Softmotion

in CoDeSys

Benutzeranleitung

Dieses Dokument ist eine Ergänzung
zum Benutzerhandbuch für das
CoDeSys Programmiersystem

Es wurden alle erdenklichen Maßnahmen getroffen, um die Richtigkeit und Vollständigkeit der vorliegenden Dokumentation zu gewährleisten. Da sich Fehler, trotz aller Sorgfalt, nie vollständig vermeiden lassen, sind wir für Hinweise und Anregungen jederzeit dankbar.

Warenzeichen
Intel ist ein eingetragenes Warenzeichen und 80286, 80386, 80486, Pentium sind Warenzeichen der Intel Corporation.

Microsoft, MS und MS-DOS sind eingetragene Warenzeichen, Windows sind Warenzeichen der Microsoft Corporation.

Herausgeber:

3S - Smart Software Solutions GmbH
Memminger Strasse 151
D-87435 Kempten
Tel. +49/ 831/ 5 40 31 – 0
Fax +49/ 831/ 5 40 31 - 50

Stand: 29.09.2006 (zu CoDeSys 2.3.7.0)
Dokument-Version: 2.8

Inhalt

1 SoftMotion Konzept und Komponentenüberblick 1-1

2 Das SoftMotion Drive Interface 2-1

2.1 Steuerungskonfiguration für SoftMotion ... 2-2
2.1.1 BusInterface ... 2-2
2.1.2 AxisGroup ... 2-2
2.1.3 Drive ... 2-4
2.1.4 Encoder ... 2-7
2.2 SM_DriveBasic.lib und automatische Codegenerierung ... 2-8
2.2.1 Mathematische Hilfsbausteine ... 2-8
2.2.2 Achsgruppen-Bausteine ... 2-8
2.2.3 Konfigurations-Bausteine ... 2-10
2.2.4 Regelungsmodus-Bausteine ... 2-11
2.2.5 Direkte Sollwertvorgabe ... 2-12
2.2.6 Virtuelle Zeit-Achse ... 2-16
2.2.7 Referenzieren über digitale Hardware-Eingänge ... 2-17
2.2.8 Diagnose-Bausteine ... 2-19
2.2.9 Encoder ... 2-20
2.2.10 Visualisierungs-Templates ... 2-20
2.3 Antriebs-Treiber <BusInterfaceBezeichnung>Drive.lib ... 2-21
2.3.1 SercosDrive.lib ... 2-21
2.3.2 SM_CAN.lib ... 2-21
2.4 Variablen der Struktur AXIS_REF ... 2-23
2.5 Parametrierung des Antriebs ... 2-27

3 Der CNC-Editor 3-1

3.1 Überblick ... 3-1
3.2 Unterstützte und erweiterte Elemente der CNC-Sprache DIN66025 ... 3-2
3.2.1 Schaltpunkte, H-Funktion ... 3-4
3.2.2 Zusatz-Funktion, M-Funktion ... 3-5
3.2.3 Verwendung von Variablen ... 3-5
3.2.4 Kreisinterpolation ... 3-5
3.2.5 Ellipseninterpolation ... 3-6
3.2.6 Splineinterpolation ... 3-6
3.2.7 Bedingte Sprünge ... 3-6
3.2.8 Variablenwerte verändern ... 3-6
3.3 Start des Editors, Einfügen und Verwalten von CNC-Programmen ... 3-7
3.4 CNC Texteditor ... 3-10
3.5 CNC Graphischer Editor ... 3-11
3.6 Kommandos und Optionen im CNC-Editor ... 3-11
3.7 Automatische Strukturbefüllung im CNC-Editor ... 3-14

4 Der CAM-Editor in CoDeSys 4-1

4.1 Überblick ... 4-1

4.2 Definition einer Kurvenscheibe für SoftMotion 4-1
4.3 Starten des CAM-Editor und Erstellen einer Kurvenscheibe 4-1
4.4 Editieren einer Kurvenscheibe 4-3
4.4.1 Allgemeine Editor-Einstellungen 4-3
4.4.2 Bearbeiten der Eigenschaften einzelner Kurvenelemente: 4-4
4.4.3 Befehle der Menüs 'Extras' und 'Einfügen' 4-6
4.5 Verwendung von Kurvenscheiben 4-9
4.5.1 Auswirkungen von Baustein-Parametern 4-9
4.5.2 Umschalten zwischen Kurvenscheiben 4-11
4.6 Kurvenscheiben-Datenstrukturen 4-12
4.6.1 Beispiel für manuell erzeugte Kurvenscheibe 4-14

5 Die Bibliothek SM_PLCopen.lib 5-1

5.1 Überblick 5-1
5.2 PLCopen-Spezifikation "Function blocks for motion control, Version 1.0" 5-1
5.3 Bausteine zur Bewegungssteuerung einzelner Achsen 5-2
5.4 Bausteine zur synchronisierten Bewegungssteuerung (multi-axis) 5-14
5.5 Zusatzbausteine 5-18

6 Die Bibliothek SM_CNC.lib 6-23

6.1 Überblick 6-23
6.2 Bausteine 6-23
6.2.1 SMC_NCDecoder 6-23
6.2.2 SMC_GCodeViewer 6-25
6.2.3 SMC_ToolCorr 6-26
6.2.4 SMC_AvoidLoop 6-28
6.2.5 SMC_SmoothPath 6-29
6.2.6 SMC_RoundPath 6-31
6.2.7 SMC_CheckVelocities 6-32
6.2.8 SMC_LimitCircularVelocities 6-33
6.2.9 SMC_Interpolator 6-34
6.2.10 SMC_GetMParameters 6-38
6.2.11 SMC_Interpolator2Dir 6-39
6.3 Hilfsfunktionen und –funktionsblöcke für Bahnrotationen, –translationen und -skalierungen 6-40
6.4 Einstellungen über globale Variablen 6-40
6.5 Strukturen der SM_CNC.lib 6-41
6.6 Bahn-Kurvenscheiben mit SMC_XInterpolator 6-47

7 Die Bibliothek SM_CNCDiagnostic.lib 7-1

7.1 Funktionsbausteine zur Analyse von SMC_CNC_REF-Daten 7-1
7.1.1 Der Funktionsblock SMC_ShowCNCREF 7-1
7.2 Funktionsbausteine zur Analyse von SMC_OutQueue-Daten 7-1
7.2.1 Der Funktionsblock SMC_ShowQueue 7-1

8 Die Bibliothek SM_Trafo.lib 8-1

8.1 Überblick 8-1
8.2 Transformations-Funktionsblöcke 8-1
8.2.1 Portal-Systeme 8-1
8.2.2 Portal-Systeme mit Werkzeugversatz 8-4
8.2.3 H-Portal-System mit stationären Antrieben 8-8
8.2.4 2-Gelenkige Scara-Systeme 8-9
8.2.5 3-Gelenkige Scara-Systeme 8-11

8.2.6 Parallel-Kinematiken ... 8-13
8.3 Räumliche Transformationen ... 8-14

9 Die Bibliothek SM_Error.lib 9-1

9.1 Funktionsblöcke ... 9-1
9.1.1 SMC_ErrorString ... 9-1
9.2 Die Enumeration SMC_Error ... 9-1

10 Die Bibliothek SM_FileFBs.lib 10-1

10.1 Überblick ... 10-1
10.2 CNC-Funktionsblöcke ... 10-1
10.3 CAM-Funktionsblöcke ... 10-3
10.4 Diagnose-Funktionsblöcke ... 10-3

11 Programmierbeispiele 11-1

11.1 Überblick ... 11-1
11.2 Beispiel: Drive Interface: Steuerungskonfiguration für Antriebe erstellen ... 11-1
11.3 Beispiel: Bewegungssteuerung einzelner Achsen in ST (single-axis) ... 11-4
11.4 Beispiel: Bewegungssteuerung einzelner Achsen in CFC mit Visualisierungs-Template (single-axis) ... 11-5
11.5 Beispiel: Kurvenscheiben-Antriebssteuerung mit Hilfe einer virtuellen Zeitachse 11-7
11.6 Beispiel: wechselnde Kurvenscheiben ... 11-8
11.7 Beispiel: Antriebssteuerung mit Hilfe des CNC-Editors ... 11-8
11.7.1 Direktes Erzeugen der OutQueue ... 11-9
11.7.2 Decodierung online mit Verwendung von Variablen ... 11-11
11.7.3 Bahnvorverarbeitung online ... 11-13
11.8 Beispiel: Dynamische SoftMotion-Programmierung ... 11-14

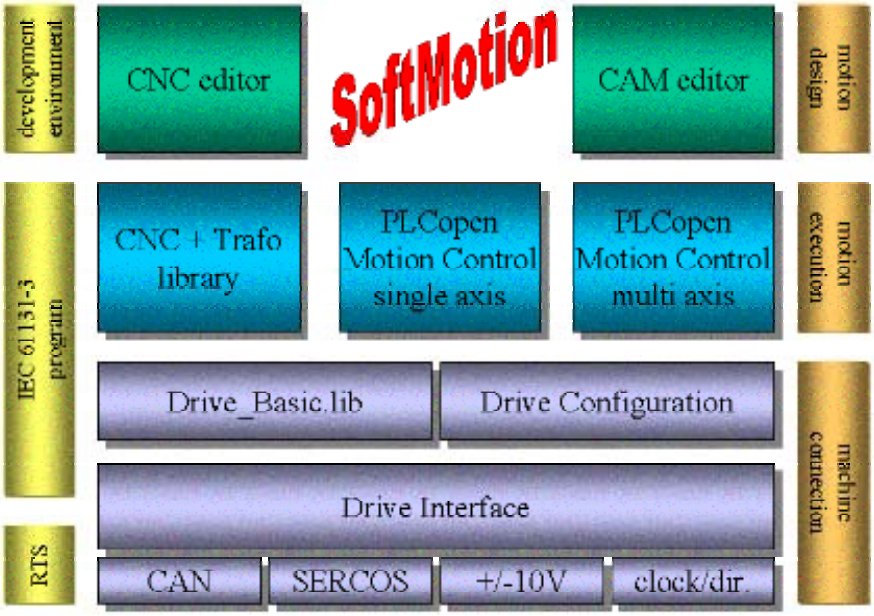
12 Index I

1 SoftMotion Konzept und Komponentenüberblick

SoftMotion ermöglicht das Realisieren von Bewegungen – von einfachen Einachs- über Kurvenscheiben- bis hin zu komplexen Bewegungen in mehreren Dimensionen unter der Entwicklungsumgebung von *CoDeSys*. Vor allem Anwendungen, bei denen nicht ausschließlich die Bewegungsfunktionen im Mittelpunkt stehen, sondern auch Ablauf- und Prozesssteuerung oder Hilfsfunktionen einen wesentlichen Teil der Applikation ausmachen, bilden einen idealen Anwendungsbereich für *SoftMotion*.

SoftMotion ist als eine Art Basis-Werkzeugkasten aufzufassen, mit Hilfe dessen – auch prozessabhängig während der Laufzeit – in vielfacher Weise Einfluss genommen werden kann, ohne dass größerer Aufwand und detailliertes Know-how für die Realisierung der gewünschten Bewegungen erforderlich ist.

Die gesamte Programmlogik wird im SPS-Programm behandelt und nur die reine Bewegungsinformation in den Bibliotheksbausteinen abgearbeitet.



SoftMotion gliedert sich in folgende Komponenten:

- **Drive Interface**
  Diese SoftMotion-Komponente ist für die Kommunikation zwischen dem Steuerungsprogramm und den Antrieben verantwortlich. Sie besteht aus der Bibliothek SM_DriveBasic.lib und Antriebs- und Bussystem-spezifischen Bibliotheken und Treibern.
  Im Konfigurationseditor in CoDeSys bildet der Programmierer die Struktur und Parametrierung der Antriebs-Hardware ab. CoDeSys erzeugt daraus mit Hilfe der DriveInterface-Bibliotheken IEC-Datenstrukturen, die die Antriebe abstrahiert darstellen. Das Drive Interface kommuniziert automatisch, d.h. ohne zusätzlichen Aufwand seitens des IEC-Programmierers, mit den Antrieben, sorgt also für laufende Aktualisierung der Antriebs-Datenstrukturen und Übertragung der darin geänderten Daten. Um steuernden Einfluss auf die Antriebs-Hardware zu nehmen, kann das IEC-Programm auf diese Datenstrukturen entweder mittels Standard-Bausteinen aus SoftMotion-Bibliotheken (SM_CNC.lib, SM_PLCOpen.lib) oder mit vom IEC-Programmierer selbst erstellten Programmen zugreifen.
  Die Sollwertvorgabe erfolgt dabei stets zyklisch, d.h. pro IEC-Taskzyklus werden Sollwerte (-positionen, -geschwindigkeiten, -beschleunigungen etc.) berechnet und vom Drive Interface zu den Antrieben übertragen. Ein „Beauftragen“ der Antriebe, wie z.B. die Vorgabe einer Zielposition, so dass der Antrieb eigenständig die Bewegung ausführt und nach dem Erreichen der Steuerung das erfolgreiche Ausführen des Auftrags mitteilt, ist nicht vorgesehen. Gründe hierfür sind u.a., dass so keine koordinierten Bewegungen mehrerer Achsen möglich wären und die zentrale Steuerung während der Bearbeitung eines Auftrags keinen Einfluss auf die Antriebe hätte.

- **CNC-Editor**
  Der CNC-Editor in CoDeSys dient der grafischen und textuellen Programmierung von mehrdimensionalen Antriebsbewegungen, angelehnt an die CNC-Sprache DIN66025. Grundsätzlich können bis zu 9-dimensionale Bewegungen realisiert werden, wobei nur zwei Dimensionen anders als linear interpoliert werden. D.h. in zwei Dimensionen können Geraden, Kreise, Kreisbögen, Parabeln, Ellipsen und Splines programmiert werden; die anderen Richtungen werden lediglich linear interpoliert. Für jede erzeugte Bahn wird von CoDeSys automatisch eine globale Datenstruktur (CNC Data) angelegt, die im IEC-Programm zur Verfügung steht.

- **CAM-Editor**
  Der in CoDeSys integrierte, graphisch bedienbare Kurvenscheiben-Editor dient der Programmierung von Kurvenscheiben (CAMs) zur Steuerung von mehrachsigen Antrieben. Aus jeder programmierten Kurvenscheibe erzeugt CoDeSys automatisch eine globale Datenstruktur (CAM data), die im IEC-Programm zur Verfügung steht.

- **CNC-Bibliotheken**
  Die Bibliothek "SM_CNC.lib", „SM_CNCDiagnostic.lib“ und „SM_Trafo.lib“ stellen dem IEC-Programmierer Bausteine zur Verfügung, mit deren Hilfe er die im CNC-Editor erzeugten bzw. zur Laufzeit geplanten Bewegungen bearbeiten, darstellen und ausführen kann.

- **PLCopen-Bibliothek**
  Die PLCopen-MotionControl-Bibliothek "SM_PLCopen.lib" besteht u.a. aus Bausteinen, mit denen die Steuerung einzelner Achsen, aber auch die synchronisierte Bewegung zweier Achsen leicht programmiert und realisiert werden kann. Neben Bausteinen zur Statusabfrage, Parametrierung und allgemeiner Bedienung enthält sie Funktionsblöcke, die eine Achse gemäß vorgegebenem Geschwindigkeits- und Beschleunigungsverhalten auf verschiedene Weisen bewegen kann. Sollen zwei Achsen synchronisiert werden, dient eine Achse als Master und steuert unter einer gewissen Vorschrift eine zweite Achse (Slave). Diese Vorschrift kann beispielsweise eine im CAM-Editor generierte Kurvenscheibe sein, die mittels vorhandener Bausteine die Slave-Achse an die Master-Achse koppelt. Desweiteren existieren Funktionsblöcke zur Programmierung elektronischer Getriebe oder Phasenverschiebungen.

