MC_AxRuntime_BkPlcMc](#) [▶ 155].
5. Optional: Bei Bedarf können an dieser Stelle verschiedene Regler aufgerufen werden. Dabei kann es sich um einen Baustein der Typen [MC_AxCtrlSlowDownOnPressure_BkPlcMc](#) [▶ 110] oder ähnliches handeln.
6. Obligatorisch: Ein Anpassungsbaustein des Typs [MC_AxRtFinish_BkPlcMc](#) [▶ 164].
7. Optional: Bei Bedarf kann an dieser Stelle ein Baustein für die automatische Inbetriebnahme aufgerufen werden.
8. Obligatorisch: Ein Ausgabebaustein des Typs [MC_AxRtDrive_BkPlcMc](#) [▶ 114].

Anstelle der Bausteine der Library können auch Applikations-Bausteine treten. Dabei ist die Notwendigkeit jedoch sorgfältig zu überprüfen und eine Kompatibilität zur Library sicher zu stellen. In einigen Anwendungsfällen kann dies notwendig werden, um einen nicht standardisierten Sensor oder Aktor anzupassen oder um ein besonderes Regelungsproblem zu lösen.

FAQ #9: Wie steuere ich eine Ventilstufe (Onboard oder extern) an?

In der [ST_TcPlcDeviceOutput](#) [▶ 100] Struktur sind die Signale **bPowerOn** und **bEnable** für die Steuerung der Endstufenversorgung und -freigabe vorgesehen. Beide Signale werden durch Bausteine vom Typ [MC_Power_BkPlcMc](#) [▶ 19] gesetzt, wenn der Eingang **Enable** gesetzt wird. Gleichzeitig setzt dieser Baustein die Software-Reglerfreigabe in [ST_TcHydAxRtData](#) [▶ 94].`nDeCtrlDWord` [▶ 223].

In der [ST_TcPlcDeviceInput](#) [▶ 98] Struktur sind die Signale **bPowerOk**, **bEnAck** und **bReady** für die Kontrolle der Endstufenversorgung, die Rückmeldung der Endstufenfreigabe und des Statussignals vorgesehen. Die von verschiedenen Herstellern bereitgestellten Signale unterscheiden sich jedoch sehr stark. Derzeit wird deshalb nur das **bPowerOk** Signal bei der Festlegung des **Status** Ausgangs des [MC_Power_BkPlcMc](#) [▶ 19] Bausteins verwendet. Steht auch hierfür kein geeignetes Signal zur Verfügung oder soll keine Überwachung realisiert werden ist [ST_TcHydAxParam](#) [▶ 88].`bDrive_DefaultPowerOk` zu setzen.

FAQ #10: Wie lege ich einen Meldungspuffer an?

Eine direkte Ausgabe von Meldungen aus den Bausteinen würde zu nicht kalkulierbaren Laufzeitschwankungen führen. Aus diesem Grund werden die Meldungen in einem Puffer abgelegt und falls dies gewünscht wird nacheinander in die Ereignisanzeige von Windows ausgegeben.

Um einen Meldungspuffer nutzen zu können ist eine Variable des Typs `ST_TcPlcMcLogBuffer` [▶ 102] anzulegen. In diesen Puffer werden die Meldungen aller Achsen aufgenommen. Es ist wichtig, unabhängig von der Anzahl der Achsen nur eine solche Variable im Projekt anzulegen. Die Adresse dieses Puffers ist den Bausteinen `MC_AxUtiStandardInit_BkPlcMc` [▶ 171] aller Achsen zusammen mit den Adressen der anderen individuellen Achsbestandteile zu übergeben. Diese Bausteine werden in der Regel im Initialisierungsteil des Projekts aufgerufen. In der Struktur `Axis_Ref_BkPlcMc` [▶ 62] wird diese Adresse im Element `pStAxLogBuffer` abgelegt und durch den Baustein überwacht.

In `nLogLevel` in `Axis_Ref_BkPlcMc` [▶ 62] wird festgelegt, bis zu welcher Bedeutsamkeit anfallende Meldungen in den Puffer aufgenommen werden. Die dabei zu verwendenden Werte [▶ 232] sind in den Globalen Variablen der Bibliothek definiert. Es ist zu beachten, dass diese Einstellung pro Achse zu erfolgen hat.

Die Bausteine der Bibliothek erkennen die oben genannten Vorbereitungen und werden die Ausgabe von Meldungen aufnehmen. Allerdings würde der Puffer bei aktivierter Meldungsabgabe nach mehr oder weniger kurzer Zeit vollständig gefüllt sein und keine weiteren Meldungen aufnehmen. Um dies zu vermeiden sind zwei Wege nutzbar.

FAQ #10.1: Weitergabe der Meldungen an die Ereignisanzeige von Windows

Um die angefallenen Meldungen aus dem LogBuffer der Library in die Ereignisanzeige von Windows zu transportieren ist zyklisch ein Baustein des Typs `MC_AxRtLoggerSpool_BkPlcMc` [▶ 176] aufzurufen. Bei jedem Aufruf wird so eine Meldung aus dem LogBuffer entfernt.

ⓘ HINWEIS! Auch auf Rechnern unter Windows CE kann eine Ereignisanzeige für die von TwinCAT erzeugten Meldungen ergänzt werden. Dazu wird dieser Dienst vom TwinCAT System Service nachgebildet. Allerdings wird in der Regel hier nur eine FlashDisk zur Verfügung stehen. Um den hier relativ kleinen Meldungsvorrat der Ereignisanzeige nicht zu überlasten sollten nur Fehler aufgezeichnet werden.

FAQ #10.2: Löschen der jeweils ältesten Meldungen

Um einen Minimalwert an aufnehmbaren Meldungen sicher zu stellen kann zyklisch ein Baustein des Typs `MC_AxRtLoggerDespool_BkPlcMc` [▶ 175] aufgerufen werden. Dieser Baustein entfernt pro Aufruf die jeweils älteste Meldung im LogBuffer, bis eine übergebene Anzahl von freien Meldungen zur Verfügung steht. Die entfernten Meldungen gehen dabei verloren.

FAQ #10.3: Erzeugen von Loggereinträgen durch die Applikation

Eine Meldung kann durch die Applikation entweder achsbezogen oder nicht achsbezogen ausgegeben werden. Hierzu stehen die Bausteine `MC_AxRtLogAxisEntry_BkPlcMc` [▶ 173] und `MC_AxRtLogEntry_BkPlcMc` [▶ 174] zur Verfügung.

FAQ #11: Wie breche ich die Überwachung einer Funktion ab?

Einige Bausteine der Bibliothek starten eine Aktivität, für deren Abarbeitung ihr zyklischer Aufruf jedoch nicht mehr zwingend erforderlich ist. Allerdings sind auch diese Bausteine entsprechend den Regeln der PLCopen Motion Control Richtlinien so ausgeführt, dass sie die Aktivität vollständig überwachen und an ihren Ausgängen darstellen. Erkennbar ist dies durch den Ausgang `Busy`, den die meisten Bausteine bereitstellen.

Den zyklischen Aufruf eines in diesem Überwachungszustand befindlichen Bausteins zu unterlassen wird zumeist erhebliche Probleme erzeugen. Der nächste Funktionsstart mit dem betreffenden Baustein wird Probleme bei der Auswertung der Flanken an seinen Eingängen haben oder er wird feststellen, dass die Achse inzwischen eine andere Funktion ausgeführt hat und ein nicht vorhandenes Problem (`CommandAborted`) signalisieren.

In älteren Versionen der Library wurde ein Baustein des Typs `MC_AxUtiCancelMonitoring_BkPlcMc()` bereitgestellt, der für einige wenige Motion Funktionen die Überwachung durch den die Funktion einleitenden Baustein abgebrochen hat. Dieser Baustein wird durch die inzwischen vollständigere Implementation der PLC Open Regeln nicht mehr benötigt.

Soll ein Baustein die Ausführung seiner Funktion nicht mehr überwachen ist es in den meisten Fällen ausreichend ihn ein- oder mehrmals mit **Execute:=FALSE** aufzurufen. Dies gilt insbesondere für [MC_MoveAbsolute_BkPlcMc \[▶ 54\]\(\)](#), [MC_MoveRelative_BkPlcMc \[▶ 57\]\(\)](#) und [MC_MoveVelocity_BkPlcMc \[▶ 59\]\(\)](#).

Anschließend kann in gleichen oder einem späteren Zyklus mit demselben Baustein oder einer Instanz desselben oder eines anderen Typs eine neue Funktionalität gestartet werden. Diese Vorgehensweise ist beliebig wiederholbar.

HINWEIS! Komplexe aus mehreren Teilaktionen zusammengesetzte Funktionen wie [MC_Home_BkPlcMc\(\)](#) erfordern den kontinuierlichen Aufruf des Bausteins da dieser die erforderlichen Abläufe selbst organisiert. ([MC_Home_BkPlcMc\(\)](#)) [▶ 51]

FAQ #12: Wie überwache ich die Kommunikation mit einem E/A-Gerät?

Sowohl [ST_TcPlcDeviceInput \[▶ 98\]](#) als auch [ST_TcPlcDeviceOutput \[▶ 100\]](#) Variablen stellen ein Element mit Namen **uiBoxState** bereit. Wenn die eingesetzten Buskoppler bzw. die Interface-Karten der Leistungsteile eine entsprechende Variable anbieten und beim eingesetzten Feldbus die Variable bei ungestörter Kommunikation den Wert 0 annimmt sollte eine Verknüpfung hergestellt werden. Diese Möglichkeit besteht z.B. beim **Beckhoff LightBus** und bei **Realtime Ethernet**. Wird für die Achse ein [MC_Power_BkPlcMc \[▶ 19\]](#) Baustein verwendet überwacht dieser den **uiBoxState** und meldet Störungen der Kommunikation. In einem solchen Fall wird die Achse in einen Stöorzustand versetzt.

EtherCAT bietet hier erheblich erweiterte Möglichkeiten.

FAQ #13: Wie versee ich die kundenspezifischen Achsparameter mit selbstgewählten Bezeichnungen?

Die [Axis_Ref_BkPlcMc \[▶ 62\]](#) Struktur unterstützt mit dem `pAuxLabels` Pointer die Verwendung eines Array von Texten, die vom `PlcMcManager` mit angezeigt werden. Diese Texte können durch den [MC_AxUtiStandardInit_BkPlcMc \[▶ 171\]](#) Baustein beim Start der Applikation aus einer Datei geladen werden. Dazu muss diesem Baustein ein die Adresse einer [ST_TcMcAuxDataLabels \[▶ 98\]](#) Variablen übergeben und eine geeignete Datei bereitgestellt werden.

Selbstverständlich ist es auch möglich, die Elemente der [ST_TcMcAuxDataLabels \[▶ 98\]](#) Variablen durch direktes Zuweisen aus der Applikation zu definieren. In diesem Fall wird die Datei nicht benötigt.

HINWEIS! Eine Reihe von Reglerbausteinen der Library definiert die Texte des Arrays automatisch.

FAQ #14: Wie steuere ich ein Stromventil an?

Im Gegensatz zu einem 4/2 oder 3/2 Proportionalwegeventil oder Servoventil wird ein Stromventil mit einem 0..10V Signal (bei vorhandener Ventilendstufe) angesteuert oder mit einem eingepprägten Strom von $0 \dots I_{\text{Nenn}}$ betätigt. In dieser Ansteuerung wird nur der Betrag der Geschwindigkeit übertragen. Die Richtung wird nicht mit dem Vorzeichen, sondern auf andere Weise transportiert. Hierfür werden zumeist digitale Signale benötigt, die für die Ansteuerung von Schaltventilen verwendet werden. In der [ST_TcPlcDeviceOutput \[▶ 100\]](#) Struktur stehen hierfür unter anderem die Elemente **bBrakeOff**, **bMovePos** und **bMoveNeg** bereit. Für die Erzeugung eines Absolut-Stellwerts ist `bDrive_AbsoluteOutput` in den Achsparametern zu setzen.

HINWEIS! Auf diesem Weg ist auch der Einsatz von klassischen Frequenzumrichtern mit Asynchronmotor, Encoder und Bremse möglich, wenn der Umrichter einen Analogeingang bereitstellt.

FAQ #15: Welche Variablen einer Achse sollten mit dem Scope aufgezeichnet werden?

Die folgende Signalzusammenstellung ist zu empfehlen:

- Immer: **Achsistposition:** [Axis_Ref_BkPlcMc.ST_TcHydAxRtData \[▶ 94\].fActPos](#): In Istwerteinheiten wie sie durch die Encoder-Skalierung festgelegt werden.

- Vor allem bei zeitgeführter Sollwertgenerierung, Getriebe- oder Gleichlaufkopplung, Kurvenscheibe: **Achssollposition**: `Axis_Ref_BkPlcMc.ST_TcHydAxRtData.fSetPos`: In Istwerteinheiten wie sie durch die Encoder-Skalierung festgelegt werden.
- Vor allem bei der Inbetriebnahme: **Geschwindigkeitswert**: `Axis_Ref_BkPlcMc.ST_TcHydAxRtData.fActVelo`: Geschwindigkeit in physikalischer Darstellung.
- Vor allem bei der Inbetriebnahme: **Restweg** oder **Zielposition**: `Axis_Ref_BkPlcMc.ST_TcHydAxRtData.fDistanceToTarget` oder `Axis_Ref_BkPlcMc.ST_TcHydAxRtData.fTargetPos`: In Istwerteinheiten wie sie durch die Encoder-Skalierung festgelegt werden.
- Nur bei aktiver Druck/Krafterfassung: **diverse Druck- und Kraftistwerte**: In `Axis_Ref_BkPlcMc.ST_TcHydAxRtData`: Nach Bedarf `fActPressure` `fActPressureA` `fActPressureB` `fActForce` `fValvePressure` `fSupplyPressure`: Drücke und Kräfte, Einheit wird durch die Parametrierung der Erfassungsbausteine festgelegt.
- Vor allem bei der Inbetriebnahme: **Geschwindigkeitsstellwert**: `Axis_Ref_BkPlcMc.ST_TcHydAxRtData.fSetVelo`: Geschwindigkeit in physikalischer Darstellung.
- Vor allem bei der Inbetriebnahme: **Reglerausgabe**: `Axis_Ref_BkPlcMc.ST_TcHydAxRtData.fLagCtrlOutput`: Geschwindigkeit in physikalischer Darstellung.

ⓘ HINWEIS! Die Signalauswahl im ScopeView wird erleichtert, wenn die Axis_Ref_BkPlcMc Variablen einen Namen erhalten, der mit aaa_ beginnt. Diese Vorgehensweise wird in den Programmbeispielen benutzt und sorgt dafür, dass die Variablen in der Symbol-Liste schnell auffindbar sind.

ⓘ HINWEIS! In der Signalzusammenstellung des ScopeViews können Kanäle zeitweise deaktiviert werden. So ist es möglich, eine umfassende Zusammenstellung vorzuhalten, aber nur die aktuell interessanten Daten aufzuzeichnen.

FAQ #16: Welche Bedeutung hat die Variable `nDebugTag` in `Axis_Ref_BkPlcMc`?

In dieser Variable hinterlegen fast alle Bausteine der Bibliothek für die Dauer ihrer Ausführung eine eindeutige Kennung. Dazu wird der vorgefundene Inhalt in einer lokalen Variablen des Bausteins gesichert und unmittelbar vor dem Verlassen des Bausteins wieder hergestellt.

Sollte es zu einem Programm-Absturz kommen und es besteht der Verdacht, dass es zu einem Problem in einem Baustein der Bibliothek gekommen ist sind die `nDebugTag` Variablen aller Achsen zu überprüfen. Ist ein Wert $\neq 0$ vorhanden war der Baustein vom Absturz betroffen und die Ursache ist zu klären. Die verwendeten numerischen Werte sind in der Bibliothek unter "Globale_Konstanten" aufgeführt. Es sollte zusätzlich der Inhalt von `ST_TcHydAxRtData` [\[► 94\]](#). `sTopBlockName` festgestellt werden. Hier ist in der Regel der Name des von der Applikation direkt aufgerufenen Bausteins zu finden.

FAQ #17: Welche Besonderheiten sind beim Einsatz von Sercos-Antrieben zu berücksichtigen?

Beim Betrieb von Sercos-Antrieben (ab V3.0.26) sind die nachfolgenden Regeln zu beachten:

- Die Sercos-Masteranschaltung (z.B. FC7501 usw.) muss im System Manager die textuelle Bezeichnung "SercosMaster" erhalten. Andernfalls ist weder eine Steuerung der Sercos Phase noch eine Parameter- und Diagnose-Kommunikation möglich.
- Es kann nur ein Sercos Strang mit der Bibliothek verwendet werden.
- Die Antriebsgeräte am Sercos Strang müssen im System Manager den textuellen Namen erhalten, unter dem sie der Bibliothek durch den Aufruf des `MC_AxUtiStandardInit_BkPlcMc()` Bausteins bekannt sind. Andernfalls ist weder eine Steuerung der Sercos Phase noch eine Parameter- und Diagnose-Kommunikation möglich.
- Die Eingangs-Variable `SystemState` [\[► 116\]](#) der Sercos-Masteranschaltung ist für jedes Antriebsgerät des Sercos Strangs zu verknüpfen.
- Beim Reset eines von mehreren Antrieben am Sercos Strang kann dieser den Feldbus unterbrechen. Die Sercos-Masteranschaltung wird dann einen entsprechenden Phasenwechsel durchlaufen. In der Regel wird der Hochlauf bis zur Phase 4 automatisch erfolgen. Anschließend wird:
 - die vom Reset adressierte Achse fehlerfrei sein sofern nicht weiterbestehende Probleme dagegen sprechen.

- sich jede andere Achse am Sercos Strang im Fehlerzustand (Feldbusausfall, Achse nicht betriebsbereit) befinden. Wenn der auslösende Reset der ersten Achse abgearbeitet ist können die anderen Achsen in der Regel ohne einen Phasenwechsel durch einen Reset in einen fehlerfreien Zustand gebracht werden.

Dieses Verhalten ist durch Eigentümlichkeiten des Sercos Feldbus festgelegt und durch die Bibliothek nicht zu beeinflussen. Es muss durch die Applikation in geeigneter Weise berücksichtigt werden.

- In Abhängigkeit von bestimmten Parametereinstellungen des Antriebsstellers werden Achsparameter automatisch ermittelt oder sind manuell vorzugeben:
 - S-0-0076, Bits 0 bis 2 legen die Wichtungsart der Lagedaten fest. Unterstützt wird:
 - a) 0 0 1 translatorische Gewichtung:
S-0-0123 definiert die Umdrehungsauflösung (Encoder-Interpolation). Aus dieser Anzahl und der Gewichtung (S-0-0077, S-0-0078) wird der Umdrehungsvorschub errechnet.
 - b) 0 1 0 rotatorische Gewichtung:
S-0-0079 definiert die Umdrehungsauflösung (Encoder-Interpolation). Der Umdrehungsvorschub ist manuell einzustellen.
 - S-0-0044, Bits 0 bis 2 legen die Wichtungsart der Geschwindigkeitsdaten fest. Unterstützt wird:
 - a) 0 0 1 translatorische Gewichtung:
Der Geschwindigkeitsstellwert wird mit dem Umdrehungsvorschub und der Umdrehungsauflösung in eine Geschwindigkeiten in Encoder-Inkrementen pro Zeit umgerechnet. Diese Information wird mit der Geschwindigkeitsauflösung (S-0-0045, S-0-0046) verrechnet und ausgegeben.
 - b) 0 1 0 rotatorische Gewichtung
Der Geschwindigkeitsstellwert wird mit dem Umdrehungsvorschub in eine Drehzahl umgerechnet und ausgegeben.
 - S-0-0091 wird mit der oben beschriebenen Umrechnungsmethode für Geschwindigkeitsstellwerte bewertet und als Bezugsgeschwindigkeit übernommen. Sollte die Maximalgeschwindigkeit den so ermittelten Wert übersteigen wird sie entsprechend begrenzt.

FAQ #18: Wie wird ein Druck oder eine Kraft ermittelt?

Zur Ermittlung eines Ist-Drucks oder einer Ist-Kraft sind pro Achse einer oder mehrere Bausteine der Typen [MC_AxRtReadPressureDiff_BkPlcMc \[► 143\]](#), [MC_AxRtReadForceDiff_BkPlcMc \[► 139\]](#), [MC_AxRtReadForceSingle_BkPlcMc \[► 141\]](#) oder [MC_AxRtReadPressureSingle_BkPlcMc \[► 145\]](#) aufzurufen. Details zur Aufrufreihenfolge sind unter [FAQ #8 \[► 216\]](#) zu finden.

Die an die Bausteine zu übergebenden AD-Wandlerwerte sind auf allozierte Variablen der Applikation zu verknüpfen. Details zur Wahl und zur Parametrierung sind in den Beschreibungen der Bausteine zu finden.

FAQ #19: Welche Besonderheiten sind beim Einsatz von AX5000 Antrieben zu berücksichtigen?

Bei AX5000 Geräten wird beim Herstellen der Kommunikation eine Reihe von IDNs aus dem Gerät gelesen und es werden verschiedene Parameter automatisch berechnet.

IDN	verwendet für Parameter
44	Bezugsgeschwindigkeit, intern: Skalierung der Geschwindigkeitsausgabe
45	intern: Skalierung der Geschwindigkeitsausgabe
46	intern: Skalierung der Geschwindigkeitsausgabe
76	Encoderinterpolation
79	Encoderinterpolation
91	Bezugsgeschwindigkeit

Die folgenden Parameter werden somit automatisch eingestellt und sind über den PlcMcManager nicht beeinflussbar:

Parameter		beeinflusst welche weiteren Parameter
Global: Bezugsgeschw.	wird aus der Maximal-Drehzahl des Geräts und dem Umdrehungsvorschub berechnet	Handgeschwindigkeiten, Max. Appl. Geschw.
Encoder: Inc.Zwischenteilung	wird aus IDN79 des Geräts gelesen	Achtung: der Umdrehungsvorschub ist als Inc.Bewertung einzugeben

FAQ #20: Wie wird eine Achse für die Nutzung des Blendings nach PLC Open vorbereitet?

in Vorbereitung

FAQ #21: Wie kann auf Register einer Klemme zugegriffen werden an der ein Encoder oder ein Ventil einer Achse angeschlossen ist?

Für die Registerkommunikation mit Klemmen an denen der Encoder oder das Ventil einer Achse angeschlossen sind wird die Verwendung von Bausteinen der Typen [MC_AxUtiReadRegDriveTerm_BkPlcMc \[► 196\]\(\)](#), [MC_AxUtiReadRegEncTerm_BkPlcMc \[► 197\]\(\)](#), [MC_AxUtiWriteRegDriveTerm_BkPlcMc \[► 204\]\(\)](#) und [MC_AxUtiWriteRegEncTerm_BkPlcMc \[► 205\]\(\)](#) empfohlen.

FAQ #22: Wie ist die ASCII Datei für eine Linearisierungstabelle aufgebaut?

Das Format einer ASCII Datei einer Linearisierungstabelle ist wie folgt festgelegt:

- Ein Linearisierungspunkt pro Zeile.
- Pro Zeile zuerst ein Geschwindigkeitswert und dann ein Ausgabewert.
- Die Geschwindigkeitswerte sind auf die Referenzgeschwindigkeit normalisiert. Sie liegen somit im Bereich -1.000 bis 1.000 einschließlich.
- Die Ausgabewerte sind auf die Vollaussteuerung normalisiert. Sie umfassen somit den Bereich -1.000 bis 1.000 vollständig.
- Vor dem ersten Wert einer Zeile dürfen White Space Characters (Leerzeichen, Tabulator) stehen.
- Zwischen den beiden Werten einer Zeile muss mindestens ein White Space Character (Leerzeichen, Tabulator) stehen.
- Zwischen den beiden Werten einer Zeile dürfen weitere White Space Characters (Leerzeichen, Tabulator) stehen.
- Als Dezimal-Trennzeichen sind Punkt und Komma zulässig.
- Zwischen einem negativen Vorzeichen und der ersten Ziffer sind keine Nichtziffern zulässig.
- Der erste Punkt legt das negative Tabellenende fest.
- Der Geschwindigkeitswert jedes weiteren Punktes muss höher (d.h. weniger negativ oder mehr positiv) sein als der seines Vorgängers.
- Es ist sinnvoll wenn der Ausgabewert eines Punktes höher (d.h. weniger negativ oder mehr positiv) ist als der seines Vorgängers, da es sonst zu einer negativen Steigung in diesem Bereich kommt. Dies würde durch einen Vorzeichenwechsel der Verstärkung eine Instabilität in einer eventuell aktiven Regelung erzeugen.
- Der Nullpunkt (d.h. beide Koordinaten des Punktes sind 0.000) ist mit anzugeben.

Beispiel: Die folgende (idealisierte) Tabelle beschreibt einen Zylinder der durch asymmetrische Wirkungsflächen (bedingt durch einseitige Kolbenstange) in negativer Bewegungsrichtung nur die halbe Geschwindigkeit der positiven Richtung erreicht. Es wird hier angenommen dass der Zylinder mit einem Nullschnitt-Ventil mit Kennlinienknick bei 40% betrieben wird

normalisierte Geschwindigkeit	normalisierte Ausgabe
-0,500	-1,000
-0,430	-0,900
-0,360	-0,800
-0,290	-0,700
-0,220	-0,600
-0,150	-0,500
-0,080	-0,400
-0,060	-0,300
-0,040	-0,200
-0,020	-0,100
0,000	0,000
0,040	0,100
0,080	0,200
0,120	0,300
0,160	0,400
0,300	0,500
0,440	0,600
0,580	0,700
0,720	0,800
0,860	0,900
1,000	1,000

FAQ #23: Wie können Kommandos des PlcMcManagers abgeblockt werden?

In einigen Situationen kann die Auslösung von Kommandos durch den PlcMcManager problematisch sein. Dies wäre z.B. der Fall, wenn eine bestimmte Abfolge von Aktionen zwingend vollständig abgearbeitet werden muss. Um in diesen Fällen die versehentlich Kommandogabe durch den PlcMcManager zu verhindern kann mit der [MC_AxRtCommandsLocked_BkPlcMc \[► 177\]](#) Funktion eine Verriegelung im Status-Doppelwort der Achse eingetragen werden. Ist diese Verriegelung aktiv wird ein vom PlcMcManager gesendetes Kommando mit einem Schreibschutz-Fehler abgelehnt.

i HINWEIS! Es ist unbedingt erforderlich die Verriegelung nach Abarbeitung der zu schützenden Aktion aufzuheben. Dies gilt auch und besonders beim Auftreten von Fehlern.

Es steht ein Beispiel zur Verfügung.

Sehen Sie dazu auch

 [MC_AxStandardBody_BkPlcMc \(V3.0\) \[► 170\]](#)

Dokumente hierzu

 [tcplcmcx_18.pro \(Resources/pro/1599851275.pro\)](#)

4.2 Globale Konstanten (ab V3.0)

Bit-Masken für Positionsnocken

Bit-Masken für Positionsnocken

Diese Masken sind von der Applikation für die Bereitstellung von digitalen Wegnocken für bActPosCams in ST_TcHydAxRtData zu verwenden.

Konstante	Beschreibung
bTcHydActPosCamPos	Zusammenfassung von bTcHydActPosCamHigh und bTcHydActPosCamUp.
bTcHydActPosCamHigh	Achse hat obere Zielposition erreicht.
bTcHydActPosCamUp	Achse befindet sich in der Nähe der oberen Zielposition.
bTcHydActPosCamDown	Achse befindet sich in der Nähe der unteren Zielposition.
bTcHydActPosCamLow	Achse hat untere Zielposition erreicht.
bTcHydActPosCamNeg	Zusammenfassung von bTcHydActPosCamLow und bTcHydActPosCamDown.

Bit-Masken für Achsstatusinformationen

Bit-Masken für Achsstatusinformationen

Diese Masken sind von der Applikation für die Abfrage von Status-Signalen in nStateDWord in ST_TcHydAxRtData zu verwenden.

Konstante	Beschreibung
dwTcHydNsDwFunctional	Achse ist betriebsbereit.
dwTcHydNsDwReferenced	Achse ist referenziert.
dwTcHydNsDwSteady	Achse ist nicht aktiv.
dwTcHydNsDwInTargRng	Die Achse befindet sich innerhalb einer von fMonPositionRange in ST_TcHydAxParam vorgegebenen Entfernung von der Zielposition.
dwTcHydNsDwInTarget	Die Achse befindet sich seit einer von fMonTargetFilter vorgegebenen Zeit ununterbrochen innerhalb einer von fMonTargetRange in ST_TcHydAxParam vorgegebenen Entfernung von der Zielposition.
dwTcHydNsDwDontTouchProtected	reserviert. nicht unterstützt.
dwTcHydNsDwStopped	Die letzte Bewegung der Achse wurde gestoppt, ohne die vorgegebene Zielposition zu erreichen.
dwTcHydNsDwBusy	Achse ist aktiv.
dwTcHydNsDwMoveUp	Achse bewegt sich in Richtung größerer Positionen.
dwTcHydNsDwMoveDown	Achse bewegt sich in Richtung kleinerer Positionen.
dwTcHydNsDwReferencing	Achse führt eine Referenzfahrt aus.
dwTcHydNsDwConstVelo	Achse bewegt sich mit konstanter Geschwindigkeit.
dwTcHydNsDwExtSetpointActive	Die Achse wird durch einen MC_AxRtSetExtGenValues_BkPlcMc [► 169] Baustein kontrolliert.
dwTcHydNsDwStartedOver	Die Achse wurde durchgestartet, d.h. das letzte akzeptierte Kommando wurde wirksam während die Achse noch eine aktive Bewegung ausführte.
dwTcHydNsDwControlActive	reserviert. nicht unterstützt.
dwTcHydNsDwErrState	Achse im Störzustand.

Bit-Masken für Achsfreigabeinformationen

Bit-Masken für Achsfreigabeinformationen

Diese Masken sind von der Applikation für die Bereitstellung von Freigabe-Signalen in nDeCtrlDWord in ST_TcHydAxRtData zu verwenden.

Konstante	Beschreibung
dwTcHydDcDwCtrlEnable	Reglerfreigabe. Diese Freigabe ist Voraussetzung für die Ausgabe von Stellwert- und Regler-Ausgaben.
dwTcHydDcDwFdPosEna	Vorschubfreigabe in positiver Richtung. Diese Freigabe ist Voraussetzung für eine Ausgabe von Stellwert- und Regler-Ausgaben in Richtung steigender Positionswerte.
dwTcHydDcDwCtrlPosEna	Zusammenfassung von dwTcHydDcDwCtrlEnable und dwTcHydDcDwFdPosEna.
dwTcHydDcDwFdNegEna	Vorschubfreigabe in negativer Richtung. Diese Freigabe ist Voraussetzung für eine Ausgabe von Stellwert- und Regler-Ausgaben in Richtung sinkender Positionswerte.
dwTcHydDcDwCtrlNegEna	Zusammenfassung von dwTcHydDcDwCtrlEnable und dwTcHydDcDwFdNegEna.
dwTcHydDcDwRefIndex	Referenziernocken.
dwTcHydDcDwAcceptBlockedDrive	reserviert. nicht unterstützt.
dwTcHydDcDwBlockedDriveDetected	reserviert. nicht vollständig unterstützt. Hinweis: Dieses Signal unterdrückt einen eventuell wirksamen Geschwindigkeitsregler.

Error-Codes

Diese Konstanten werden für ErrorID-Ausgänge von Bausteinen und nErrorCode in ST_TcHydAxRtData zu verwenden.