- **Dateidienst-Bibliothek**
  Die Bibliothek "SM_FileFBs.lib" baut auf der System-Bibliothek „SysLibFile.lib“ auf und kann deshalb nur auf Steuerungen, die die Bibliothek unterstützen verwendet werden.

- **Fehler-Bibliothek**
  Die Bibliothek "SM_Error.lib" beinhaltet alle Fehlerausgaben, die die Bausteine der anderen Bibliotheken erzeugen können. Außerdem können mit ihr aus den numerischen Fehlervariablen deutsche und englische Fehlertexte erzeugt werden.

Portabilität
Ausgenommen manche zum DriveInterface gehörenden Treiber, die direkt Hardware-Komponenten bedienen, sind alle SoftMotion-Laufzeit-Komponenten in IEC1131-3 programmiert. Dadurch wird eine maximale Plattformunabhängigkeit erreicht.

Für ein schnelles Verständnis der SoftMotion-Komponenten wird dem Leser empfohlen, die jeweiligen Beispiele zu studieren.

2 Das SoftMotion Drive Interface

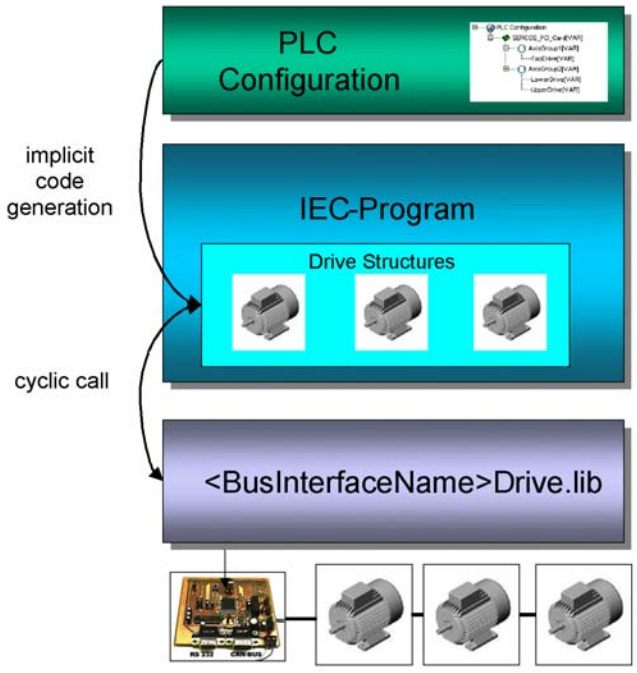
Das *Softmotion Drive Interface* ist eine genormte Schnittstelle, die dem Programmierer die Möglichkeit bietet, das abstrahierte Bild eines Antriebs in das IEC-Programm einzubinden, dort zu konfigurieren und anzusprechen. Es sorgt automatisch für die Aktualisierung und Übertragung der Daten, die für die Steuerung der Antriebs-Hardware nötig sind. So können Antriebe nicht nur leicht ausgetauscht und IEC-Programme wiederverwendet werden, es werden dem Applikateur auch die Schwierigkeiten und Unbequemlichkeiten der Antriebsanbindung erspart.

Das Drive Interface arbeitet mit folgenden Komponenten:

- Der CoDeSys **Steuerungskonfigurator**: Hier ist auf Basis einer entsprechenden Konfigurationsdatei die Struktur der zu bedienenden Antriebe abzubilden und zu parametrieren. Diese Struktur wird dann mit Hilfe der Drive-Interface Bibliotheken über implizit erzeugte und belegte (System-)Variablen für die Applikation (IEC-Programm) zugänglich gemacht.

- Die interne Bibliothek **SM_DriveBasic.lib** stellt IEC-Datenstrukturen und globale Variablen bereit, die die im Steuerungskonfigurator dargestellten Antriebe, Achsgruppen sowie BusInterfaces repräsentieren.

- Der Antriebs-Treiber, d.h. die für die verwendeten Antriebe und das Bussystem vom Steuerungshersteller zu liefernde spezifische Bibliothek **<BusInterfaceBezeichnung>Drive.lib** (z.B. SercosDrive.lib): bietet spezielle Funktionen, die für den Datenaustausch zwischen den Strukturen und der jeweiligen Hardware erforderlich sind (siehe 2.3).



2.1 Steuerungskonfiguration für SoftMotion

(Sehen Sie hierzu auch das Programmierbeispiel in Kapitel 11.2)

Um die Struktur der Antriebe abzubilden, stehen in der CoDeSys-Steuerungskonfiguration i.d.R. folgende Elemente zur Verfügung (die entsprechende Konfigurationsdatei muss im Verzeichnis für Konfigurationsdateien vorliegen !):

BusInterface:	Feldbus-Interface, über welches mit den Antriebe kommuniziert wird
Axisgroup:	physikalisch zusammenhängende Gruppe (Achsgruppe) von Antrieben (Drives)
Drive:	Antrieb
Encoder:	Geber

Die Businterfaces, Achsgruppen und Antriebe können mit beliebigen, aber eindeutigen IEC 61131-3 Identifiern benannt werden:

Jedes dieser Konfigurationsobjekte kann über Dialoge konfiguriert werden, sofern die Konfigurationsbeschreibung des Targets dies unterstützt. Neben der konfortablen Konfiguration über Dialoge können die Parameter ausserdem in einer Konfigurationsliste („Modulparameter“) gesetzt werden. Dort befinden sich auch target-spezifische Parameter, die mit „MS:“ beginnen.

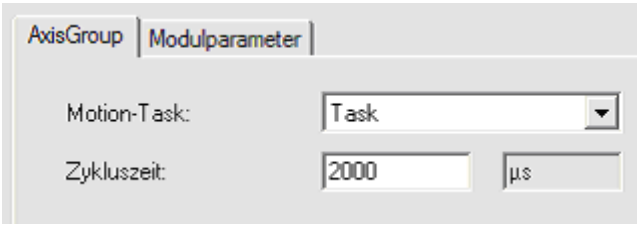
2.1.1 BusInterface

Hier wird in der Regel nur der Kommunikationtreiber ausgewählt.

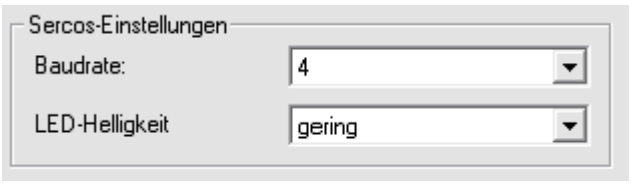
2.1.2 AxisGroup

Hier wird die Task angegeben, in welcher die Kommunikation zu den Antrieben stattfindet, und – falls es sich um keine zyklische Task, sondern eine ereignisgesteuerte handelt – deren Zykluszeit.

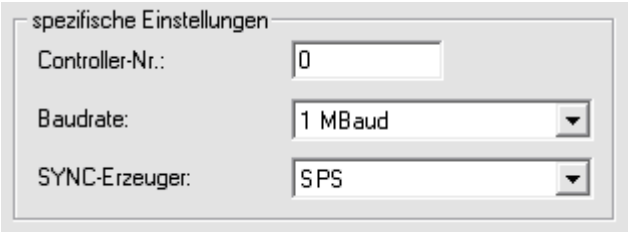
Bei Systemen ohne Taskkonfiguration bleibt das Feld leer.



Bei **Sercos**-Interfaces werden weitere spezifische Einstellungen gemacht, und zwar die Baudrate und die Helligkeit der LED.



Auch für **CAN-Achsgruppen** sind spezifische Einstellungen zu machen:



Neben der Baudrate legt man mit der Controller-Nr. die Nummer des verwendeten CAN-Controllers fest (man beachte, dass, sollte man eine Steuerung mit 2 CAN-Kanälen und zudem die 3S-CANopen.lib verwenden, sich diese automatisch Controller 0 nimmt, man also für die Antriebe den Kanal 1 wählen muss).

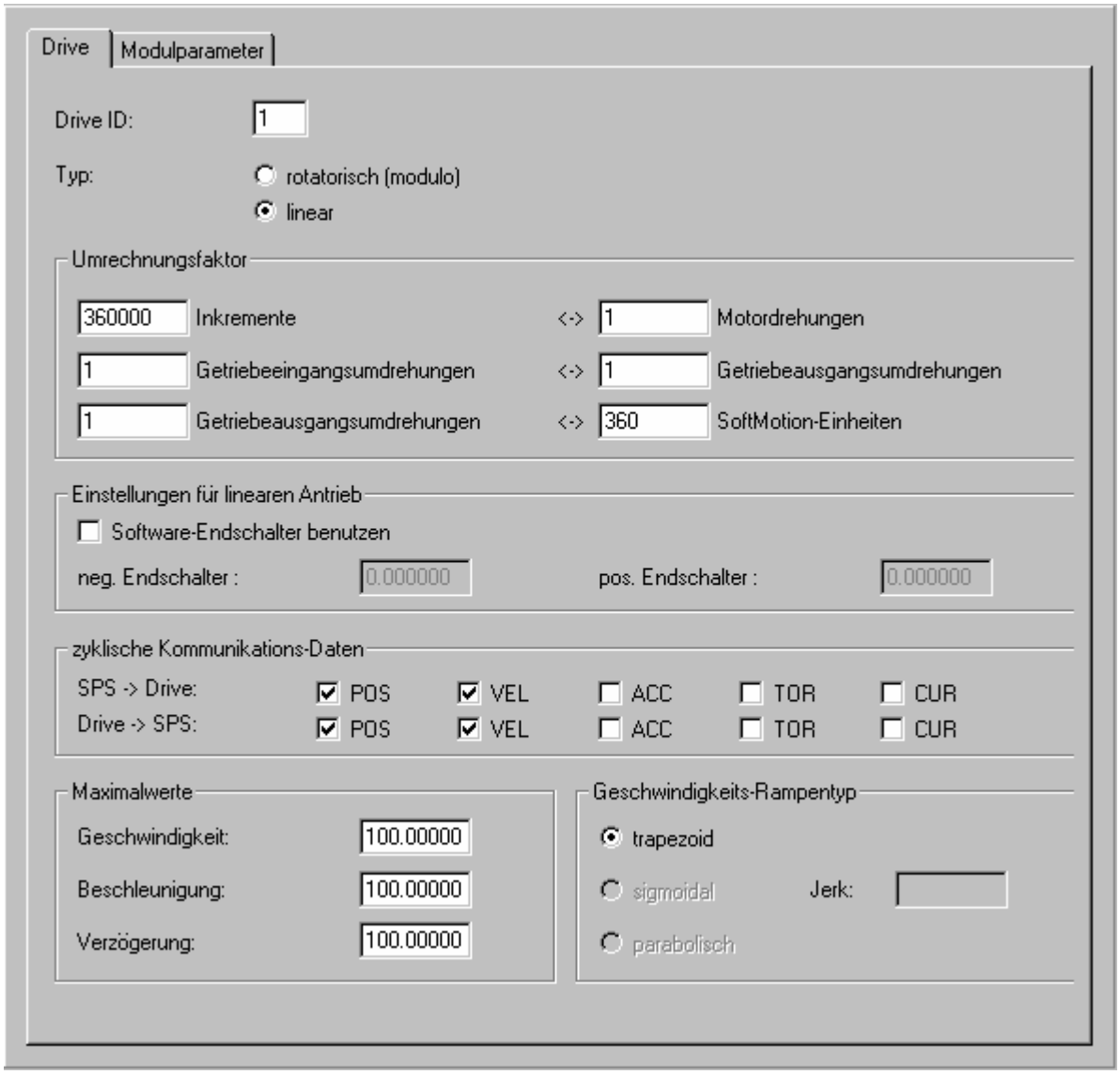
Bei SYNC-Erzeuger kann man zwischen drei verschiedenen Methoden zur Synchronisation zwischen Antrieben und SPS wählen:

- SPS: Hier tritt die SPS als Synchronisationsmaster auf. Der Anwender definiert die Motion-Task i.d.R. als zykliche Task. Diese task ruft den Treiber auf, welcher sofort ein SYNC-Telegramm versendet. Diese Methode ist die einfachste, kann aber bei einer Steuerung mit hohem Jitter und Antrieben, die auf eine große Genauigkeit des SYNC Telegramms fordern, zu Problemen führen.
- 1.Antrieb: Hier erzeugt der erste Antrieb (falls er dieses Feature unterstützt) das SYNC-Telegram. Die Motion-task in der SPS ist dann in der Regel auf das Ereignis <AxisGroup>.bSync definiert, wartet also, bis ein SYNC-Telegramm empfangen wurde und stösst daraufhin die Taskabarbeitung an.
- SYNC_Device: Diese Methode wird angewendet, wenn die beiden oberen nicht möglich sind. In den Bus wird ein zusätzliches Device mit CAN ID 127 installiert, welches fähig ist, SYNC-Telegramme zeitgenau zu erzeugen (Index: 1005h, Bit30).

Alle diese Einstellungen können zudem im Reiter „Modulparameter“ eingesehen und verändert werden.

sTask	String, der mit dem Namen der Task übereinstimmt, in der die Datenübertragung dieser Achsgruppe stattfinden soll.
dwCycle	Zykluszeit der unter 'sTask' definiertenTask in µs (muss nur angegeben werden, wenn die Steuerung keine Tasks unterstützt und automatisch PLC_PRG aufruft (Default-Task))
wParam1 ... wParam4	Karten-/Antriebsspezifische Parameter vom Typ WORD
dwParam1 .. dwParam4	Karten/Antriebsspezifische Parameter vom Typ DWORD

2.1.3 Drive



In diesem Dialog wird die Antriebs-ID (**Drive ID**) festgelegt. Außerdem wird der Antriebs**typ** ausgewählt: linear oder rotatorisch (Modulo).

Im Bereich **Umrechnungsfaktor** bestimmt man die Umrechnung zwischen den vom Antrieb empfangenen Ganzzahl-Positionswerten (erste Zeile links) und den technischen Einheiten, die man im IEC-Programm verwendet (erste Zeile rechts). Dabei kann außerdem noch ein Getriebe berücksichtigt werden (zweite und dritte Zeile). Im obigen Beispiel würde ein Antrieb, der für eine Umdrehung 3600000 Inkremente erzeugt, so skaliert werden, dass die technischen Einheiten in Winkelgrad vorliegen.

Im Bereich **Einstellungen für..** können je nach gewähltem Antriebstyp (s.o.) für **lineare Antriebe** Software-Endschalter definiert werden bzw. muss für **rotatorische Antriebe** der Modulo-Bereich festgelegt werden.

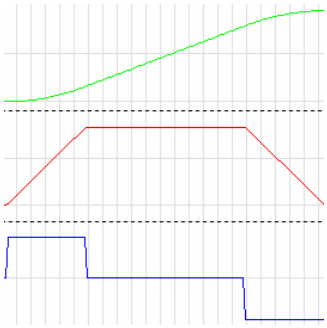
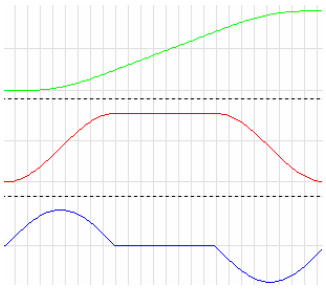
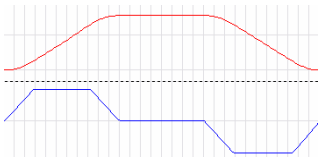
Im Bereich **zyklische Kommunikations-Daten** wird ausgewählt, welche Soll- und Ist-Daten zyklisch zwischen Steuerung (SPS) und Antrieb (Drive) übertragen werden sollen.

Unter **Maximalwerte** werden diejenigen Grenzwerte gesetzt , die von SMC_ControlBy-Bausteinen verwendet werden, um festzustellen, ob ein Sprung vorliegt (siehe Kap. 2.2.5, Beschreibung SMC_ControlAxisByPos).

Bei **Geschwindigkeits-Rampentyp** spezifiziert der Anwender (falls von den eingebundenen Bibliotheken unterstützt), den Geschwindigkeits-Profiltyp für die bewegungserzuegenden Einach- und Master/Slave-Bausteine. „trapezoid“ hat ein trapezförmiges Geschwindigkeitsprofil (abschnittsweise konstante Beschleunigung), „sigmoidal“ ein sin²-förmiges Geschwindigkeitsprofil (stetige Beschleunigung), „parabolisch“ ein stetiges, trapezoides Beschleunigungsprofil und damit parabolisches Geschwindigkeitsprofil zur Folge.

Für die Modi „sigmoidal“ und „parabolisch“ muss zudem der Ruck (**Jerk**) festgelegt werden.

Folgende Grafiken zeigen, wie sich die unterschiedlichen Rampen-Modi bei einer Positionierung auswirken. Dabei ist die Position grün, die Geschwindigkeit rot und die Beschleunigung blau dargestellt.

Beim **trapezförmigen** Geschwindigkeitsmodus ist zu erkennen, dass in der Beschleunigung Sprünge auftreten.	
Beim **sigmoidalen** Geschwindigkeitsmodus sind diese Sprünge eliminiert. Der Bewegungsverlauf ist während der Bewegungsphase eindeutig festgelegt, wodurch auch der Ruck (Jerk) nicht begrenzt werden kann. Die Festelegung desselben wird lediglich verwendet, wenn der Antrieb zu Beginn bereits einen aktuellen Beschleunigungswert ungleich null besitzt. Dann nämlich wird die Beschleunigung ruckbegrenzt auf null gefahren, bevor die eigentliche Bewegung gestartet wird. Im Vergleich zum trapezförmigen Geschwindigkeitsprofil dauert diese Bewegung länger.	
Beim **parabolischen** Geschwindigkeitsverlauf nimmt die Beschleunigung ein stetiges, trapezförmiges Profil an, dessen Steigung durch den Ruck (Jerk) begrenzt wird. Für die Geschwindigkeit ergibt sich ein stetiges, parabolisches Profil. Nur bei diesem Profil kann der Ruck wirklich begrenzt werden.	

Alle diese Einstellungen können zudem im Reiter „Modulparameter“ eingesehen und verändert werden:

wId	ID des Antriebs in der AxisGroup (WORD)
wControlType	vordefinierte Steuerungs- und Rückmeldearten (WORD); folgende Auswahlmöglichkeiten: (<Sendedaten> -> <Rückmeldedaten>) TOR -> --- Drehmoment (Torque) -> --- VEL -> VEL Geschwindigkeit -> Geschwindigkeit VEL -> POS Geschwindigkeit -> Position POS -> POS Position -> Position POS, VEL -> POS, VEL Position,Geschwindigkeit -> Geschwindigkeit,Position VEL -> --- Geschwindigkeit -> --- CONFIGURABLE manuelle Konfiguration über wCyclicDataS1, ..S2, ..S3 und wCyclicDataR1, ..R2, ..R3 (s.u.)
wCyclicDataS1 wCyclicDataS2 wCyclicDataS2 wCyclicDataR1 wCyclicDataR2 wCyclicDataR3	Definition der Sende (..S..)- und Rückmeldedaten (..R..), wenn wControlType auf 'CONFIGURABLE' eingestellt ist; Auswahlmöglichkeiten sind abhängig vom jeweiligen Antriebs-Treiber. Grundsätzlich möglich: Act/SetPosition Position Act/SetVelocity Geschwindigkeit Act/SetTorque Drehmoment (Torque) Act/SetCurrent Strom Act/SetUserDef benutzerspezifische Definition
dwRatioTechUnitsDenom iRatioTechUnitsNum	Nenner und Zähler für Umrechnungsfaktor von Busdaten in technische Einheiten [u]; (DWORD bzw. INT)
iMovementType	Bewegungstyp; Auswahlmöglichkeiten: linear ("linear") oder rotatorisch ("rotary")
fPositionPeriod	Periode für rotatorische Achsen in technischen Einheiten; Man beachte, dass diese umgerechnet in Inkremente (durch 'dwRatioTechUnitsDenom' und 'iRatioTechUnitsNum') nicht größer sein darf als 16#80000000.
fSWMaxVelocity	Maximale Geschwindigkeit für Software-Überprüfung
fSWMaxAcceleration	Maximale Beschleunigung für Software-Überprüfung
fSWMaxDeceleration	Maximale Bremsung für Software-Überprüfung
bSWLimitEnable	Software-Positionsüberprüfung einschalten (nur lineare Antriebe), die bewirkt, dass Achse in Fehlerzustand gesetzt wird, wenn sie das Positionsfenster verlässt.
fSWLimitNegative	negative Positionsgrenze (nur lineare Antriebe)
fSWLimitPositive	positive Positionsgrenze (nur lineare Antriebe)
bHWLimitEnable	Hardware-Positionsüberprüfung einschalten (nur lineare Antriebe), die bewirkt, dass Achse in Fehlerzustand gesetzt wird, wenn sie das Positionsfenster verlässt.

Für **Sercos**-Antriebe steht ein zusätzlicher Dialog zur Verfügung:

Bei Device-Typ kann zwischen Antrieb und I/O-Device gewählt werden, da es keine Standard-CoDeSys-Unterstützung für Sercos-I/Os gibt. Wählt man I/O-Device, so werden einige Parameter, die der Master normalerweise überträgt, weggelassen.

Außerdem können (neben den Standard-Werten POS, VEL, ACC, TOR, CUR**) zusätzliche zyklische Kommunikationsdaten** übertragen werden. Dazu gibt man Sercos-Parameternummer (IDN) und –länge an.

Bei **PackProfile-Überprüfung** kann getestet werden, ob die gemachten Einstellungen gemäß PackProfile-Standard gemacht worden sind. Dazu wird zwischen den Profilen BasicA, BasicB und Extended unterschieden. Man beachte, dass diese Prüfung auf den Offline-Daten durchgeführt werden kann. Sollte der bei Sercos mögliche zusätzlichen Konfigurationsmechanismus (ASCII-File auf der Steuerung einlesen) verwendet werden, kann dies das Ergebnis ändern. Deshalb ist in der SercosDrive.lib eine zusätzliche Online-Überprüfung (siehe Sercosdrive.pdf) implementiert. Zusammen mit der Sercosdrive.lib werden zudem XML-Files ausgeliefert, die in diesem Dialog importiert werden können, und alle zulässigen PackProfile-Parameter enthalten.

Daneben können Parameter, die während des Hochlaufs zum Antrieb geschrieben werden sollen, definiert werden (**Initialisierungsdaten**). Dazu existiert je eine Liste für Parameter, die in Phase2, Phase3 und am Beginn von Phase4 an den Antrieb geschickt werden sollen. Mit der richtigen Parameterliste kann so erreicht werden, dass der Antrieb am Anfang der Applikation vollständig initialisiert wird, z.B. wenn er ausgetauscht werden musste (siehe auch 2.5)

Über Einträge in der Liste „**gesperrt**“ kann verhindert werden, dass einzelne Parameter vom Treiber automatisch übertragen werden.

Alle diese Einstellungen können mit einer **Konfigurationsbezeichnung** versehen und in ein XML-File abgelegt (Schaltfläche “**speichern**“) bzw. aus einem XML-File gelesen (Schaltfläche “**laden**“) werden.

Für **CAN**-Antriebe gibt es ebenso einen speziellen Dialog, in welchem Parameter eingetragen werden, die während des Hochlaufs zum Antrieb geschrieben werden sollen. Auch diese können in XML-Files abgelegt und aus solchen geladen werden.

2.1.4 Encoder

Beim Übersetzen wird für jeden konfigurierten Encoder eine Datenstruktur vom Typ SMC_ENCODER_REF angelegt, die mit folgenden Werten konfiguriert wird. Diese Datenstruktur dient als Eingang für den Baustein SMC_Encoder, der aus der Konfiguration und den Echtzeit-Encoder-Werten eine virtuelle Achse bildet. Je nach BusInterface-Typen muss die aktuelle Encoder-Position manuell (Dummydrive) beim Baustein SMC_Encoder eingegeben werden, oder sie wird automatisch zusammen mit den Istwerten der Antriebe über den Bus eingelesen.

wEncoderId	ID des Encoders (WORD)
dwRatioTechUnitsDenom iRatioTechUnitsNum	Nenner und Zähler für Umrechnungsfaktor von Busdaten (Antriebs-Inkrementen) in technische Einheiten (in der Applikation verwendeten Einheiten, Softmotion-Einheiten) [u]; (DWORD bzw. INT)
iMovementType	Gebertyp; Auswahlmöglichkeiten: linear ("linear") oder rotatorisch ("rotary")
fPositionPeriod	Periode für rotatorische Achsen; hängt von den Umrechnungsfaktoren 'dwRatioTechUnitsDenom' und 'iRatioTechUnitsNum' ab
bSWLimitEnable	Software-Positionsüberprüfung einschalten (nur lineare Antriebe), die bewirkt, dass später über SMC_Encoder verknüpfte Achse in Fehlerzustand gesetzt wird, wenn sie das Positionsfenster verlässt.
fSWLimitNegative	negative Positionsgrenze (nur lineare Encoder)
fSWLimitPositive	positive Positionsgrenze (nur lineare Encoder)

dwEncoderModulo	Wert, bei dem der Encoder-Umbruch stattfindet, Bitbreite des Encoders: (0: 32 Bit, 16#10000: 16 Bit, 16#1000: 12 Bit) etc.

Daneben können ggf. zusätzliche herstellerspezifische Parameter (diese beginnen mit „P:“) vorhanden sein, die in den Parametern dwParam1..dwParam8 von SMC_ENCODER_REF abgelegt werden.

2.2 SM_DriveBasic.lib und automatische Codegenerierung

Wenn die Bibliothek "**SM_DriveBasic.lib**" in der IEC1131-Applikation in CoDeSys eingebunden ist, werden aus dem Antriebs-Abbild, das in der Steuerungskonfiguration mit den entsprechenden Parametern eingegeben wurde, automatisch Strukturobjekte erstellt, auf die das IEC-Programm zugreifen kann. (Siehe hierzu auch Kapitel 2.3 zur Beschreibung der AXIS_REF Struktur.)

In der IEC1131-Applikation **muss** außerdem die zur verwendeten Steuerung passende, herstellerspezifische Bibliothek mit dem Namen **<BusInterfaceBezeichnung>Drive.lib** eingebunden sein. Diese unterstützt die hardware-spezifische Drive-Interface-Funktionalität. Die BusInterface-Bezeichnung ergibt sich aus dem in der Steuerungskonfiguration in den Modulparametern des Bus-Interfaces bei 'InterfaceType' gewählten Eintrag. Von dessen String-Bezeichnung wird sie aus dem linken Teil bis zum ersten Leerzeichen gewonnen (Beispiel : "CAN (Peak)" => "CAN" => herstellerspezifische Bibliothek heißt: "CANDrive.lib").

Beim Starten der Applikation sorgt der implizite Aufruf der Funktionen **<BusInterfaceBezeichnung>DriveExecute_Start** und **<BusInterfaceBezeichnung>DriveInit** am Taskanfang und **<BusInterfaceBezeichnung>DriveExecute_End** am Taskende für die Übertragung und Wartung der AXIS_REF-Strukturvariablen. Die globale Variable **g_strBootupError** enthält für den Fall, dass bei der Initialisierung der Antriebe Fehler auftreten, eine von der Bibliothek <BusInterfaceBezeichnung>Drive.lib erzeugte Fehlerbeschreibung.

Die Bibliothek SM_DriveBasic.lib enthält zu ihrer Hauptfunktion, der Repräsentation der Antriebe im IEC-Programm, einige Hilfs-Bausteine:

2.2.1 Mathematische Hilfsbausteine

Die Funktion **SMC_sgn** gibt den Vorzeichenwert des Eingangs zurück, also –1 bei negativer, +1 bei positiver und 0 bei Null-Eingabe.

Die Funktion **SMC_fmod** berechnet den Modulo-Wert des Eingangs x zur Periode m. Der Rückgabewert liegt stets im Intervall [0, m[.

Die Funktion **SMC_atan2** berechnet und gibt den Winkel α zurück, der folgende Gleichungen löst:

$\sin(\alpha) * f = \text{Sinus}$ und $\cos(\alpha) * f = \text{Cosinus}$.

Im Gegensatz zur gewöhnlichen ATAN-Funktion deckt der Wertebereich hier das ganze Intervall [0; 2π[ab.

2.2.2 Achsgruppen-Bausteine

Zykluszeitmessung

Häufig ist es für die Diagnose wichtig, die exakte Zykluszeit einer Achsgruppe messen zu können, insbesondere wenn die Task aktiv die Kommunikation steuert (wie etwa, wenn die SPS als CAN-SYNC-Erzeuger auftritt). In der Achsgruppen-Datenstruktur findet man die Variable <AxisGroup>.**bDebug** (BOOL), die die Zeitmessung steuert. Steht sie auf TRUE, wird in <AxisGroup>.**udiTaskMinTime** und <AxisGroup>.**udiTaskMaxTime** die minimale bzw. maximale Zeit, die zwischen zwei Task-Aufrufen vergangen ist, angegeben.

SMC_IsAxisGroupReady

Diese Funktion gibt über eine BOOLsche Variable zurück, ob der Hochlauf, der für jede Achsgruppe implizit beim Start des IEC-Programms durchgeführt wird, abgeschlossen ist, und diese mit ihren Achsen betriebsbereit ist (TRUE), oder ob der Hochlauf noch nicht abgeschlossen oder ein Fehler aufgetreten ist (FALSE).

SMC_GetAxisGroupState

Dieser Funktionsblock gibt genauer über den Zustand der Achsgruppe Aufschluß:

Eingänge (VAR_INPUT) des Bausteins:

bEnable : BOOL

Ist dieser Eingang TRUE, liefert der Baustein Informationen über den Zustand der Achsgruppe.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

AxisGroup : SMC_AXISGROUP_REF

Achsgruppe, für die Informationen benötigt wird.

Ausgänge (VAR_OUTPUT) des Bausteins:

bDone : BOOL

TRUE, sobald gültige Daten auf den Ausgängen.

wState : WORD

Interne Zustandsvariable der Achse.

bStartingUp : BOOL

Achsgruppe wird hochgefahren, d.h. die Antriebe werden konfiguriert. (0 <= wState <= 99)

bNormalOperation: BOOL

Achsgruppe im normalen Betrieb. (wState = 100)

bResetting: BOOL

Achsgruppe wird gerade reinitialisiert. (200 <= wState <= 210)

bErrorDuringStartUp: BOOL

Während des Hochlaufs trat ein Fehler auf. (wState >= 1000)

pErrorDrive: POINTER TO AXIS_REF

Zeiger auf die Fehler-verursachende Achse. Nur gültig wenn bErrorDuringStartUp = TRUE.

Mithilfe dieser Ausgabe kann die fehlerhafte Achse durch Setzen der Variable bDisableDriveInAxisGroup zur Laufzeit aus der Achsgruppe gelöscht werden, diese mittels SMC_ResetAxisGroup frisch initialisiert werden, und mit den restlichen, funktionierenden Achsen fortgefahren werden, sofern die nötigen Redundanzen in der Maschine existieren

SMC_ResetAxisGroup

Mit diesem Funktionsblock kann eine komplette Achsgruppe reinitialisiert werden.

Eingänge (VAR_INPUT) des Bausteins:

bExecute : BOOL

Ist dieser Eingang TRUE, startet der Baustein mit der Reinitialisierung der Achsgruppe.

bKeepRatioSettings: BOOL

Ist dieser Eingang TRUE, werden die bisherigen Getriebeeinstellungen (dwRatioTechunitsDenom und iRatioTechUnitsNum), der Modulowert (fPositionPeriod) und der Achstyp (iMovementType, linear/rotatorisch) beibehalten und nicht durch die in der Steuerungskonfiguration eingestellten Werte ersetzt.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

AxisGroup : SMC_AXISGROUP_REF

Achsgruppe, die reinitialisiert werden soll.