Konstante	Hexadezimal	Dezimal	Beschreibung
dwTcHydAdsErrNoError	0	0	Kein Fehler.
dwTcHydAdsErrUnknownPort	16#0006	6	ADS Port nicht bekannt. Mögliche Ursachen: <ul style="list-style-type: none"> AMS NetID / ADS Port adressieren das falsche Laufzeitsystem oder den falschen Rechner in der adressierten PLC läuft ein anderes Projekt die Applikation ruft keinen <u>MC_AxAdsCommServer_BkPlcMc [► 187]()</u> Baustein auf
dwTcHydAdsErrUnknownTarget	16#0007	7	Target Maschine nicht bekannt. Mögliche Ursachen: <ul style="list-style-type: none"> AMS NetID / ADS Port adressieren das falsche Laufzeitsystem oder den falschen Rechner das Zielsystem ist nicht gestartet TwinCAT ist nicht gestartet die Verbindung ist elektrisch / mechanisch unterbrochen bei Kommunikation via Ethernet: Die TCP/IP Verbindung arbeitet nicht
dwTcHydAdsErrInvalidIdxGroup	16#0702	1794	Unzulässiger IndexGroup. Mögliche Ursachen: <ul style="list-style-type: none"> AMS NetID / ADS Port adressieren das falsche Laufzeitsystem oder den falschen Rechner in der adressierten PLC läuft ein anderes Projekt Softwarefehler der Applikation (falsche Kombination von ADS Port / IdxGroup / IdxOffset)
dwTcHydAdsErrInvalidIdxOffset	16#0703	1795	Unzulässiger IndexOffset. Mögliche Ursachen: <ul style="list-style-type: none"> AMS NetID / ADS Port adressieren das falsche Laufzeitsystem oder den falschen Rechner in der adressierten PLC läuft ein anderes Projekt Softwarefehler der Applikation (falsche Kombination von ADS Port / IdxGroup / IdxOffset) versuchter Zugriff auf ein Array Element mit unzulässigem Index (out of bounds)
dwTcHydAdsErrRdWrNotPermitted	16#0704	1796	Zugriff (Schreiben, Lesen) nicht zulässig. Mögliche Ursachen: <ul style="list-style-type: none"> es wurde ein Schreibzugriff auf eine nicht beschreibbare Variable angefordert
dwTcHydAdsErrInvalidSize	16#0705	1797	Nicht zulässige Größe (Anzahl Bytes). Mögliche Ursachen: <ul style="list-style-type: none"> Softwarefehler der Applikation (falsche Kombination von ADS Port / IdxGroup / IdxOffset)

Konstante	Hexadezimal	Dezimal	Beschreibung
dwTcHydAdsErrIllegalValue	16#0706	1798	Nicht zulässiger Wert. Mögliche Ursachen: <ul style="list-style-type: none"> • der übertragene Wert liegt außerhalb von absoluten Grenzen des Parameters • der übertragene Wert liegt außerhalb von Grenzen des Parameters die durch andere bereits gültig gewordene Parameter festgelegt sind
dwTcHydAdsErrNotReady	16#0707	1799	Nicht betriebsbereit. Mögliche Ursachen: <ul style="list-style-type: none"> • ein MC_Power_BkPlcMc Baustein wurde durch seinen Enable Eingang veranlasst eine nicht betriebsbereite Achse zu aktivieren
dwTcHydAdsErrBusy	16#0708	1800	Bereits aktiv. Mögliche Ursachen: <ul style="list-style-type: none"> • die Achse könnte einen Auftrag nicht akzeptieren weil sie bereits mit einer anderen Aufgabe beschäftigt ist
dwTcHydAdsErrNoFile	16#070C	1804	reserviert: Datei fehlt / nicht zugreifbar.
dwTcHydAdsErrSyntax	16#070D	1805	Syntax in Kommando oder Datei unzulässig. Mögliche Ursachen: <ul style="list-style-type: none"> • beim Lesen einer im ASCII Format gespeicherten Kennlinien-Datei wurden nicht zulässige Zeichen oder Zeichenkombinationen erkannt • beim Lesen einer im ASCII Format gespeicherten Kennlinien-Datei wurden unvollständige Angaben erkannt
dwTcHydAdsErrTimeout	16#0745	1861	Zeitüberschreitung. Mögliche Ursachen: <ul style="list-style-type: none"> • in einer Kommunikation ist die Antwort nicht innerhalb einer vorgesehenen Zeit eingetroffen <ul style="list-style-type: none"> ◦ die Zeit ist zu kurz gewählt ◦ die Verbindung ist unterbrochen • der Prozess hat die Abarbeitung des Kommandos verhindert oder über das vorgesehene Maß hinaus verzögert • die vorgegebenen Parameter des Kommandos haben den Zeitbedarf über das vorgesehene Maß hinaus erhöht
dwTcHydErrCdNotCompatible	16#4040	16448	Achse ist zur geforderten Funktion nicht kompatibel. Mögliche Ursachen: <ul style="list-style-type: none"> • Softwarefehler der Applikation
dwTcHydErrCdIllegalOutputNumber	16#4104	16644	Ausgangsnummer außerhalb des zulässigen Bereichs. Mögliche Ursachen: <ul style="list-style-type: none"> • ein MC_ReadDigitalOutput_BkPlcMc oder MC_WriteDigitalOutput_BkPlcMc Baustein wurde mit einem unzulässigen Parameter aufgerufen.
dwTcHydErrCdNotSupport	16#4107	16647	Funktion oder Kommando wird nicht unterstützt. Mögliche Ursachen: <ul style="list-style-type: none"> • Softwarefehler der Applikation
dwTcHydErrCdCycleTime	16#4205	16901	Zykluszeit (fCycleTime in ST_TcHydAxParam) nicht zulässig. Mögliche Ursachen: <ul style="list-style-type: none"> • Parametrierfehler

Konstante	Hexadezimal	Dezimal	Beschreibung
dwTcHydErrCdMissingEnc	16#4210	16912	Es besteht keine Verbindung zu einem Encoder-Interface (pStDeviceInput und/oder pStDeviceOutput in Axis_Ref_BkPlcMc [► 62]). Mögliche Ursachen: <ul style="list-style-type: none"> • Softwarefehler der Applikation (der MC_AxUtiStandardInit_BkPlcMc Baustein wurde nicht aufgerufen oder nicht mit der Adresse einer ST_TcPlcDeviceInput und einer ST_TcPlcDeviceOutput Struktur versorgt)
dwTcHydErrCdMissingDrive	16#4212	16914	Es besteht keine Verbindung zu einem Drive-Interface (pStDeviceInput und/oder pStDeviceOutput in Axis_Ref_BkPlcMc [► 62]). Mögliche Ursachen: <ul style="list-style-type: none"> • Softwarefehler der Applikation (der MC_AxUtiStandardInit_BkPlcMc Baustein wurde nicht aufgerufen oder nicht mit der Adresse einer ST_TcPlcDeviceInput und einer ST_TcPlcDeviceOutput Struktur versorgt)
dwTcHydErrCdCannotSynchronize	16#421A	16922	Start-Distanz bei Aufruf eines MC_GearInPos_BkPlcMc() Bausteins nicht ausreichend. Mögliche Ursachen: <ul style="list-style-type: none"> • die Achse steht bei Aktivierung des Bausteins zu nah am Synchronisierungspunkt • die Dynamik-Parameter der Achse sind nicht ausreichend
dwTcHydErrCdIllegalGearFactor	16#421B	16923	Die Parameter einer Getriebekopplung sind nicht zulässig. Mögliche Ursachen: <ul style="list-style-type: none"> • der Parameter des Bausteins ist nicht zulässig
dwTcHydErrCdSoftEnd	16#4222	16930	Die Zielposition liegt jenseits eines aktiven Software-Endschalters und ist somit nicht zulässig.
dwTcHydErrCdLowDist	16#4228	16936	Die Fahrstrecke ist unzulässig klein.
dwTcHydErrCdIllegalStartType	16#4239	16953	Unzulässiger Start-Typ.
dwTcHydErrCdCommandBufferOverflow	16#423F	16959	Kommando Puffer ist voll.
dwTcHydErrCdEncLostCam	16#4253	16979	reserviert. nicht unterstützt.
dwTcHydErrCdCtrlEnaLost	16#4260	16992	Reglerfreigabe wurde während Bewegung entzogen. Mögliche Ursachen: <ul style="list-style-type: none"> • auf Grund eines Signals der Maschinen-Logik wurde der Achse die Freigabe zu einem unerwarteten Zeitpunkt entzogen • Softwarefehler der Applikation
dwTcHydErrCdEncNoCamFound	16#429C	17052	reserviert. nicht unterstützt.
dwTcHydErrCdEncNoCamEnd	16#429D	17053	reserviert. nicht unterstützt.
dwTcHydErrCdEncNoSyncPulse	16#429E	17054	reserviert. nicht unterstützt.
dwTcHydErrCdAcc	16#4309	17161	Die Beschleunigung ist nicht zulässig.
dwTcHydErrCdDec	16#430A	17162	Die Verzögerung ist nicht zulässig.
dwTcHydErrCdJerk	16#430B	17163	Die Ruckbegrenzung ist nicht zulässig.
dwTcHydErrCdPtrPlcMc	16#4345	17221	Es besteht keine Verbindung zu einem der benötigten Achs-Interfaces (pStDeviceInput oder pStDeviceOutput in Axis_Ref_BkPlcMc [► 62]).

Konstante	Hexadezimal	Dezimal	Beschreibung
dwTcHydErrCdPtrMcPlc	16#4346	17222	Es besteht keine Verbindung zu einem der benötigten Achs-Interfaces (pStDeviceInput oder pStDeviceOutput in Axis_Ref_BkPlcMc [► 62]).
dwTcHydErrCdCtrlEna	16#4356	17238	Bewegung ohne Reglerfreigabe unzulässig.
dwTcHydErrCdNegFdEna	16#4357	17239	Bewegung in Richtung sinkender Positionen ohne Vorschubfreigabe in negativer Richtung unzulässig.
dwTcHydErrCdPosFdEna	16#4358	17240	Bewegung in Richtung steigender Positionen ohne Vorschubfreigabe in positiver Richtung unzulässig.
dwTcHydErrCdSetVelo	16#4359	17241	Die geforderte Geschwindigkeit ist nicht zulässig.
dwTcHydErrCdPehTimeout	16#435C	17244	Die Achse erreicht das Zielfenster nicht innerhalb der vorgegebenen Zeit.
dwTcHydErrCdNotMoving	16#435D	17245	Die Achse bewegt sich nicht oder nicht in die richtige Richtung.
dwTcHydErrCdConsequential	16#43A0	17312	Folgefehler: Die Achse ist durch ein Problem bei einer anderen Achse ebenfalls in den Störfzustand versetzt worden.
dwTcHydErrCdEncType	16#4401	17409	Der Parametertyp ist ungültig.
dwTcHydErrCdEncScaling	16#4406	17414	Die Inkrementskalierung ist unzulässig.
dwTcHydErrCdEncSyncDist	16#4414	17428	Der Abstand zwischen Latch_Enable und Sync-Puls ist zu klein.
dwTcHydErrCdEncSetActPos	16#4422	17442	Beim Istwertsetzen ist ein Problem aufgetreten.
dwTcHydErrCdPtrPlcEncln	16#4442	17474	Die Achse besitzt keinen Zeiger auf ein Encodereingangsinterface
dwTcHydErrCdPtrPlcEncOut	16#4443	17475	Die Achse besitzt keinen Zeiger auf ein Encoderausgangsinterface
dwTcHydErrCdEncUnderrun	16#4450	17488	Von einigen Encoder-Typen gemeldet: Die Istposition hat die untere Zählgrenze des Encoders passiert.
dwTcHydErrCdEncOverrun	16#4451	17489	Von einigen Encoder-Typen gemeldet: Die Istposition hat die obere Zählgrenze des Encoders passiert.
dwTcHydErrCdEncHdwFailed	16#4464	17508	Antriebssteller oder Encoder melden einen Hardwarefehler
dwTcHydErrCdSsi	16#4470	17520	Beim Betrieb eines SSI-Encoders wurde ein Fehler erkannt.
dwTcHydErrCdPosLag	16#4550	17744	Der Schleppabstand überschreitet eine aktive Grenze.
dwTcHydErrCdDriveType	16#4601	17921	Der in nDrive_Type eingestellte Wert ist nicht zulässig.
dwTcHydErrCdRefVelo	16#4605	17925	Bezugsgeschwindigkeit (fRefVelo in ST_TcHydAxParam) nicht zulässig.
dwTcHydErrCdStepperStalled	16#4636	17974	Eine Stall-Situation wurde erkannt.
dwTcHydErrCdPtrPlcDriveIn	16#4642	17986	Die Achse besitzt keinen Zeiger auf ein Antriebseingangsinterface
dwTcHydErrCdPtrPlcDriveOut	16#4643	17987	Die Achse besitzt keinen Zeiger auf ein Antriebsausgangsinterface
dwTcHydErrCdDriveNotReady	16#4650	18000	Leistungsteil nicht betriebsbereit
dwTcHydErrCdTblEntryCount	16#4A02	18946	Die Anzahl der Tabelleneinträge (Zeilen) ist nicht zulässig
dwTcHydErrCdTblInvalidMasterStep	16#4A04	18948	Die Tabelle enthält Einträge mit einer unzulässigen Master-Schrittweite.
dwTcHydErrCdTblNoInit	16#4A10	18960	Die Tabelle ist nicht initialisiert

Konstante	Hexadezimal	Dezimal	Beschreibung
dwTcHydErrCdTblIllegalIndex	16#4A13	18963	Tabellenindex nicht zulässig.
dwTcHydErrCdTblLineCount	16#4A15	18965	Die Anzahl der Tabelleneinträge ist zu groß
dwTcHydErrCdNotStartable	16#4B01	19201	Achse ist in einem Zustand, der einen Start nicht zulässt.
dwTcHydErrCdFuncTimeout	16#4B07	19207	Funktion wurde nicht in der vorgegebenen Zeit fertig gemeldet.
dwTcHydErrCdNotReady	16#4B09	19209	Achse ist in einem nicht betriebsbereiten Zustand.
dwTcHydErrCdHomingType	16#4F00	20224	Referenziermethode (nEnc_HomingType in ST_TcHydAxParam) nicht zulässig.
dwTcHydErrCdEncCutOff	16#4F01	20225	Die Grenzfrequenz der Istwerterfassung wurde überschritten.
dwTcHydErrCdIllegalDistance	16#4F02	20226	Distanz ist unzulässig: Null oder negativ.
dwTcHydErrEncDisconnected	16#4FF0	20464	Encoderhardware ist abgekoppelt. Mögliche Ursachen: <ul style="list-style-type: none"> • die Feldbusverbindung ist unterbrochen • die Stromversorgung des Geräts ist nicht verfügbar • das Gerät ist ausgefallen • ein in der Feldbusverbindung zwischen der Steuerung und dem Gerät angeordnetes anderes Gerät ist nicht mit Strom versorgt oder ausgefallen
dwTcHydErrDriveDisconnected	16#4FF1	20465	Antriebshardware ist abgekoppelt. Mögliche Ursachen: <ul style="list-style-type: none"> • die Feldbusverbindung ist unterbrochen • die Stromversorgung des Geräts ist nicht verfügbar • das Gerät ist ausgefallen • ein in der Feldbusverbindung zwischen der Steuerung und dem Gerät angeordnetes anderes Gerät ist nicht mit Strom versorgt oder ausgefallen
dwTcHydErrDistanceInsufficient	16#4FF2	20466	Fahrweg nicht ausreichend.

Gerätespezifische Error-Codes des Bausteins MC_Power_BkPlcMc

Gerätespezifische Error-Codes des Bausteins MC_Power_BkPlcMc

Diese Werte erscheinen am **ErrorID** Ausgang eines MC_Power_BkPlcMc Bausteins, wenn ein Fehler vom externen Gerät gemeldet wird.

Konstante	Hexadezimal	Dezimal	Beschreibung
dwTcHydErrCdAX2000MainPwrTmOut	16#0001	1	Nur bei AX2000: Keine Rückmeldung durch das Netzschütz (Zeitüberschreitung beim Warten auf ST_TcPlcMcAx2000In.bPowerOk).
dwTcHydErrCdAX2000MainPwrFault	16#0002	2	Nur bei AX2000: Fallende Flanke an Rückmeldung des Netzschütz (ST_TcPlcMcAx2000In.bPowerOk).
dwTcHydErrCdAX2000PwrStageTmOut	16#0003	3	Nur bei AX2000: Keine Rückmeldung von AX-Endstufe (Zeitüberschreitung beim Warten auf ST_TcPlcMcAx2000In.DriveState[3].6, kein Ready).
dwTcHydErrCdAX2000PwrStageFault	16#0004	4	Nur bei AX2000: Fallende Flanke von AX-Endstufe (ST_TcPlcMcAx2000In.DriveState[3].6, kein Ready).
dwTcHydErrCdAX2000ReportsError	16#0005	5	Nur bei AX2000: Fehlermeldung durch AX-Gerät (ST_TcPlcMcAx2000In.DriveState[3].7 oder ST_TcPlcMcAx2000In.DriveError<>0).
dwTcHydErrCdAX2000ErrorI2T	16#0006	6	Nur bei AX2000: I ² T-Fehlermeldung von AX-Endstufe (ST_TcPlcMcAx2000In.DriveState[0].0).
dwTcHydErrCdAX2000ErrorChopper	16#0007	7	Nur bei AX2000: Bremswiderstand der AX-Endstufe defekt (ST_TcPlcMcAx2000In.DriveState[0].1).
dwTcHydErrCdAX2000ErrorWatchDog	16#0008	8	Nur bei AX2000: Watchdog (Timeout bei Kommunikation) der AX-Endstufe hat angesprochen (ST_TcPlcMcAx2000In.DriveState[0].3).
dwTcHydErrCdAX2000ErrorPwrLine	16#0009	9	Nur bei AX2000: Versorgungsfehler durch AX-Endstufe gemeldet (ST_TcPlcMcAx2000In.DriveState[0].4).
dwTcHydErrCdAX2000ConnectionLost	16#000A	10	Nur bei AX2000: Die Verbindung zum AX-Gerät ist unterbrochen oder erheblich gestört (ST_TcPlcMcAx2000In.BoxState<>0).
dwTcHydErrCdAX2000ConnectionTmOut	16#000B	11	Nur bei AX2000: Die Kommunikation mit dem AX-Gerät konnte nicht aufgenommen werden (Zeitüberschreitung).
dwTcHydErrCdKL2531OverTemp	16#0001	1	Nur bei KL2531/KL2541: Die KL2531/KL2541-Klemme meldet Übertemperatur-Alarm.
dwTcHydErrCdKL2531UnderVoltage	16#0002	2	Nur bei KL2531/KL2541: Die KL2531/KL2541-Klemme meldet eine zu geringe Versorgungsspannung auf der Power Rail.
	16#0003	3	Nur bei KL2531/KL2541: Reserviert.
dwTcHydErrCdKL2531OpenLoadA	16#0004	4	Nur bei KL2531/KL2541: Die KL2531/KL2541-Klemme meldet Drahtbruch auf A-Seite.
dwTcHydErrCdKL2531OpenLoadB	16#0005	5	Nur bei KL2531/KL2541: Die KL2531/KL2541-Klemme meldet Drahtbruch auf B-Seite.
dwTcHydErrCdKL2531OverCurrentA	16#0006	6	Nur bei KL2531/KL2541: Die KL2531/KL2541-Klemme meldet Überstrom an Endstufe A.
dwTcHydErrCdKL2531OverCurrentB	16#0007	7	Nur bei KL2531/KL2541: Die KL2531/KL2541-Klemme meldet Überstrom an Endstufe B.
dwTcHydErrCdKL2531NotReady	16#0008	8	Nur bei KL2531/KL2541: Die Klemme meldet ein Endstufenproblem (Enabled, nicht Ready).
dwTcHydErrCdKL2531ConnectionLost	16#000A	10	Nur bei KL2531/KL2541: Die Verbindung zum Klemme ist unterbrochen oder erheblich gestört (ST_TcPlcMcDriveIn.uiBoxState<>0).
dwTcHydErrCdKL2531ConnectionTmOut	16#000B	11	Nur bei KL2531/KL2541: Die Kommunikation mit der Klemme konnte nicht aufgenommen werden (Zeitüberschreitung).

ADS-Codes

ADS-Codes

Diese Konstanten werden von den MC_AxAdsReadDecoder und MC_AxAdsWriteDecoder Bausteinen akzeptiert.

IndexGroup	IndexOffset	Typ	R/W	Beschreibung
16#4000 + Achsisindex	2	STRING()	R	Achsname als Text.
	4	UDINT	R	Zykluszeit in Mikrosekunden.
	16#10003	UDINT	R	Encodertyp: nEnc_Type aus ST_TcHydAxParam.
	16#10006	LREAL	R	Inkrementbewertung: fEnc_IncWeighting aus ST_TcHydAxParam.
	16#30003	UDINT	R	Drivetyyp: nDrive_Type aus ST_TcHydAxParam.
16#4100 + Achsisindex	1	UDINT	R	Errorcode: nErrorCode aus ST_TcHydAxRtData.
	16#10002	LREAL	R	Istposition: fActPos aus ST_TcHydAxRtData.
	16#10005	LREAL	R	Istgeschwindigkeit: fActVelo aus ST_TcHydAxRtData.
16#4200 + Achsisindex	1	-	W	Achsreset ausführen.
	16#10	-	W	Referenzfahrt starten.
	16#21	Struktur	W	Achsbewegung starten.
	16#FFFF0001	-	W	Parameter speichern.
	16#FFFF0002	-	W	Parameter laden.
16#4300 + Achsisindex	16#81	UDINT	R	Statusdoppelwort: nStateDWord aus ST_TcHydAxRtData.
	16#B1	UDINT	R	Errorcode: nErrorCode aus ST_TcHydAxRtData.
16#F000 + Achsisindex	1	Struktur	R	Die ST_TcHydAxRtData Variable der Achse.
	2	Struktur	R/W	Die ST_TcHydAxParam Variable der Achse.
16#800F0000 + Achsisindex	E_TcMCPParameter [► 74]		R/W	Parameter und Aktualwerte der Achse.
16#FFFFFFFF	0	String()	R	Kennung des Servers.
	1	UINT	R	Major version der Library.
	2	UINT	R	Minor version der Library.
	3	UINT	R	Release der Library.
	4	UINT	R	Anzahl der unterstützten Achsen

Array Dimensions

Array Dimensions

Die folgenden Konstanten werden für die Dimensionierung von Feldern verwendet und können von der Applikation genutzt werden.

Konstante	Beschreibung
ciBkPlcMc_CamSwitchRef_MinIdx	Lower boundary index on an array[] of CAMSWITCH_REF BkPlcMc [▶ 63], supplied to blocks of type MC_DigitalCamSwitch BkPlcMc [▶ 41]
ciBkPlcMc_CamSwitchRef_MaxIdx	Upper boundary index on an array[] of CAMSWITCH_REF BkPlcMc [▶ 63], supplied to blocks of type MC_DigitalCamSwitch BkPlcMc [▶ 41]
ciBkPlcMc_TrackRef_MinIdx	Lower boundary index on an array[] of TRACK_REF BkPlcMc [▶ 103], supplied to blocks of type MC_DigitalCamSwitch BkPlcMc [▶ 41]
ciBkPlcMc_TrackRef_MaxIdx	Upper boundary index on an array[] of TRACK_REF BkPlcMc [▶ 103], supplied to blocks of type MC_DigitalCamSwitch BkPlcMc [▶ 41]

Logger Levels

Logger Levels

Die folgenden Konstanten werden für die Festlegung des Levels verwendet, ab der Meldungen in die Logger-Funktion der Library aufgenommen werden.

Konstante	Beschreibung
dwTcHydLogLevel_None	Keine Aufzeichnung
dwTcHydLogLevel_Errors	Nur Fehlermeldungen
dwTcHydLogLevel_Warnings	Fehlermeldungen und Warnungen
dwTcHydLogLevel_Actions	Fehlermeldungen, Warnungen und Aktivitäten

Logger Sources

Logger Sources

Die folgenden Konstanten werden für die Festlegung der Quelle einer Meldung in der Logger-Funktion der Library verwendet.

Konstante	Beschreibung
dwTcHydLogSource_Library	Ein Baustein der Hydraulik-Library
dwTcHydLogSource_LibExt_2R2V	Ein Baustein der 2R2V Library
dwTcHydLogSource_Application	Ein Baustein der Applikation
dwTcHydLogSource_ApplicationFramework	Ein Baustein einer Applikation-Plattform

Logger Argument Types

Logger Argument Types

Die folgenden Konstanten werden für die Festlegung des Typs eines optionalen Parameters einer Meldung in der Logger-Funktion der Library verwendet.

Konstante	Beschreibung
dwTcHydLogArgType_DInt	Die Meldung enthält einen Parameter vom Typ DINT. Im Text der Meldung muss ein Platzhalter der Form %d erscheinen.
dwTcHydLogArgType_LReal	Die Meldung enthält einen Parameter vom Typ LREAL. Im Text der Meldung muss ein Platzhalter der Form %f erscheinen.
dwTcHydLogArgType_String	Die Meldung enthält einen Parameter vom Typ STRING. Im Text der Meldung muss ein Platzhalter der Form %s erscheinen.

4.3 Ventil

Das Ventil ist in der Regel das Stellglied, wodurch die Achse geregelt wird. Unter den Stetigventilen wird zwischen:

- Servoventil
- Proportionalventil
- Regelventil

unterschieden.

Servoventil

Diese Ventile steuern über kleine elektrische Signale große Ölströme

- Ein kleiner Torque-Motor regelt das angeschlossene Steueröl, wodurch der Schieber der Hauptstufe verstellt wird.
- Oft mehrstufig ausgelegt
- hohe Reaktionsfähigkeit und Regelbarkeit

Proportionalventil

Ein Spulenstrom erzeugt eine proportionale Kraft, die den Schieber des Ventils gegen die Kraft einer Feder verstellt.

Im Vergleich zum Servoventil:

- Größere Sprungantwortzeit
- Größerer Stromverbrauch
- Größere Hysterese
- Robuster gegenüber Verschmutzung
- Günstiger im Preis

Regelventil:

Ist ein Proportionalventil, bei dem die Schieberposition gemessen und nachgeführt wird, dadurch:

- Kleinere Sprungantwortzeit
- Kleinere Hysterese
- Weniger Lastrückwirkung
- Aufwändiger und teurer als Proportionalventile
- Elektronik auf dem Ventil oder im Schaltschrank

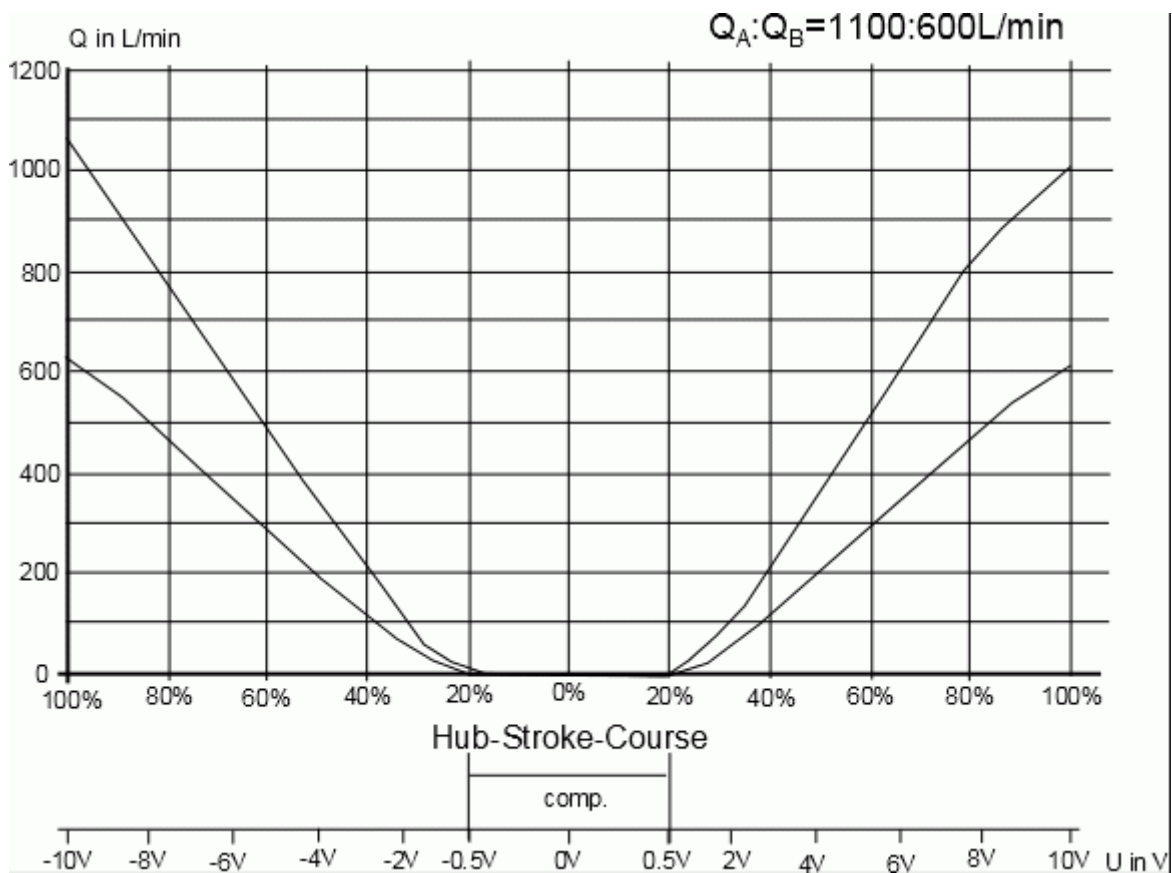
Grundlagen zum Lesen von Ventildatenblättern

Das Stetigventil ist in der Regel das Stellglied vom Regler und von Hersteller zu Hersteller und von Typ zu Typ sehr unterschiedlich gefertigt. Um die Ausgabeskalierung auf die Gegebenheiten dieses Ventils anzupassen muss bei der Inbetriebnahme das Ventildatenblatt des Stetigventils vorliegen. An einem Ventil gibt es verschiedene Hydraulische Anschlüsse. A und B sind die Ventilausgänge wobei A an die Zylinderseite mit der größeren Kolbenfläche und B an die Zylinderseite mit der kleineren Kolbenfläche angeschlossen wird. P und T stellen die Versorgungsanschlüsse darstellen. P ist die Druckleitung und T ist die Rückleitung in den Tank.

☐ HINWEIS! In der Hydraulikbibliothek wird unter A-Seite immer die Seite verstanden, die positiv fahrend ist und B-Seite ist die negativ fahrende Seite.

Häufig muss der Ventilschieber einen gewissen Hub vornehmen, bevor ein Ölfluss zu erkennen ist. Dieser Hub wird im Datenblatt des Ventils unter Überdeckung gelistet.

HINWEIS! In einigen Fällen handelt es sich laut Datenblatt um ein überdecktes Ventil, dessen Überdeckung in der Elektronik des Ventils kompensiert wird.



Die Volumenstromkennlinie gibt die wichtigsten Informationen des Ventils wieder. In der Abbildung oben ist zu erkennen, dass der Kolben selbst eine Überdeckung von 20% aufweist, diese aber in der Elektronik des Ventils auf 5% verringert wurde. Dadurch ist keine Überdeckungskompensation über die Hydraulikbibliothek vorzunehmen.

HINWEIS! Nur weil die Überdeckungskompensation im Ventil vorgenommen worden ist, handelt es sich nicht um ein Nullschnittventil und somit ist eine Achse auch hier nur in Grenzen lageregelungsfähig.

Es ist zu erkennen, dass der Ölstrom in die A-Kammer des Kolbens größer ist als der Ölstrom in die B-Kammer. Diese Asymmetrie ist eine Flächenkompensation im Ventil, in diesem Fall von 11:6.

4.4 Programm-Beispiele (ab V3.0)

HINWEIS! Beim Öffnen der SPS Projekte erscheint in der Regel die Meldung, dass eine oder mehrere Bibliotheken nicht geladen werden konnten.

Es wird angeboten, den Pfad zur Bibliothek anzupassen. Von dieser Möglichkeit können Sie nur dann Gebrauch machen, wenn die Bibliothek in einer ausreichend aktuellen Version auf Ihrem Rechner verfügbar ist. Andernfalls sollten Sie das Angebot ablehnen und das Projekt als reines Anschauungsmaterial nutzen. Ein Kompilieren und Starten ist dann natürlich nicht möglich.

HINWEIS! Ist die Bibliothek auf Ihrem Rechner vorhanden besteht die Möglichkeit, die Beispiele als ZIP Datei zu speichern und zu entpacken. Die enthaltene PRO Datei ist im PlcControl zu öffnen. In der Regel ist der Pfad zu den Bibliotheken zu aktualisieren.

Im Hauptprogramm wird ein MC_AxUtiStandardInit_BkPlcMc Baustein aufgerufen. Die als Achs-Name und Datei-Pfad übergebenen Strings sind an die Verhältnisse auf Ihrem Rechner anzupassen. Die Dateien vom Typ DAT sind im hierdurch festgelegten Verzeichnis abzulegen. Es wird eine vorbereitete Scope-Einstellung als SCP Datei bereitgestellt, die alle wesentlichen Daten erfasst.

SPS Projekt	Projekt (ZIP)	Beschreibung
TcPlcMcEx_1.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599853451.zip	Demonstration der Bausteine MC AxUtiStandardInit BkPlcMc [▶ 171] , MC Power BkPlcMc [▶ 19] , MC AxStandardBody BkPlcMc [▶ 170] und MC AxAdsCommServer BkPlcMc [▶ 187] . Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_2.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599855627.zip	Demonstration der Benutzung von Arrays für Projekte mit mehreren Achsen. Fünf Achsen. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_3.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599857803.zip	Demonstration der Bausteine MC AxRtEncoder BkPlcMc [▶ 123] , MC AxRuntime BkPlcMc [▶ 155] , MC AxCtrlSlowDownOnPressure BkPlcMc [▶ 110] (Ablösender Druckregler), MC AxRtFinish BkPlcMc [▶ 164] und MC AxRtDrive BkPlcMc [▶ 114] . Demonstration der Nutzung von fCustomerData [▶ 88] für Projekt-Parameter. Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_5.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599859979.zip	Demonstration der Bausteine MC MoveAbsolute BkPlcMc [▶ 54] , MC MoveVelocity BkPlcMc [▶ 59] , MC MoveRelative BkPlcMc [▶ 57] und MC Stop BkPlcMc [▶ 60] . Der Arbeitszyklus wird durch Setzen der globalen Variablen bStart im PlcControl ausgelöst. Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_6.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599862155.zip	Demonstration des Generator-Typs iTcMc ProfileTimeRamp [▶ 160] . Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_7.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599864331.zip	Demonstration des Bausteins MC SetOverride BkPlcMc [▶ 31] . Über globale Variablen (bOverrideSinusoidal, fOverrideCycleTime, fOverrideMinValue, fOverrideMaxValue) kann der Verlauf, die Periodendauer und die Begrenzungen eines Signalerzeugers festgelegt werden, der den Override verändert. Für die Erzeugung des Overrides werden MC FunctionGeneratorFD BkPlcMc [▶ 147] , MC FunctionGeneratorTB BkPlcMc [▶ 148] und MC FunctionGeneratorSetFrg BkPlcMc [▶ 147] Bausteine eingesetzt. Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_8.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599866507.zip	Demonstration des Bausteins MC DigitalCamSwitch BkPlcMc [▶ 41] . Eine Achse. Bedienung über PlcControl und PlcMcManager.