Ausgänge (VAR_OUTPUT) des Bausteins:

bDone : BOOL

TRUE, wenn der Vorgang abgeschlossen ist.

bError : BOOL

Fehler trat auf.

nErrorID: SMC_ERROR

Beschreibung des Fehlers.

SMC_DisableAllDrives

Mit dieser Funktion wird die Variable bDisableDriveInAxisGroup jedes Antriebs gesetzt. Das hat zur Folge, dass beim nächsten Ausführen der Funktion SMC_ResetAxisGroup kein Antrieb mehr zur Verfügung steht, sondern ausschließlich die Basis-Kommunikationsmechanismen der Achsgruppe ausgeführt werden.

Eingänge (VAR_INPUT) der Funktion:

pAG : POINTER TO SMC_AXIS_GROUP

Zeiger auf die Achsgruppe.

Rückgabewert der Funktion:

TRUE.

SMC_EnableAllDrives

Im Gegensatz zu SMC_DisableAllDrives werden mit dieser Funktion wieder alle Antriebe eingeschaltet und durch den Aufruf von SMC_ResetAxisGroup hochgefahren.

Eingänge (VAR_INPUT) der Funktion:

pAG : POINTER TO SMC_AXIS_GROUP

Zeiger auf die Achsgruppe.

Rückgabewert der Funktion:

TRUE.

2.2.3 Konfigurations-Bausteine

SMC_ChangeGearingRatio

Mithilfe dieses Bausteins kann das IEC-Programm zur Laufzeit das Getriebeverhältnis und die Antriebsart ändern.

Nach dem Ausführen dieses Bausteins sollte die Achsgruppe mittels SMC_ResetAxisGroup (bKeepRatioSettings=TRUE) neu gestartet werden, da nur so garantiert werden kann, dass alle Variablen korrekt initialisiert sind!

Eingänge (VAR_INPUT) des Bausteins:

bExecute : BOOL

Mit einer steigenden Flanke wird der Baustein aktiv.

dwRatioTechUnitsDenom : DWORD

iRatioTechUnitsNum: DWORD

Umrechnungsverhältnis zwischen SoftMotion-Einheit und Inkrementen (siehe 2.1)

fPositionPeriod: LREAL

Positionsperiode, Modulowert (nur für rotatorische Antriebe) (siehe 2.1)

iMovementType: INT

o: rotatorische Achse, 1: lineare Achse.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Antrieb, dessen Getriebeverhältnis geändert werden soll.

Ausgänge (VAR_OUTPUT) des Bausteins:

bDone : BOOL

TRUE, sobald die Aktion ausgeführt ist.

bError : BOOL

TRUE, wenn Fehler auftrat.

nErrorID : SMC_Error

Beschreibt den Fehler.

2.2.4 Regelungsmodus-Bausteine

SMC_SetControllerMode

Mithilfe dieses Bausteins kann – falls vom Antrieb unterstützt – in einen anderen Regelungsmodus geschaltet werden.

Eingänge (VAR_INPUT) des Bausteins:

bExecute : BOOL

Mit einer steigenden Flanke wird der Baustein aktiv.

nControllerMode: SMC_CONTROLLER_MODE

Gewünschter Regelungsmodus: SMC_torque (Drehmoment), SMC_velocity (Geschwindigkeit), SMC_position (Position), SMC_current (Strom)

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF (VAR_IN_OUT)

Antrieb, dessen Regelungsmodus geändert werden soll.

Ausgänge (VAR_OUTPUT) des Bausteins:

bDone : BOOL (VAR_OUTPUT)

TRUE, sobald die Aktion ausgeführt ist.

bError : BOOL (VAR_OUTPUT)

TRUE, wenn Fehler auftrat.

nErrorID : SMC_Error (VAR_OUTPUT)

Beschreibt den Fehler.

2.2.5 Direkte Sollwertvorgabe

SoftMotion gibt dem Anwender die Möglichkeit, Sollwerte für Position, Geschwindigkeit und Drehmoment/Kraft (siehe SMC_SetTorque) direkt vorzugeben und auf die Achse zu schreiben. Die dazu bereitgestellten Bausteine können z.B. dazu verwendet werden, selbst berechnete Profile auf die Achse zu schreiben.

Ein weiteres wichtiges Anwendungsgebiet ist die CNC-Steuerung, wo Sollwerte in X/Y/Z erst durch einen Transformationsbaustein laufen und auf Sollwerte für die einzelnen Achsen umgerechnet werden müssen. Um diese Sollwerte dann an die Achse zu senden und ggf. Grenzwert-Überwachungen durchzuführen, stehen folgende Bausteine zur Verfügung.

Die Bausteine SMC_FollowPosition/Velocity schreiben vorgegebene Sollwerte ohne weitere Überprüfung auf die Achsstruktur. Der Baustein SMC_CheckLimits überprüft die aktuellen Sollwerte auf Einhaltung der Grenzen hin.

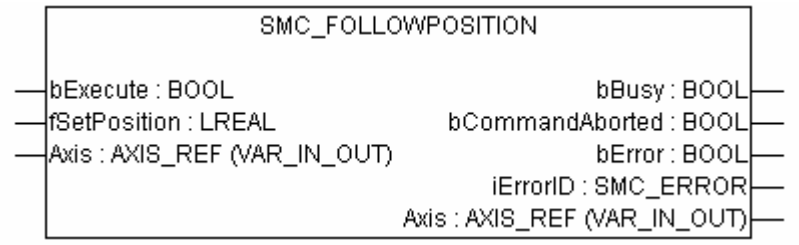
Die Bausteine SMC_ControlAxisByPos/Vel sind speziell für die CNC gedacht und bieten zusätzliche Funktionalität: Solange die Grenzen nicht überschritten werden, arbeiten sie wie SMC_FollowPosition/Velocity.

Tritt aber ein Sollwert-Sprung auf (was z.B. durch das sequentielle Abfahren zweier CNC-Programme entsteht, bei denen Endpunkt und Startpunkt nicht übereinstimmen), halten sie den Interpolator an, positionieren auf die aktuelle Interpolator-Sollposition und lösen die Sperre wieder, worauf der Interpolator weiterfährt.

Diese Bausteine sind also dafür gedacht Positions-Sollwertsprünge auszugleichen. Einen solchen detektieren sie, indem sie die Einhaltung der Maxima (von Geschwindigkeit und Beschleunigung) überprüfen. Die Überschreitung der Grenzen kann aber auch dann auftreten, wenn die Geschwindigkeit des CNC-Programms oder der Override-Wert am Interpolator zu hoch, oder die Maximalwerte der Achse in der Steuerungskonfiguration zu niedrig (oder gar nicht) eingestellt wurden. Resultieren würde eine ruckelige Fahrt, da der Interpolator, sobald die Bahngeschwindigkeit so hoch ist, dass eine Achsgrenze überschritten wird, immer wieder gebremst wird und die Achsen einzeln auf die entsprechende Position verfahren werden. Dann würde der Interpolator wieder anfahren um kurz darauf wieder gebremst zu werden.

SMC_FollowPosition

Dieser Baustein schreibt Positions-Sollwerte auf die Achse und führt dabei keinerlei Überprüfungen durch.



Eingänge des Bausteins:

bExecute: BOOL

Durch eine steigende Flanke beginnt der Baustein, die in fSetPosition eingegebenen Sollwerte auf die Axis-Datenstruktur zu schreiben. Er blebit so lange aktiv, bis er durch einen anderen Baustein (z.B. MC_Stop) unterbrochen wird.

fSetPosition: LREAL

Soll-Position in technischen Einheiten.

Ausgänge des Bausteins:

bBusy : BOOL (Default: FALSE)

Mit TRUE zeigt der Baustein an, dass er aktiv ist und Sollwerte auf die Achse schreibt.

bCommandAborted : BOOL (Default: FALSE)

Bei TRUE wurde der Baustein durch einen anderen abgebrochen.

bError : BOOL (Default: FALSE)

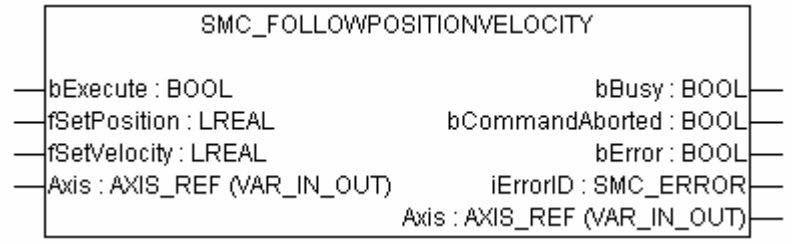
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

iErrorID : SMC_Error (INT)

Fehlernummer

SMC_FollowPositionVelocity

Dieser Baustein schreibt Positions- und Geschwindigkeits-Sollwerte auf die Achse und führt dabei keinerlei Überprüfungen durch. Der Anwender hat dafür Sorge zu tragen, dass die Werte sinnvoll zusammenpassen.



Eingänge des Bausteins:

bExecute: BOOL

Durch eine steigende Flanke beginnt der Baustein, die in fSetPosition eingegebenen Sollwerte auf die Axis-Datenstruktur zu schreiben. Er blebit so lange aktiv, bis er durch einen anderen Baustein (z.B. MC_Stop) unterbrochen wird.

fSetPosition: LREAL

Soll-Position in technischen Einheiten.

fSetVelocity: LREAL

Soll-Geschwindigkeit in technischen Einheiten pro Sekunde.

Ausgänge des Bausteins:

bBusy : BOOL (Default: FALSE)

Mit TRUE zeigt der Baustein an, dass er aktiv ist und Sollwerte auf die Achse schreibt.

bCommandAborted : BOOL (Default: FALSE)

Bei TRUE wurde der Baustein durch einen anderen abgebrochen.

bError : BOOL (Default: FALSE)

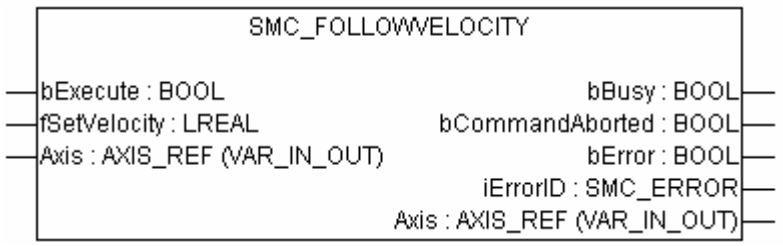
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

iErrorID : SMC_Error (INT)

Fehlernummer

SMC_FollowVelocity

Dieser Baustein schreibt Geschwindigkeits-Sollwerte auf die Achse und führt dabei keinerlei Überprüfungen durch.



Eingänge des Bausteins:

bExecute: BOOL

Durch eine steigende Flanke beginnt der Baustein, die in fSetPosition eingegebenen Sollwerte auf die Axis-Datenstruktur zu schreiben. Er blebit so lange aktiv, bis er durch einen anderen Baustein (z.B. MC_Stop) unterbrochen wird.

fSetVelocity: LREAL

Soll-Geschwindigkeit in technischen Einheiten pro Sekunde.

Ausgänge des Bausteins:

bBusy : BOOL (Default: FALSE)

Mit TRUE zeigt der Baustein an, dass er aktiv ist und Sollwerte auf die Achse schreibt.

bCommandAborted : BOOL (Default: FALSE)

Bei TRUE wurde der Baustein durch einen anderen abgebrochen.

bError : BOOL (Default: FALSE)

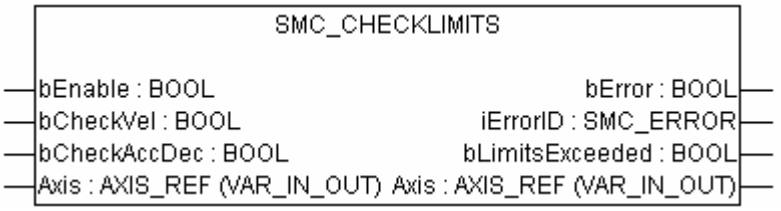
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

iErrorID : SMC_Error (INT)

Fehlernummer

SMC_CheckLimits

Dieser Baustein überprüft, ob die aktuellen Sollwerte des Antriebs die in der Steuerung eingestellten Maxima überschreiten und zeigt dieses über den Ausgang bLimitsExceeded an.



Eingänge des Bausteins:

bEnable: BOOL

Während bEnable gesetzt ist, wird die Überprüfung durchgeführt.

bCheckVel: BOOL

Wenn gesetzt, wird die Soll-Geschwindigkeit überprüft.

bCheckAccDec: BOOL

Wenn gesetzt, werden Soll-Beschleunigung und -Verzögerung überprüft.

Ausgänge des Bausteins:

bError : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

iErrorID : SMC_Error (INT)

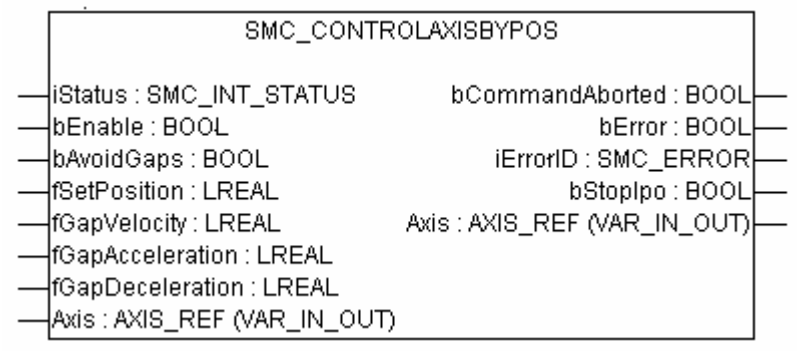
Fehlernummer

bLimitsExceeded : BOOL (Default: FALSE)

TRUE bedeutet, dass die aktuellen Sollwerte die in der Steuerungskonfiguration festgelegten und in Axis unter fSWMaxVelocity, fSWMaxAcceleration bzw. fSWMaxDeceleration gespeichert Werte überschreiten.

SMC_ControlAxisByPos

Dieser Baustein schreibt Sollpositionen in eine Antriebsstruktur und überwacht selbige auf Sprünge.



Eingänge des Bausteins:

iStatus: SMC_INT_STATUS

Zustand des Interpolations-Bausteins. Wird mit gleichnamigem Ausgang von SMC_Interpolator verbunden.

bEnable: BOOL

Kontrolliert die Achse, solange der Eingang TRUE ist.

bAvoidGaps: BOOL

TRUE: Baustein überwacht Position und Geschwindigkeit. Überschreitet die Geschwindigkeit den in der Achse gespeicherten Grenzwert fSWMaxVelocity (konfiguriert im Drive-Dialog unter „Maximalwerte“), setzt der Baustein den Ausgang bStopIpo und bewegt die Achse mit den Parametern fGapVelocity, fGapAcceleration und fGapDeceleration an diese Position und löscht den Ausgang bStopIpo wieder.

fSetPosition: LREAL

Soll-Position der Achse. Typischerweise ein Ausgang eines Transformationsbausteins.

fGapVelocity, fGapAcceleration, fGapDeceleration: LREAL

Bewegungsparameter für die Überbrückung eines Sprungs.

Ausgänge des Bausteins:

bCommandAborted : BOOL (Default: FALSE)

Bei TRUE wurde der Baustein durch einen anderen abgebrochen.

bError : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

iErrorID : SMC_Error (INT)

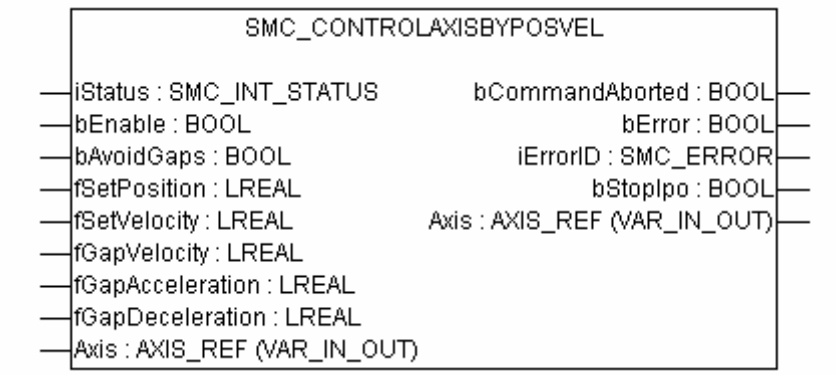
Fehlernummer

bStopIpo : BOOL (Default: FALSE)

Bei TRUE hat der Baustein einen Geschwindigkeits- oder Positionssprung entdeckt, und positioniert gerade auf die neue Position. Dieser Ausgang sollte deswegen mit dem EmergencyStop-Eingang des SMC_Interpolators verbunden werden, sodass der Interpolator wartet, bis die Achse in richtiger Position steht.

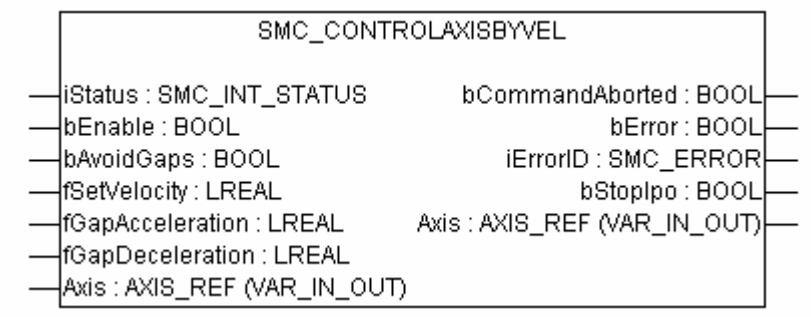
SMC_ControlAxisByPosVel

Dieser Baustein funktioniert ähnlich wie SMC_ControlAxisByPos, nur dass zusätzlich die Geschwindigkeit vorgegeben werden kann.

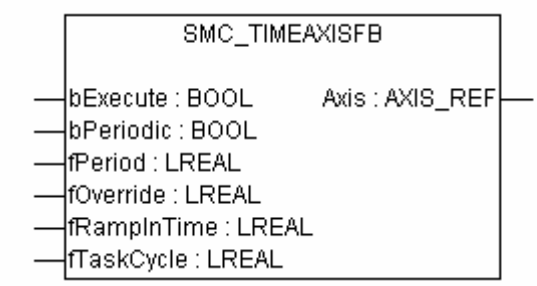


SMC_ControlAxisByVel

Dieser Baustein funktioniert ähnlich wie SMC_ControlAxisByPos, nur dass die Achse nicht über die Position, sondern über die Geschwindigkeit gesteuert wird.



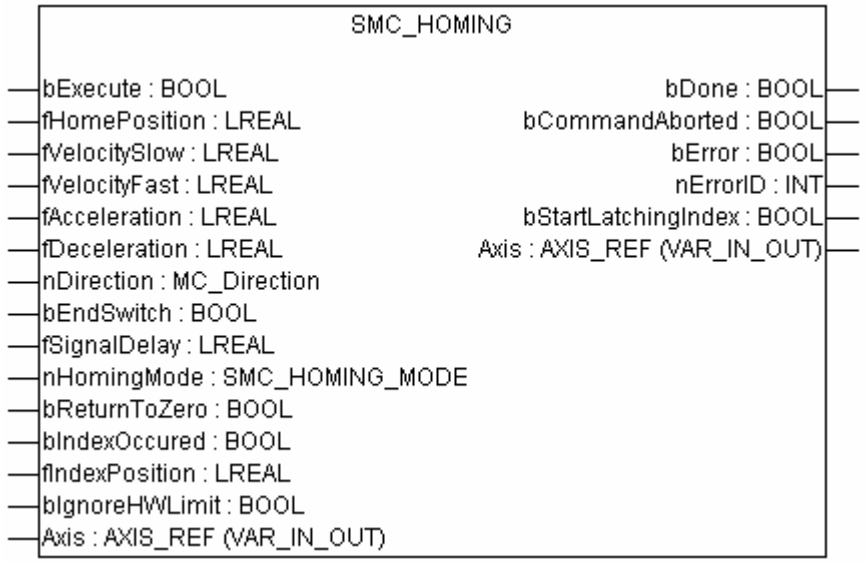
2.2.6 Virtuelle Zeit-Achse



Dieser Funktionsbaustein erzeugt eine Zeitachse, die er im Ausgang *Axis* (AXIS_REF) zur Verfügung stellt. Mit einer steigenden Flanke im Eingang bExecute beginnt die Sollposition der Zeitachse bei 0 beginnend in Sekunden hochzuzählen. Ist der Eingang bPeriodic gesetzt, so beginnt sie beim Erreichen der Zeit *fPeriod* erneut von 0. Der Eingang *fOverride* gibt einen Zeitmultiplikator an, der standardmäßig auf 1 gesetzt ist. Eine 2 würde doppelt so schnelles Ablaufen der Zeit bedeuten. *fRampInTime* legt fest, wie lange nach der Übernahme der neuen Sollwerte der Baustein Zeit hat, auf den neuen Override zu rampen. Im Eingang *fTaskCycle* muss die Zykluszeit der Task in Sekunden stehen, von welcher aus der Baustein aufgerufen wird.

2.2.7 Referenzieren über digitale Hardware-Eingänge

SMC_Homing



Dieser Funktionsbaustein ist in der Lage, die Referenzfahrt einer Achse auszuführen. Als Einschalter fungiert dabei ein BOOLscher Wert, der typischerweise ein Hardware_Input ist.

Wurde der Baustein mit steigender Flanke in *bExecute* gestartet, bewegt er die Achse mit *fVelocityFast* in die Richtung, die durch nDirection spezifiziert wurde, solange, bis *bEndSwitch* = FALSE, also der Referenzschalter geschlossen wird. Dann wird die Achse abgebremst und mit *fVelocitySlow* in die entgegen gesetzter Richtung gefahren. An der Stelle, an der sich der Referenzschalter öffnet (*bEndSwitch* = TRUE), wird die Referenzposition gesetzt (*fHomePosition*) und der Antrieb gestoppt.

Eingänge des Bausteins:

bExecute : BOOL (Default: FALSE)

Die Referenzfahrt des Antriebs soll bei steigender Flanke gestartet werden.

fHomePosition : REAL

Angabe der absoluten Position auf der Referenzposition [u].

fVelocitySlow, fVelocityFast : REAL

Soll-Geschwindigkeiten für Phase 1 und 2 in [u/s].

fAcceleration, fDeceleration : REAL

Soll-Beschleunigung und Bremsung in [u/s²].

nDirection : MC_Direction (Default: negative)

Richtung der Referenzfahrt: zulässige Werte: positive/negative.

bEndSwitch : BOOL (Default: TRUE)

Referenzschalter: TRUE (offen), FALSE (geschlossen).

fSignalDelay : REAL (Default: 0.0)

Übertragungszeit des Referenzschalters in s. Wird eine Zeit >0 angegeben, so verwendet der Baustein nicht die Position, bei der bEndSwicth TRUE wurde als Referenzposition, sondern die Position, die die Achse fSignalDelay Sekunden zuvor hatte.

nHomingMode : SMC_HOMING_MODE (Default: FAST_BSLOW_S_STOP)

FAST_BSLOW_S_STOP:

Der Antrieb wird in die angegebene Richtung mit der Geschwindigkeit fVelocityFast gefahren (FAST), bis der Eingang *bEndSwicth* FALSE wird, dann wird gestoppt und mit *fVelocitySlow* in die entgegen gesetzte Richtung bewegt (BSLOW) bis *bEndSwitch* wieder TRUE wird. An dieser Stelle wird der Referenzpunkt gesetzt (S) und gestoppt (STOP).

FAST_BSLOW_STOP_S:

Im Gegensatz zu FAST_BSLOW_S_STOP wird hier nach dem Freifahren erst gestoppt und dann der Referenzpunkt gesetzt.

FAST_BSLOW_I_S_STOP:

Im Gegensatz zu FAST_BSLOW_S_STOP wird hier nach dem Freifahren ein Index-Impuls (bIndexOccured=TRUE) und dessen Position fIndexPosition abgewartet, ddie als Referenzpunkt gesetzt wird. Dann erst wird gestoppt.

FAST_BSLOW_S_STOP/ **FAST_BSLOW_STOP_S** / **FAST_BSLOW_I_S_STOP:**

Diese Modi funktionieren genau wie die oben beschriebenen, mit dem Unterschied, dass beim erreichen des Referenzschalters nicht umgekehrt wird, sondern in derselben Richtung weitergefahren wird. Man beachte, dass in diesen Modi der Eingang bIgnoreHWLimits aus Sicherheitsgründen FALSE sein muss.

bReturnToZero: BOOL (Default: FALSE)

Ist dieses Flag gesetzt, positioniert der Baustein nach Abschluss der in *nHomingMode* festgelegten Prozedur auf den Nullpunkt.

bIndexOccured: BOOL (Default: FALSE)

Nur für nHomingMode **FAST_BSLOW_I_S_STOP:** Zeigt an, ob Index-Puls aufgetreten ist.

fIndexPosition: REAL (Default: 0.0)

Nur für nHomingMode **FAST_BSLOW_I_S_STOP: Gelatchte Position des** Index-Pulses.

Wenn dieser Eingang TRUE ist, wird die Hardware-Kontrolle der Endschalter ausgesetzt. Verwenden Sie diese Option, wenn Sie für Hardware-End- und Referenzschalter den gleichen physikalischen Schalter verwenden.

bIgnoreHWLimit: BOOL (Default: FALSE)

Wenn dieser Eingang TRUE ist, wird die Hardware-Kontrolle der Endschalter ausgesetzt. Verwenden Sie diese Option, wenn Sie für Hardware-End- und Referenzschalter den gleichen physikalischen Schalter verwenden.

Ausgänge des Bausteins:

bDone : BOOL (Default: FALSE)

Bei TRUE ist der Antrieb referenziert und im Stillstand.

bCommandAborted : BOOL (Default: FALSE)

Bei TRUE wurde der Befehl durch einen anderen abgebrochen.

bError : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

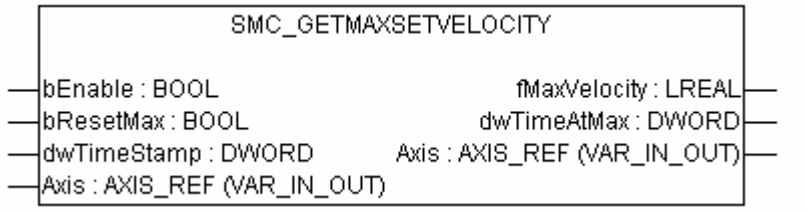
nErrorID : SMC_Error

Fehlernummer.

2.2.8 Diagnose-Bausteine

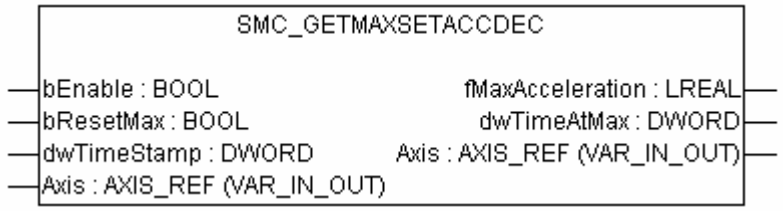
SMC_GetMaxSetVelocity

Dieser Funktionsbaustein ist in der Lage, die betragsmäßige Maximal(soll)geschwindigkeit einer Achse zu messen. Die Messung wird durchgeführt, wenn *bEnable* TRUE ist, und auf 0 zurückgesetzt, solange *bResetMax* TRUE ist. Mit *dwTimeStamp* hat man die Möglichkeit, ein beliebiges DWORD (z.B. Aufrufszähler), welches mit einem neuen Maximalwert übernommen und ausgegeben wird.



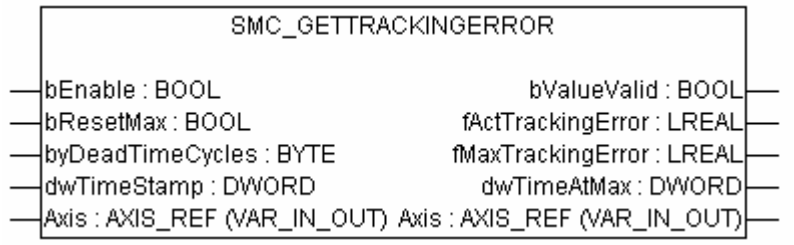
SMC_GetMaxSetAccDec

Dieser Baustein funktioniert analog zu SMC_GetMaxSetVelocity und bestimmt die betragsmäßig größte Beschleunigung oder Bremsung.



SMC_GetTrackingError

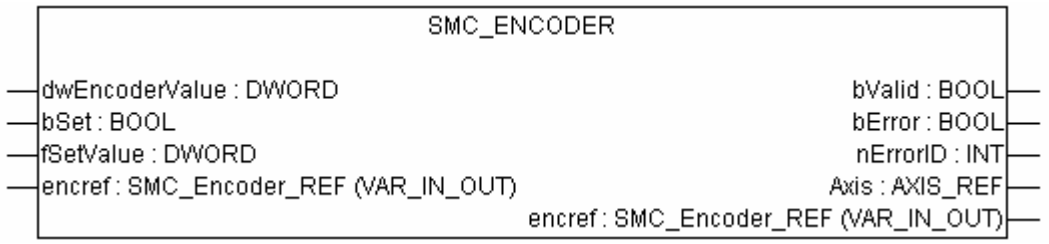
Dieser Baustein misst in Abhängigkeit von der Totzeit, die durch die Kommunikation über einen Feldbus entstehen kann und in Zykluszahlen angegeben wird (*byDeadTimeCycles*), den aktuellen und maximalen Schleppfehler. Wie bei SMC_GetMaxSetVelocity kann ein Zeitstempel (*dwTimeStamp*) verwendet werden, um den Zeitpunkt des Maximums zu messen.



2.2.9 Encoder

Über die Steuerungskonfiguration besteht die Möglichkeit, Encoder in eine Achsgruppe einzubinden und zu konfigurieren. Die damit angelegten Datenstrukturen vom Typ SMC_ENCODER_REF müssen von einer Instanz des Bausteins SMC_Encoder verarbeitet werden. Dieser liefert als Ausgabe eine AXIS_REF-Datenstruktur, die – sobald der Ausgang bValid die Gültigkeit der Daten bestätigt - als Eingang für alle anderen Funktionsbausteine (z.B. MC_CamIn, MC_GearIn, MC_TouchProbe) dienen kann.

Über den boolschen Eingang bSet kann der aktuelle Wert des Encoders auf den Eingang fSetValue gesetzt werden.



2.2.10 Visualisierungs-Templates

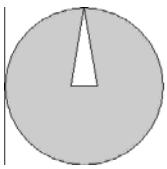
Für die zwei Typen von Antrieben (linear/rotatorisch) ist in dieser Bibliothek je ein Visualisierungs-Template enthalten, welches mit einer Achsstruktur (AXIS_REF) verknüpft werden kann und die Ist-Position des Antriebs visualisiert:

LinDrive

Für einen Linearantrieb wird obiges Bild dargestellt. Der Schlitten wird entsprechend seiner aktuellen Position relativ zur unteren und oberen Positionsgrenze platziert und erhält eine blaue Farbe, sobald er sich in Regelung befindet. Voraussetzung für die Verwendung des Templates ist, dass die Parameter fSWLimitPositive und fSWLimitNegative gesetzt sind.

Das Template **LinDrive_V** stellt den Antrieb in vertikaler Form dar.

RotDrive



Für einen Rotationsantrieb wird obiges Bild dargestellt. Die aktuelle Position wird durch die Stellung des Pfeils dargestellt und erhält eine blaue Farbe, sobald sich der Antrieb in Regelung befindet. Voraussetzung für die Verwendung des Templates ist, dass der Parameter fPositionPeriod gesetzt ist.