SPS Projekt	Projekt (ZIP)	Beschreibung
TcPlcMcEx_9.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599868683.zip	Demonstration des Bausteins <u>MC_SetOverride_BkPlcMc</u> [▶ 31]. Gezeigt wird das Verhalten bei abrupten Änderungen usw. Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_10.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599870859.zip	Demonstration der Bausteine <u>MC_AxUtiAutoIdent_BkPlcMc</u> [▶ 181], <u>MC_AxRtFinishLinear_BkPlcMc</u> [▶ 165], <u>MC_LinTableExportToAsciiFile_BkPlcMc</u> , <u>MC_LinTableImportFromAsciiFile_BkPlcMc</u> , <u>MC_LinTableExportToBinFile_BkPlcMc</u> und <u>MC_LinTableImportFromBinFile_BkPlcMc</u> . Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_11.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599873035.zip	Demonstration der Bausteine <u>MC_Stop_BkPlcMc</u> [▶ 60], <u>MC_EmergencyStop_BkPlcMc</u> [▶ 44] und <u>MC_ImmediateStop_BkPlcMc</u> [▶ 53]. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_12.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599875211.zip	Demonstration des PLCopen Command Buffering. Bausteine <u>MC_MoveAbsolute_BkPlcMc</u> [▶ 54], <u>MC_MoveRelative_BkPlcMc</u> [▶ 57], <u>MC_Stop_BkPlcMc</u> [▶ 60] und <u>MC_AxRtCmdBufferExecute_BkPlcMc</u> . Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_13.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599877387.zip	Demonstration der Bausteine <u>MC_AxUtiSlewRateLimiter_BkPlcMc</u> [▶ 179], <u>MC_AxUtiPT1_BkPlcMc</u> [▶ 178] und <u>MC_AxUtiSlidingAverage_BkPlcMc</u> [▶ 180]. Keine Achsen. Bedienung über PlcMcManager.
TcPlcMcEx_14.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599879563.zip	Demonstration der Bausteine <u>MC_FunctionGeneratorTB_BkPlcMc</u> und <u>MC_FunctionGeneratorFD_BkPlcMc</u> . Keine Achsen. Bedienung über PlcControl.
TcPlcMcEx_15.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599881739.zip	Demonstration des Bausteins <u>MC_AxCtrlPressure_BkPlcMc</u> [▶ 106]. Eine Achsen. Bedienung über PlcControl.
TcPlcMcEx_16.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599883915.zip	Demonstration der Bausteine <u>MC_AxUtiStandardInit_BkPlcMc</u> [▶ 171], <u>MC_Power_BkPlcMc</u> [▶ 19], <u>MC_AxStandardBody_BkPlcMc</u> [▶ 170] und <u>MC_AxAdsPtrArrCommServer_BkPlcMc</u> [▶ 188]. Eine Achse. Bedienung über PlcControl und PlcMcManager.

SPS Projekt	Projekt (ZIP)	Beschreibung
TcPlcMcEx_18.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599886091.zip	Demonstration der Bausteine MC_AxRtCommandsLocked_BkPlcMc [▶ 177] . Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_100.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599888267.zip	Demonstration der Bausteine MC_GearIn_BkPlcMc [▶ 45] und MC_GearOut_BkPlcMc [▶ 49] . Fünf Achsen. Die Achse 5 ist eine virtuelle Achse (Mittelpunkt von Achse 1 und 2), die Achsen 1 und 2 werden über Getriebefunktionen an Achse 5 gekoppelt und entkoppelt. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_101.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599890443.zip	Demonstration der Bausteine MC_CamTableSelect_BkPlcMc [▶ 40] , MC_CamIn_BkPlcMc [▶ 37] , MC_CamOut_BkPlcMc [▶ 39] , MC_AxTableFromAsciiFile_BkPlcMc und MC_AxTableToAsciiFile_BkPlcMc . Fünf Achsen. Die Achse 5 ist eine virtuelle Achse (Mittelpunkt von Achse 1 und 2), die Achsen 1 und 2 werden über Camming Tabellen an Achse 5 gekoppelt und entkoppelt. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_103.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599892619.zip	Demonstration der Bausteine MC_GearInPos_BkPlcMc [▶ 47] und MC_GearOut_BkPlcMc [▶ 49] . Eine Achse. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_104.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599894795.zip	Demonstration einer Gleichlaufregelung für ein Zweiachs-Gantry unter Verwendung eines virtuellen Masters. Drei Achsen. Bedienung über PlcControl und PlcMcManager.
TcPlcMcEx_105.pro	http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/zip/1599896971.zip	Demonstration der Kennlinienermittlung für ein Zweiachs-Gantry mit den Bausteinen MC_AxUtiAutoIdent_BkPlcMc [▶ 181] und MC_AxUtiAutoldentSlave_BkPlcMc . Drei Achsen. Bedienung über PlcControl und PlcMcManager.

4.5 Der PlcMcManager

Der PlcMcManager unterstützt die Inbetriebnahme und den Test von Achsen, die mit Hilfe der Hydraulik-Bibliothek automatisiert werden. Er visualisiert den Ist-Zustand, ermöglicht den Zugriff auf Parameter und das Auslösen von Kommandos.

ⓘ HINWEIS! Der PlcMcManager ist nicht für die Bedienung von Maschinen und Anlagen vorgesehen. Er ersetzt keine Bedienoberfläche.

Sicherheitshinweis



Achtung

Unerwartetes Maschinenverhalten

Der PlcMcManager kann durch von ihm ausgelöste Kommandos die automatisch ablaufenden Aktionen und Reaktionen der Steuerungs-Software behindern oder in eine unerwartete oder unerwünschte Richtung beeinflussen. Dabei kann es zu unerwarteten und gefährlichen Bewegungen kommen.

Installation

Eine lizenzfreie Kopie des <http://infosys.beckhoff.com/content/1031/tcplcLibhydraulics30/Resources/exe/1599899147.exe> wird mit der Bibliothek bzw. der Dokumentation zur Verfügung gestellt. Wählen Sie einen geeigneten Pfad an und erstellen Sie anschließend auf dem Desktop des PCs eine Verknüpfung. Wird eine solche Verknüpfung nicht erstellt kann der PlcMcManager nur aus dem Explorer gestartet werden.

Ausführen des PlcMcManagers

Wurde das Tool auf dem PC gespeichert kann es durch Doppelklicken im Explorer oder der Verknüpfung auf dem Desktop gestartet werden.

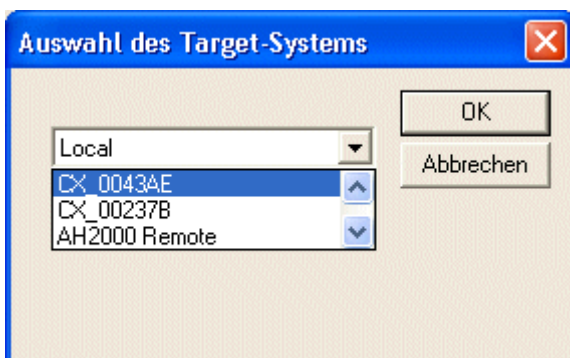
Anzeigen einer Parameter-Datei

In der Menü-Leiste finden Sie unter **Online** den **Offline-Dateimodus**. Hier wird ein Dialog zur Auswahl einer Achsparameter-Datei vom Typ DAT angeboten. Wird eine Datei geöffnet werden die Parameter der Achse soweit möglich dargestellt wie im Online-Betrieb.

HINWEIS! Es werden keine Istzustände von Achsen dargestellt und es können keinerlei Kommandos an Achsen ausgelöst werden. Dies gilt auch dann, wenn die angezeigten Parameter zu einer Achse gehören, auf die ein Zugriff möglich wäre.

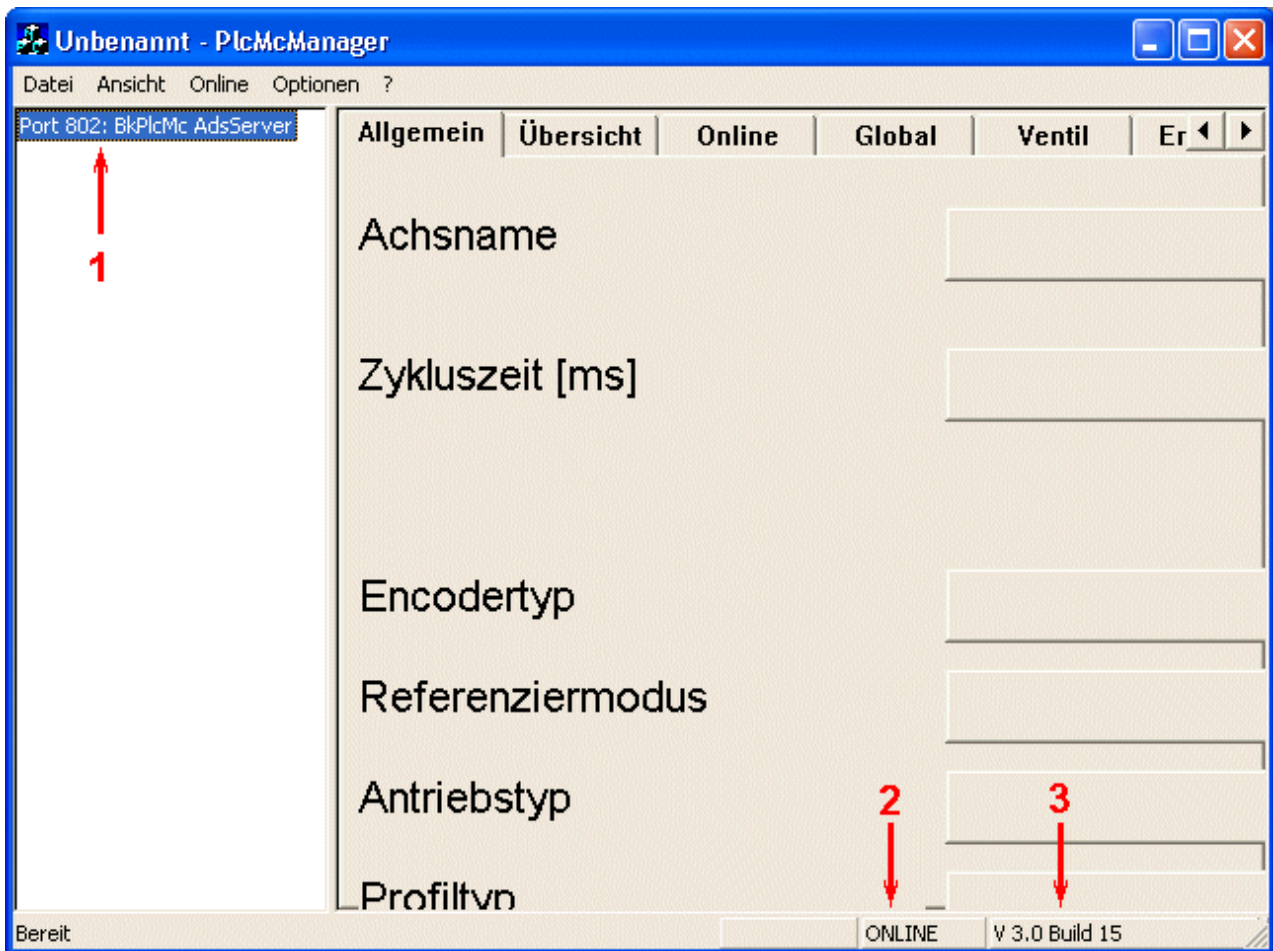
Online-Betrieb

Befindet sich das Laufzeit-System mit den Bibliotheks-Bausteinen nicht auf dem PC, auf dem der PlcMcManager ausgeführt wird ist zuerst das Zielsystem anzuwählen. In der Menü-Leiste finden Sie dafür unter **Online** den Dialog **Target**. Hier werden die Rechner aufgelistet, die im **TwinCAT System Service** auf dem Reiter **AMS Router** als **Remote Computer** eingetragen sind.



Durch die Auswahl eines **Remote Computer** wird die Kommunikation mit dem Laufzeitsystem automatisch aktiviert. Befindet sich das Laufzeit-System mit den Bibliotheks-Bausteinen auf dem PC, auf dem der PlcMcManager ausgeführt wird kann die Kommunikation mit dem Laufzeitsystem über die Menü-Leiste unter **Online** mit **Einloggen** aktiviert werden.

Dies kann vor allem bei Verwendung einer Netzwerkverbindung einige Sekunden dauern. Anschließend sollten die unten gezeigten Details erscheinen.



1. Hier werden der Port und der Server angezeigt, über den mit dem Laufzeitsystem kommuniziert wird.
2. Der ONLINE Modus wird angezeigt.
3. Die Versions-Information der von der PLC-Applikation verwendeten Bibliothek wird angezeigt.

Sollten diese Details auch nach einigen Sekunden nicht erscheinen ist keine Verbindung zustande gekommen. Dies kann eine Reihe von Ursachen haben:

- Es wurde kein Ziel-System angewählt obwohl die Applikation nicht auf demselben Rechner läuft der den PlcMcManager ausführt.
- Die Applikation läuft nicht auf dem angewählten Ziel-System.
- Es besteht keine Verbindung zum angewählten Ziel-System.
- Der PC auf dem der PlcMcManager ausgeführt wird hat keine Zugriffsrechte auf dem angewählten Ziel-System.
- Die PLC ist nicht gestartet.

Sollte zu diesem Zeitpunkt ein Dialog mit einer Fehlermeldung erscheinen ist die Verbindung zum Zielsystem gestört (Timeout / Zeitüberschreitung) oder der PlcMcManager und die in der Applikation verwendete Bibliothek sind nicht kompatibel. Letzteres ist in der Regel darauf zurück zu führen, dass eine neue Bibliotheks-Version verwendet wird ohne dass der PlcMcManager aktualisiert wurde.

Durch Doppelklicken des auf der linken Seite angezeigten Servers erscheinen die Achsen der Applikation als Liste. Durch Anklicken einer Achse wird diese angewählt, ihr Status wird zyklisch aktualisiert und ihre Parameter sind zugreifbar. Sollte aus irgendeinem Grund die Kommunikation abbrechen kann durch Anklicken einer Achse ein Neustart der Kommunikation ausgelöst werden.

4.6 Setup

4.6.1 Achs-Inbetriebnahme (ab V3.0)

Die folgende Achs-Inbetriebnahme beschreibt, wie ein großer Teil an Achsen mit dieser Bibliothek in Betrieb genommen werden kann. Bei Achsen mit speziellen Sonderventilen wie sie in der Kunststofftechnik bei Einspritzachsen vorzufinden sind, sollte nicht nach dieser Inbetriebnahme vorgegangen werden, um unvorhersehbare Effekte zu vermeiden.

Grundsätzlich teilt sich die Inbetriebnahme in drei Bereiche auf:

- Allgemeine Sicherheitshinweise
- Voreinstellungen bevor die Achse mit Energie versorgt wird
- Ermittlung der Bezugs- und Regelparameter

Gerade der Bereich "Voreinstellung" benötigt genauere Informationen zum Ventil sowie Zylinder und dem Aufbau der Hydraulik. Aus diesem Grund sollten vor der Inbetriebnahme Datenblätter zu Ventil und Zylinder vorliegen und der Hydraulikplan der Anlage verfügbar sein.

Allgemeines

Während einer Inbetriebnahme sind eine Reihe von [Regeln \[▶ 242\]](#) und Vorgaben zu beachten.

Voreinstellungen

Bevor die Anlage mit Energie versorgt werden darf, müssen einige Voreinstellungen im "PLCManager" vorgenommen werden. Die folgenden Punkte beschreiben jeweils einen Reiter im PlcMc-Manager, welcher jeweils nacheinander abgearbeitet werden muss:

- [Allgemein \[▶ 242\]](#)
- [Encoder \[▶ 244\]](#)
- [Ventil \[▶ 255\]](#)
- [Regler \[▶ 251\]](#)
- [Monitor \[▶ 250\]](#)
- [Global \[▶ 247\]](#)
- [Status \[▶ 253\]](#)

Wurden alle oben erklärten Punkte abgearbeitet, kann unter der Berücksichtigung von "Allgemeines" die Anlage mit Energie versorgt werden. Im vorherigen Abschnitt wurden alle Vorbereitungen getroffen um die Achse im nächsten Schritt mit Energie zu versorgen.


Inbetriebnahme

Beim Zuschalten von Energie auf die Achse darf sich die Achse außer einer kleinen Drift kaum bewegen. Um die Inbetriebnahme erfolgreich abschließen zu können müssen anschließend die weiteren Punkte abgearbeitet werden:

- [Encoder-Einstellungen \[▶ 258\]](#)
- [Sollwertgenerator \[▶ 270\]](#)
- [Ausgabeeanpassung \[▶ 265\]](#)
- [Regloptimierung \[▶ 280\]](#)

4.6.2 Achs-Inbetriebnahme: Allgemeines

Sicherheitshinweis

 Achtung	<p>Unerwartete Achsbewegungen</p> <p>Bei der Inbetriebnahme von Achsen ist die von der Maschine ausgehende Gefahr um ein Vielfaches höher als im regulären Betrieb. Durch nicht oder falsch eingestellte Parameter, unterbrochene Kontrollstrukturen oder Wirkungsumkehr durch Verpolung in Software, Elektrik, Hydraulik, Sensorik, Aktuatorik oder Mechanik kann es jederzeit zu Bewegungen in unerwarteter Richtung, in nicht vorgesehenen Bereichen oder mit unerwarteter Beschleunigung oder Geschwindigkeit kommen.</p>
-----------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Die folgenden Punkte sind unbedingt zu beachten:

- Beachten Sie die Hinweise zu dieser Dokumentation.
- Weisen Sie alle Personen in der Umgebung der Maschine auf die besondere Gefahrenlage hin.
- Stellen Sie sicher, dass keine Personen die Maschine betreten oder sich ihr nähern.
- Entfernen Sie vor allem zu Beginn der Inbetriebnahme alle nicht unmittelbar erforderliche Einrichtungen oder Werkzeuge.
- Falls möglich sollten Bereiche durch mechanische Sperren (Block) gesichert werden, in denen die Gefahr einer Beschädigung besteht.
- Stellen Sie sicher, dass keine anderen als die jeweils in Betrieb genommenen Maschinenteile aktiv werden.
- Falls möglich sollten Sie abhängig von der getesteten Funktion den Betriebsdruck absenken.
- Weisen Sie das gesamte eingesetzte Personal darauf hin, dass keinerlei Veränderungen oder sonstige Arbeiten (elektrisch, mechanisch, hydraulisch) an der Maschine ohne **detaillierte** Abklärung mit dem Inbetriebnehmer durchgeführt werden dürfen.
- Es sind über die oben beispielhaft genannten Maßnahmen hinaus alle geltenden Vorschriften zur Personen- und Maschinensicherheit zu beachten.

....

4.6.3 Voreinstellungen

Bevor die Achse in Betrieb genommen werden kann, müssen einige Voreinstellungen für diese Achse vorgenommen werden. Die folgenden Punkte sollten beachtet werden:

- Der MC_AxUtiStandardInit_BkPlcMc Baustein
- Der Reiter Allgemein im PlcMcManager

Der MC_AxUtiStandardInit_BkPlcMc Baustein

Für jede Achse ist ein `MC_AxUtiStandardInit_BkPlcMc` [▶ 171] Baustein aufzurufen. Er übernimmt nicht nur die Verwaltung und Überwachung der Pointer auf die E/A-Interfaces und diverse optionale Achsbestandteile, sondern dient auch der Festlegung des Achsnamens und des Dateipfades.

```
FUNCTION_BLOCK MC_AxUtiStandardInit_BkPlcMc
VAR_INPUT
    AxisName:      STRING(255);
    PathName:      STRING(255);
    .....
END_VAR
```

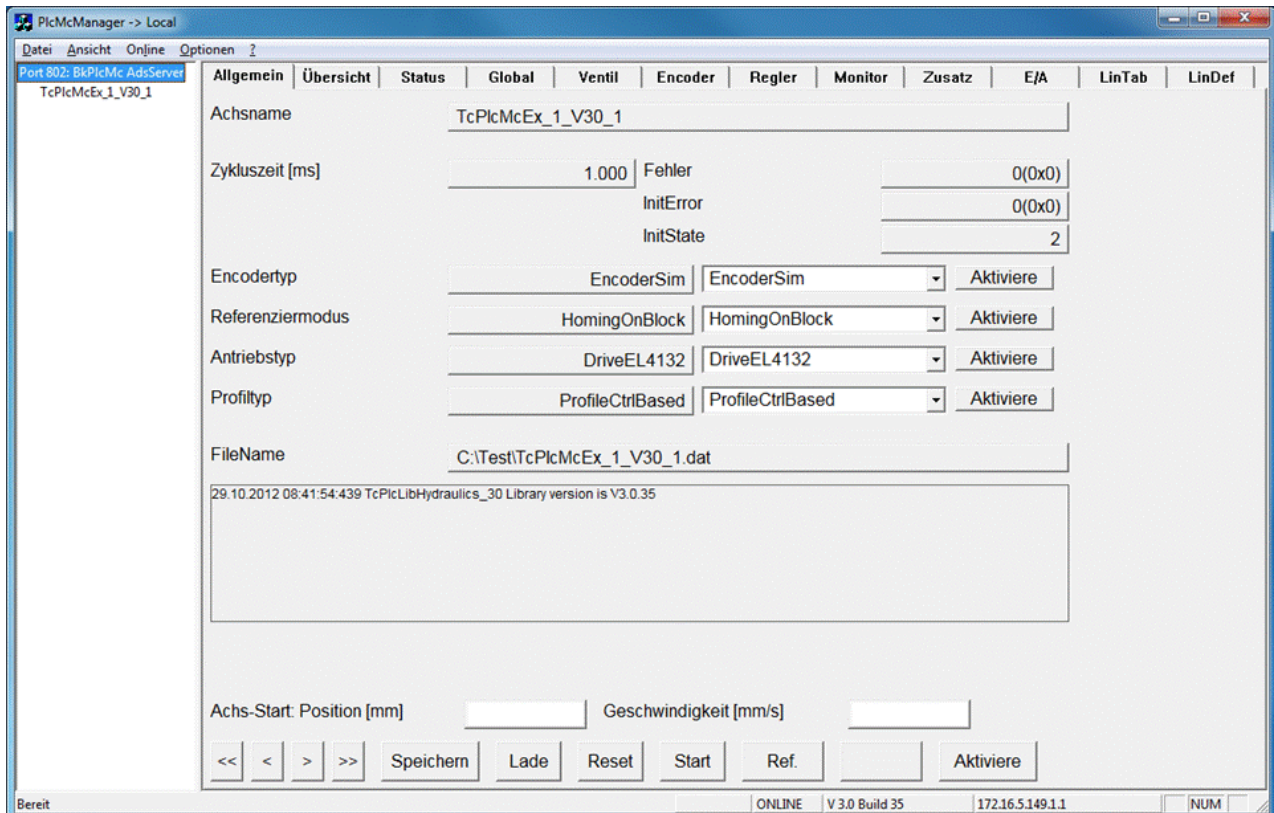
Durch die Variable **Pathname** wird der Pfad innerhalb des Dateisystems festgelegt. Hierüber kann auf alle Dateien dieser Achse zugegriffen werden. Die Verwendung von Konstanten ist hier dringend zu empfehlen. Das nachstehende Beispiel zeigt den Aufbau eines Pfadnamens für einen PC unter Windows XP, XPe oder 2000. Dabei ist der Backslash \ am Ende zwingend erforderlich.

```
VAR_GLOBAL CONSTANT
    cnst_ParamFilePath: STRING(255):='C:\Projekt\Daten\';
END_VAR
```

Wenn zum Beispiel auf CX-Rechnern ein Windows CE genutzt wird, ist ein Pfadname zu verwenden, wie er nachstehend gezeigt wird.

```
VAR_GLOBAL CONSTANT
    cnst_ParamFilePath: STRING(255):='Hard Disk\Daten\';
END_VAR
```

Der PlcMc-Manager Reiter "Allgemein"



Achsnamen:

Name der Achse.

Zykluszeit:

Zykluszeit für die Regelung einer Achse (minimal 100us möglich).

Fehler:

Zeigt den aktuellen Fehler dieser Achse an. Unter "Error-Codes" kann die Fehlerbeschreibung [► 222] zu dieser Fehlernummer ermittelt werden.

InitError:

Fehler bei der Initialisierung der Achse.

InitState:

Zeigt den Status der Initialisierung an. Eine zwei signalisiert eine erfolgreiche Initialisierung. Eine minus zwei signalisiert eine fehlgeschlagene Initialisierung. Mögliche Ursachen sind:

- Falscher Datenpfad
- Dateiname nicht korrekt

Um eine fehlende Datei mit Standardwerten unter dem eingetragenen Pfad anzulegen, muss der Button "Speichern" und anschließend der Button "Lade" aktiviert werden. Daraufhin sollte "InitState" zwei sein.

Encodertyp:

Ein Encoder liefert in Verbindung mit einer Eingangsbaugruppe die Istposition einer Achse in einer diskretisierten Form. Diese Information muss aus den Daten der Baugruppe ermittelt und in eine Ortsangabe in üblichen Einheiten (in der Regel in mm) umgerechnet werden. Gleichzeitig sind Status-Signale des Encoders oder der Baugruppe zu beobachten und (soweit technisch möglich) die Verbindung zwischen der Baugruppe und der Steuerung zu überwachen. All diese Aufgaben werden vom [MC_AxRtEncoder_BkPlcMc \[▶ 123\]](#) Baustein übernommen. Damit der von der PLC-Applikation aufgerufene [MC_AxRtEncoder_BkPlcMc\(\)](#) Baustein den zur Baugruppe passenden Unterbaustein aufrufen kann, muss der Encodertyp ausgewählt und durch den "Aktiviere"-Button übernommen werden. Der aktuell verwendete Encodertyp wird links vom Eingabefeld angezeigt.

Referenziertyp:

Bei Inkrementalencodern muss eine [Referenzierung \[▶ 247\]](#) durchgeführt werden. Umfang und Zusammensetzung der Referenzierparameter (Homing-Parameter) hängen vom ausgewählten Referenziermodus ab.

Antriebstyp:

Die Verbindung zum Antrieb wird über eine Schnittstelle hergestellt, die aus einer Hardware (Klemme, Feldbusinterface) und einer dazu passenden Software besteht. Die Bibliothek stellt den Baustein [MC_AxRtDrive_BkPlcMc \[▶ 114\]](#) zur Verfügung. Damit dieser den passenden Unterbaustein aufrufen kann, muss der passende Antriebstyp ausgewählt und mit dem "Aktiviere"-Button rechts neben dem Eingabefeld verwendet werden. Der aktuell verwendete Antriebstyp wird links vom Eingabefeld angezeigt.

Profiltyp:

Die aktive Positionierung einer Achse erfolgt durch einen Profil-Generator. In der Bibliothek wird diese Aufgabe durch den [MC_AxRuntime_BkPlcMc \[▶ 155\]](#) Baustein übernommen. Damit die Achse das gewünschte Profil erzeugt, muss hier der passende Generatortyp ausgewählt werden. Das Standardprofil ist "ProfileCtrlBased".

FileName:

Pfad inklusive Dateinamen.

Loggerausgabe:

Informiert über den Zustand der Achse. Es ist möglich über den Baustein [MC_AxUtiStandardInit_BkPlcMc \[▶ 171\]](#) den Loggerlevel zu verändern.

Speichern:

Speichert alle aktuellen Einstellungen unter "FileName".

Lade:

Laden der unter "FileName" gespeicherten Einstellungen.

Reset:

Zurücksetzen der Achse bei einem Fehler.

Ref.:

Auslösen einer Referenzierung mit Default-Werten (davon wird an dieser Stelle abgeraten).

Aktiviere:

Die eingestellten Änderungen werden in die PLC übernommen.

<<|<|>|>>:

Handbewegungen zum Verfahren der Achse. Die gleiche Funktion kann über die Tasten F1 bis F4 ausgeführt werden.

4.6.3.1 Voreinstellung Encoder

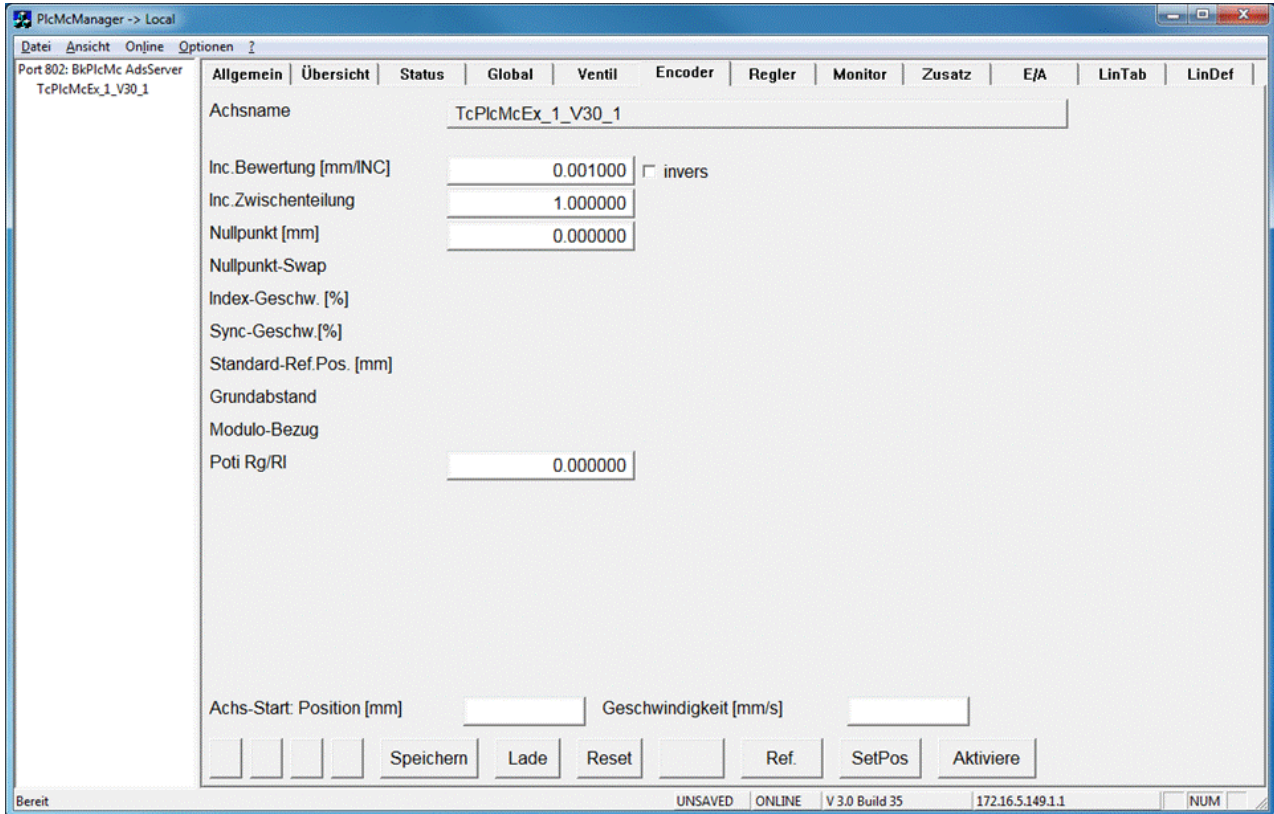
Abhängig vom ausgewählten Encodertyp muss einer der folgenden Voreinstellungen durchgeführt werden:

- Analoge Encoder
- Feldbusencoder
- Antriebe mit integrierter Positionsrückmeldung
- Nockenschaltwerke

- Encoder mit speziellen Schnittstellen

Analoge Encoder

Zu dieser Gruppe von Encodern gehören alle Messsysteme, die ein analoges Signal (z. B. 0..10V) für die Steuerung bereitstellen.



Inc. Bewertung [mm/INC] & Inc. Zwischenteilung:

In Inc. Bewertung [mm] Eingabe der gesamten Messlänge des Encoders und in Inc. Zwischenteilung die Anzahl der Inkremente des Wertebereichs der Eingangs-Elektronik (in der Regel 32767). Sobald die Achse im weiteren Verlauf der Inbetriebnahme kontrolliert gefahren werden kann, sind die ermittelten Werte zu verifizieren.

invers:

Zunächst ist festzulegen, welche Bewegungsrichtung der Achse als "positiv", "steigend" oder "aufwärts" zu betrachten ist. Bei dieser Festlegung ist die physikalische Zählrichtung des Encoders ohne jede Bedeutung. Allerdings muss festgestellt werden, ob der Encoder zur festgelegten Zählrichtung invers arbeitet. Zur Überprüfung kann dies aus dem Schaltplan ermittelt werden oder es besteht die Möglichkeit den Encoder manuell (ohne Antriebsenergie) zu bewegen.



Achtung

Achsbewegung vermeiden

Es wird nicht empfohlen die Achsen in diesem Zustand zu bewegen, da grundlegende Achseinstellungen noch nicht vorgenommen worden sind.

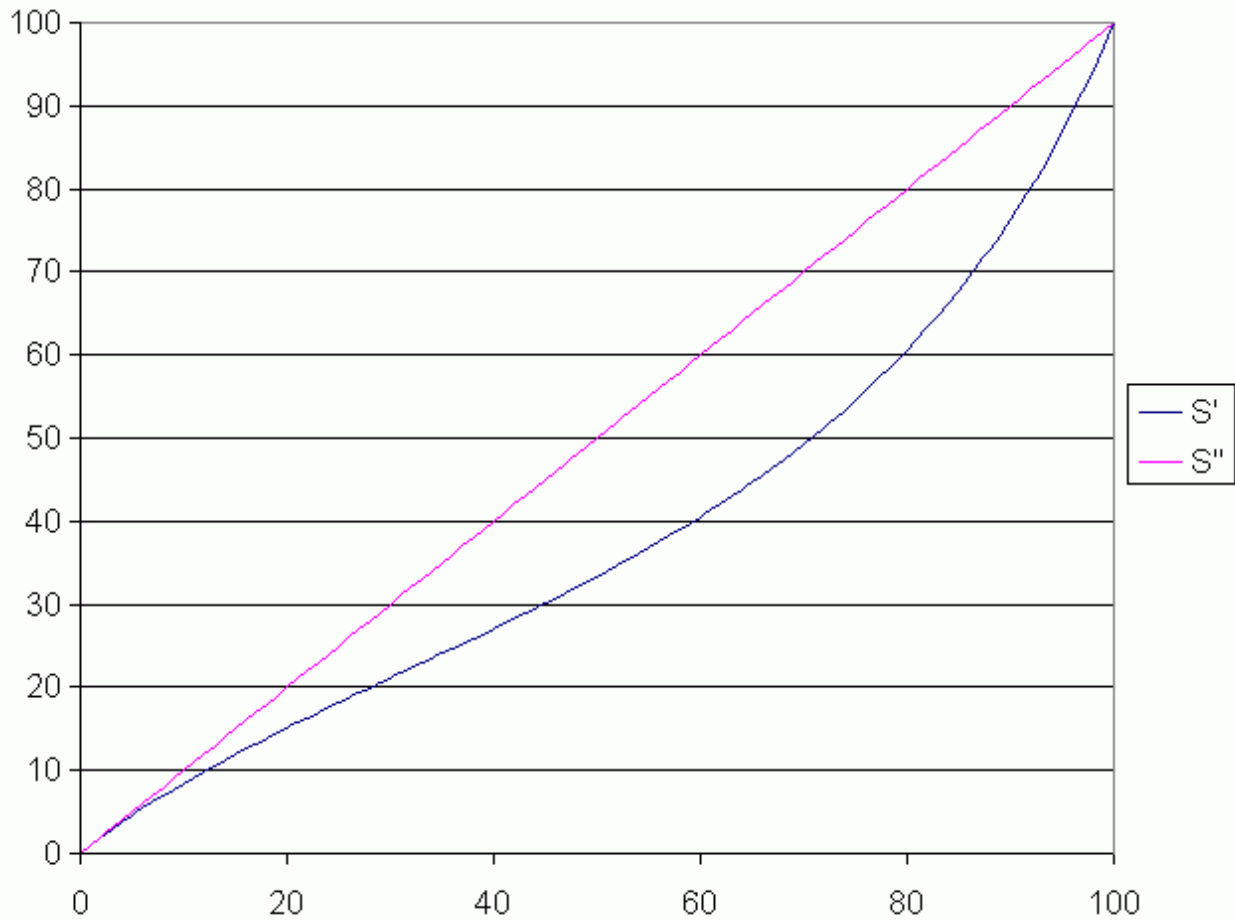
Stimmt die Zählrichtung der Istposition nicht mit der Richtung der mechanischen Bewegung überein ist der Haken bei "invers" zu setzen. Nach dem Betätigen des "Aktiviere"-Button wird in ST TcHydAxParam ▶ 88] die Variable bEnc_Reversed TRUE.

Nullpunkt:

Bei Absolutencodern liegt der Nullpunkt des Encoders an einem Ort, der durch die Konstruktion des Encoders und seine Anbringung in der Maschine festgelegt wird. Um den Nullpunkt der Achse frei wählen zu können, wird ein Positionsoffset addiert. Diese Einstellung wird erst vorgenommen, wenn die Achse verfahren werden kann.

Poti Rg/RI:

Wenn als Weggeber ein Potentiometer verwendet wird und dieses ohne eingebauten oder zwischengeschalteten Verstärker direkt am Analogeingang betrieben wird, ist die ohmsche Belastung durch den Eingangswiderstand zu berücksichtigen. Bleibt der Einfluss unkompensiert ist ein nichtlinear wegabhängiger Fehler feststellbar. Die nachstehende Grafik stellt diesen Effekt überhöht dar. Bei typischen Werten für den Widerstand des Potentiometers und des Analogeingangs sind absolute Maximalfehler zwischen 1 und 5% des Fahrweges zu erwarten.



Dieser Fehler kann kompensiert werden, indem das Verhältnis des Potentiometer-Gesamtwiderstands zum Lastwiderstand eingegeben wird.

Feldbusencoder

In Vorbereitung

Antriebe mit integrierter Positionsrückmeldung

In Vorbereitung

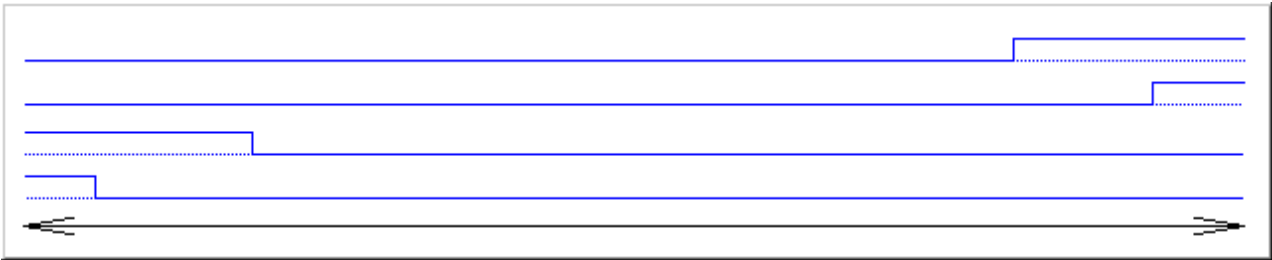
Encoder mit speziellen Schnittstellen

In Vorbereitung

Digitale Positionsnocken

Digitale Positionsnocken

In einigen Anwendungen wird kein Encoder zur Positionsermittlung eingesetzt, sondern ein Satz von üblicherweise vier digitalen Sensoren. Um diese auszuwerten, ist als [Encodertyp \[▶ 242\]](#) `iTcMc_EncoderDigCam` vorzugeben. Die Nocken müssen, bezogen auf den Fahrweg, das unten stehende Verhalten zeigen. Insbesondere dürfen die Nocken nicht überfahrbar sein. Andernfalls kann es insbesondere nach dem Einschalten zur Ausgabe von zu hohen Stellwerten und somit zum Überfahren des Ziels kommen.



HINWEIS! Diese Art der Positionsermittlung ist nur sinnvoll kombinierbar mit einem Generator vom Typ `iTcMc_ProfileTimeRamp`. [\[▶ 160\]](#)

4.6.3.2 Referenziertyp

Bei Incrementalencodern muss eine Referenzierung durchgeführt werden. Umfang und Zusammensetzung der Referenzierparameter (Homing-Parameter) hängt vom gewählten Referenziermodus ab:

HomingOnBlock: Die Achse wird mit `ST_TcHydAxParam [▶ 88].fEnc_RefIndexVelo` in der durch `ST_TcHydAxParam [▶ 88].bEnc_RefIndexPositive` festgelegten Richtung bewegt. Wird für eine Zeitdauer von 2 Sekunden keine Bewegung festgestellt gilt der Festanschlag (Block) als erreicht. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

HomingOnExec: Die steigende Flanke am Homing-Baustein setzt den Istwert der Achse unmittelbar auf den Wert der Referenzposition.

HomingOnIndex: Die Achse wird mit `ST_TcHydAxParam [▶ 88].fEnc_RefIndexVelo` in der durch `ST_TcHydAxParam [▶ 88].bEnc_RefIndexPositive` festgelegten Richtung bewegt. Wird in `ST_TcHydAxRtData.nDeCtrlDWord` die Referenznocke (Bit 5, `dwTcHydDcDwRefIndex`) erkannt stoppt die Achse. Anschließend wird sie mit `ST_TcHydAxParam [▶ 88].fEnc_RefSyncVelo` in der durch `ST_TcHydAxParam [▶ 88].bEnc_RefSyncPositive` festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

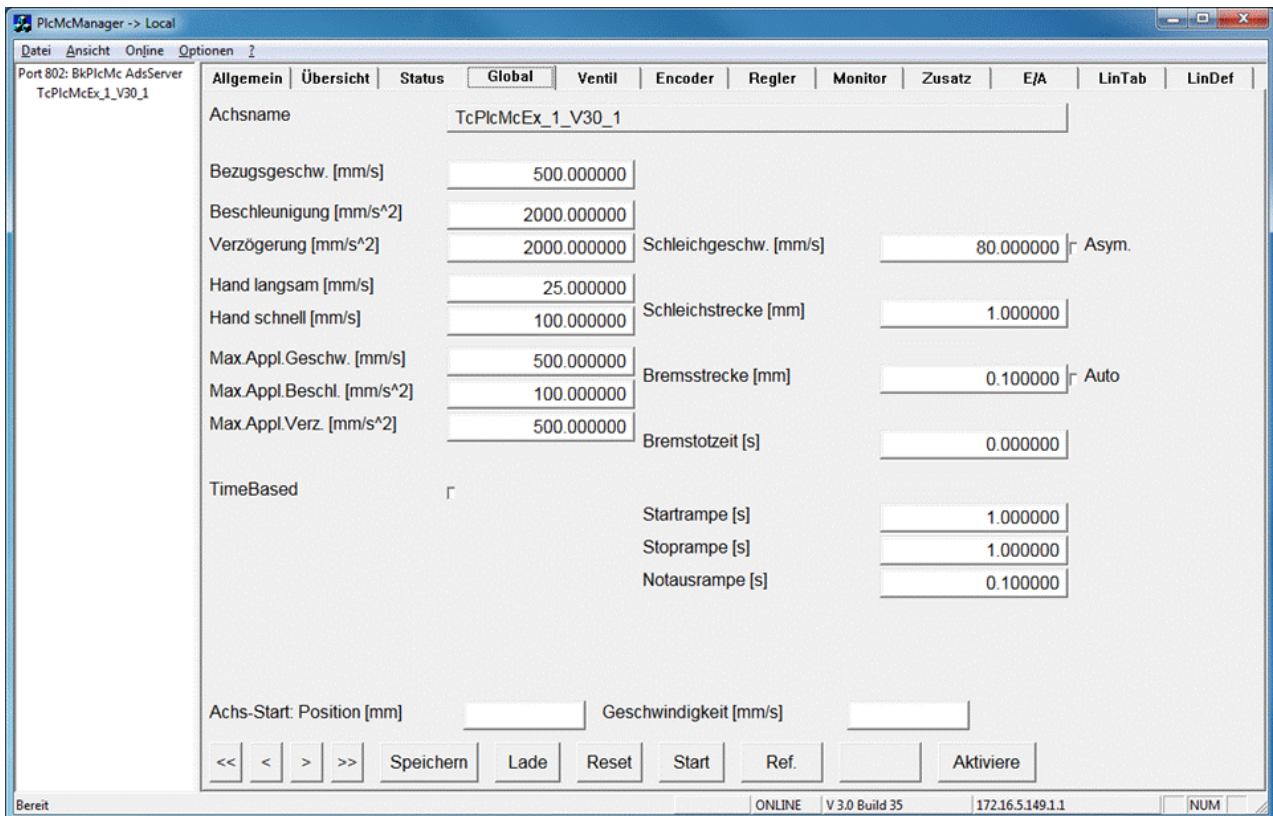
HomingOnMultiSync: reserviert.

HomingOnSync: Die Achse wird mit `ST_TcHydAxParam [▶ 88].fEnc_RefIndexVelo` in der durch `ST_TcHydAxParam [▶ 88].bEnc_RefIndexPositive` festgelegten Richtung bewegt. Wird in `ST_TcHydAxRtData.nDeCtrlDWord` die Referenznocke (Bit 5, `dwTcHydDcDwRefIndex`) erkannt stoppt die Achse. Anschließend wird sie mit `ST_TcHydAxParam [▶ 88].fEnc_RefSyncVelo` in der durch `ST_TcHydAxParam [▶ 88].bEnc_RefSyncPositive` festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Dann wird das Hardware-Latch des Encoders aktiviert und die Achse so lange weiter bewegt, bis das Latch gültig wird. Nach dem Stoppen der Achse wird der Istwert der Achse auf einen Wert gesetzt, der aus der Referenzposition und der seit dem Sync-Puls des Encoders zurückgelegten Strecke errechnet wird.

Der ausgewählte Typ wird in `ST_TcHydAxParam [▶ 88].nEnc_HomingType` aus `E_TcMcHomingType [▶ 73]` eingestellt.

4.6.3.3 Global

Um die Achse beim ersten Einschalten halbwegs gut zu kontrollieren sollten die Bezugswerte der Achsen angemessen gewählt werden.



Bezugsgeschw.: Für einige Antriebs-Typen (AX2000B750 ▶ 116], AX5000B110 ▶ 116]) wird die **Bezugsgeschwindigkeit** aus Werten errechnet, die aus dem Gerät ausgelesen werden. Dazu muss im "Encoder" der Umdrehungsvorschub in die Achse richtig eingestellt worden sein. Für hydraulik-Achsen kann hier als erste Näherung ein Wert vorgegeben werden, der aus Datenblattangaben des Ventils und des Zylinders errechnet werden.

$$V_{ref} := 16666 * Q_{nom} / A_{min}$$

mit V_{ref} := Bezugsgeschwindigkeit in mm pro Sekunde; Q_{nom} := nomineller Volumenstrom des Ventils in Liter pro Minute; A_{min} := kleinere der beiden Wirkungsflächen in mm^2 . Die Bezugsgeschwindigkeit gibt die gewünschte Geschwindigkeit bei Vollaussteuerung wieder.

Beschleunigung: Hier kann zumindest vorerst ein sehr großer Wert wie z.B. 20000.0 vorgegeben werden.

Verzögerung: Hier kann zumindest vorerst ein sehr großer Wert wie z.B. 20000.0 vorgegeben werden.

Hand schnell: Hier sollte ein Wert vorgegeben werden, der eine ausreichende Reaktionsmöglichkeit sicher stellt, jedoch nicht größer als 100 mm/s ist. Dieser Wert wird im weiteren Verlauf nachgestellt.

Hand langsam: Hier sollte ein Wert vorgegeben werden, der im Verhältnis zwei bis fünf Mal kleiner als **Hand schnell** ist.

Max. Appl. Geschw.: Geschwindigkeit die in der Applikation maximal erlaubt ist. Darf nicht größer als die Referenzgeschwindigkeit sein.

Max Appl. Beschl.: Muss immer kleiner als der **Beschleunigung** sein. In diesem Schritt 2* **Bezugsgeschw.**

Max. appl. Verz.: Muss immer kleiner als der **Verzögerung** sein. In diesem Schritt 1* **Bezugsgeschw.**

TimeBased: Aktiviert eine zeitgesteuerte Profilgenerierung. Der Großteil an hydraulischen Achsen muss genau positionieren aber nicht mit genau der vorgegebenen Sollgeschwindigkeit verfahren werden. Nur wenn die vorgegebene Geschwindigkeit auch von der Achse gefahren werden soll, muss während eines Positionswechsels ständig der Lageregler aktiv sein und somit TimeBased aktiviert werden.



Hinweis

Die zeitgesteuerte Positionierung kann bei schwach gedämpften Achsen zu Instabilitäten führen.

Schleichgeschw.: Geschwindigkeit von Abschnitt 3 der Profilerzeugung. Während der Bremsphase wird auf diese Geschwindigkeit abgebremst. Durch die häufig großen Massen wird erst über die Schleichgeschwindigkeit ein exaktes Positionieren der Achse möglich. Sollte auf 10% der Bezugsgeschwindigkeit gestellt werden.

Schlechstrecke: Für diese Strecke gibt der Profilergenerator die Geschwindigkeit **Schleichgeschw** aus. Sollte auf 10 mm gestellt werden.

Bremsstrecke: Muss kleiner als die Schleichstrecke eingestellt sein, da die Schleichstrecke die Bremsstrecke mit enthält. Während der Bremsstrecke wird der Lageregler aktiv. Sollte in diesem Schritt auf wenige mm eingestellt werden.

Bremstotzeit: Bei einigen Achsen schwankt in der Schleichphase die Istgeschwindigkeit stark. Wenn dies nicht durch die Lageregelung ausgeglichen wird kann dadurch die Genauigkeit im Ziel beeinträchtigt sein. In diesem Fall kann ein Teil der Bremsstrecke in Form einer Zeit dargestellt werden. Diese Zeit wird mit der Istgeschwindigkeit verrechnet und als zusätzliche dynamische Bremsstrecke addiert. **HINWEIS! Ein zu großer Wert kann zu einem Anhalten vor dem Ziel führen. Es sollte höchstens die halbe Bremsstrecke in dieser Weise erzeugt werden.**

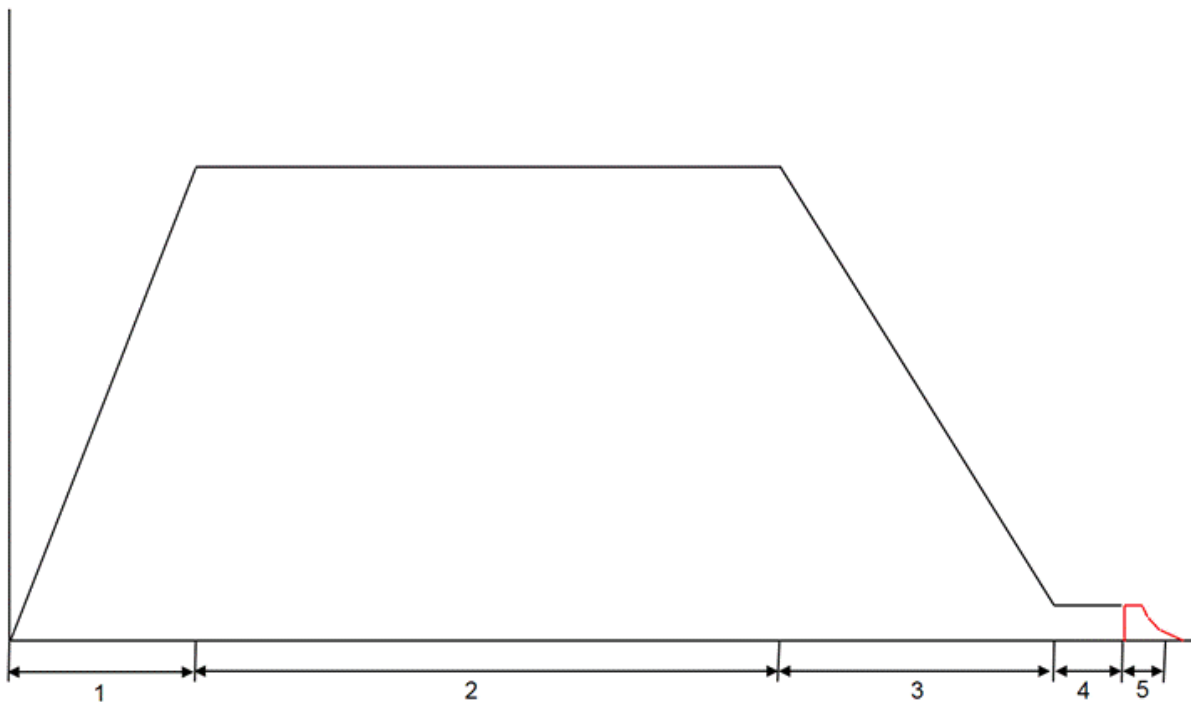
Startrampe: In Vorbereitung

Stoprampe: In Vorbereitung

Notausrampe: In Vorbereitung

Die Weggesteuerte Achsgenerierung:

Die Profilerzeugung einer Weggesteuerten Achse teilt sich in drei Teilabschnitte:



1. Beschleunigungsphase:

In dieser Phase wird der Stellwert als zeitgeführte Rampe aufgebaut mit $V_{Stell}=A*t$. Die Phase endet wenn die vorgegebene Fahrgeschwindigkeit oder der Beginn der Bremsphase erreicht wurde. Bei Zeitsteuerung wird die gleiche Formel benutzt. Allerdings wird bei der Wegsteuerung keine Sollposition festgelegt und keine Lageregelung vorgenommen.

2. Konstantgeschwindigkeit:

In dieser Phase wird ein konstanter Stellwert in Höhe der geforderten Fahrgeschwindigkeit ausgegeben mit $V_{Stell}=V_{fahr}$. Die Phase endet wenn der Beginn der Bremsphase erreicht wurde. Bei Zeitsteuerung wird die gleiche Formel benutzt. Allerdings wird bei der Wegsteuerung keine Sollposition festgelegt und keine Lageregelung vorgenommen.

3. Bremsphase:

In dieser Phase wird ein Stellwert in Abhängigkeit des Restwegs ausgegeben mit $V_{Stell}=\sqrt{2*D*s}$. Die Phase endet wenn der Beginn der Schleichphase erreicht wurde. Diese Phase beeinflusst wesentlich die Genauigkeit und das Verhalten der Achse im Ziel.

4. Schleichphase:

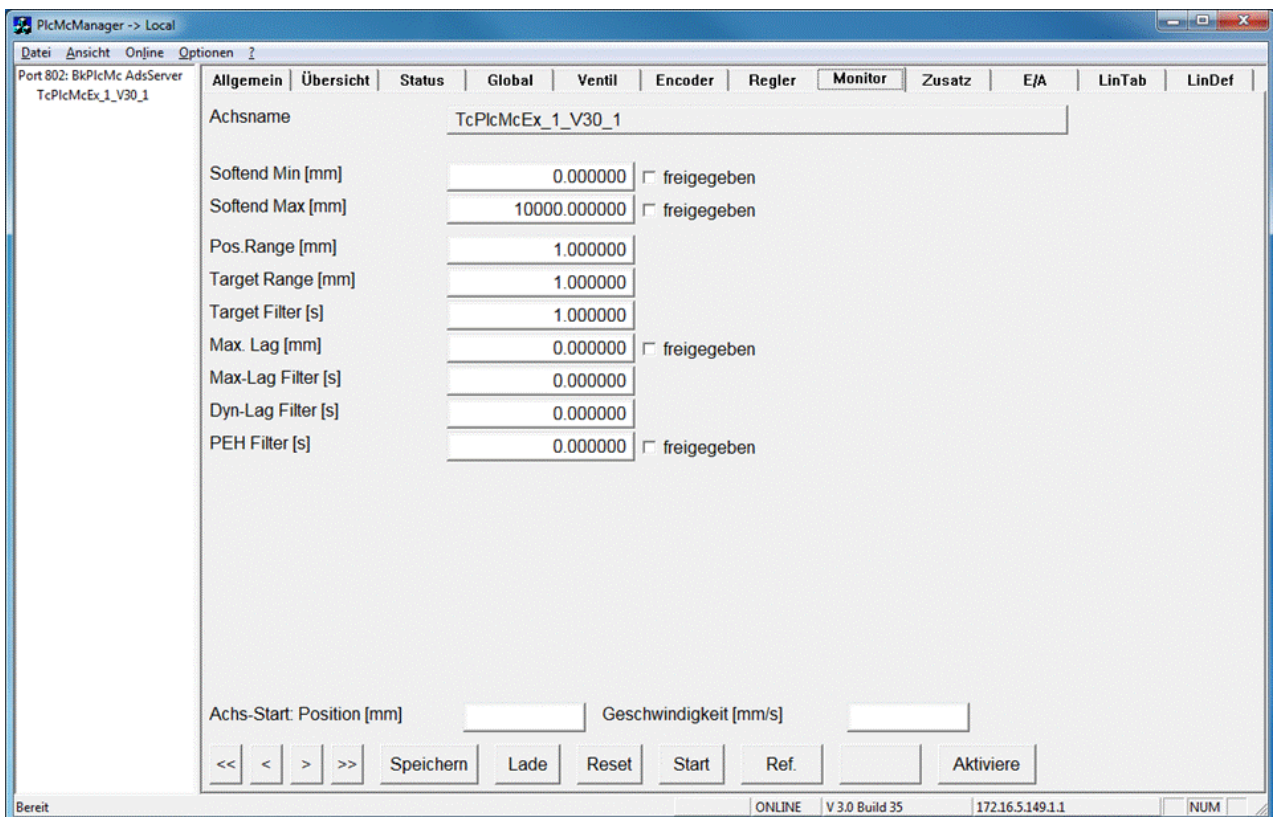
In dieser Phase wird ein konstanter durch Achsparameter festgelegter Stellwert ausgegeben mit $V_{Stell}=V_{creep}$. Die Phase endet mit dem Zielübergang. Durch einen nichtaktiven Lageregler folgt die Achse dem Profil, jedoch mit einem Schleppabstand. Um den Schleppabstand vor der Positionierung gering zu halten, ist die Schleichphase erforderlich. Die Länge der Schleichphase hängt davon ab wie schnell die Achse der Schleichgeschwindigkeit folgt.

5. Zielübergang:

Das Verhalten in dieser Phase ist mit dem Ruheverhalten der Achse identisch mit $V_{ctrl}=kP*s$. Die Phase endet mit dem Erreichen des Zielfensters. Anschließend wird das Ruheverhalten fortgesetzt bis ein neuer Auftrag gegeben wird.

4.6.3.4 Monitor

Die Parameter auf dem Monitor-Reiter des PlcMcManagers dienen zur Konfiguration verschiedener Überwachungsfunktionen.



Software-Endschalter

Software-Endschalter

Softend Min: Unterer Softwareendschalter zur Begrenzung des Fahrwegs. Über "Freigabe" kann dieser Endschalter aktiviert werden, wodurch kein Baustein eine Zielposition akzeptiert, die sich jenseits der Endlage befindet. Bausteine ohne definierte Zielposition (MC_MoveVelocity_BkPlcMc, NC_MoveJoysticked_BkPlcMc usw.) werden die in Bewegungsrichtung liegende aktive Endlage als Zielposition verwenden. Ist dies nicht gewünscht ist der Endschalter zu deaktivieren.

Softend Max: Oberer Softwareendschalter zur Begrenzung des Fahrwegs. Über "Freigabe" kann dieser Endschalter aktiviert werden, wodurch kein Baustein eine Zielposition akzeptiert, die sich jenseits der Endlage befindet. Bausteine ohne definierte Zielposition (MC_MoveVelocity_BkPlcMc, NC_MoveJoysticked_BkPlcMc usw.) werden die in Bewegungsrichtung liegende aktive Endlage als Zielposition verwenden. Ist dies nicht gewünscht ist der Endschalter zu deaktivieren.

Zielfenster-Überwachung

Zielfenster-Überwachung

Pos Range: Softwarenocke zur Erkennung, dass die Maschine ihr Ziel fast erreicht hat, um weitere Prozesse zu starten. Der PosRange sollte immer größer als der TargetRange gewählt werden. Ist diese Softwarenocke für den vorliegenden Prozess nicht erforderlich, kann diese groß gewählt werden.

Target Range: die Istposition muss für **Target Filter** ununterbrochen kleiner oder gleich diesem Parameter sein. Beide Werte sollten der Applikation angemessene Werte übergeben bekommen. Die Fertigmeldung des Fahrweggenerators der Achse erfolgt erst, wenn sich die Achse im PosRange und im TargetRange für mindestens TargetFilter und in der BrakeDistance befindet. Anschließend erfolgt die Lageregelung.

MAX Lag: Wenn der Schleppabstand eine Schwelle überschreitet kann ein Achs-Fehler ausgelöst werden. Dieser Parameter stellt den statischen (von der Geschwindigkeit unabhängigen) Teil der Schwelle dar.

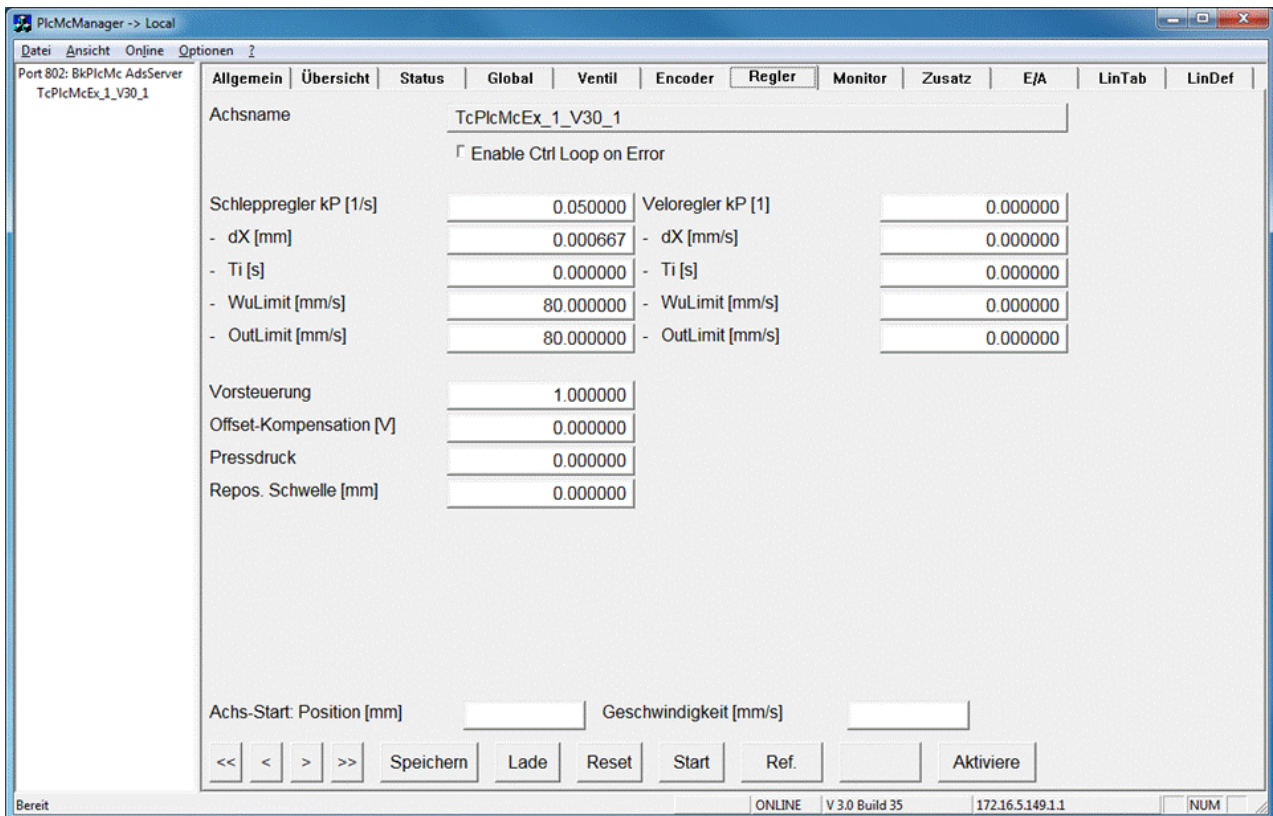
Max-Lag Filter: Erst wenn der Schleppabstand für die hier eingestellt Zeit ununterbrochen die Schwelle überschreitet wird ein Fehler ausgelöst.

Dyn-Lag Filter: Eine hier eingestellt Zeit wird mit der Sollgeschwindigkeit verrechnet und bildet den dynamischen (von der Geschwindigkeit abhängiger Anteil) Teil der Schwelle für die Überwachung dar.

PEH Filter: Diese Überwachung prüft, ob die Achse nach Erreichen des Bremsfensters (Ende der aktiven Bewegung) zumindest einmalig das Zielfenster erreicht. Wenn diese Überwachung aktiviert ist, wird ein Fehler erkannt und gemeldet, wenn ein Baustein (z.B. MC_MoveAbsolute_BkPlcMc) seinen Auftrag nicht *Done* melden kann.

4.6.3.5 Regler

Im Stillstand ist bei einer Achse der Lageregler aktiv, um ein Driften der Achse von der Sollposition zu verhindern. Gerade bei einer falschen bzw. nicht optimal eingestellten Achse können bei aktivem Lageregler unerwünschte Effekte auftreten. Aus diesem Grund sind die Einstellungen des Reglers zu kontrollieren.



Enable Ctr Loop on Error: Einige Fehlersituationen erlauben eine Lageregelung der Achse. Dies ist immer dann gegeben, wenn sowohl der Encoder als auch der Antrieb der Achse funktionstüchtig sind. Um dies zu nutzen, muss diese Eigenschaft aktiviert werden.

Schleppregler :

- **kp:** Verstärkungsfaktor für Schlepp- und Lageregler. Muss in diesem Schritt zunächst auf Null gestellt werden, da gerade bei falscher Encoder- bzw. Ventilinvertierung der Antrieb in seine Endlage rast. Dies liegt an der entstandenen Mitkopplung (eine wachsende Ausgabe lässt den Antrieb sich weiter vom Sollwert entfernen).
 - **HINWEIS!** Die Reglerausgabe kann auf die Schleichgeschwindigkeit der Strecke begrenzt werden.
- **dX:** Erst auf einen Fehler, der diesen Wert übersteigt reagiert der I-Anteil. Als Minimum kann hier die Encoderauflösung angegeben werden. Bei einer Eingabe von Null wird automatisch 2/3 der Encoderauflösung eingetragen, da eine Regelung genauer als die Istwerterfassung nicht möglich ist und nur zusätzliche Unruhe in das System bringt.
- **Ti:** Der I-Anteil für Schlepp- und Lageregler. Auch dieser muss in diesem Schritt auf Null gesetzt werden, da gerade bei falscher Encoder- bzw. Ventilinvertierung der Antrieb in seine Endlage rast. Dies liegt an der entstandenen Mitkopplung (eine wachsende Ausgabe lässt den Antrieb sich weiter vom Sollwert entfernen).
- **WuLimit:** Begrenzung des I-Anteils. Der I-Anteil kann nicht größer als der angegebene Wert werden. Der WuLimit muss immer kleiner als OutLimit gewählt werden, um dem P-Anteil immer eine Reaktionsmöglichkeit zu garantieren. Wird diese Einstellung nicht beachtet, kann es zu instabilem Verhalten kommen.
- **OutLimit:** Begrenzung der Ausgabe. Die Ausgabe des Reglers kann nicht größer als der angegebene Wert werden. Der OutLimit muss immer größer als der WuLimit gewählt werden, um dem P-Anteil immer eine Reaktionsmöglichkeit zu garantieren. Wird diese Einstellung nicht beachtet, kann es zu instabilem Verhalten kommen.