2.3 Antriebs-Treiber <BusInterfaceBezeichnung>Drive.lib

- Antriebs-Treiber sind für die Kommunikation zwischen IEC-Programm, speziell den AXIS_REF-Strukturen, und den Antrieben verantwortlich. Sie sind CoDeSys-Bibliotheken und beinhalten mindestens die drei unter 0 erwähnten Funktionen. Diese Bibliotheken werden typischerweise vom Steuerungshersteller mitgeliefert, und müssen in das Projekt eingebunden sein.

- DummyDrive.lib ist ein Beispiel für Antriebs-Treiber-Bibliotheken und wird mit den SoftMotion-Bibliotheken mitgeliefert. Auch wenn diese Bibliothek keine echten Antriebe bedient, funktioniert sie nach dem gleichen Prinzip.

2.3.1 SercosDrive.lib

Mit dieser Bibliothek, die wiederum die externe Bibliothek SercosBase.lib als Schnittstelle zur Hardware verwendet, können alle Sercos-konformen Antreibe gesteuert werden.

Ähnlich wie bei CAN existieren Funktionsbausteine für das Lesen und Schreiben von Parametern:

- **SMC_ReadSercosParameter**
- **SMC_WriteSercosParameter**
- **SMC_ReadSercosList**
- **SMC_WriteSercosList**
- **SMC_ReadSercosString**

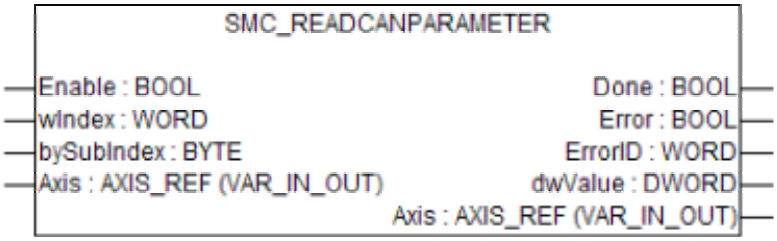
Der exakte Funktionsumfang wird im Dokument SercosDrive.pdf beschrieben.

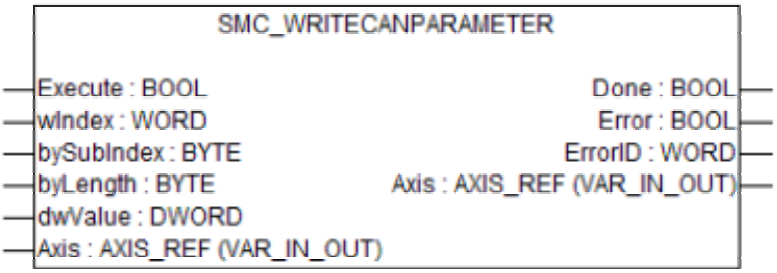
2.3.2 SM_CAN.lib

- Für jeden angebundenen CAN-Antrieb wird - im Gegensatz zu Sercos - ein eigener Treiber benötigt.

- Gemeinsam für alle CAN-Antriebe kann jedoch – sofern dies im cfg-File eingestellt ist – in der Steureungskonfiguration beim Achsgruppen-Dialog die Baudrate und die Nummer des CAN-Controllers (beginnend bei 0) festgelegt werden. Um den Bus deterministisch zu halten, werden pro CAN-Kanal entweder I/Os oder Antriebe, niemels jedoch beides zusammen betrieben. Sollte die 3S-CANopen-Library verwendet werden, nimmt diese automatisch die ersten CAN-Controller und so kann für die Achsgruppe ein anderer reserviert werden.

- Auf der Bibliothek SM_CAN.lib bauen alle von 3S erstellten CAN-Bibliotheken auf. In ihr sind zwei Bausteine enthalten, die für den Anwender praktische Bedeutung haben, da man mit ihnen auf einfache Art Parameter des Antriebs auslesen oder beschreiben kann: **SMC_ReadCANParameter** und **SMC_WriteCANParameter**. In ihrer Funktionsweise gleichen sie den beiden Bausteinen MC_ReadParameter und MC_WriteParameter:





Eingänge der Bausteine:

Enable : BOOL

TRUE: Parameter wird fortlaufend gelesen.

Execute : BOOL

Steigende Flanke: Einmaliges Schreiben des Parameters wird angestoßen.

wIndex, **bySubIndex : BYTE**

Index und Subindex des Parameters, der gelesen bzw. geschrieben werden soll.

Axis : AXIS_REF (VAR_IN_OUT)

Antrieb, dessen Parameter gelesen/geschrieben werden soll.

Ausgänge der Bausteine:

dwValue : DWORD

Gelesener Wert, bzw. zu schreibender Wert.

Done : BOOL

TRUE: Aktion erfolgreich beendet.

Error : BOOL

TRUE: Fehler trat auf.

ErrorID : WORD

Fehlernummer gemäß SMC_Error.

2.4 Variablen der Struktur AXIS_REF

Für jeden in der Steuerungskonfiguration angelegten Antrieb (Drive) wird von CoDeSys beim Übersetzen eine Variable der Struktur AXIS_REF angelegt (definiert in der SM_DriveBasic.lib, siehe Kapitel 2.3). Diese Struktur bildet die Schnittstelle zwischen Applikation und Antriebsinterface. Über sie werden sowohl zyklische als auch azyklische Daten ausgetauscht.

Die meisten Variablen der Struktur sind für den Anwender nicht von Bedeutung, sondern werden vom System intern verwendet. Der Anwender sollte stets Funktionsblöcke verwenden und nie direkt auf die Struktur zugreifen, zumindest nicht schreibend.

Nr.	Name	Datentyp	Initwert	Kommentar
1000	nAxisState	INT	standstill	Achsenstatus: 0: power_off 1: errorstop 2: stopping 3: standstill 4: discrete_motion 5: continuous_motion 6: synchronized_motion 7: homing
1001	wControlType	WORD	PLC-Config*	Ziffer, die anzeigt, welche Komponenten der Struktur zyklisch gesendet und empfangen werden. 0: definiert durch param 1002-1008 1: SetTorque 2: SetVelocity,ActVelocity 3: SetVelocity,ActPosition 4: SetPosition,ActPosition 5: SetVelocity,SetPosition, ActVelocity,ActPosition 6: SetVelocity
1002 1003 1004	wCyclicDataS1 wCyclicDataS2 wCyclicDataS3	WORD	PLC-Config* oder Init-FB von <BusInterfaceBezeichnung>Drive.lib	Anzahl der 3S Parameter, die pro Zyklus gesendet werden sollen
1006 1007 1008	wCyclicDataR1 wCyclicDataR2 wCyclicDataR3	WORD	PLC-Config* or Init-FB of <BusInterfaceBezeichnung>Drive.lib	Parameternummern der 3S Parameter, die pro Zyklus empfangen werden sollen
1010	bRegulatorOn	BOOL	FALSE	Regler (Leistung) zuschalten/abschalten
1011	bDriveStart	BOOL	FALSE	Bremse aktivieren/deaktivieren
1012	bCommunication	BOOL	FALSE	TRUE: Antrieb antwortet
1013	wCommunicationState	WORD	0	Interne Verwendung
1015	bRegulatorRealState	BOOL	FALSE	Dustan des Reglers
1016	bDriveStartRealState	BOOL	FALSE	Zustand der Bremse
1020	wAxisGroupId	WORD	PLC-Config*	Index der Achsgruppe in der Konfiguration
1021	wDriveId	WORD	PLC-Config*	Knotennummer des Antriebs auf dem Feldbus

Nr.	Name	Datentyp	Initwert	Kommentar
1022	iOwner	INT	0	Id-Number des momentanen Owners (FB)
1023	iNoOwner	INT	0	Anzahl der früheren und momentanen Owner
1024	bMovedInThisCycle	BOOL	FALSE	wurde Antrieb in diesem IEC-Zyklus bewegt
1025	fTaskCycle	REAL	PLC-Config*	Task-Zykluszeit in s
1026	bRealDrive	BOOL	PLC-Config*	TRUE: erzeugt durch Config; FALSE: erzeugt durch IEC-Programm
1030	bError	BOOL	FALSE	Fehler ist aufgetreten
1031	wErrorID	WORD	0	Fehlernummer
1032	bErrorAckn	BOOL	FALSE	Bestätige Fehler
1035	fbeFBError	ARRAY [0..g_SMC_NUMBER_FB_ERRORS] OF SMC_FBERROR	0	FB-Fehlernummer-Speicher
1051	dwRatioTechUnitsDenom	DWORD	PLC-Config*	Konvertierung technischer Einheiten in Inkremente: Denominator
1052	iRatioTechUnits Num	INT	PLC-Config*	Konvertierung technischer Einheiten in Inkremente: Numerator
1053	nDirection	MC_Direction	positive	-1 : negativ (fSetVelocity < 0), 1: positiv
1054	fScalefactor	REAL	1	Konvertierung von Bus-Einheit in technische Einheit per auf dem Bus empfangener Einheit
1055	fFactorVel	REAL	1	Konvertierung von Bus-Einheit in techn. Inherited/s
1056	fFactorAcc	REAL	1	Konvertierung von Bus-Einheit in techn. Einheit/s^2
1057	fFactorTor	REAL	1	Konvertierung von Bus-Einheit in Nm oder N
1058	fFactorJerk	REAL	1	Konvertierung von Bus-Einheit in techn. Einheit/s^3
1060	iMovementType	INT	1	0: Rotatorisch (modulo); 1: Linear
1061	fPositionPeriod	REAL	1000	Länge der Periode für rotatorische Antriebe in techn.Einheiten
1091	byControllerMode	BYTE	3	1: Drehmomentkontrolle 2: Geschwindigk.kontrolle 3: Positionskontrolle
1092	byRealControllerMode	BYTE	0	tatsächlicher Regelmodus

Nr.	Name	Datentyp	Initwert	Kommentar
1100/1	fSetPosition	REAL	0	Vorgegebene Position in techn. Einheiten
1101	fActPosition	REAL	0	Aktuelle Position in techn. Einheiten
1105	fAimPosition	REAL	0	Zielposition (für manche MC_FBs)
1106	fMarkPosition	REAL	0	interne Positionsmarke
1107	fSavePosition	REAL	0	interne Position am Zyklusanfang
1110,11	fSetVelocity	REAL	0	Vorgegebene Geschwindigkeit in techn. Einheiten/sec
1111,10	fActVelocity	REAL	0	Aktuelle Geschw. in techn. Einheiten/sec
1112,9	fMaxVelocity	REAL	100	Maximale Geschw. in techn. Einheiten/sec
1113	fSWMaxVelocity	REAL	100	maximale Geschw. für implizite Bewegungen in techn. Einheiten/sec
1115	bConstantVelocity	BOOL	FALSE	Achse fährt mit konstanter Geschwindigkeit
1116	fMarkVelocity	REAL	0	interne Geschwindigkeitsmarke
1117	fSaveVelocity	REAL	0	interne Geschwindigkeit am Zyklusanfang
1120	fSetAcceleration	REAL	0	Vorgegebene Beschleunigung in techn. Einheiten/sec^2
1121	fActAcceleration	REAL	0	Aktuelle Beschleunigung in techn. Einheiten/sec^2
1122,13	fMaxAcceleration	REAL	100	Maximale Beschleunigung in techn. Einheiten/sec^2
1123	fSWMaxAcceleration	REAL	100	maximale Beschleunigung für implizite Bewegungen in techn. Einheiten/sec
1125	bAccelerating	BOOL	FALSE	Achse beschleunigt gerade
1126	fMarkAcceleration	REAL	0	interne Beschleunigungsmarke
1127	fSaveAcceleration	REAL	0	interne Beschleunigung am Zyklusanfang
1130	fSetDeceleration	REAL	0	Vorgegebene Abbremsung in techn. Einheiten/sec^2
1131	fActDeceleration	REAL	0	Aktuelle Abbremsung in techn. Einheiten/sec^2
1132,15	fMaxDeceleration	REAL	100	Maximale Abbremsung in techn. Einheiten/sec^2
1133	fSWMaxDeceleration	REAL	100	maximale Abbremsung für implizite Bewegungen in techn. Einheiten/sec

Nr.	Name	Datentyp	Initwert	Kommentar
1135	bDecelerating	BOOL	FALSE	Achse bremst gerade ab
1137	fSaveDeceleration	REAL	0	interne Abbremsung am Zyklusanfang
1140	fSetJerk	REAL	0	Vorgegebener Jerk in techn. Einheiten/sec^3
1141	fActJerk	REAL	0	aktueller Jerk in techn. Einheiten/sec^3
1142,16	fMaxJerk	REAL	100	maximaler Jerk in techn. Einheiten/sec^3
1143	fSWMaxJerk	REAL	100	maximaler Jerk für implizite Bewegungen in techn. Einheiten/sec
1146	fMarkJerk	REAL	0	interne Jerk-Marke
1147	fSaveJerk	REAL	0	interner Jerk am Zyklusanfang
1150	fSetCurrent	REAL	0	Vorgegebene Stromstärke (A)
1151	fActCurrent	REAL	0	Aktuelle Stromstärke (A)
1152	fMaxCurrent	REAL	100	Maximale Stromstärke (A)
1153	fSWMaxCurrent	REAL	0	maximale Stromstärke für implizite Bewegungen in techn. Einheiten/sec
1160	fSetTorque	REAL	0	Vorgeg. Drehmoment in Nm bzw. N (linear)
1161	fActTorque	REAL	0	Aktuelles Drehmoment in Nm bzw. N (linear)
1162	fMaxTorque	REAL	0	Maximales Drehmoment in Nm bzw. N (linear)
1200,2	fSWLimitPositive	REAL	0	Positionsgrenze positiver Richtung in techn. Einheiten
1201,3	fSWLimitNegative	REAL	0	Positionsgrenze in negativer Richtung in techn. Einheiten
1202	fCaptPosition	REAL	0	Capture position in techn. Einheiten
1204	bSWEndSwitchActive	BOOL	FALSE	SW-Endschalter aktiv
1205	bSWLimitEnable	BOOL	FALSE	Software-Endschalter aktivieren
1206	bHWLimitEnable	BOOL	FALSE	Hardware-Endschalter aktivieren/deaktivieren
1207	bCaptureOccurred	BOOL	FALSE	Capture hat stattgefunden (durch Schreiben bestätigen)
1208	bStartCapturing	BOOL	FALSE	Capture für aktuellen Trigger starten/beenden
1210	bStartReference	BOOL	FALSE	TRUE: Referenzlauf starten

Nr.	Name	Datentyp	Initwert	Kommentar
1211	fReference	REAL	0	Referenzposition
1215	fOffsetPosition	REAL	0	Nullpunktsverschiebung
1220	fFirstCapturePosition	REAL	0	Fenster-Start-Position für Capture
1221	fLastCapturePosition	REAL	0	Fenster-End-Position für Capture
1222	tiTriggerInput	TRIGGER_REF		Beschreibung des Capture-Eingangs
1223	bCaptureWindowActive	BOOL	FALSE	Capture auf Fenster beschränkt
1230	dwPosOffsetForResiduals	DWORD	0	interne Variable zur Restwertbehandlung
1231	dwOneTurn	DWORD	0	interne Variable zur Restwertbehandlung
1232	fLastPosition	REAL	0	interne Variable zur Restwertbehandlung
1234	iRestNumerator	INT	0	interne Variable zur Restwertbehandlung
1235	iTurn	INT	0	interne Variable zur Restwertbehandlung
1236	dwBusModuloValue	DWORD	0	interne Variable zur Restwertbehandlung
1237	dwPosOffsetForResidualsHoming	DWORD	0	interne Variable zur Restwertbehandlung
1300	bDisableDriveInAxisGRoup	BOOL	FALSE	Antrieb aus Achsgruppe löschen
1301	bErrorDuringStartup	BOOL	FALSE	gesetzt, wenn Fehler während des Hochlaufs
	pMS	POINTER TO BYTE	0	Zeiger auf Antriebsspezifische Struktur <BusInterfaceBezeichnung>_AXIS_REF

Jede AXIS_REF-Strukturvariable verhält sich entsprechend der PLCopen-Spezifikation „Function blocks for motion control“, Version 1.0.