Veloregler:

Der Geschwindigkeitsregler und der Lageregler sind über ein Integral miteinander verkoppelt. Aus diesem Grund kann eine ungünstige Einstellung im Geschwindigkeitsregler schnell zu instabilem Verhalten führen.

- **kp:** Verstärkungsfaktor für Geschwindigkeitsregler. Muss auf Null gestellt werden.

- **dX:** Erst auf einen Fehler größer diesem Wert reagiert der I-Anteil. Als Minimum kann hier der Diskretisierungsfehler der Klemme angenommen werden (Encoderlänge / Max. Incemente).
- **Ti:** Der I-Anteil für Geschwindigkeitsregler. Auch dieser muss in diesem Schritt auf Null gesetzt werden.
- **WuLimit:** Begrenzung des I-Anteils. Der I-Anteil kann nicht größer als der angegebene Wert werden. Der WuLimit muss immer kleiner als OutLimit gewählt werden, um dem P-Anteil immer eine Reaktionsmöglichkeit zu garantieren. Wird diese Einstellung nicht beachtet, kann es zu instabilem Verhalten kommen.
- **OutLimit:** Begrenzung der Ausgabe. Die Ausgabe des Reglers kann nicht größer als der Angegebene Wert werden. Der OutLimit muss immer größer als der WuLimit gewählt werden, um dem P-Anteil immer eine Reaktionsmöglichkeit zu garantieren. Wird diese Einstellung nicht beachtet, kann es zu instabilem Verhalten kommen.

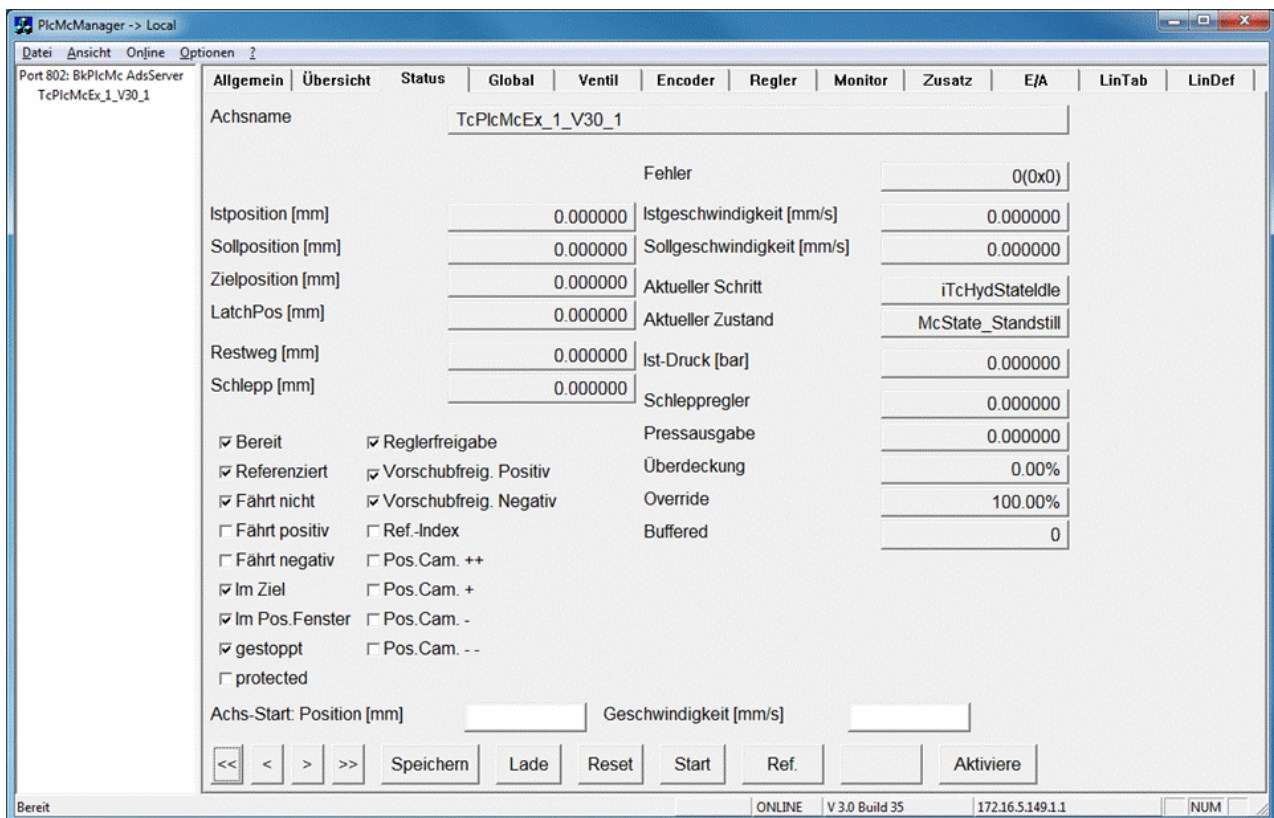
Weitere Einstellungen:

- **Vorsteuerung:** Gewichtet die Sollgeschwindigkeit zwischen null und eins. Null bedeutet keine Vorsteuerung, somit ist nur der Lageregler aktiv und eins bedeutet, dass die komplett berechnete Vorsteuerung ausgegeben wird. In der Regel sollte der Wert auf eins stehen. **📌 HINWEIS! Wenn eine weggesteuerte Sollwertgenerierung verwendet wird steht dieser Parameter fest auf 1.**
- **Offset-Kompensation:** Um eine Offset-Spannung beim Analog-Ausgang zu kompensieren. Dieser sollte in diesem Zustand auf jeden Fall auf Null stehen.
- **Pressdruck:** In einigen Anwendungsfällen soll ein Proportionalventil nach dem Abschluss einer Bewegung durch eine konstante Ausgabe einen **Pressdruck** am Hartanschlag erzeugen. An diesem Punkt der Inbetriebnahme muss der Wert auf Null stehen.
- **Repos Schwelle:** Bei nicht lageregelbaren Achsen kann es sinnvoll sein, die Achse zu Re-Positionieren, wenn sie aus der Zielposition bewegt wurde. An dieser Stelle sollte der Wert auf Null stehen.

4.6.3.6 Status

Nachdem alle vorherigen Schritte erfolgreich durchgeführt wurden, kann die Achse mit Antriebsenergie versorgt werden. Außer einer kleinen Drift darf sich die Achse nicht bewegen. Um eine aktive Fahrbewegung durchführen zu können, muss eine Reglerfreigabe erfolgen und folgende Punkte beachtet werden:

- keine Fehlermeldung
- Meldung *Bereit*,
- Meldung *Referenziert*,
- Meldung *Fährt nicht*,
- Meldung *gestoppt*,
- Meldung *Reglerfreigabe*,
- Meldung *Vorschubfreig. Positiv*,
- Meldung *Vorschubfreig. Negativ*,
- Aktueller Schritt sollte iTcHydStateldle sein.



Fehler: Zeigt einen aktuellen Fehler der Achse an. Es sollte kein Fehler vorliegen.

Istposition: Aktuelle Position der Achse.

Sollposition: Sollposition der Achse für den Schleppregler. **Istposition** und **Sollposition** sollten nahe beieinander liegen.

Zielposition: Vom außen durch einen Fahrauftrag vorgegebene Position.

Latch Pos: In Vorbereitung

Restweg: Der restliche zurückzulegende Weg bis zur Zielposition (Zielposition - Istposition)

Schlepp: Der restliche zurückzulegende Weg bis zur berechneten Sollposition. Auf diesen Fehler reagiert der Lageregler (Sollposition - Istposition)

Istgeschwindigkeit: Aktuelle Geschwindigkeit der Achse.

Sollgeschwindigkeit: Vorgegebene Geschwindigkeit der Achse.

Aktueller Schritt: Gibt den aktuellen Zustand der Statemachine von dieser Achse wieder.

Aktueller Zustand: Gibt den aktuellen Zustand der Statemachine von dieser Achse wieder.

Ist-Druck: In Vorbereitung

Schleppregler: Ausgabe des Schleppreglers in %

Pressausgabe: In Vorbereitung

Überdeckung: Anteilige Ausgabe der Überdeckung in %

Override: In Vorbereitung

Digitale Statusmeldungen:

Bereit: In Vorbereitung

Referenziert: Rückmeldung, dass die Achse referenziert wurde (nur bei Incrementalencodern von Bedeutung)

Fährt nicht: Die Achse hat einen Fahrauftrag und befindet sich in Lageregelung.

Fährt positiv: Die Achse fährt mit positiver Geschwindigkeit.

Fährt negativ: Die Achse fährt mit negativer Geschwindigkeit.

Im Ziel: Wird aktiv, wenn sich die Achse nach einem Fahrprofil im "*pos. Range*" (unter dem Reiter Monitor) befindet. Sobald die Achse aus diesem Range driftet fällt das Flag ab.

Im Pos. Fenster: Wird aktiv, wenn sich die Achse nach einem Fahrprofil einmal im "*pos. Range*" (unter dem Reiter Monitor) befindet. Sobald die Achse aus diesem Range driftet fällt das Flag ab.

gestoppt: In Vorbereitung

protected: In Vorbereitung

Reglerfreigabe: Rückmeldung, dass der Lageregler aktiviert wurde. Sollte in diesem Fall aktiv sein.

Vorschubfreig. Positiv: Vorschubfreigabe in positive Richtung. Sollte in diesem Fall aktiv sein.

Vorschubfreig. Negativ: Vorschubfreigabe in negative Richtung. Sollte in diesem Fall aktive sein.

Ref.-Index: In Vorbereitung

Pos.Cam + +: In Vorbereitung

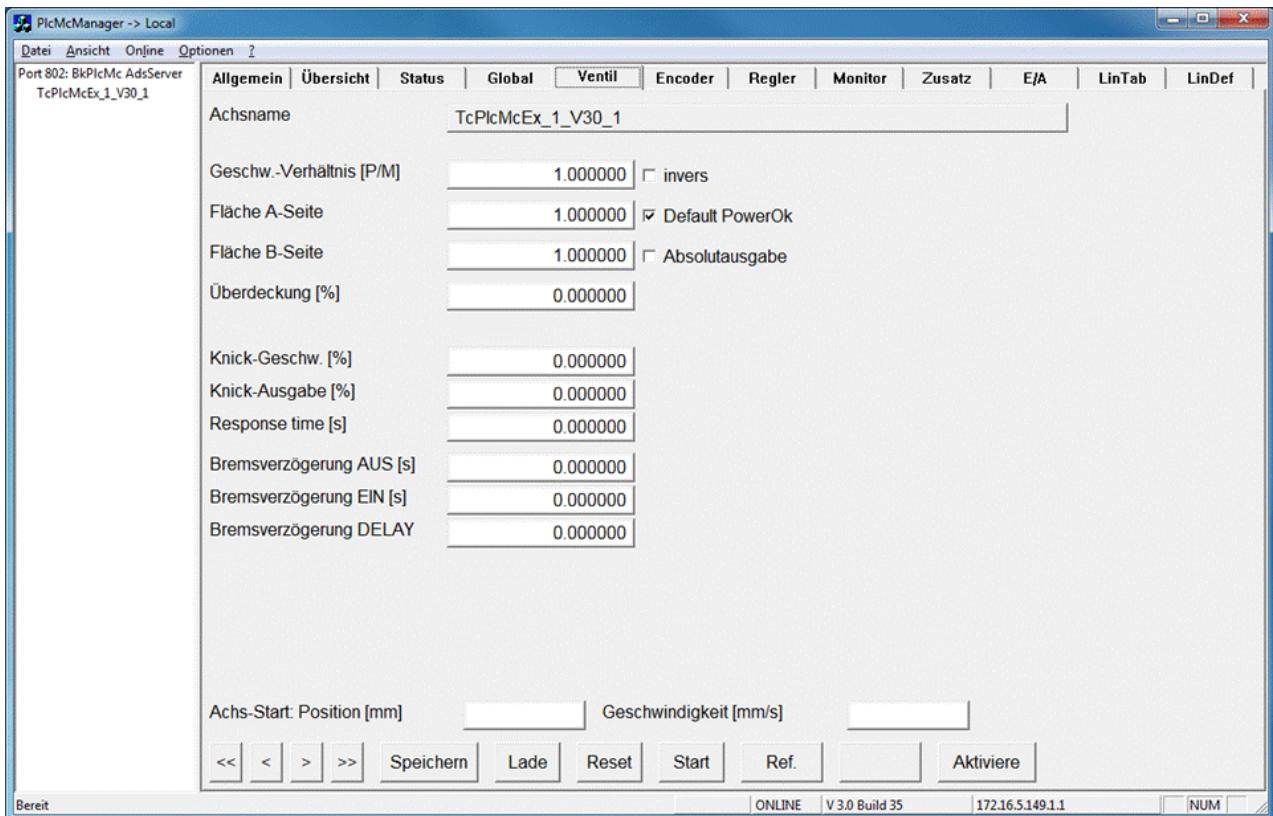
Pos.Cam + : In Vorbereitung

Pos.Cam - : In Vorbereitung

Pos.Cam- - : In Vorbereitung

4.6.3.7 Voreinstellung Ventil/ Antriebssteller

Die Einstellungen zum Ventil befinden sich unter dem Reiter "Ventil" im PlcMc-Manager. Hier werden Voreinstellungen bezüglich Bezugsgeschwindigkeit, Überdeckung und Flächenverhältnisse vorgenommen. Gerade für die Ermittlung der Überdeckung und des Geschwindigkeitsverhältnisses müssen Daten zum Ventil und Zylinder vorliegen.



Hydraulischer Antrieb:

invers: Invertierung der Ventilausgabe. In der Hydraulik ist die A-Seite eines Zylinders die Seite mit der größeren Fläche. Für die Hydraulikbibliothek ist die A-Seite eines Zylinders die Seite, die mit Druck beaufschlagt werden muss, um positiv fahren zu können. Wenn die größere Fläche mit Druck beaufschlagt werden muss, um auch in positive Richtung fahren zu können, darf keine Invertierung vorgenommen werden.

Default Power Ok: Einige Antriebsteller geben eine Rückmeldung, dass die Aktivierung des Antriebsstellers erfolgreich war. Bei Deaktivierung wird eine solche aktive Rückmeldung vom Antriebssteller erwartet, bevor die Achse freigegeben wird.

Absolutausgabe: Ist nur zu aktivieren, falls die Stellwertausgabe nur positiv sein darf. Die Richtungsumkehrung muss in diesem Fall über andere Wege erfolgen. Ist in der Regel abzuwählen.

Geschw.-Verhältnis [P/M]: Dieser Wert wird erst in der weiteren Inbetriebnahme genau festgelegt. Der Parameter kompensiert die unterschiedlichen Geschwindigkeiten beim positiven und negativen Verfahren der Achse. Bei 1.0 wird ein richtungsunabhängiges Verhalten erwartet.

Überdeckung: Die Überdeckung des Ventils ist aus dem Datenblatt vom Ventilhersteller zu entnehmen. Bei Nullschnittventilen ist der Parameter 0.0, ansonsten bewegt sich der Wert i. d. R. zwischen 0.0 und 20 %. Ist die Überdeckung zwischen P-A und P-B asymmetrisch, so müssen statt einer allgemeinen Ventilüberdeckung eine spezielle für jede Richtung eingegeben werden. Hierzu ist unter dem Reiter "Global" der Haken bei "Asym." zu setzen. Die hier eingestellten Werte sind Voreinstellungen und werden im weiteren Verlauf der Inbetriebnahme weiter optimiert.



Hinweis

Überdeckung

Ist die Überdeckung richtungsabhängig stark unterschiedlich, ist höchste Vorsicht geboten. Um kein Fehlverhalten der Achse zu verursachen, sollte die kleinere Überdeckung in beide Seiten eingetragen werden. Im weiteren Verlauf der Inbetriebnahme nähert man sich vorsichtig den optimalen Überdeckungswerten an.

HINWEIS! Ist die Überdeckung nicht bekannt, so ist es möglich, diese Einstellung in einem späteren Schritt nachzuholen.

Fläche A-Seite: Zylinderfläche der A-Seite (positiv fahrende Fläche) in mm². Wird zur Berechnung der angeforderten Ölmengen sowie zur Umrechnung eines gemessenen Druckes in eine Kraft benötigt.

Fläche B-Seite: Zylinderfläche der B-Seite (negativ fahrende Fläche) in mm². Wird zur Berechnung der angeforderten Ölmengen sowie zur Umrechnung eines gemessenen Druckes in eine Kraft benötigt.

Knick-Geschw.: Bei Ventilkennlinien mit einem Knick in der Kennlinie ist die Geschwindigkeit / Ölmenge des Knickes in % hier einzutragen.

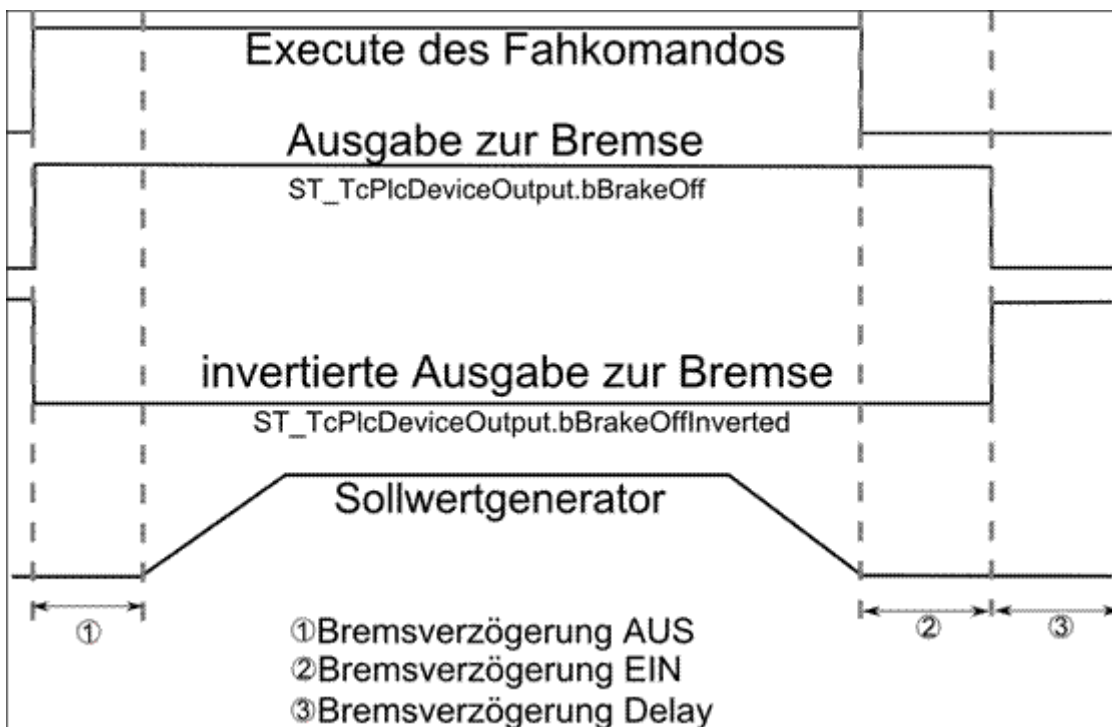
Knick-Ausgabe: Bei Ventilkennlinien mit einem Knick in der Kennlinie ist die Ausgabe der Klemme des Knickes in % hier einzutragen.

Response time: Sprungantwortzeit des gesamten Systems. Durch die Massenträgheit entsteht beim Abbremsen auf die gewünschte Position ein nicht linearer Verlauf der Geschwindigkeit. Soll der Bremsvorgang linear verlaufen, muss hier die Sprungantwortzeit des Systems eingetragen werden. Dieser Parameter ist sehr schwer zu ermitteln und beansprucht das Ventil erheblich. Aus diesem Grund sollte der Wert, falls nicht unbedingt erforderlich, auf Null gesetzt werden.

Bremsverzögerung AUS: Nachdem ein Fahrbefehl vorliegt, beginnt der Sollwertgenerator nach dieser Verzögerung mit der Ausgabe.

Bremsverzögerung EIN: Nach dem ein Fahrbefehl beendet wurde, wird diese Zeit abgewartet, bevor der Ausgang zur Bremse gesetzt wird.

Bremsverzögerung DELAY: Bevor der nächste Fahrauftrag für diese Achse angenommen werden darf, muss diese Zeit abgewartet werden. Fahraufträge die innerhalb dieser Zeit erfolgen werden abgelehnt.



Elektrischer Antrieb:

(Ist noch in Vorbereitung. NICHT freigegebene Erklärungen)

invers:

Default Power Ok: Einige Antriebstersteller geben eine Rückmeldung, dass die Aktivierung des Antriebsstellers erfolgreich war. Bei Deaktivierung wird eine aktive Rückmeldung vom Antriebssteller erwartet, bevor die Achse freigegeben wird.

Absolutausgabe: Ist nur zu aktivieren, falls die Stellwertausgabe nur positiv sein darf. Die Richtungsumkehrung muss in diesem Fall über andere Wege erfolgen. Ist in der Regel abzuwählen.

Geschw.-Verhältnis [P/M]: Dieser Wert hat bei elektrischen Achsen keine Bedeutung und sollte auf Eins gestellt werden.

Überdeckung: Der Wert muss auf Null gestellt werden, außer beim Einsatz von Frequenzumrichtern.

Fläche A-Seite: Hat bei elektrischen Achsen keine Bedeutung. Muss auf Eins gestellt werden.

Fläche B-Seite: Hat bei elektrischen Achsen keine Bedeutung. Muss auf Eins gestellt werden.

Knick-Geschw.: Hat bei elektrischen Achsen keine Bedeutung. Muss auf Null gestellt werden.

Knick-Ausgabe: Hat bei elektrischen Achsen keine Bedeutung. Muss auf Null gestellt werden.

Response time: In der Regel ist hier 0 einzustellen. In wenigen Fällen kann mit diesem Parameter das Bremsverhalten verbessert werden. Vor allem bei zu hohen Werten kann sich die Achse unerwartet verhalten oder Kavitation auftreten. In letzterem Fall besteht das Risiko, das Ventil oder andere Bauteile zu beschädigen.

Bremsverzögerung AUS: Nachdem ein Fahrbefehl vorliegt, beginnt der Sollwertgenerator nach dieser Verzögerung mit der Ausgabe.

Bremsverzögerung EIN: Nach dem ein Fahrbefehl beendet wurde, wird diese Zeit abgewartet, bevor der Ausgang zur Bremse gesetzt wird.

Bremsverzögerung DELAY: Bevor der nächste Fahrauftrag für diese Achse angenommen werden darf, muss diese Zeit abgewartet werden. Fahraufträge die innerhalb dieser Zeit erfolgen werden abgelehnt.

Inc. Bewertung:

Inc. Zwischenteilung:

4.6.4 Inbetriebnahme

4.6.4.1 Encoder und Druckmessdosen

Die Inbetriebnahme eines Encoders umfasst die Wegskalierung, die Zählrichtung und bei Absolutencodern die Nullpunktverschiebung. Bei Inkrementalencodern sind Parameter für die Referenzierung (Homing) vorzugeben.

Zählrichtung

Zunächst ist festzulegen, welche Bewegungsrichtung der Achse als "positiv", "steigend" oder "aufwärts" zu betrachten ist. Bei dieser Festlegung ist die physikalische Zählrichtung des Encoders ohne jede Bedeutung. Allerdings muss festgestellt werden, ob der Encoder zur festgelegten Zählrichtung invers arbeitet. Dazu muss die vom Encoder gemeldete Position zumindest eine minimale Änderung wiedergeben. Es ist zu überprüfen, ob die Änderungsrichtung der ermittelten Positionsänderung zur Richtung der mechanischen Bewegung passt.

- Wenn die Möglichkeit besteht sollte die Achse manuell (ohne Antriebsenergie) bewegt werden.
- Eventuell kann der Encoder unabhängig von der Maschine bewegt werden. Dabei ist zu berücksichtigen, dass die Bewegung des feststehenden Teils des Encoders einer Bewegung der Achse in der entgegengesetzten Richtung entsprechen kann.
- Bestehen diese Möglichkeiten nicht muss die Achse mit Antriebsenergie bewegt werden. Da die Parameter der Achse noch nicht sinnvoll eingestellt sind ist hierbei höchste Vorsicht geboten.
 - Stellen Sie sicher, dass sich keine anderen Achsen oder Einrichtungen in Bewegung setzen können.
 - Wenn die Möglichkeit besteht, die Last der Achse durch Demontage von Übertragungsmitteln (Riemen, Hebel, Stangen) vollständig oder teilweise zu entkoppeln ohne dass dadurch die Verbindung zum Encoder aufgehoben wird, sollte dies unbedingt getan werden.

- Zumindest die Voreinstellungen der Antriebs-Inbetriebnahme [▶ 264] sollte erfolgt sein. Ist dies nicht möglich geben Sie als Bezugsgeschwindigkeit und Max.Appl.Geschw. einen willkürlichen Wert (z.B. 100) und als Handfahrgeschwindigkeit 10 bzw. 25 vor.
- Stellen Sie das Flächenverhältnis auf 1 und die Parameter für Überdeckung und Knick-Kompensation auf Null. Stellen Sie die Response Time und alle Brems-Zeiten auf Null. Deaktivieren Sie die Überwachung der Leistungszuschaltung (Default PowerOk).
- Setzen Sie alle Regler (Defaultregler, Applikationsregler) außer Funktion. Die Achse sollte ausschließlich durch den Stellwert bewegt werden. Stellen Sie sicher, dass kein automatischer Nullpunktsabgleich vorgenommen wird. Stellen sie die Offset-Kompensation, die Pressdruckausgabe und die Repos-Schwelle auf Null.
- Deaktivieren Sie die Schleppüberwachung und die Software-Endschalter.
- Vergewissern Sie sich, dass sich keine Personen oder Gegenstände im Gefahrenbereich befinden. Sichern Sie den Bereich gegen Betreten. Weisen Sie alle Personen in der Umgebung auf die besondere Gefahrensituation hin. Achten Sie darauf, dass Sie selbst z.B. durch eine zweite Person gesichert sind.
- Schalten Sie jetzt die Energie zu. Von einer kleinen Drift abgesehen darf sich die Achse nicht bewegen.
- Im Online-Bild sollte Sie Achse auf die F2-Taste mit einer Bewegung in negativer Richtung und auf die F3-Taste mit einer Bewegung in positiver Richtung reagieren. Es ist hier nicht die Reaktion des Achs-Istposition gemeint, sondern die mechanische Bewegung. Reagiert die Achse nicht kann dies auf eine Überdeckung des Ventils zurück zu führen sein. Dies sollte im Datenblatt des Herstellers vermerkt sein. In diesem Fall sind die Taste F1 (statt F2) und F4 (statt F3) zu benutzen. Sollte sich die Achse in entgegengesetzter Richtung bewegen ist die Ausgabepolarität (PlcMcManager: Ventil-Reiter, "invers", rechts neben "Flächenverhältnis A/B") zu ändern.
- Wenn die Achse mit einer Bewegung in der richtigen Orientierung reagiert sollte die Istposition diese Bewegung wiedergeben. Stimmt die Zählrichtung der Istposition nicht mit der Richtung der mechanischen Bewegung überein ist die Einstellung von bEnc_Reversed (PlcMcManager: Encoder-Reiter, "invers", rechts neben "Inc.Bewertung [mm/INC]") zu ändern. Siehe auch Beispiel 1 [▶ 261].

Skalierung der Istposition

Die Skalierung wird als Wertepaar eingestellt: Eine Positionsdivergenz **fEnc_IncWeighting** und die entsprechende Anzahl der erfassten Weginkremente **fEnc_IncInterpolation**. Siehe auch Beispiel 2 [▶ 263]. Abhängig von Aufbau, Interface und Wirkungsweise des Encoders bieten sich mehrere Vorgehensweisen an:

- Bei einigen Encoder-Typen (AX2000B750 [▶ 127], AX5000B110 [▶ 128]) wird **fEnc_IncInterpolation** aus dem Gerät ausgelesen. Hier ist der Vorschub pro Umdrehung der Motorwelle als **fEnc_IncWeighting** einzugeben.
- Bei einigen Encoder ist die Inkrementgewichtung vom Hersteller angegeben und ohne Genauigkeitsverlust eingebbar. Dies ist z.B. für SSI-Encodern typisch. Hier kann das **fEnc_IncWeighting** direkt (z.B. 0.01mm) eingegeben werden. Die **fEnc_IncInterpolation** ist dann auf 1.0 einzustellen.
- Bei rotatorischen Encodern ist die Anzahl der Inkremente pro Umdrehung oft eine Potenz von 2, also 2048 oder 8192. Der Vorschub pro Umdrehung ist jedoch oft eine ganze Zahl. Wird hier die Inkrementgewichtung ermittelt ergibt sich oft eine kleine Zahl mit sehr vielen Nachkommastellen. Dieser Wert ist ohne Genauigkeitsverlust nicht eingebbar. Hier sollte vorzugsweise der Vorschub pro Umdrehung als **fEnc_IncWeighting** und die Inkremente pro Umdrehung als **fEnc_IncInterpolation** angegeben werden.
- Bei Analog-Meßsystemen kann eine beliebige Wegstrecke als **fEnc_IncWeighting** und die dazu gehörende Anzahl der ADC-Inkremente als **fEnc_IncInterpolation** eingegeben werden. Zu Beginn der Inbetriebnahme kann oft eine ungefähre Einstellung vorgenommen werden. Dazu wird als Wegstrecke die Messlänge des Encoders und als Anzahl der Inkremente der Wertebereich der Eingangs-Elektronik (in der Regel 32768) verwendet werden. Sobald die Achse im weiteren Verlauf der Inbetriebnahme kontrolliert gefahren werden kann ist die Ermittlung der Werte genauer durchzuführen. Dazu sollte zunächst an einen mechanischen Anschlag gefahren werden. Der Wert des AD-Wandlers ist zu

notieren und die mechanische Position zu markieren. Diese Vorgehensweise ist an einem zweiten mechanischen Anschlag zu wiederholen. Der mechanische Weg ist als **fEnc_IncWeighting** und die dazu gehörende Differenz der ADC-Inkrementen als **fEnc_InclInterpolation** einzugeben.

ⓘ HINWEIS! Der Unterschied zwischen Pulsen und Inkrementen ist zu beachten. Beispiel: Ein Inkremental-Encoder mit 10mm Weg und 2048 Pulsen pro Umdrehung wird an eine KL5101 angeschlossen. Diese Klemme verwendet (Defaulteinstellung) die Methode der Vierfachauswertung und erzeugt somit 8192 Inkremente pro Umdrehung.

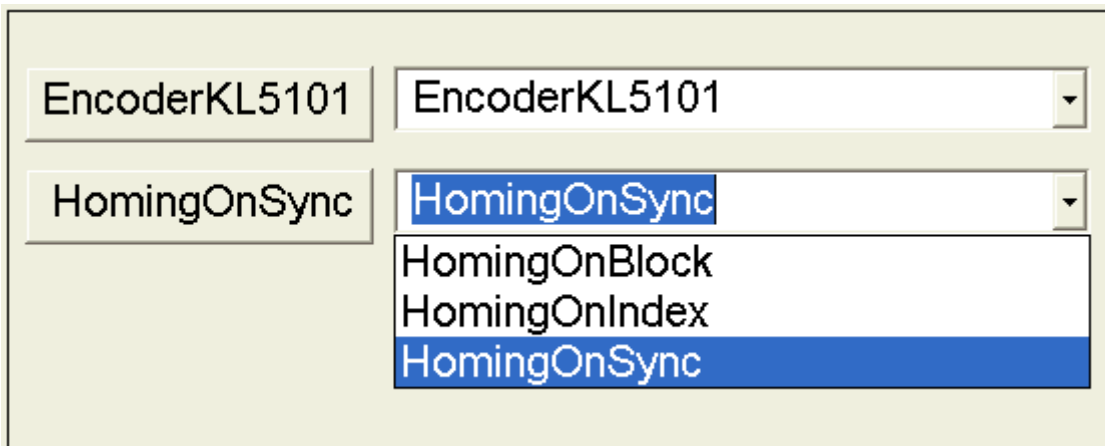
Nullpunktverschiebung

Bei Absolutencodern liegt der Nullpunkt des Encoders an einem Ort, der durch die Konstruktion des Encoders und seine Anbringung in der Maschine festgelegt wird. Um den Nullpunkt der Achse frei wählen zu können wird ein Positionsoffset addiert. Dieser Offset kann entweder direkt (PlcMcManager: Encoder-Reiter, (PlcMcManager: Encoder-Reiter, "dritter Eintrag von oben) eingegeben oder automatisch errechnet werden. Dazu wird die Achse zum Beispiel an einen mechanischen Anschlag gefahren, der einer definierten Position entspricht. Diese Position wird im PlcMcManager wie die Zielposition einer Bewegung eingegeben (PlcMcManager: "Achse-Start: Position [mm]") und mit dem Button "SetPos" wirksam gemacht. Siehe auch [Beispiel 1](#) [[▶ 261](#)].

ⓘ HINWEIS! Nach dem Setzen der Istposition sind die Parameter der Achse zu speichern.

Parameter für die Referenzierung (Homing)

Umfang und Zusammensetzung der Homing-Parameter hängt vom gewählten Referenziermodus ab. Dieser wird auf dem Allgemein-Reiter der Achse eingestellt. Dabei wird links die aktuelle Einstellung gezeigt. Rechts davon kann über eine Optionsliste eine neue Wahl vorgenommen werden. Um den gewählten Modus zu aktivieren ist die "Aktiviere"-Schaltfläche direkt rechts neben der neuen Auswahl zu betätigen. Dadurch wird in [ST_TcHydAxParam](#) [[▶ 88](#)].nEnc_HomingType ein Wert aus [E_TcMcHomingType](#) [[▶ 73](#)] eingestellt.



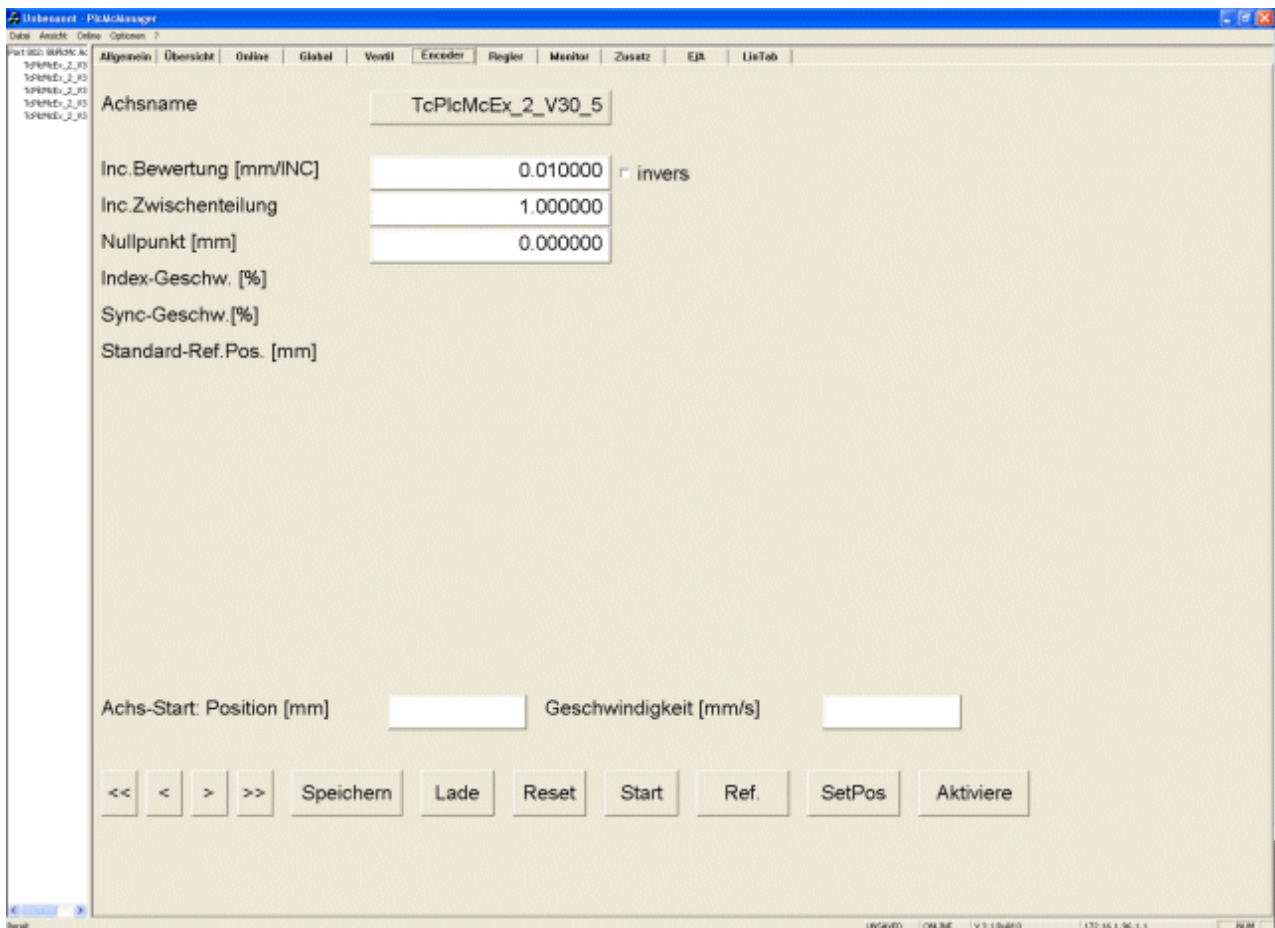
HomingOnBlock: Die Achse wird mit [ST_TcHydAxParam](#) [[▶ 88](#)].fEnc_RefIndexVelo in der durch [ST_TcHydAxParam](#) [[▶ 88](#)].bEnc_RefIndexPositive festgelegten Richtung bewegt. Wird für eine Zeitdauer von 2 Sekunden keine Bewegung festgestellt gilt der Festanschlag (Block) als erreicht. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

HomingOnIndex: Die Achse wird mit [ST_TcHydAxParam](#) [[▶ 88](#)].fEnc_RefIndexVelo in der durch [ST_TcHydAxParam](#) [[▶ 88](#)].bEnc_RefIndexPositive festgelegten Richtung bewegt. Wird in [ST_TcHydAxRtData.nDeCtrlDWord](#) die Referenznocke (Bit 5, dwTcHydDcDwRefIndex) erkannt stoppt die Achse. Anschließend wird sie mit [ST_TcHydAxParam](#) [[▶ 88](#)].fEnc_RefSyncVelo in der durch [ST_TcHydAxParam](#) [[▶ 88](#)].bEnc_RefSyncPositive festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Der Istwert der Achse wird auf den Wert der Referenzposition gesetzt.

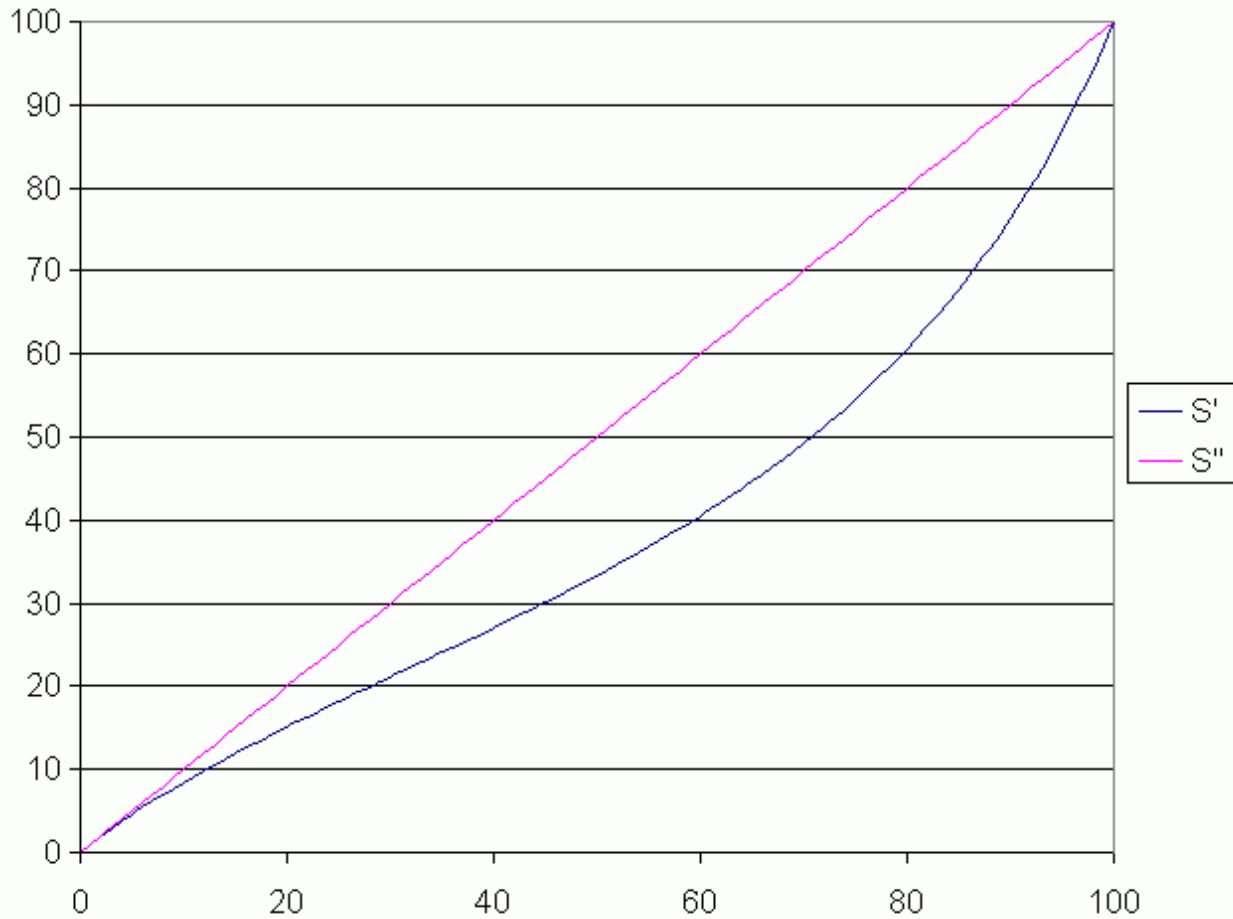
HomingOnSync: Die Achse wird mit [ST_TcHydAxParam](#) [[▶ 88](#)].fEnc_RefIndexVelo in der durch [ST_TcHydAxParam](#) [[▶ 88](#)].bEnc_RefIndexPositive festgelegten Richtung bewegt. Wird in [ST_TcHydAxRtData.nDeCtrlDWord](#) die Referenznocke (Bit 5, dwTcHydDcDwRefIndex) erkannt stoppt die

Achse. Anschließend wird sie mit `ST_TcHydAxParam [► 88].fEnc_RefSyncVelo` in der durch `ST_TcHydAxParam [► 88].bEnc_RefSyncPositive` festgelegten Richtung bewegt, bis die Referenznocke wieder verlassen wird. Dann wird das Hardware-Latch des Encoders aktiviert und die Achse so lange weiter bewegt, bis das Latch gültig wird. Nach dem Stoppen der Achse wird der Istwert der Achse auf einen Wert gesetzt, der aus der Referenzposition und der seit dem Sync-Puls des Encoders zurückgelegten Strecke errechnet wird.

Beispiel 1: Encoder-Reiter, Absolutencoder



Wenn als Weggeber ein Potentiometer verwendet wird und ohne eingebauten oder zwischengeschalteten Verstärker direkt am Analogeingang betrieben wird, ist die ohmsche Belastung durch den Eingangswiderstand zu berücksichtigen. Bleibt der Einfluss unkompensiert ist ein nichtlinear wegabhängiger Fehler feststellbar. Die nachstehende Grafik stellt diesen Effekt überhöht dar. Bei typischen Werten für den Widerstand des Potentiometers und des Analogeingangs sind absolute Maximalfehler zwischen 1 und 5% des Fahrweges zu erwarten.

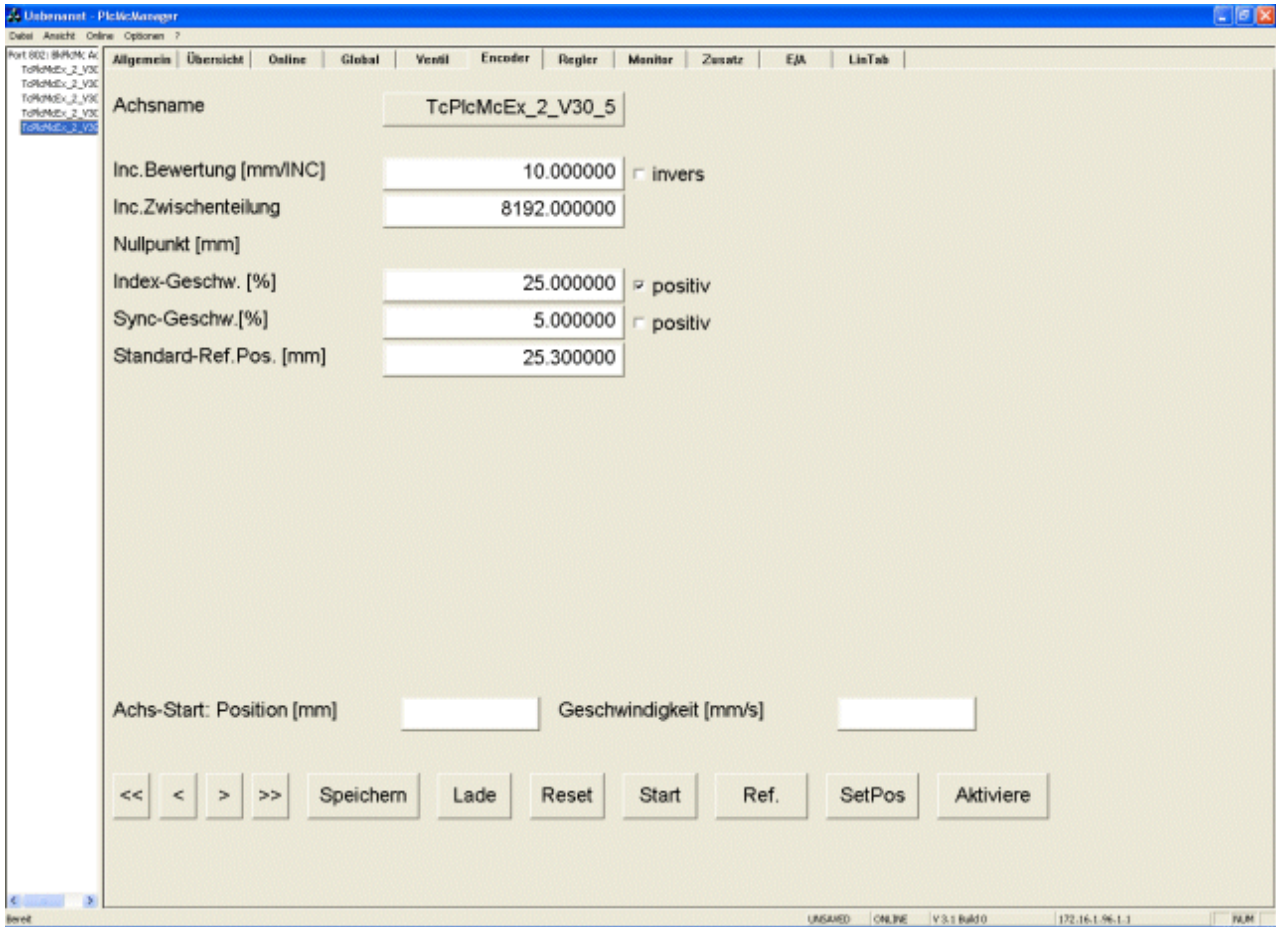


Dieser Fehler kann kompensiert werden indem das Verhältnis des Potentiometer-Gesamtwiderstands zum Lastwiderstand eingegeben wird.

Inc.Bewertung [mm/INC]	0.001000	<input type="checkbox"/> invers
Inc.Zwischenteilung	1.000000	
Nullpunkt [mm]	0.000000	
Index-Geschw. [%]		
Sync-Geschw.[%]		
Standard-Ref.Pos. [mm]		
Grundabstand		
Modulo-Bezug		
Poti Rg/RI	0.000000	

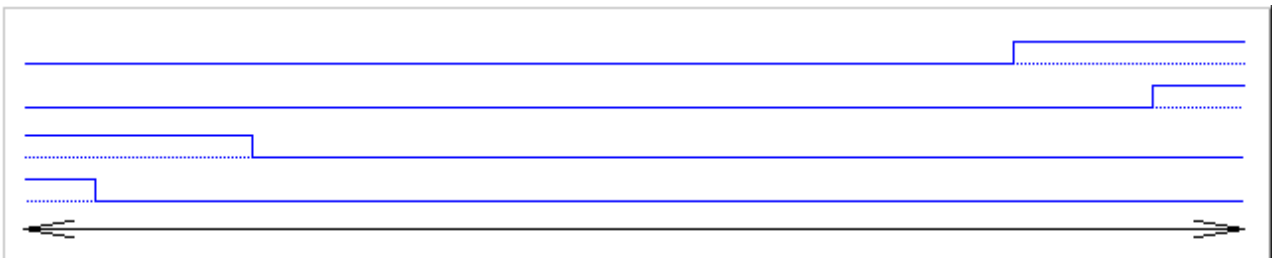
Wenn das Potentiometer einen Widerstand von 5kOhm und der Eingang einen Widerstand von 100kOhm aufweist ist also ein Wert von 0.05 einzutragen.

Beispiel 2: Encoder-Reiter, Inkrementalencoder



Sonderfall digitale Positionsnocken

In einigen Anwendungen wird kein Encoder zur Positionsermittlung eingesetzt sondern ein Satz von üblicherweise vier digitalen Sensoren. Um diese auszuwerten ist als Encodertyp `iTcMc_EncoderDigCam` [► 132] vorzugeben. Die Nocken müssen über den Fahrweg dargestellt das unten stehende Verhalten zeigen. Insbesondere dürfen die Nocken nicht überfahrbar sein. Andernfalls kann es insbesondere nach dem Einschalten zur Ausgabe von zu hohen Stellwerten und somit zum Überfahren des Ziels kommen.



HINWEIS! Diese Art der Positionsermittlung ist sinnvoll nur mit einem Generator vom Typ `iTcMc_ProfileTimeRamp` kombinierbar. Zur Inbetriebnahme siehe unter `Profiltyp iTcMc_ProfileTimeRamp`. [► 273]

Druck- und Kraftmessung

Die Erfassung von Kräften unter Verwendung von Druckmessdosen erfolgt mit Hilfe von [MC_AxRtReadForceDiff_BkPlcMc \[▶ 139\]](#) oder [MC_AxRtReadForceSingle_BkPlcMc \[▶ 141\]](#) Bausteinen. Wird ein Druck erfasst oder eine Kraft mit direkten Kraftsensoren ermittelt sind [MC_AxRtReadPressureDiff_BkPlcMc \[▶ 143\]](#) oder [MC_AxRtReadPressureSingle_BkPlcMc \[▶ 145\]](#) Bausteine zu verwenden. Mögliche Vorgehensweisen für die Inbetriebnahme sind in der Dokumentation der Bausteine beschrieben.

4.6.4.2 Antriebe

in Vorbereitung

Es sind einige Sonderfälle zu beachten: [Frequenzumrichter \[▶ 274\]](#).

Bezugsgeschwindigkeit: Hier ist die höchste erreichbare Achsgeschwindigkeit einzutragen.

Beschleunigung, Verzögerung: Diese Parameter legen absolute Begrenzungen fest. Die weiter unten beschriebenen Applikations-Parameter werden auf diese Werte begrenzt.

Handfahrgeschwindigkeit langsam, schnell: Diese Geschwindigkeiten werden z.B. vom PlcMcManager bei Handfahrten (Tasten F1 - F4) benutzt.

Max. Appl. Geschwindigkeit: Hier kann die von einem Applikationsbaustein anforderbare Fahrgeschwindigkeit begrenzt werden. Dieser Parameter kann maximal den Wert der **Bezugsgeschwindigkeit** annehmen.

Max. Appl. Beschleunigung, Verzögerung: Diese Parameter werden verwendet um die Parametrierung von Bausteinen wie z.B. [MC_MoveAbsolute_BkPlcMc \[▶ 54\]](#) zu begrenzen oder zu ersetzen.

Voreinstellungen

Die hier beschriebenen Einstellungen sind als provisorisch anzusehen. Sie sollen es ermöglichen, die Achse überhaupt halbwegs kontrolliert bewegen zu können. Dies ist erforderlich um andere wichtige Einstellungen (Encoder, Positionierung) vornehmen zu können.

Bezugsgeschwindigkeit:

Für einige Antriebs-Typen ([AX2000B750 \[▶ 116\]](#), [AX5000B110 \[▶ 116\]](#)) wird die **Bezugsgeschwindigkeit** aus Werten errechnet, die aus dem Gerät ausgelesen werden.

Für hydraulik-Achsen kann hier als erste Näherung ein Wert vorgegeben werden, der aus Datenblattangaben des Ventils und des Zylinders errechnet wird.

$$V_{\text{ref}} := 16666 * Q_{\text{nom}} / A_{\text{min}}$$

mit

V_{ref} := Bezugsgeschwindigkeit in mm pro Sekunde

Q_{nom} := Nomineller Volumenstrom des Ventils in Liter pro Minute

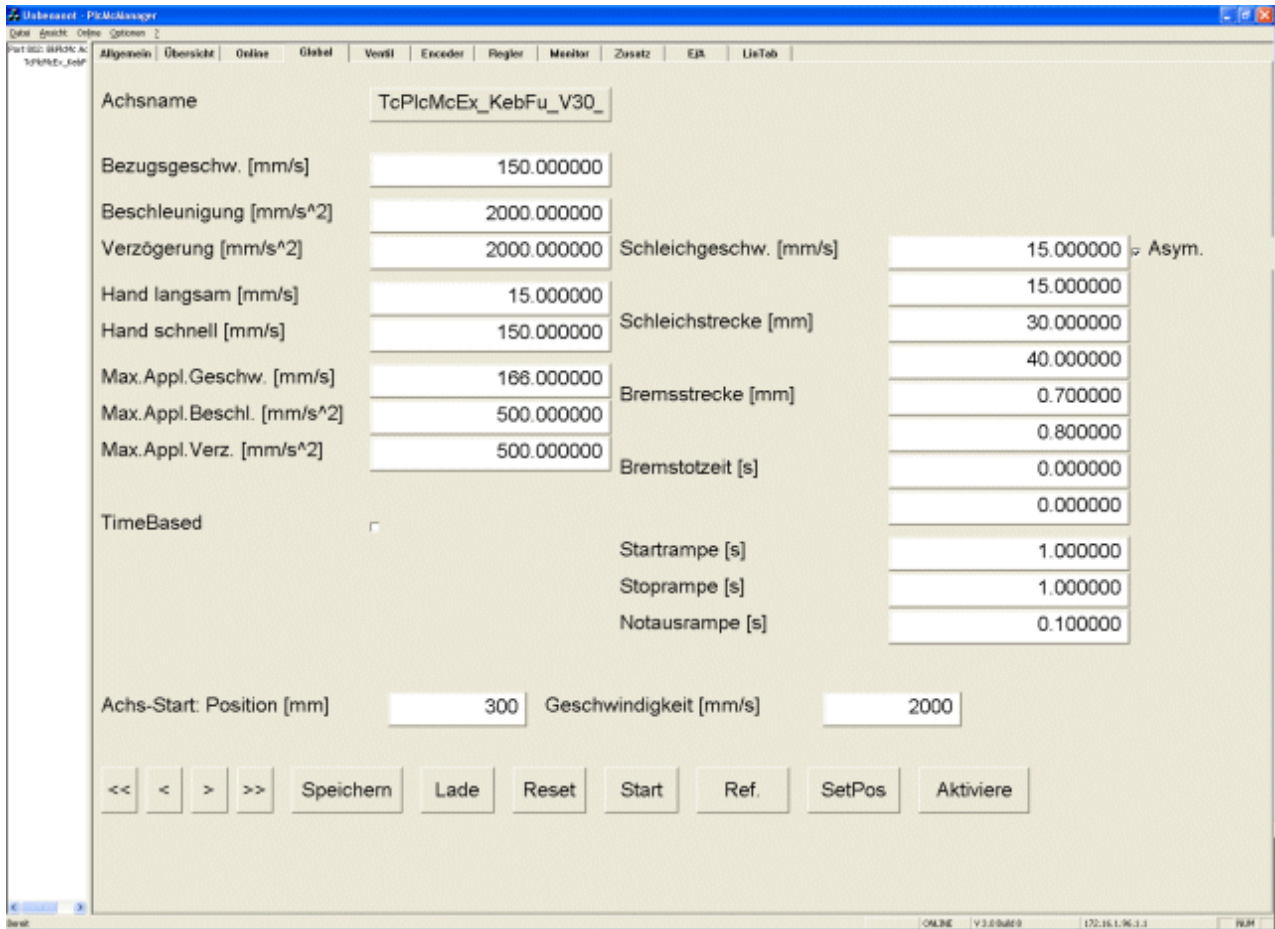
A_{min} := Kleinere der beiden Wirkungsflächen in mm²

Beschleunigung, Verzögerung: Hier kann zumindest vorerst ein sehr großer Wert wie z.B. 20000.0 vorgegeben werden.

Handfahrgeschwindigkeit langsam, schnell: Hier sollten Werte vorgegeben werden, die eine ausreichende Reaktionsmöglichkeit sicherstellen. Dies sollte bei Einstellungen von wenigen 100mm/s gegeben sein. Das Verhältnis von schnell zu langsam sollte dabei etwa vier bis fünf zu eins sein.

Max. Appl. Beschleunigung, Verzögerung: Hier ist vorerst ein kleiner Wert einzustellen. Dabei ist die Hochlaufzeit ($V_{\text{ref}} / \text{Acc}$) eine gute Kennzahl. Liegt sie unter eine Sekunde ist die Beschleunigung eher hoch. Zeiten von über fünf Sekunden kennzeichnen eine sehr weiche Einstellung. Als Voreinstellung sollte eine Hochlaufzeit von etwa einer bis höchstens zwei Sekunden angestrebt werden.

Beispiel 1: Global-Reiter bei Profiltyp iTcMc_ProfileCtrlBased



4.6.4.3 Bezugswerte der Vorsteuerung

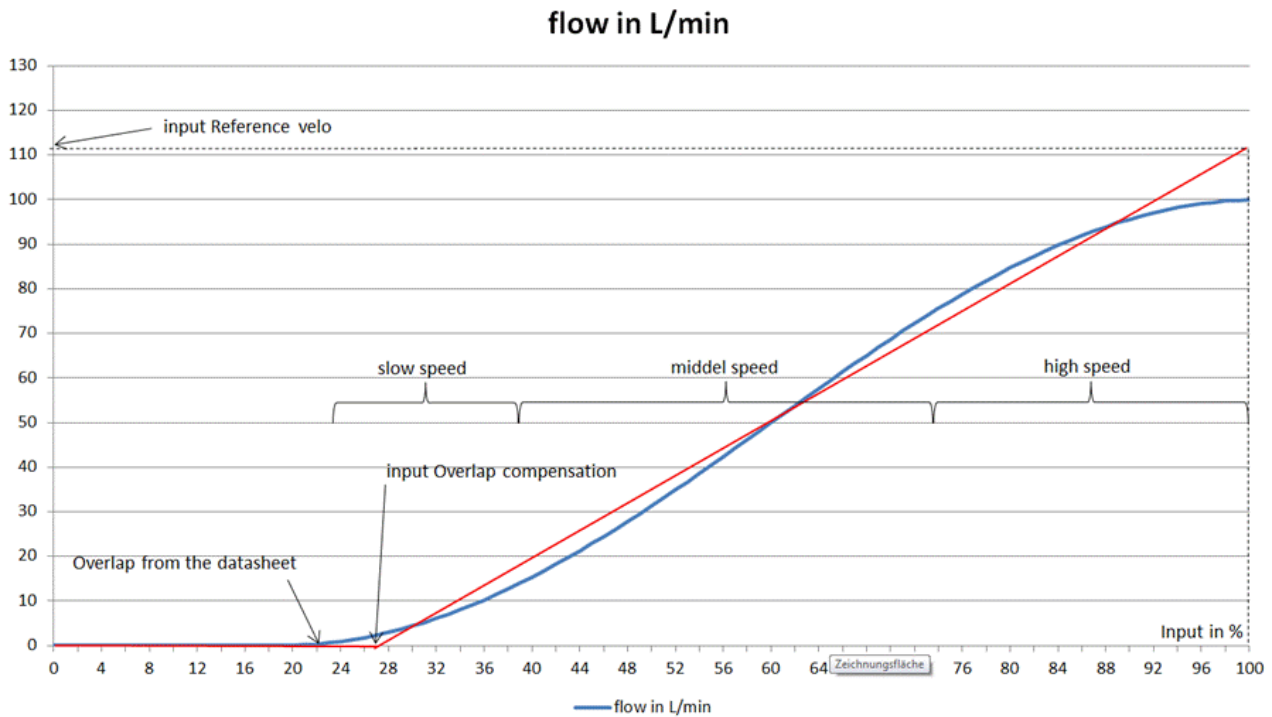
Um die Dynamik der Achse zu optimieren sind verschiedene Einstellungen vorzunehmen. Es gibt mehrere Vorgehensweisen um die Achse zu optimieren. Wird die Achse über ein Ventil geregelt besteht die Möglichkeit von:

- Linearisierung der Kennlinie
Die Kennlinie wird über wenige Kennwerte angenähert.
- Automatische Kennlinienermittlung
Die Kennlinie der Achse wird automatisch ermittelt.

Linearisierung der Kennlinie

Bei der Linearisierung wird eine Ventilkennlinie über wenige Parameter angenähert.

- Überdeckung (Overlap)
- Bezugsgeschwindigkeit



Anpassung der Überdeckung:

HINWEIS! Die Überdeckungskompensation und die Knick-Kompensation sind inkompatible. Soll die Überdeckungskompensation genutzt werden muss die Knick-Kompensation Null sein!

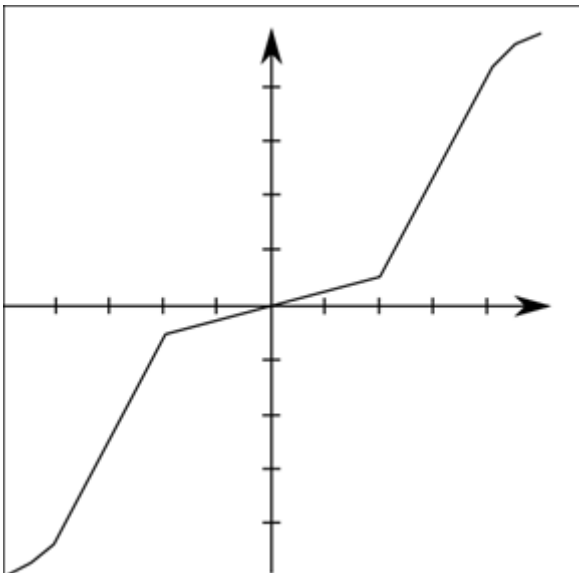
In der Regel ist die Überdeckung aus den Ventildatenblättern so, dass ab diesem Punkt gerade ein Ölfluss einsetzt, aber die Achse sich nicht wesentlich bewegt. Um die richtige Überdeckung für diese Achse zu ermitteln sind folgende Schritte durchzuführen:

- Die Sollgeschwindigkeit langsam erhöhen, bis eine Reaktion an der Istgeschwindigkeit zu erkennen ist.
- Werden unterschiedliche Geschwindigkeitssollwerte benötigt um die Achse in positiver bzw. negativer Richtung aus dem Stillstand in Bewegung zu versetzen deutet dies auf ein nicht symmetrisches Ventil hin. In diesem Fall muss in den Globalen Einstellungen der Haken "Asym" gesetzt werden. Nun kann die Parametrierung für das Ventil in positive und negative Richtungen vorgenommen werden.
- Die Ausgabe in Prozent, bei der sich die Achse bewegt, muss im Karteireiter "Ventil" unter "Überdeckung" eingetragen werden. Bei einem nicht symmetrischen Ventil, ist darauf zu achten, dass die Eintragung im richtigen Feld vorgenommen wird, wobei im ersten Feld die Überdeckung für die positive Richtung und im zweiten Feld für die negative Richtung erwartet wird.
- Nach dieser Optimierung muss die Achse bei verschiedenen kleinen Geschwindigkeiten reagieren. Dabei ist nicht wichtig, ob die Achse mit der richtigen Geschwindigkeit reagiert.

Anpassung der Knick-Kompensation

HINWEIS! Die Knick-Kompensation und die Überdeckungskompensation sind inkompatible. Soll die Knick-Kompensation genutzt werden muss die Überdeckungskompensation Null sein!

Diese Einstellung ist nur bei Ventilen vorzunehmen, bei denen die Volumenstromkennlinie aus zwei annähernd linearen Kennlinien mit unterschiedlicher Steigung zusammengesetzt ist.



Diese Einstellung darf nur vorgenommen werden, wenn es sich um ein Nullschnittventil mit geknickter Kennlinie handelt, also die Überdeckung null ist! Die "Knick-Ausgabe" und die "Knick-Geschw." müssen in Prozent angegeben werden. Beide Angaben sind aus der Volumenstromkennlinie des Ventil-Datenblattes zu entnehmen. Die "Knick-Geschw." lässt sich durch $Q_{\text{knick}} / Q_{\text{nom}}$ berechnen und "Knick-Ausgabe" wird oft schon in Prozent angegeben.

Anpassung der Bezugsgeschwindigkeit:

- Um die Bezugsgeschwindigkeit zu ermitteln, muss die Schleichstrecke auf mehrere 10mm gestellt werden und die Schleichgeschwindigkeit auf 10% der Bezugsgeschwindigkeit. Die Schleichgeschwindigkeit sollte beim Verändern der Bezugsgeschwindigkeit mit angepasst werden.
- Die Sollgeschwindigkeit muss langsam erhöht werden und dabei muss kontrolliert werden, ob die Achse der vorgegeben Geschwindigkeit folgt.
 - ▣ **HINWEIS! Diese Messung darf nur mit der kleinen Kolbenfläche ausgewertet werden!**
- Um gerade bei mittleren Geschwindigkeiten vor dem mechanischen Anschlag anhalten zu können, kann auch mit absoluten Verfahrenswegen gearbeitet werden. Hierzu muss in Position der Zielpunkt und in Geschwindigkeit die Sollgeschwindigkeit angegeben werden. Über Start wird die Bewegung aktiviert. Für eine einfache Auswertung sollte das beigefügte Scope mitverwendet werden um anschließend die Geschwindigkeit auslesen zu können.



Achtung

Achsbewegung

Die Softwareendschalter sollten der Steuerung vorher übergeben worden sein, damit die Achse nicht bei voller Fahrt gegen den Endanschlag fährt.

▣ **HINWEIS! Falls die geforderte Geschwindigkeit nicht erreicht wird muss die Beschleunigung bzw. Verzögerung angepasst werden.**

- Ist die Sollgeschwindigkeit wesentlich größer als die Istgeschwindigkeit, so muss die „Reference velo“ verkleinert werden.
Ist die Sollgeschwindigkeit wesentlich kleiner als die Istgeschwindigkeit, so muss die „Reference velo“ vergrößert werden
- Stimmen die mittleren bis hohen Soll- und Istgeschwindigkeit annähert überein, so ist die passende Bezugsgeschwindigkeit gefunden.