2.5 Parametrierung des Antriebs

Viele wichtige Konfigurationsdaten sind im Antrieb gespeichert. Zwar bietet SoftMotion die Möglichkeit, Parameterwerte in der Steuerungskonfiguration vorzugeben, die während der Hochlaufphase übertragen werden (siehe 2.1.3), allerdings ist dies für den normalen Anwender schwierig, da er in der Regel nicht weiß, welche Parameter zu übertragen sind und welchen Wert diese bekommen sollen. So sind häufig für die Erstinbetriebnahme (auch bei mehreren seriengleichen Maschinen) und beim Austausch der Antriebe die antriebsspezifischen Tools notwendig. In SoftMotion ist deshalb eine Funktionalität integriert, die dies zumindest teilweise übernimmt.

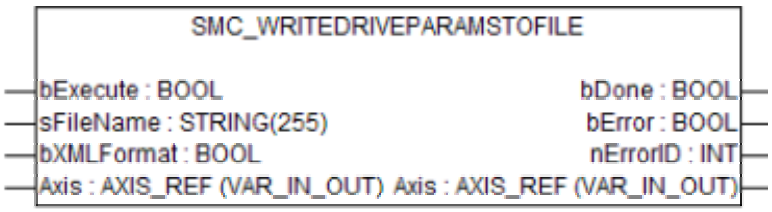
Deshalb gibt es folgende Varianten:

(A) Der Anwender konfiguriert den Antrieb mit dem Tool des Antriebsherstellers. Er geht mit einem SoftMotion-Projekt, bei dem die Antriebe registriert sind online, und ruft einen Baustein auf, der alle Antriebsparameter liest und in ein XML-File speichert, welches der Anwender von der Steuerung zurückliest, und im entsprechenden Konfigurationsdialog einliest. Dann sind alle nötigen Parameter im **Projekt gespeichert** und werden bei jedem Start übertragen. Weder bei der Erstinbetriebnahme von weiteren Maschinen, noch beim Ersetzen eines defekten Antriebs würde das Inbetriebnahme-Tool gebraucht werden.

(B) Der Anwender konfiguriert den Antrieb mit dem Tool des Antriebsherstellers. In seiner Applikation sieht er vor, dass der Anwender über einen FB die Parameter des Antriebs in ein ASCII-File auf die **Steuerung speichern** kann. Im Gegensatz zu A liegen die Parameter in einem File auf der Steuerung und nicht innerhalb der Applikation.

Beide Lösungen haben Vor- und Nachteile: Um bei A einen Parameter nachträglich zu ändern, muss das Projekt neu downgeloadet werden. Bei B funktioniert dies, dafür muss beachtet werden, dass bei jeder Neuinbetriebnahme entweder die Parameterdateien mit auf der Steuerung abgelegt werden, oder das Drive-Inbetriebnahmetool verwendet werden muss.

SMC_WriteDriveParamsToFile



Dieser Baustein liest alle Konfigurations-Parameter des Antriebs aus und speichert diese in einer Datei. Da er einen Dateizugriff durchführt, der evtl. mehrere ms die Abarbeitung der Applikation blockiert, darf er i.d.R. nicht in der Motion-Task aufgerufen werden, sondern sollte in einer niedrigerprioren Task ausgeführt werden.

Welche Parameter ausgelesen werden sollen, erfährt der Baustein vom Antriebs-Treiber, der dies entweder vom Antrieb selbst erfahren kann (Sercos), oder eine Standard-Parameterliste enthält (bei CAN: siehe globale Variablenliste der Standard-3S-Treiber). Bei CAN-Antrieben kann eine eigene Liste gleichen Formats erstellt und dem Antrieb über folgende Zuweisung zugeteilt werden:

```
<Drive>_MS.pParameterlist := ADR(<NeueList>);
```

Ein-/Ausgänge des Bausteins:

bExecute: BOOL

Mit TRUE wird der Baustein gestartet.

sFileName: STRING(255)

Dateiname.

bXMLFormat: BOOL

Ist diese Variable TRUE, dann wird die Datei im XML-Format erstellt (und kann anschließend im Antriebs-Dialog importiert werden), ansonsten im Text-Format (welches auf der Steuerung liegen und vom Antriebstreiber während des Hochlaufs eingelesen und versendet werden kann).

Axis: AXIS_REF

Antrieb, dessen Parameter ausgelesen werden sollen.

bDone: BOOL

Aktion abgeschlossen.

bError: BOOL

Fehler trat auf.

nErrorID: INT

Fehlernummer (siehe 9.2)

3 Der CNC-Editor

3.1 Überblick

Der CNC-Editor in CoDeSys erlaubt es, mehrdimensionale Bewegungen gleichzeitig graphisch und textuell angelehnt an die CNC-Sprache **DIN66025** zu programmieren. Sehen Sie zur CNC-Sprache Kapitel 3.2, zum Texteditor Kapitel 3.4 und zum Grafikeditor Kapitel 3.5. Programmierbeispiele finden Sie in Kapitel 1.

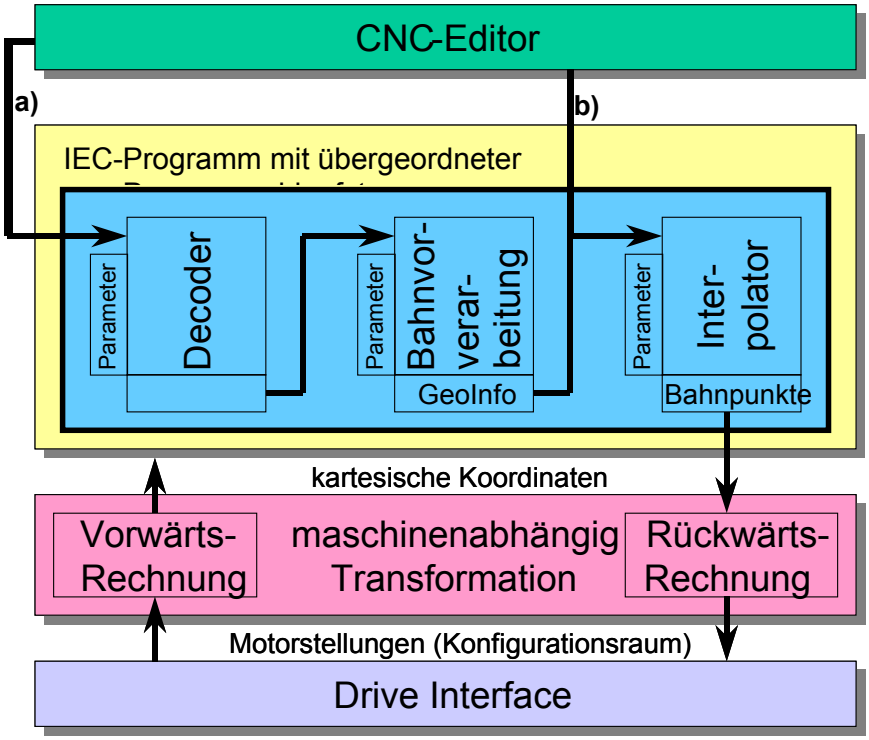
Grundsätzlich können bis zu 9-dimensionale Bewegungen realisiert werden, wobei nur zwei Dimensionen anders als linear interpoliert werden. D.h. in zwei Dimensionen können Geraden, Kreise, Kreisbögen, Parabeln und Splines programmiert werden; die anderen Richtungen werden lediglich linear interpoliert.

Für jede erzeugte Bahn wird von CoDeSys automatisch eine globale Datenstruktur (**CNC Data**) angelegt, die im IEC-Programm zur Verfügung steht. Dies geschieht auf verschiedene Arten:

a) Das CNC-Programm wird als **Array aus G-Code-Worten** abgelegt und wird zur Laufzeit des SPS-Programms mit Hilfe eines Decoder-Bausteins entschlüsselt, so dass für die einzelnen Bahnobjekte GEOINFO-Strukturobjekte zur Verfügung stehen. Diese können dann mit Hilfe von Bahnvorverarbeitungsmodulen (z.B. Werkzeugradiuskorrektur) bearbeitet werden, anschließend interpoliert, transformiert und dem Drive Interface wieder zur Kommunikation an die Hardware übergeben werden. Sehen Sie hierzu die Bausteine der SM_CNC.lib, Kapitel 6.

b) Das CNC-Programm wird **als Liste (OUTQUEUE-Struktur) von GEOINFO-Strukturobjekten** in eine Datenstruktur geschrieben und kann so direkt in den Interpolator eingegeben werden. Im Vergleich zu a) spart man dadurch den Aufruf des Decoder- und der Bahnvorverarbeitungs-Funktionsbaustein. Dafür hat man auch nicht die Möglichkeit, das Programm zur Laufzeit zu veändern.

c) Das CNC-Programm wird in der Form a) bzw. b) in das **Dateisystem** der Steuerung geschrieben und zur Laufzeit Schritt für Schritt ausgelesen und umgesetzt. Diese Methode eignet sich vor allem für große Programme, die nicht komplett im Speicher gehalten werden können.



3.2 Unterstützte und erweiterte Elemente der CNC-Sprache DIN66025

Um dem Programmierer eine einfache Möglichkeit zu bieten, eine geometrische Bahn zu erstellen, unterstützt SoftMotion Teile der CNC-Sprache DIN66025. Da das gesamte SoftMotion-Konzept in die viel mächtigere Sprache IEC61131 eingebettet ist, **werden nur die Teile der DIN66025 unterstützt, die zur Erstellung einer Bahn dienen**.

Vorgeschriebener Aufbau eines CNC-Programms:

```
%
N<zahl> G<zahl> ...
...
N<zahl> G<zahl> ...
```

Beispiel:

```
%1 example
N10 G01 X100 Y100 E100 F100 E-200
N20 G01 Z40 F20
N30 G03 X-100 R200 F100
...
```

Ein SoftMotion-CNC-Programm muss mit einem Prozent-Zeichen **%** beginnen. In derselben Zeile kann – getrennt durch Leerzeichen oder TAB – ein **Programmname** stehen. Das eigentliche CNC-Programm setzt sich aus mehreren *Sätzen* zusammen.

Jeder **Satz** (Zeile) besteht aus beliebig vielen **Worten**.

Ein Wort besteht aus einem Buchstaben (**Wortkennung**) und einer nachfolgenden Zahl (z.B. G01; siehe auch untenstehende Liste). Groß- und Kleinschreibung sind ebenso belanglos wie führende Nullen (G01 = g1).

Das erste Wort jedes Satzes bildet die **Satz-Nummer** (N<zahl>), z.B. "N01", die sich aus Satz-Buchstabe und Satz-Zahl zusammensetzt. Die Worte eines Satzes sind durch Leerzeichen oder TABs getrennt. Die Satz-Nummer hat zur Zeit keinerlei Bedeutung, wird aber aus Konformitätsgründen erwartet. Die restlichen Worte jedes Satzes werden von rechts nach links abgearbeitet. Dabei bewirken alle Worte außer den **Fahrbefehlen** (G<zahl>, z.B. "G02"; siehe auch untenstehende Liste), dass die Wort-Zahl in eine Variable gemäß dem Wort-Buchstaben geschrieben wird, auf welche schließlich der Fahrbefehl zugreift.

Jeder Satz darf nur einen Fahrbefehl enthalten, der direkt rechts neben der Satznummer stehen muss. Wird der Fahrbefehl in einem Satz nicht eingetragen, so wird automatisch der im letzten Satz verwendete Fahrbefehl ergänzt.

Jeder Fahrbefehl kann als Wegobjekt (Gerade, Kreisbogen,…) verstanden werden. Die Geschwindigkeit, mit welcher die Bahnobjekte interpoliert werden, richtet sich grundsätzlich nach der eingestellten Sollgeschwindigkeit, -beschleunigung und –bremsung. Der Interpolator hat sicherzustellen, dass diese Grenzwerte nicht überschritten werden. Die Geschwindigkeit während des Übergangs zweier benachbarter Objekte wird nach folgenden Regeln bestimmt:

- Eines der beiden Objekte ist eine Positionierung (G0): Übergangsgeschwindigkeit = 0
- Der Winkel zwischen den Tangenten der beiden Objekte am Übergang sind größer als die Winkeltoleranz: Übergangsgeschwindigkeit = 0
- Ansonsten: Die Übergangsgeschwindigkeit ist die kleinere Sollgeschwindigkeit der beiden Wegbjekte

Grundsätzlich bewirkt ein Fahrbefehl, dass von der Zielposition des letzten Fahrbefehls ausgehend zur in diesem Fahrbefehl spezifizierten Zielposition interpoliert wird. Der erste Fahrbefehl startet an der (im Decoder oder CNC-Editor) spezifizierten Start-Position. Wurde diese nicht festgelegt, beginnt er bei X=0, Y=0, Z=0. Zusätzlich besteht die Möglichkeit, mittels G92 die Position im CNC-Programm zu setzen. Dies ist sowohl am Anfang des CNC-Programms möglich (dort bewirkt es das Setzen der Startposition) als auch in der Mitte des Programms, wo es einen Sprung der Soll-Position auf die mit G92 definierte Position zur Folge hat. Bei mehreren aufeinander folgenden G92-Befehlen überwiegt der letzte; die vorherigen werden also übersprungen. Will man aber sicherstellen, dass auch die vorigen G92-Positionen (für einen Zyklus lang) ausgegeben werden, fügt man dazwischen den Befehl G1 mit identischen Koordinaten ein. Dieses Mittel setzt man ein, wenn die Bahn zwischen diesen Punkten nicht von Interesse ist, sondern die Sollposition auf schnellstmöglichem Weg dorthin gelangen soll. Die Bausteine SMC_ControlAxisByPos detektieren dann einen Sprung in den

Sollwerten, halten den Interpolator an und interpolieren jede Achse einzeln auf schnellstmöglichem Weg dorthin. Beispiel:

```
G92 X100 Y100 (Sollposition auf 100/100 setzen)
G1 X100 Y100 (einmalige Ausgabe der Position sicherstellen)
G92 X50 Y100 (Sollposition auf 50/100 setzen)
```

Ein mit dem Zeichen ‚/' beginnender Satz wird bei eingeschalteter **Satzunterdrückung** übersprungen.

Zeichen, die innerhalb zweier runder Klammern stehen, werden als **Kommentare** aufgefasst und haben keinen Einfluss auf die programmierte Bahn. Verschachtelte Kommentare werden nicht unterstützt.

Alle Zahlenwerte außer die des Fahrbefehls (G<zahl>) und der Schaltpunkt-Nummer (H<zahl>) können Fließkommawerte sein.