▣ **HINWEIS! Die Bezugsgeschwindigkeit muss nicht mit der realen maximalen Geschwindigkeit der Achse übereinstimmen.**

Anpassen des Geschwindigkeitsverhältnisses:

In der Regel stimmen die mittleren Geschwindigkeiten für die Richtung mit der großen Kolbenfläche nicht mit der Istgeschwindigkeit überein. Das liegt an den unterschiedlichen Zylinderflächen. Zum Ermitteln der „velo ratio“ muss in beide Richtungen ein Fahrprofil mit einer mittlerer Geschwindigkeit durchgeführt werden. Dieses Verhältnis ist im Reiter Ventil unter „velo ratio“ einzutragen.

Anpassen der Schleichphase:

Die Ziel- und Schleichphase ist wesentlich für die Positioniergenauigkeit und Performance der Achse verantwortlich. Diese Phase dient dazu die weggesteuerte Achse vor dem Ziel wieder besser zu kontrollieren.

- Die **Schleichgeschwindigkeit** ist so zu wählen, dass die Achse mit dieser Vorgabe noch sicher in die korrekte Richtung fährt. Wird ein zu hoher Wert verwendet verschlechtert dies die Wiederholgenauigkeit und/oder das Überspringen im Ziel. Ein zu geringer Wert kann dazu führen, dass die Achse vor dem Ziel stehen bleibt oder sogar zurückdriftet und nicht als 'im Ziel' gemeldet wird.
- Als **Schleichstrecke** ist ein Weg vorzugeben, innerhalb dessen sich die Istgeschwindigkeit nach dem Abbremsen auf Schleichgeschwindigkeit auch unter ungünstigen Voraussetzungen in etwa eingependelt hat. Ein zu hoher Wert wirkt sich negativ auf den Zeitbedarf jeder Bewegung aus. Ein zu niedriger Wert führt dazu, dass die Achse zumindest unter ungünstigen Voraussetzungen das Ziel überfährt.
- Unterschreitet die Reststrecke den Wert von **Bremsstrecke** geht der Generator in den Ruhezustand über. Die Vorgehensweise bei der Inbetriebnahme hat zu berücksichtigen, ob dann ein Lageregler aktiv wird:
 - Ist eine Regelverstärkung [► 280] > 0.0 eingestellt wird der Lageregler automatisch aktiv. Zunächst ist die Bremsstrecke eher groß zu wählen und die Regelverstärkung so einzustellen, dass die Achse im Ziel nicht übermäßig schwingt. Anschließend ist die Bremsstrecke auf einen angemessenen Wert zu verkürzen. Der nachstehende Scopeview zeigt die Wirkung unterschiedlich langer Bremsstrecken.

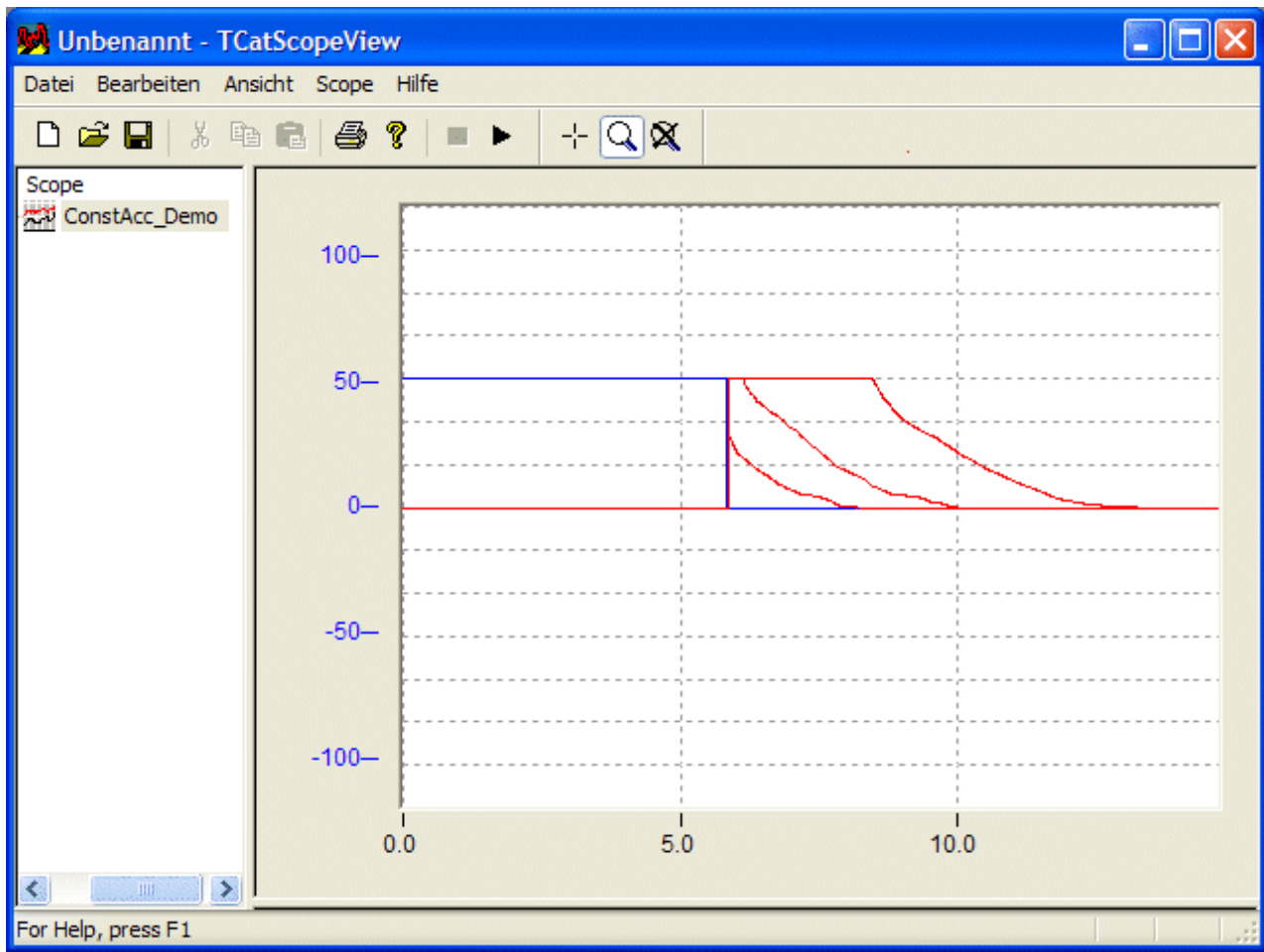
Die untere rote Kurve zeigt die Wirkung einer ungünstigen Kombination von Bremsstrecke und Lageregelung. Hier wird die Ausgabe sprunghaft geändert.

Die obere Kurve zeigt eine unnötig lange Bremsstrecke. Die Ausgabe des Lagereglers wird auf die Schleichgeschwindigkeit begrenzt und konnte auch durch den Fahrgenerator bereitgestellt werden. Eine kürzere Bremsstrecke würde eine kürzere Schleichstrecke ermöglichen und so die Positionierzeit verkürzen.

Die mittlere Linie ist optimal, da es keinen sprunghaftigen Übergang zwischen den beiden Phasen gibt und der Lageregler die Achse schnell ins Ziel bringt.

- Wird im Ruhezustand kein Lageregler aktiv (Regelverstärkung [► 280]= 0.0) ist die Bremsstrecke so zu wählen, dass die Achse nach dem Ausschwingen im Mittel im Ziel steht. Dazu sind wiederholte Fahrten durchzuführen und der mittlere Fehler im Ziel zu beurteilen. Sollte die Achse in Fahrtrichtung betrachtet im Mittel vor dem Ziel stehen bleiben ist die Bremsstrecke zu verkürzen, andernfalls zu verlängern.
- Nachdem ein zufrieden stellendes Verhalten im Ziel erreicht wurde kann die Positionierleistung der Achse optimiert werden. Dazu ist jetzt schrittweise Beschleunigung und Verzögerung zu erhöhen. In der Praxis wird die Istgeschwindigkeit der Sollgeschwindigkeit mit einer erkennbaren Verzögerung folgen. In der Beschleunigungsphase ist dies nicht unbedingt wichtig da keine Schleppüberwachung vorgenommen wird. Während der Bremsphase wird die Achse verzögert reagieren und mit einem gewissen Fehler der Bremsrampe folgen. Während der Bremsphase führt der auftretende Zeitverzug dazu, dass die Achse zu Beginn der Schleichphase nicht auf Schleichgeschwindigkeit abgebremst hat.
- Sollte die Wiederholgenauigkeit im Ziel zu schlecht sein kann dies durch zu große Streuungen der Istgeschwindigkeit während der Schleichfahrt verursacht werden. Dies kann durch die Vorgabe einer **Bremstotzeit** verbessert werden.

☐ HINWEIS! Die Bremsstrecke ist ein Teil der Schleichstrecke, d.h. die Schleichstrecke muss immer größer als die Bremsstrecke eingestellt werden.



Während einer aktiven Bewegung beobachtet der Profil-Generator ständig der richtungsspezifische Zielfenstermarken. Wird der Nocken erkannt geht der Generator automatisch in die Zielfahrt über. Dabei wird durch `fStopRamp` in `Axis.ST TchHydAxParam [▶ 88]` festgelegt, wie schnell der Stellwert auf `fCreepSpeed` abgesenkt wird. Den Parameter finden Sie auf dem Global-Reiter des `PlcMcManagers`.

Sobald der Zielfenstermarken erkannt wird geht der Generator in den Ruhezustand über. Dabei kann die Ausgabe eines Ruhestellwerts erfolgen um z.B. das Wegdriften der Achse zu verhindern oder um sie an einem Festanschlag anzupressen. Dazu ist ein entsprechender Wert auf dem Ventil-Reiter des `PlcMcManagers` einzustellen. Ist eine solche Ausgabe nicht gefordert ist der Parameter auf 0 zu setzen.

HINWEIS! Die Schleichgeschwindigkeit kann ab Version V3.0.8 richtungsabhängig gemacht werden. Dazu ist das Kontrollfeld "Asym." (für asymmetrische Zielfahrt) zu markieren und diese Einstellung zu aktivieren. Dadurch wird ein zweiter Satz von Eingabefeldern sichtbar. Zu Beginn der Inbetriebnahme sollte diese Möglichkeit zunächst abgewählt werden und die Parameter für die Bewegung in positiver Richtung optimiert werden. Anschließend können die Parameter für die negative Fahrtrichtung bei Bedarf nachoptimiert werden.

Kontrolle der Achsparameter:

Nach dieser Inbetriebnahme muss die Achse kleine und mittlere Geschwindigkeiten gut folgen. Bei hohen Geschwindigkeiten ist ein gutes Folgen der Achse oft nicht mehr möglich.

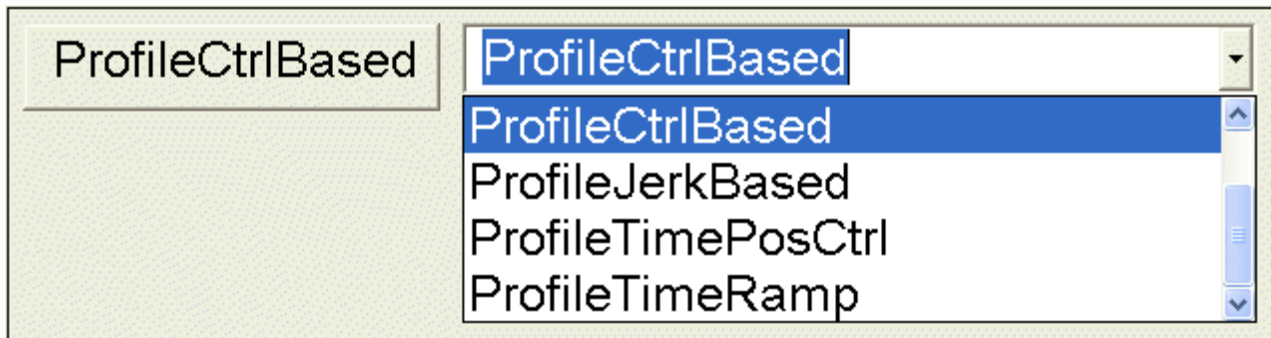
Wird die Geschwindigkeit langsam zu erhöhen, muss die Achse der Sollgeschwindigkeit Annäherung folgen. Wichtig ist, dass die Beschleunigung gering ist um Dynamikeffekte zu vermeiden.

4.6.4.4 Positionierung

Die aktive Positionierung einer Achse erfolgt durch einen Profil-Generator. In der Library wird diese Aufgabe durch den [MC_AxRuntime_BkPlcMc](#) [► 155] Baustein übernommen. Auf dem [Allgemein-Reiter](#) [► 242] des PlcMcManagers ist der gewünschte Generator-Typ vorzugeben.

HINWEIS! Die Inbetriebnahme einer Achse wird durch eine Aufzeichnung mit dem TwinCAT ScopeView wesentlich erleichtert. Siehe dazu [FAQ #15](#). [► 218]

HINWEIS! Die hier angebotenen Profilgeneratoren bieten zum Teil nicht alle von der PLCopen beschriebenen Möglichkeiten und Funktionalitäten. Es sollten die Hinweise in der Beschreibung des [MC_AxRuntime_BkPlcMc](#) Bausteins beachtet werden.



Profiltyp [iTcMc_ProfileCtrlBased](#) [iTcMc_ProfileCtrlBased](#) [► 158]

in Vorbereitung

Beschleunigung-Phase

In der Beschleunigungsphase einer Bewegung wird der Stellwert der Achse mit einer zeitgeführten Rampe aufgebaut. Diese Rampe wird durch das Verhältnis von Bezugsgeschwindigkeit und Beschleunigung festgelegt. Näheres dazu finden Sie unter "Inbetriebnahme: [Antrieb](#) [► 264]".

Die Beschleunigungsphase endet, wenn der Stellwert den vorgegebenen Wert (z. B. Velocity Parameter eines [MC_MoveAbsolute_BkPlcMc](#) Bausteins) erreicht hat oder der Profil-Generator die Zielfahrt einleitet.

Zielfahrt

Während einer aktiven Bewegung beobachtet der Profil-Generator ständig die verbleibende Strecke bis zur Zielposition. Ist diese Reststrecke kleiner oder gleich der unter Berücksichtigung der Parameter benötigten Anhaltstrecke geht der Generator automatisch in die Zielfahrt über. Die dabei verwendeten Parameter finden Sie auf dem [Global-Reiter](#) des PlcMcManagers.

HINWEIS! Die Parameter **Schleichgeschwindigkeit**, **Schleichstrecke**, **Bremsstrecke** und **Bremstzeit** können ab Version V3.0.8 richtungsabhängig gemacht werden. Dazu ist das Kontrollfeld "Asym." (für asymmetrische Zielfahrt) zu markieren und diese Einstellung zu aktivieren. Dadurch wird für die genannten Parameter ein zweiter Satz von Eingabefelder sichtbar. Zu Beginn der Inbetriebnahme sollte diese Möglichkeit zunächst abgewählt werden und die Parameter für die Bewegung in positiver Richtung optimiert werden. Anschließend können die Parameter für die negative Fahrtrichtung bei Bedarf nachoptimiert werden.

Empfohlene Vorgehensweise bei der Inbetriebnahme:

1. Vorbedingungen:
 - Es sind die erforderlichen **Sicherheitsmaßnahmen** durchzuführen.
 - Die Encoder-Inbetriebnahme muss erfolgt sein.
 - Zumindest die [Voreinstellungen der Antriebs-Inbetriebnahme](#) [► 264] sollte erfolgt sein.

- Die Achse muss bewegt werden können.
2. Voreinstellungen:
- Zunächst ist die **Bremstotzeit** auf 0.0 zu stellen.
 - Die Beschleunigung der Achse sollte auf einen eher kleinen Wert eingestellt werden.
 - Die Verzögerung der Achse sollte auf einen eher kleinen Wert eingestellt werden.
 - Die **Schleichstrecke** ist vorerst so groß zu wählen, dass die Schleichfahrt überdeutlich zu erkennen ist.
 - Ist eine Lageregelung im Stillstand [▶ 280] möglich und gewünscht sollte die Regelverstärkung auf einen kleinen Wert gestellt werden um ein Driften zu verhindern.
 - Die **Bremsstrecke** ist auf einen eher großen Wert (einige mm) einzustellen.
3. Optimierung
- Die **Schleichgeschwindigkeit** ist so zu wählen, dass die Achse mit dieser Vorgabe noch sicher in die korrekte Richtung fährt. Wird ein zu hoher Wert verwendet verschlechtert dies die Wiederholgenauigkeit und/oder das Überspringen im Ziel. Ein zu geringer Wert kann dazu führen, dass die Achse vor dem Ziel stehen bleibt oder sogar zurückdriftet und nicht als 'im Ziel' gemeldet wird.
 - Als **Schleichstrecke** ist ein Weg vorzugeben, innerhalb dessen sich die Istgeschwindigkeit nach dem Abbremsen auf Schleichgeschwindigkeit auch unter ungünstigen Voraussetzungen in etwa eingependelt hat. Ein zu hoher Wert wirkt sich negativ auf den Zeitbedarf jeder Bewegung aus. Ein zu niedriger Wert führt dazu, dass die Achse zumindest unter ungünstigen Voraussetzungen das Ziel überfährt.
 - Unterschreitet die Reststrecke den Wert von **Bremsstrecke** geht der Generator in den Ruhezustand über. Die Vorgehensweise bei der Inbetriebnahme hat zu berücksichtigen, ob dann ein Lageregler aktiv wird:

Ist eine Regelverstärkung [▶ 280] > 0.0 eingestellt wird der Lageregler automatisch aktiv. Zunächst ist die Bremsstrecke eher groß zu wählen und die Regelverstärkung so einzustellen, dass die Achse im Ziel nicht übermäßig schwingt. Anschließend ist die Bremsstrecke auf einen angemessenen Wert zu verkürzen. Der nachstehende Scope View zeigt die Wirkung unterschiedlich langer Bremsstrecken.

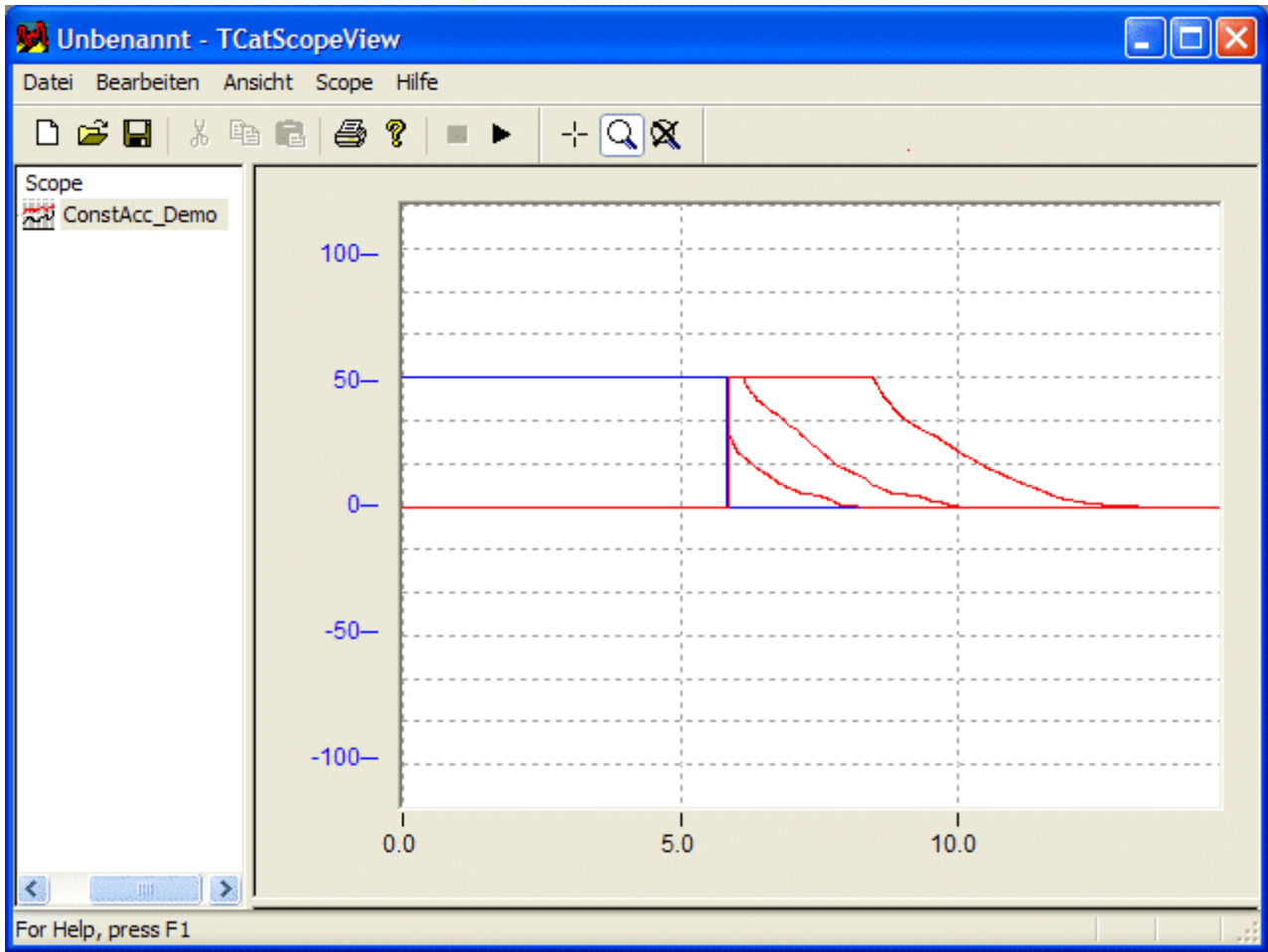
Die untere rote Kurve zeigt die Wirkung einer ungünstigen Kombination von Bremsstrecke und Lageregelung. Hier wird die Ausgabe sprunghaft geändert.

Die obere Kurve zeigt eine unnötig lange Bremsstrecke. Die Ausgabe des Lagereglers wird auf die Schleichgeschwindigkeit begrenzt und konnte auch durch den Fahrgenerator bereitgestellt werden. Eine kürzere Bremsstrecke würde eine kürzere Schleichstrecke ermöglichen und so die Positionierzeit verkürzen.

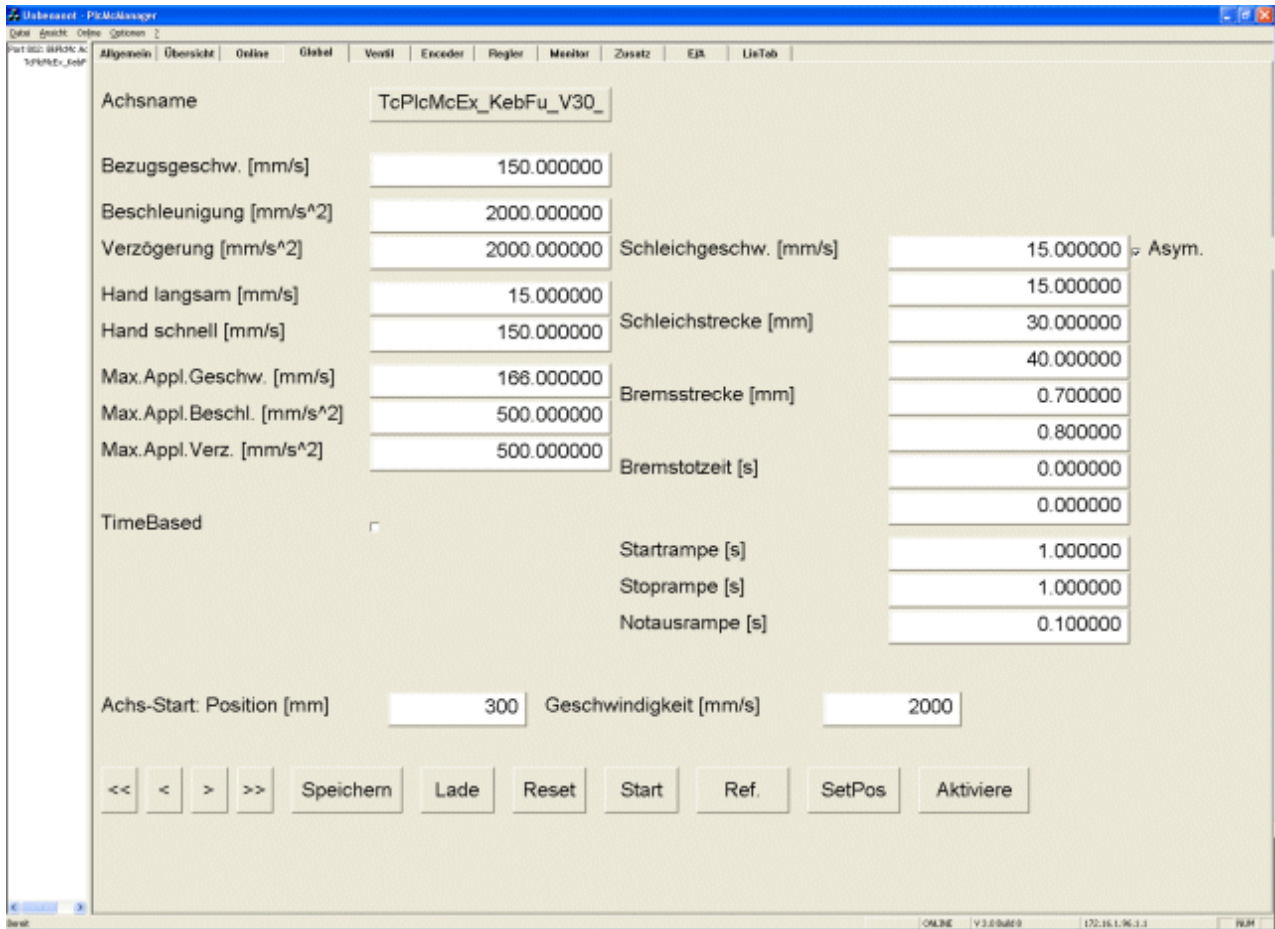
Wird im Ruhezustand kein Lageregler aktiv (Regelverstärkung [▶ 280]= 0.0) ist die Bremsstrecke so zu wählen, dass die Achse nach dem Ausschwingen im Mittel im Ziel steht. Dazu sind wiederholte Fahrten durchzuführen und der mittlere Fehler im Ziel zu beurteilen. Sollte die Achse in Fahrtrichtung betrachtet im Mittel vor dem Ziel stehen bleiben ist die Bremsstrecke zu verkürzen, andernfalls zu verlängern.

- Nachdem ein zufrieden stellendes Verhalten im Ziel erreicht wurde kann die Positionierleistung der Achse optimiert werden. Dazu ist jetzt schrittweise Beschleunigung und Verzögerung zu erhöhen. In der Praxis wird die Istgeschwindigkeit der Sollgeschwindigkeit mit einer erkennbaren Verzögerung folgen. In der Beschleunigungsphase ist dies nicht unbedingt wichtig da keine Schleppüberwachung vorgenommen wird. Während der Bremsphase führt der auftretende Zeitverzug dazu, dass die Achse zu Beginn der Schleichphase nicht auf Schleichgeschwindigkeit abgebremst hat.

Sollte die Wiederholgenauigkeit im Ziel zu schlecht sein kann dies durch zu große Streuungen der Istgeschwindigkeit während der Schleichfahrt verursacht werden. Dies kann durch die Vorgabe einer **Bremstotzeit** verbessert werden.



Beispiel 1: Global-Reiter bei Profiltyp iTcMc_ProfileCtrlBased



Profiltyp iTcMc_ProfileJerkBased

in Vorbereitung

Profiltyp iTcMc_ProfileTimeRamp

Der Profiltyp [iTcMc_ProfileTimeRamp \[► 160\]](#) kann sinnvoll nur in Kombination mit dem Encodertyp [iTcMc_EncoderDigCam \[► 132\]](#) verwendet werden. Siehe auch [Sonderfall digitale Positionsnocken \[► 263\]](#).

Beschleunigungs-Phase

In der Beschleunigungsphase einer Bewegung wird der Stellwert der Achse mit einer zeitgeführten Rampe aufgebaut. Diese Rampe wird durch `fStartRamp` in [Axis.ST_TcHydAxParam \[► 88\]](#) festgelegt. In der hier eingestellten Zeit wird der Stellwert auf die Bezugsgeschwindigkeit erhöht. Ist eine geringere Fahrgeschwindigkeit gefordert wird diese in einer proportional verkürzten Zeit erreicht. Den Parameter finden Sie auf dem Global-Reiter des PlcMcManagers.

Die Beschleunigungsphase endet, wenn der Stellwert den vorgegebenen Wert (z. B. Velocity Parameter eines MC_MoveAbsolute_BkPlcMc Bausteins) erreicht hat oder der Profil-Generator die Zielfahrt einleitet.

Zielfahrt

Während einer aktiven Bewegung beobachtet der Profil-Generator ständig der richtungsspezifische Zielfensternocken. Wird der Nocken erkannt geht der Generator automatisch in die Zielfahrt über. Dabei wird durch `fStopRamp` in [Axis.ST_TcHydAxParam \[► 88\]](#) festgelegt, wie schnell der Stellwert auf `fCreepSpeed` abgesenkt wird. Den Parameter finden Sie auf dem Global-Reiter des PlcMcManagers.

Sobald der Zielnocken erkannt wird geht der Generator in den Ruhezustand über. Dabei kann die Ausgabe eines Ruhestellwerts erfolgen um z.B. das Wegdriften der Achse zu verhindern oder um sie an einem Festanschlag anzupressen. Dazu ist ein entsprechender Wert auf dem Ventil-Reiter des PlcMcManagers einzustellen. Ist eine solche Ausgabe nicht gefordert ist der Parameter auf 0 zu setzen.

☐ HINWEIS! Die Schleichgeschwindigkeit kann ab Version V3.0.8 richtungsabhängig gemacht werden. Dazu ist das Kontrollfeld "Asym." (für asymmetrische Zielfahrt) zu markieren und diese Einstellung zu aktivieren. Dadurch wird ein zweiter Satz von Eingabefeldern sichtbar. Zu Beginn der Inbetriebnahme sollte diese Möglichkeit zunächst abgewählt werden und die Parameter für die Bewegung in positiver Richtung optimiert werden. Anschließend können die Parameter für die negative Fahrtrichtung bei Bedarf nachoptimiert werden.

Sonderfälle

[Frequenzumrichter](#) [▶ 274] mit analoger oder digitaler Schnittstelle.

4.6.4.5 Sonderfälle

4.6.4.5.1 Positionierung mit Frequenzumrichtern

Wird die Kombination aus einem Frequenzumrichter und einem Asynchronmotor als Antrieb eingesetzt sind einige Besonderheiten zu berücksichtigen. Diese sind zum Teil auf die speziellen Eigenschaften der Komponenten, zum Teil aber auch auf die Art der Verbindung mit der Steuerung zurück zu führen. Als Profiltyp sollte bevorzugt **iTcMc_ProfileCtrlBased** eingesetzt werden.

Frequenzumrichter mit Analog-Schnittstelle

in Vorbereitung

Frequenzumrichter mit Digital-Schnittstelle

Diese Geräte werden ausschließlich über digitale Signale gesteuert. Als nDrive_Type ist **iTcMc_DriveLowCostInverter** einzustellen. Bei der [Verknüpfung](#) [▶ 122] zu den E/As und der Verdrahtung zum Gerät sind zwei Varianten denkbar, die eventuell Anpassungen der bereitgestellten Signale erfordern.

Frequenzen und Geschwindigkeiten

Im Umrichter sind zwei Festfrequenzen zu programmieren, durch die eine Eilgang- und eine Schleichgang-Geschwindigkeit definiert werden. Die Eilgang-Frequenz liegt typischerweise im Bereich 50 bis 70 Hz. Für die Schleichgang-Frequenz ist ein Wert zu wählen, der in jeder zu erwartenden Lastsituation sicher eine Bewegung in die korrekte Richtung erzeugt. Typische Werte liegen im Bereich 3 bis 10 Hz.

☐ HINWEIS! Ein unnötig hoher Wert verschlechtert die Positioniergenauigkeit.

Um die Frequenz in eine Geschwindigkeit umzurechnen ist die folgende Formel zu verwenden:

$$V := F / N_{pp}$$

mit:

V = Geschwindigkeit in mm/s

F = Frequenz in Hz

N_{pp} = Polpaarzahl

Durch Umrechnung der Eilgang-Frequenz ergibt sich die Bezugs- und Maximal-Geschwindigkeit. Der erhaltene Wert ist als **fRefVelo** und als **fMaxVelo** einzutragen. Aus der Schleichgang-Frequenz ist die Schleichgeschwindigkeit **fCreepSpeed** zu errechnen.

☐ HINWEIS! Die so errechneten Geschwindigkeiten vernachlässigen den Schlupf. Im realen Betrieb ergibt sich eine Beschleunigung.

Beschleunigung

Die Rampen werden bei dieser Betriebsart durch den Umrichter vorgegeben. Dazu sind als Beschleunigungszeit und Verzögerungszeit geeignete Werte einzustellen, die vom Umrichter unter allen zu erwartenden Lastsituationen bewältigt werden können. In den Achsparametern kann für **fAcc** und **fMaxAcc** ein beliebiger Wert eingetragen werden, da er nicht verwendet wird. Aus Gründen der Nachvollziehbarkeit sollte jedoch der eine Wert eingestellt werden, der wie folgt berechnet wird:

$$fAcc := fMaxAcc := fRefVelo / Beschleunigungszeit_im_Umrichter$$

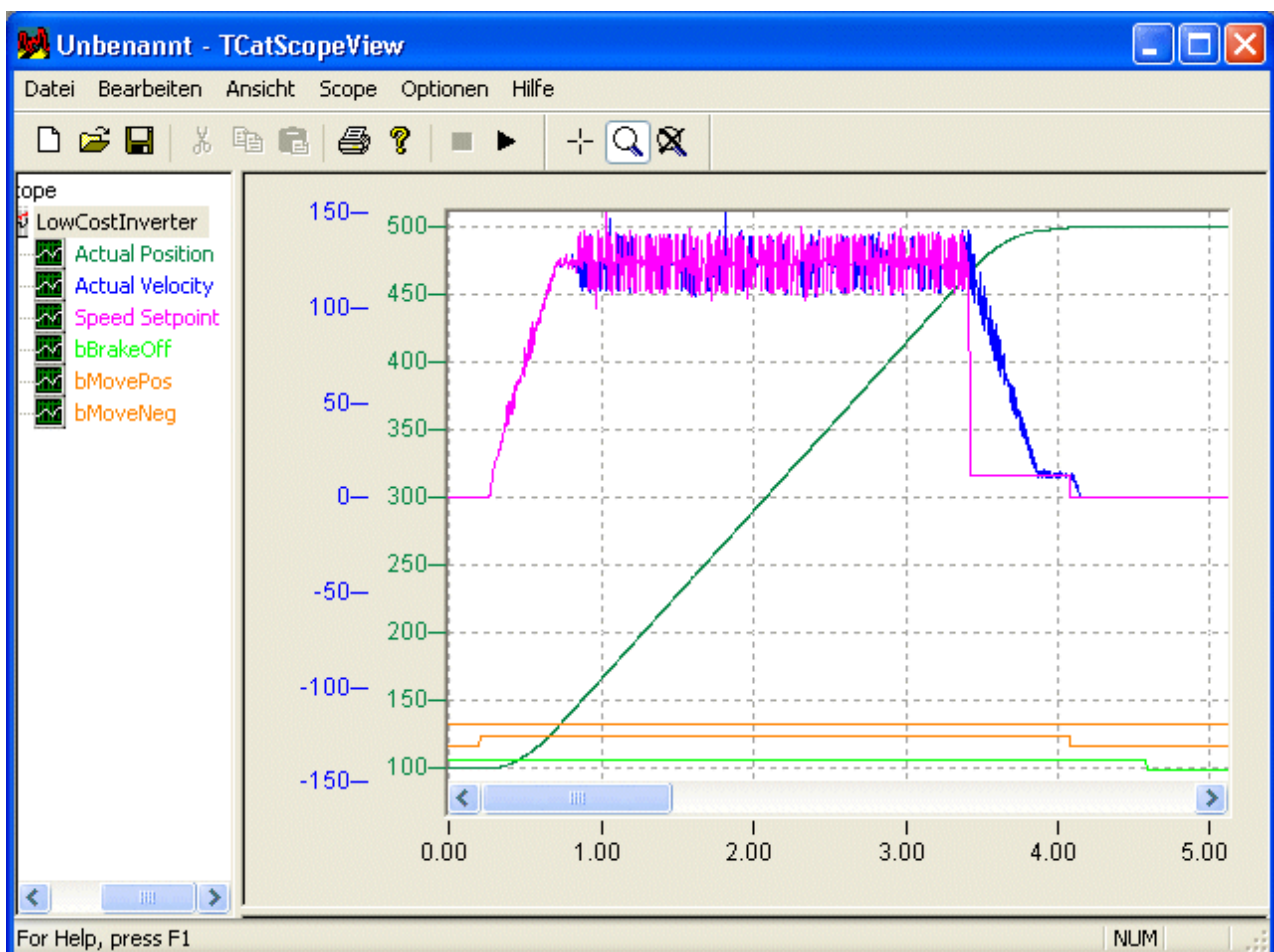
Verzögerung

Für die Verzögerung **fDec** und **fMaxDec** sowie die Schleichstrecke **fCreepDistance** sind zwei Inbetriebnahme-Strategien möglich.

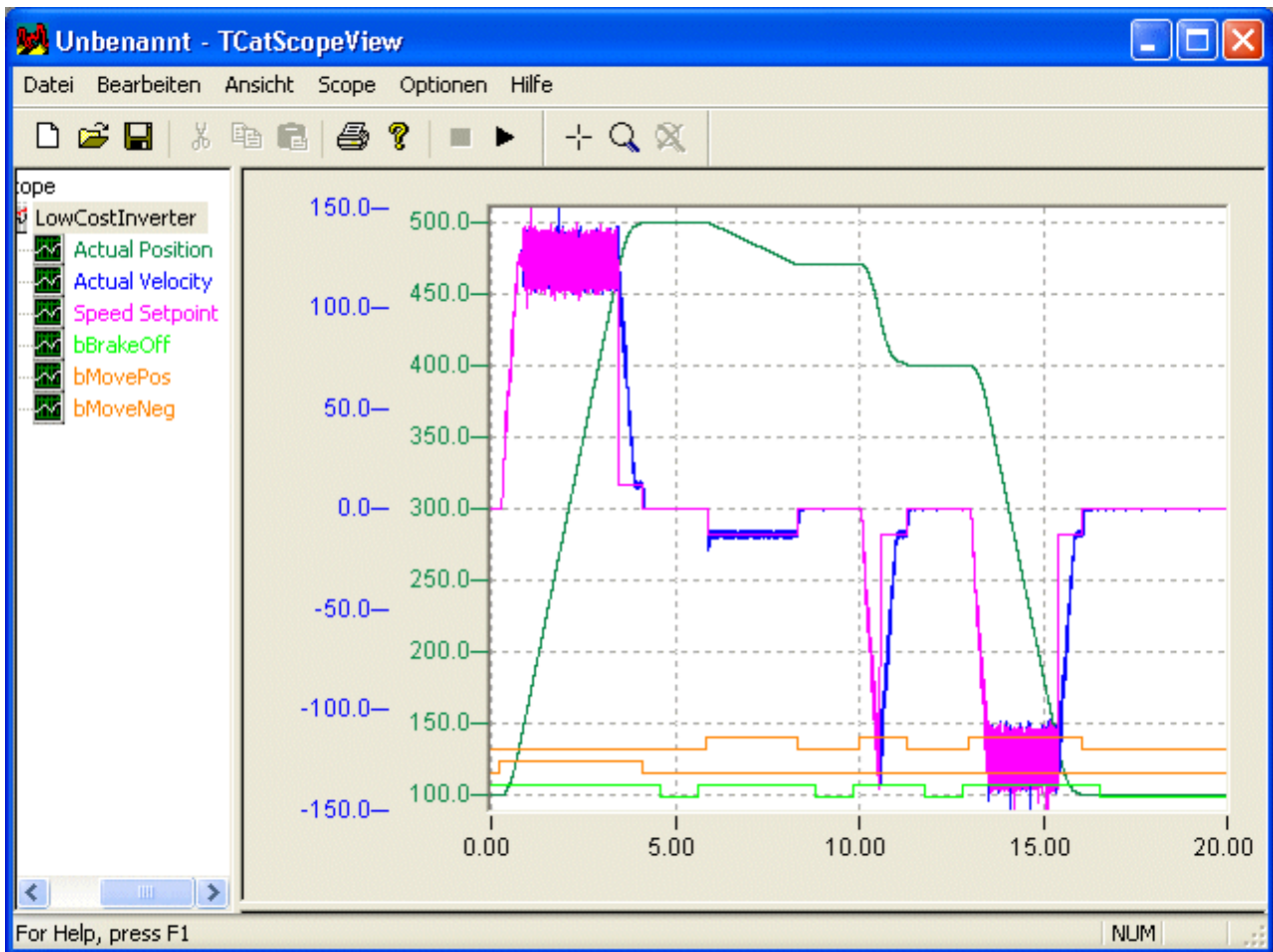
Die konventionelle Eil/Schleich-Einstellung

Es wird ein sehr hoher Wert für die Verzögerung eingestellt. Um einen solchen Wert zu wählen kann zum Beispiel das 10-fache der Bezugsgeschwindigkeit durch die Verzögerungszeit des Umrichters dividiert werden. Für die **fCreepDistance** wird ein Wert gewählt, bei dem sich auch im ungünstigsten Fall (maximale Geschwindigkeit wird erreicht, maximale Last, maximale in Bewegungsrichtung wirkende Prozesskraft) eine sicher erkennbare Kriechstrecke ergibt.

Beispiel:

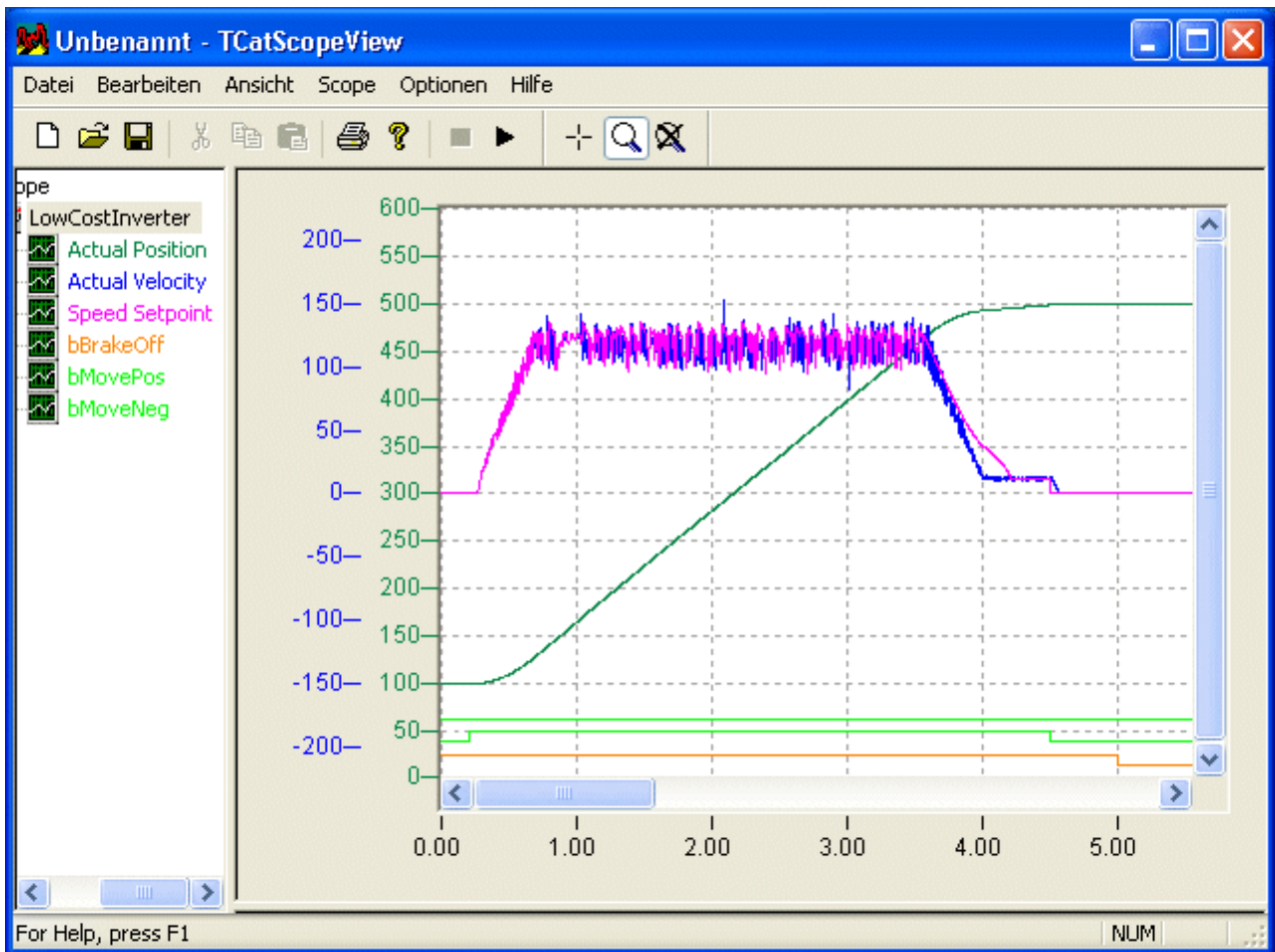


Der Vorteil dieser Vorgehensweise ist die recht einfache Inbetriebnahme. Allerdings werden bereits etwas kürzere Fahrwege nur oder zu einem unnötig großen Teil im Schleichgang gefahren.

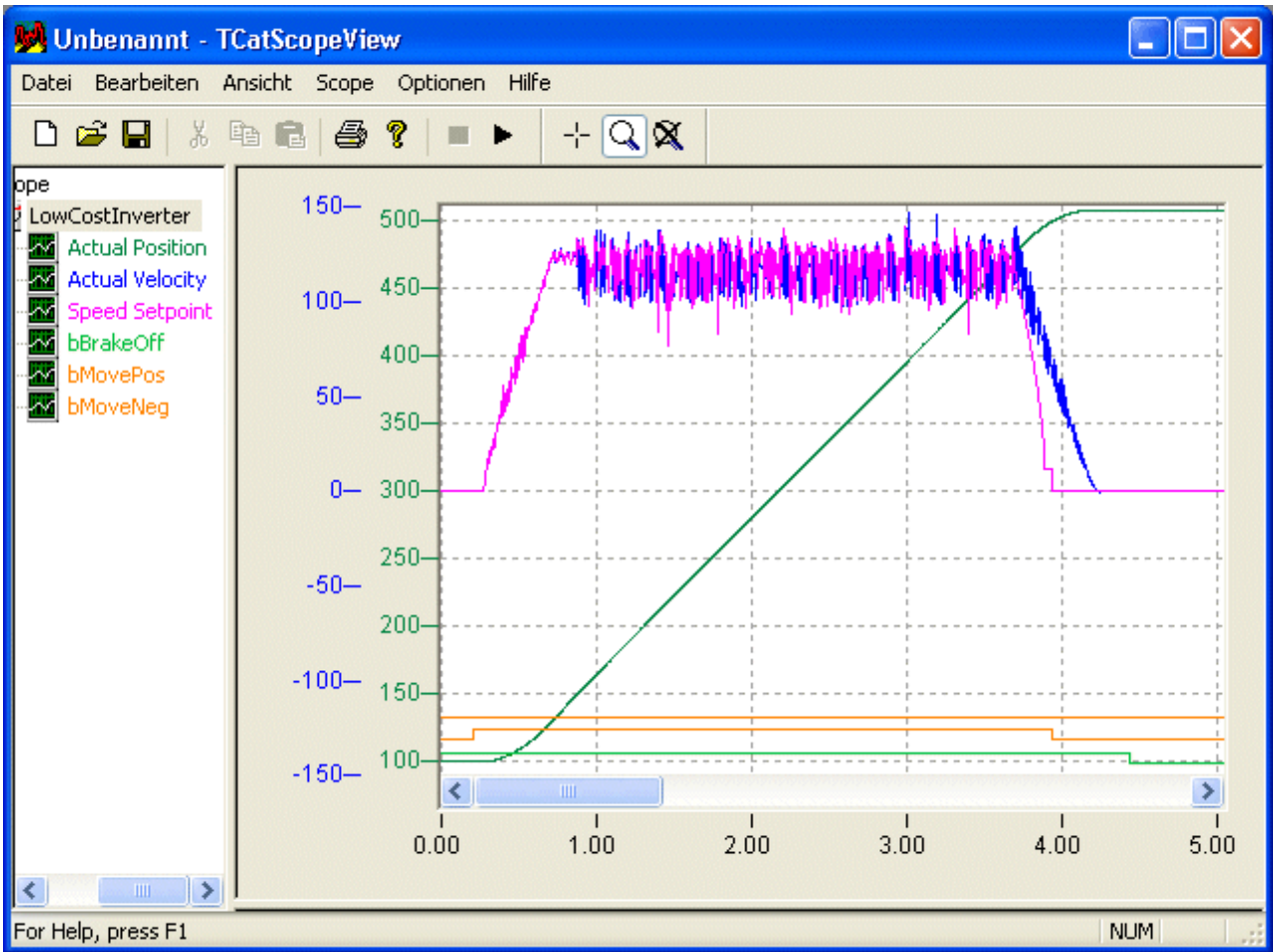


Die vorausschauende Eil/Schleich-Einstellung

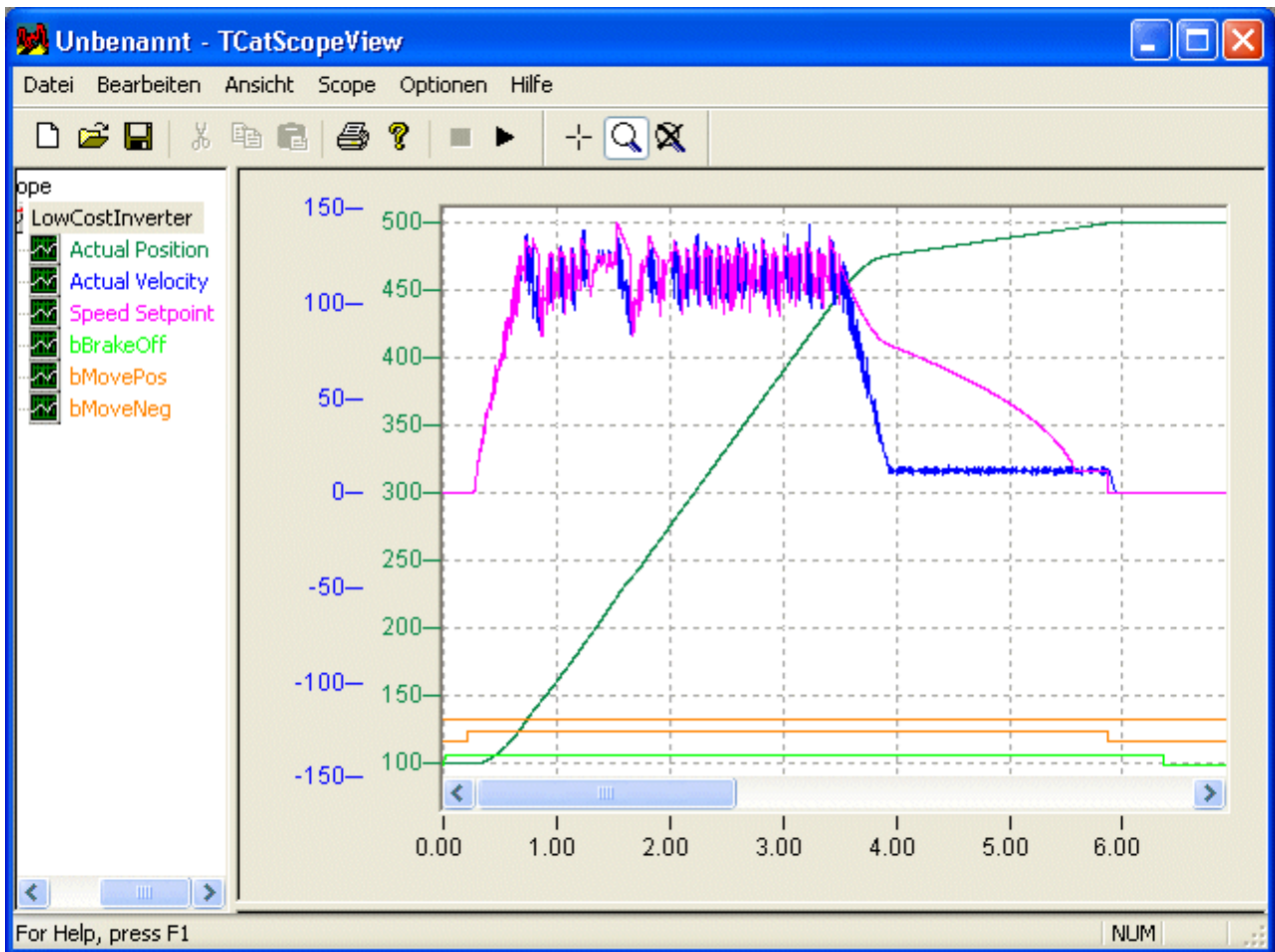
Ausgehend von einer konventionellen Einstellung wie oben beschrieben wird jetzt die Einstellung für **fDec** so lange reduziert bis im ungünstigsten Fall der Geschwindigkeitsstellwert nicht mehr erkennbar steiler verläuft als die Istgeschwindigkeit. Der nachstehende Scope View zeigt ein Beispiel mit einer noch ausreichenden Reserve. Anschließend kann der Wert in **fCreepDistance** wie im Beispiel gezeigt so reduziert werden, dass noch ein sicher erkennbarer Schleichgang stattfindet.



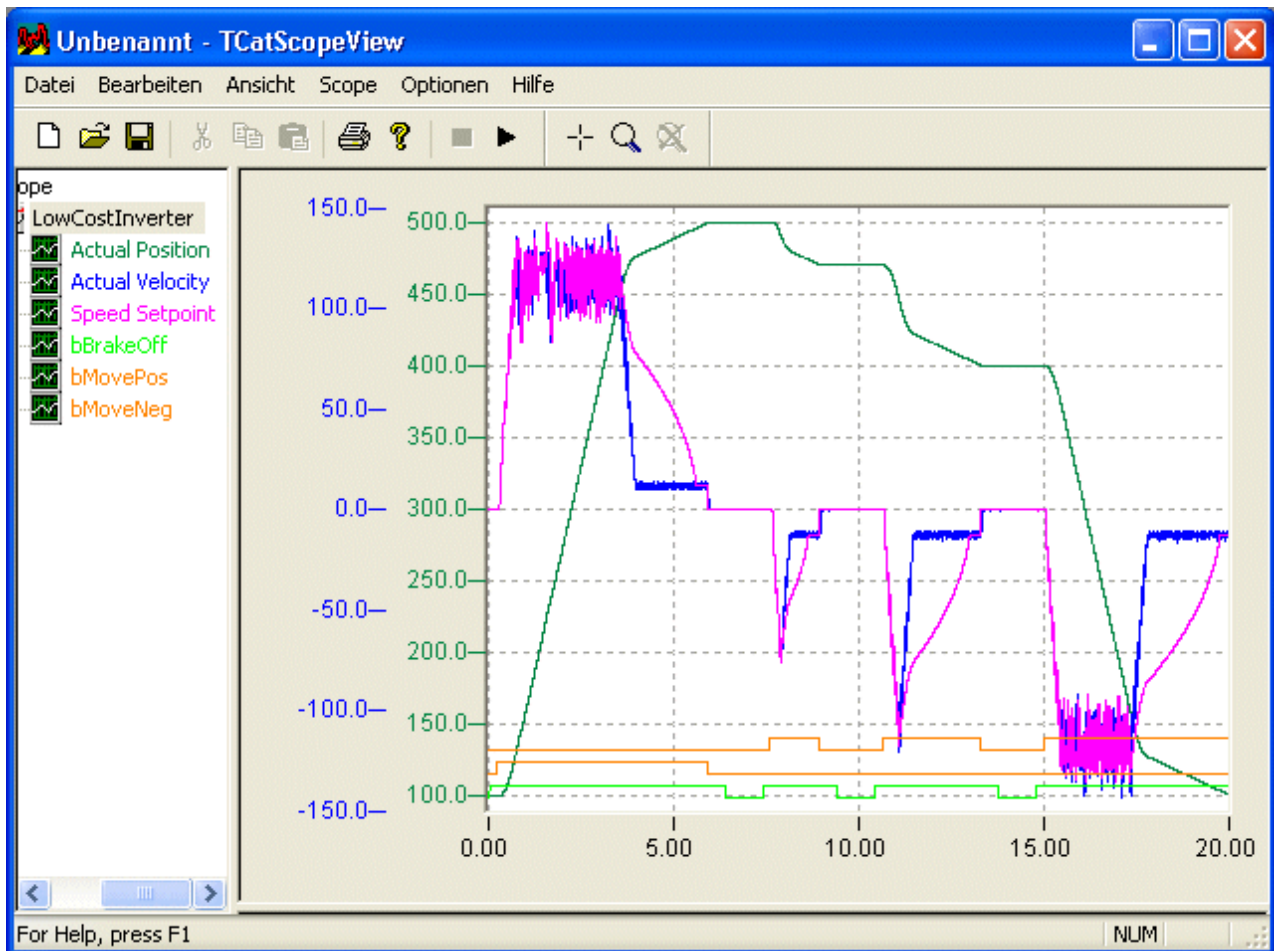
Sollte die oben beschriebene Vorgehensweise nicht unter den ungünstigsten Bedingungen vorgenommen werden besteht die Gefahr, dass sich das unten dargestellte Verhalten ergibt. In diesem Beispiel überfährt die Achse das Ziel um mehr als 10 mm.



Ist die Einstellung korrekt vorgenommen wird die Achse in anderen als der ungünstigsten Situation das unten dargestellte Verhalten zeigen. Die auffällige horizontale Verformung der Bremsrampe zeigt kein Problem an. Vielmehr wird hier die Anpassung der internen Stellwertgenerierung an das tatsächliche Bremsverhalten der Achse sichtbar.



Bei einer Einstellung nach der hier beschriebenen Strategie wird auch eine etwas kürzere Strecke nicht ausschließlich im Schleichgang gefahren.



4.6.4.6 Regelung und Überwachung

in Vorbereitung

Standard-Regler der Achse

Der Standard-Regler wird aktiv, wenn sich die Achse im Ruhezustand befindet und eine Reglerfreigabe vorliegt. Der Übergang von einer aktiven Bewegung in den Ruhezustand [► 270] wird durch den verwendeten Profil-Generator [► 242] definiert.

Wenn im Stillstand eine Lageregelung erfolgen soll ist in **Schleppregler** die Verstärkung vorzugeben.

Einige Fehlersituationen erlauben eine Lageregelung der Achse. Dies ist immer dann gegeben wenn sowohl der Encoder als auch der Antrieb der Achse funktionstüchtig sind. Um dies zu nutzen ist **Enable Ctr Loop on Error** zu setzen.

HINWEIS! Die Ausgabe des Standard-Reglers wird auf die Schleichgeschwindigkeit begrenzt. So wird in der Zielfahrt beim Übergang aus der Schleichphase in den Ruhezustand eine abrupte Änderung der Ausgabe verhindert.

Eine Offset-Spannung des verwendeten Analog-Ausgangs kann mit der **Offset-Kompensation** ausgeglichen werden. Die Eingabe erfolgt in Volt.

In einigen Anwendungsfällen soll ein Proportional-Ventil nach dem Abschluss einer Bewegung durch eine konstante Ausgabe einen **Pressdruck** am Hartanschlag erzeugen. Der hier eingestellte Wert wird mit einer Polarität ausgegeben, die durch die letzte aktive Fahrbewegung festgelegt ist. Erreicht die Achse das vorgegebene Fahrziel nicht wird diese Ausgabe nicht wirksam.

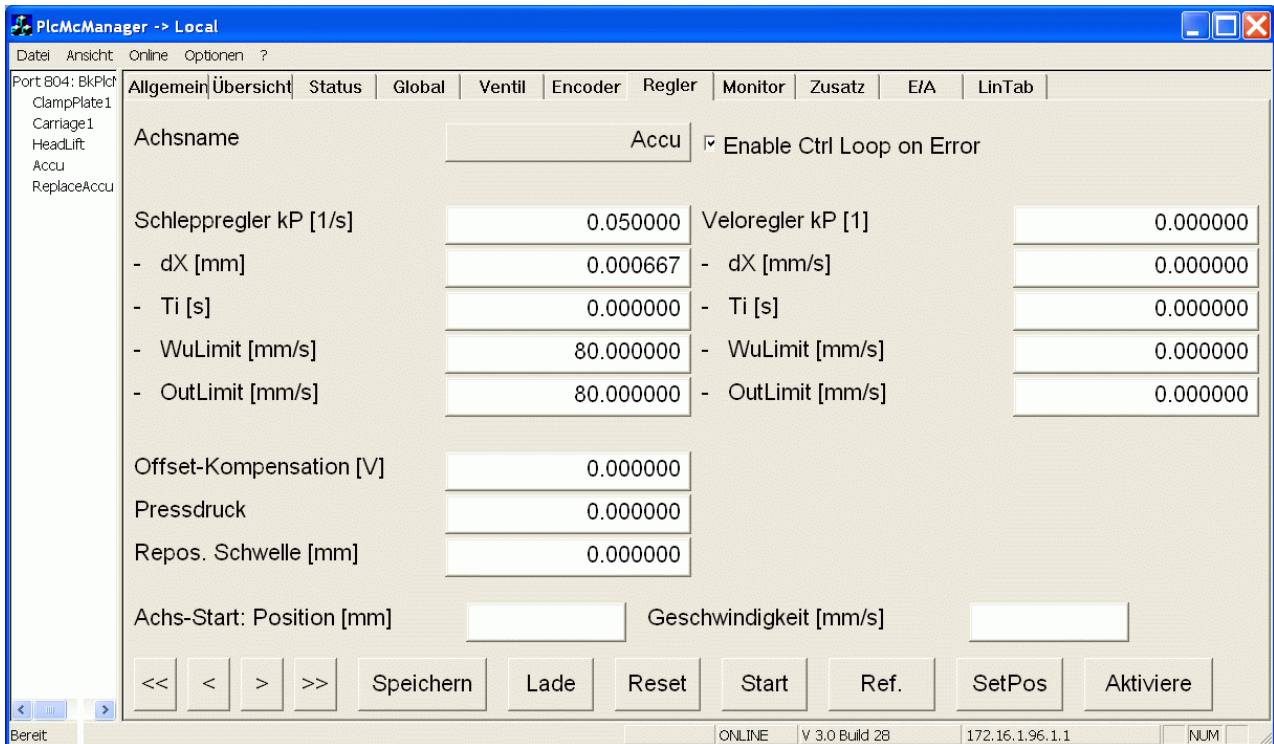
HINWEIS! Ein Reset der Achse löscht die Ausgabe ab.

Bei nicht lageregelbaren Achsen kann es sinnvoll sein, die Achse zu Re-Positionieren wenn sie aus der Zielposition bewegt wurde. Soll dies automatisch und ohne weitere Aktivität der Applikation erfolgen kann in **Repos. Schwelle** die Abweichung eingestellt werden, deren Überschreitung eine solche Reaktion auslösen soll.

HINWEIS! Diese Funktion darf grundsätzlich nicht mit einer **Pressdruck-Ausgabe** kombiniert werden. Andernfalls würde die Achse bei der Zielanfahrt ein instabiles Verhalten zeigen.

HINWEIS! Wird diese Funktion aktiviert muss **Repos. Schwelle** größer als die **Bremstrecke** eingestellt werden. Andernfalls würde die Achse bei der Zielanfahrt ein instabiles Verhalten zeigen.

Beispiel 1: Regler-Reiter



Monitor-Funktion

Die Parameter auf dem Monitor-Reiter des PlcMcManagers dienen zur Konfiguration verschiedener Überwachungsfunktionen.

Software-Endschalter

Die Parameter **Softend Min** und **Softend Max** legen Begrenzungen für den Fahrweg der Achse fest. Ihre Aktivität kann **freigegeben** oder gesperrt werden. Sind sie aktiv wird kein Baustein eine Zielposition akzeptieren, die sich jenseits der Endlage befindet. Bausteine ohne definierte Zielposition (MC_MoveVelocity_BkPlcMc, NC_MoveJoysticked_BkPlcMc usw.) werden eine in Bewegungsrichtung liegende aktive Endlage als Zielposition verwenden.

Zielfenster-Überwachung

Diese Parameter werden von verschiedenen Profil-Generatoren verwendet, um das Erreichen der Zielposition in ausreichender Genauigkeit festzustellen. Das Ziel gilt als erreicht wenn:

1. die Reststrecke kleiner oder gleich dem Parameter **Pos.Range** ist und
2. die Reststrecke seit **Target Filter** ununterbrochen kleiner oder gleich dem Parameter **Target Range** ist und
3. der Stellwertgenerator in den Ruhezustand geschaltet wurde.

PEH-Überwachung

Diese Überwachung prüft, ob die Achse nach Erreichen des Bremsfensters (Ende der aktiven Bewegung) zumindest einmalig das Zielfenster erreicht. Wenn diese Überwachung aktiviert ist wird ein Fehler erkannt und gemeldet, wenn ein Baustein (z.B. MC_MoveAbsolute_BkPlcMc) seinen Auftrag nicht **Done** melden kann.

Schlepp-Überwachung

Der Schleppabstand ist die Differenz zwischen einer Soll- und einer Istposition. Bei einer weggesteuerten Positionierung gibt es während einer aktiven Bewegung zwar eine Zielposition, jedoch keine zeitlich veränderliche Sollposition. Aus diesem Grund ist eine Schlepp-Überwachung nur in bestimmten Situationen sinnvoll möglich:

- im Stillstand als Lagefehler-Überwachung.
- bei abhängiger Sollwertgenerierung:
 - Getriebefunktionen (Gearing)
 - Kurvenscheiben (Camming)
 - externe Sollwertgenerierung z.B. durch MC_MoveJoysticked_BkPlcMc oder Applikation.
- bei Umschaltung auf zeitgesteuerter Sollwertgenerierung.

Nur in diesen Situationen wird auf dem Online Reiter ein Schlepp angezeigt.

Für die Fehlererkennung wird zunächst ein Schwellwert gebildet. Dazu wird die Sollgeschwindigkeit der Achse mit **Dyn-Lag Filter [s]** multipliziert und dann **Max. Lag [mm]** addiert. Überschreitet der Betrag des Schleppts diese Grenze für mindestens **Max-Lag Filter [s]** und ist die Überwachung **freigegeben** wird die Achse mit ErrorID **dwTchYdErrCdPosLag** in den Fehlerzustand versetzt.

Beispiel 2: Monitor-Reiter