Wortkennungen :

D	Werkzeugradius (für Korrektur G40-42 bzw. Eckverrundung G50-51) bzw. Variablenwert (G36/G37)
E	max. Beschleunigung (>0) / Verzögerung (<0) [Wegeinheiten/s^2]
F	Geschwindigkeit, mit der gefahren werden soll [Wegeinheiten/sec]
G	Fahrbefehl (s.u.)
H	Schaltpunkt einschalten (>0) / ausschalten (<0)
I	X-Koordinate vom Kreis-/Ellipsenmittelpunkt (G02/G03/G08/G09) oder X-Koordinate vom Parabel-Tangentenschnittpunkt
J	Y-Koordinate vom Kreis-/Ellipsenmittelpunkt (G02/G03/G08/G09) oder Y-Koordinate vom Parabel-Tangentenschnittpunkt
K	Richtung der Ellipsenhauptachse im mathematischen Sinn (0° O, 90° N,...) bzw. Sprungbedingung (G20) bzw. dT1-Parameterwert (M-Funktion)
L	absolute Schaltpunktposition gemessen vom Wegobjektanfang (>0) / -ende (<0) bzw. Sprungziel (G20) bzw. dT2-Parameterwert (M-Funktion)
M	Zusatzfunktion
O	relative Schaltpunktposition [0..1] bzw. zu verändernde Variable (G36/G37) oder M-Parameter-Datenstruktur (M)
P	Zielwert der Zusatzachse P
Q	Zielwert der Zusatzachse Q
R	Kreisradius (G02/G03) (alternativ zu I,J) oder Längenverhältnis Nebenachse/Hauptachse (G08/G09) [0..1]
S	S-Profil für lineare Achsen ein-(>0)/ausschalten(<0) 3: Z-Achse, 7: P-Achse, 8: Q-Achse, 9: U-Achse, 10: V-Achse, 11: W-Achse
U	Zielwert der Zusatzachse U
V	Zielwert der Zusatzachse V
W	Zielwert der Zusatzachse W
X	X-Koordinate vom Zielpunkt
Y	Y-Koordinate vom Zielpunkt
Z	Zielwert der Zusatzachse Z

Fahrbefehle :

G00	direkte Bewegung ohne Werkzeugeingriff, Positionierung
G01	lineare (gerade) Bewegung mit Werkzeugeingriff
G02	Kreis(-stück) im Uhrzeigersinn
G03	Kreis(-stück) gegen den Uhrzeigersinn
G05	Punkt einer Kardinalspline
G06	Parabel
G08	Ellipse(-nstück) im Uhrzeigersinn
G09	Ellipse(-nstück) gegen den Uhrzeigersinn
G20	Bedingter Sprung (nach L wenn K<>0)
G36	Wert (D) auf Variable (O) schreiben
G37	Variable (O) um Wert(D) inkrementieren.
G40	Ende der Werkzeugradiuskorrektur
G41	Beginn der Werkzeugradiuskorrektur links der Fahrrichtung
G42	Beginn der Werkzeugradiuskorrektur rechts der Fahrrichtung
G50	Ende der Eckverrundung/-schelifung
G51	Beginn der Eckverschleifung
G52	Beginn der Eckverrundung
G60	Ende der Schleifenvermeidung
G61	Beginn der Schleifenvermeidung
G90	nachfolgende Koordinatenangaben (für X/Y/Z/P-W) absolut (Standard)
G91	nachfolgende Koordinatenangaben (für X/Y/Z/P-W) relativ
G92	Setzen der Position ohne Bewegung
G98	nachfolgende Koordinatenangaben von I/J absolut
G99	nachfolgende Koordinatenangaben von I/J relativ zum Startpunkt (Standard)

Man beachte, dass die Bibliothek "SM_CNC.lib" für ein fehlerfreies Übersetzen des Projekts eingebunden sein muss.

3.2.1 Schaltpunkte, H-Funktion

Die **Schaltpunkt**-Funktionalität (oder **H-Funktion**) gibt dem Programmierer die Möglichkeit, binäre wegabhängige Schalter zu bedienen. Grundsätzlich muss immer zuerst die Nummer des Schaltpunkts spezifiziert werden (H<nummer>), dann muss die Position des Schaltpunkts im Objekt entweder absolut über das Wort L<position> oder relativ über das Wort O<position> festgelegt werden. In folgendem Beispiel wird der Schaltpunkt2 an der Position X=40/Y=25 (nach dem ersten Viertel des Objekts) ausgeschaltet.

```
N90 G1 X20 Y20
N100 G1 X100 Y40 H-2 O0.25
```

Hinweise: 1. Pro Wegobjekt kann nur eine begrenzte Anzahl Schaltpunkt-Schaltaktionen (MAX_SWITCHES) durchgeführt werden.

2. Eine Schaltpunktposition kann nur im CNC-Texteditor eingefügt werden! Sie erscheint im grafischen Editor als grüner Punkt auf der Kurve.

3.2.2 Zusatz-Funktion, M-Funktion

Mittels der **Zusatz-Funktionen** oder **M-Funktionen** kann ein binärer Ausgang gesetzt werden, der eine anderweitige Aktion startet. Im Unterschied zu den Schaltpunkten wird dabei so lange an der aktuellen Position verharrt, bis die M-Funktion durch Setzen eines Einganges bestätigt wird. Dies wird häufig dazu verwendet, wenn die weitere Abarbeitung des Programms von anderen Prozessen abhängt. Folgende Zeile würde die M-Funktion 10 starten und so lange warten, bis diese bestätigt wird:

```
N90 M10
```

Es besteht die Möglichkeit, über K und L zwei Parameter einzugeben, welche in das entsprechende SMC_GeoInfo-Objekt kopiert werden. Die Applikation hat dann zur Laufzeit, wenn der Interpolator auf einer M-Funktion steht und wartet, die Möglichkeit über den Baustein **SMC_GetMParameters** auf diese Werte zuzugreifen.

Der im folgenden beschriebene Mechanismus kann nur verwendet werden, wenn die Bahn online verarbeitet wird (mit SMC_NCDecoder):

Um zusätzliche Parameter zu übergeben, kann man die in der Bibliothek SM_CNC.lib angelegte globale Datenstruktur gSMC_MParameters (Typ: SMC_M_PARAMETERS) verwenden. Diese entspricht den Parametern dP1..dP8.

Will man anstelle der globale Datenstruktur gSMC_MParameters eine andere, selbst angelegt verwenden, so kann man diese über Ovar übergeben. Beispielsweise enthält g_myMParams (Typ: SMC_M_PARAMETERS) die zu übergebenden Parameter:

```
N150 M13 O$g_myMParams$
```

3.2.3 Verwendung von Variablen

Zusätzlichhat man die Möglichkeit, anstelle fester Zahlenwerte **Variablen** zu verwenden. Diese müssen zwischen zwei $-Zeichen stehen (z.B. R$g_fVar$). Man beachte in diesem Kontext, dass das Verwenden von Variablen nur dann funktioniert, wenn das Programm als Programm-Variable übersetzt und online vom Decoder-Baustein verarbeitet wird. Die Variablen werden zu dem Zeitpunkt ersetzt, an dem der Decoder die entsprechende Zeile verarbeitet.
Wird das CNC-Programm stattdessen als OutQueue übersetzt wird, funktioniert der Variablen-Mechanismus nicht, da die Bahn dann offline erzeugt und als unveränderliche Datenstruktur der Applikation übergeben wird. In diesem Fall – wie auch zur offline-Anzeige – ersetzt der Editor Variablen mit deren Initialwerten.
Die Verwendung von Variablen in online eingelesenen G-Code-Programmen bedarf zusätzlicher Vorbereitung (siehe SMC_VARLIST).

3.2.4 Kreisinterpolation

Zur Beschreibung eines Kreisbogens müssen die Zielkoordinaten (X/Y) festegelegt werden. Zur Festlegung der Krümmung hat man zweierlei Möglichkeiten: man gibt entweder der Radius (R) des Kreisbogens oder dessen Mittelpunktskoordinaten (I/J) ein.

Bei der Variante mit dem Radius beachte man, dass nur Bögen, die weniger als 180° (Halbkreis) umspannen, möglich sind, da diese Methode (außer im Fall des Halbkreises) immer zwei mögliche Lösungen beschreibt, eine kleiner, eine größer als ein Halbkreis. Das System wählt immer die, die kleiner als ein Halbkreis ist.

Beispiel für einen Halbkreis:

```
N10 G1 X100 Y100
N20 G2 X200 Y100 R50
```

Will man einen Kreisbogen mit mehr als 180° Öffnungswinkel, so muss man ihn über (I/J) festlegen. Diese Methode ist eindeutig, ausgenommen in dem Fall, in dem Start- und Endpunkt des Kreises identisch sind. Dort könnte entweder ein Null- oder ein Vollkreis entstehen. Das System fügt in diesem Fall einen Vollkreis ein.

Man beachte auch, dass mit der Methode (I/J) entscheidend ist, ob die Koordinaten I/J (siehe G98/G99) relativ oder absolut angegeben werden. Werden I und J nicht korrekt angegeben (Abstand zwischen Mittelpunkt und Start- bzw. Endpunkt muss identisch sein), dann ist kein Kreisbogen möglich und das System ersetzt diesen durch eine Gerade.

Beispiel für denselben Halbkreis über relativen Mittelpunkt spezifiziert:

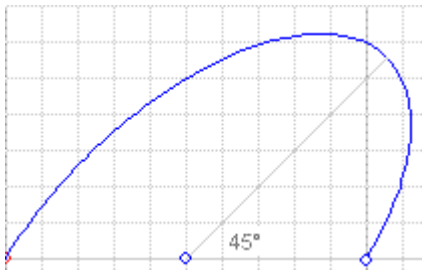
```
N10 G1 X100 Y100
N15 G99
N20 G2 X200 Y100 I50 J0
```

Beispiel für denselben Halbkreis über absoluten Mittelpunkt spezifiziert:

```
N10 G1 X100 Y100
N15 G98
N20 G2 X200 Y100 I150 J100
```

3.2.5 Ellipseninterpolation

Zur Beschreibung eines Ellipsenbogens müssen die Zielkoordinaten (X/Y), der Ellipsenmittelpunkt (I/J), die Ellipsenhauptachsenrichtung (K) sowie das Längenverhältnis zwischen Haupt- und Nebenachse (R) spezifiziert werden.



Beispiel:

```
N10 G0 X100 Y100
N15 G98
N20 G8 X200 Y100 I150 J100 K45 R0.5
```

3.2.6 Splineinterpolation

Zur Beschreibung von Splines genügt es, die Zielkoordinaten des nächsten Splineabschnittes einzugeben. Der Splineabschnitt wird vom System automatisch so berechnet, dass der Endvektor des vorigen Elements und der Startvektor des Splines, sowie der Endvektor des Splines und der Startvektor des folgenden Elements übereinstimmen, sodass kein Knick in der Bahn entsteht.

3.2.7 Bedingte Sprünge

Über den Befehl G20 kann ein bedingter Sprung realisiert werden. Dazu wird die Zeilennummer des Sprungziels (L) und die Bedingung für den Sprung (K) eingegeben. Wird keine Sprungbedingung angegeben, so wird automatisch die implizite Decoder-Variable verwendet. Der Sprung wird immer dann ausgeführt, wenn die Bedingung ungleich null ist.

3.2.8 Variablenwerte verändern

Mit den Befehlen G36 und G37 steht dem Anwender die Möglichkeit zur Verfügung, Variablen zu schreiben oder zu verändern. Die Variable, die verändert werden soll, gibt der Anwender über Ovar an. Mit D gibt man den Wert vor, der auf die Variable geschrieben (G36) bzw. addiert (G37) werden soll.

Folgendes Beispiel setzt die globale Variable g_i auf 5:

```
N1000 G36 O$g_i$ D5
```

Folgendes Beispiel bewirkt, dass die Zeilen 1010 und 1020 fünf Mal verfahren werden:

```
N1000 G36 O$g_i$ D5
N1010 G1 X100 F100 E100 E-100
N1020 G1 X0
N1030 G37 O$g_i$ D-1
N1040 G20 L1010 K$g_i$
```

Man beachte, dass dieser Mechanismus nur funktioniert, wenn die Bahn online verarbeitet wird, weil nur dann Variablen verwendet werden können! Im CNC-Editor funktioniert dieser Mechanismus also nicht. Stattdessen kann folgendermassen vorgegangen werden:

Wird bei O keine Variable spezifiziert, so wird eine **implizite Decoder-Variable** (Typ: INT) verwendet. Dieser Mechanismus funktioniert auch offline im Editor. Man beachte aber, dass damit nur eine Variable zur Verfügung steht und man so keine verschachtelten Sprünge und Schleifen programmieren kann.

Beispiel:

```
N1000 G36 D5
N1010 G1 X100 F100 E100 E-100
N1020 G1 X0
N1030 G37 D-1
N1040 G20 L1010
```

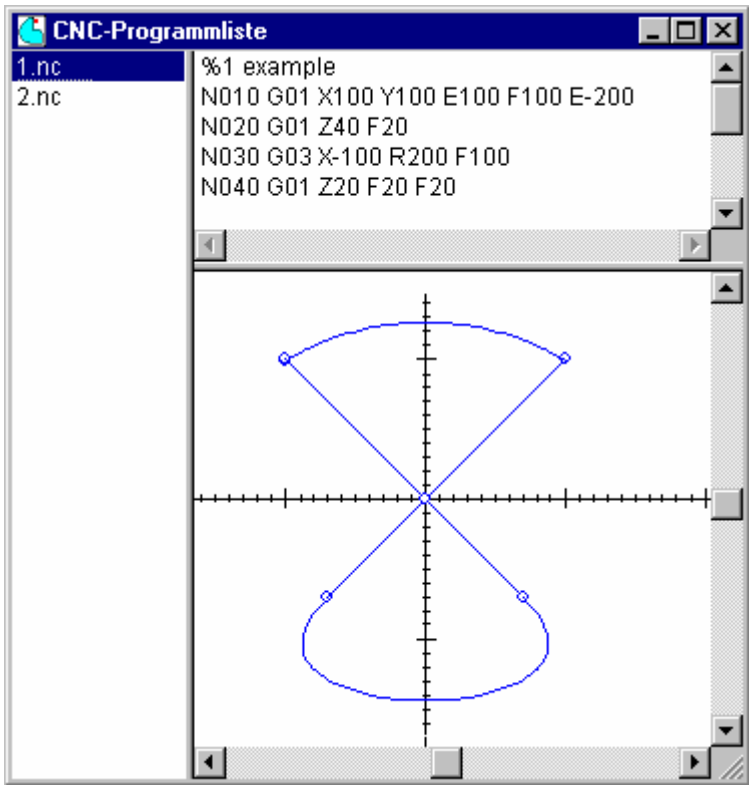
3.3 Start des Editors, Einfügen und Verwalten von CNC-Programmen

Der CNC-Editor wird unter 'Ressourcen' im Object-Organizer gestartet. Es öffnet sich ein dreigeteiltes Fenster mit dem Titel '**CNC-Programmliste**'. In der linken Spalte steht eine Liste mit den Bezeichnungen der erstellten Programme. Die rechte obere Fensterausschnitt dient als Texteditor, in dem das CNC-Programm nach DIN66025 eingegeben werden kann (siehe Kapitel 3.2). Rechts unten wird dieses graphisch dargestellt und kann dann sowohl im Text- als auch im Grafikbereich verändert werden, wobei es jeweils im anderen Editor automatisch aktualisiert wird.

In der Menüleiste wird das Menü 'Einfügen' durch das Menü '**CNC-Programm**' ersetzt, solange der CNC-Editor aktiv ist.

Neues CNC-Programm

Wählen Sie den Befehl **'Neues CNC-Programm'** im Menü 'CNC-Programm' oder im Kontextmenü, wenn der Fokus in der CNC-Programmlistenspalte steht. Es erscheint der Dialog 'Programmname', in dem die neue Programmbezeichnung vergeben wird. Standardmäßig werden die CNC-Programme mit der Bezeichnung "_CNC<n>" versehen, n ist eine mit 1 beginnende fortlaufende Nummerierung. Sie können den im Dialog vorgegebenen Programmnamen editieren, die Eingabe eines bereits vergebenen wird jedoch nicht akzeptiert. Nach Schließen des Dialogs mit OK erscheint das neue Programm markiert in der Liste. Im Texteditor wird die erste Programmzeile angezeigt: bei neu angelegtem Programm "% comment", der graphische Editor ist in diesem Fall noch leer.



Das aktuell markierte Programm kann nun sowohl im Texteditor (siehe Kapitel 3.4) wie im graphischen Editor (siehe Kapitel 3.5) bearbeitet werden.

CNC-Programm umbenennen

Markieren Sie das Programm in der CNC-Programmliste und wählen Sie den Befehl '**CNC-Programm umbenennen**' im Menü 'CNC-Programm' oder im Kontextmenü. Der Dialog 'Programmname' öffnet, wo der Name editiert werden kann.

Löschen

Markieren Sie das Programm in der CNC-Programmliste und wählen Sie den Befehl '**CNC-Programm löschen**' im Menü 'CNC-Programm' oder im Kontextmenü. Das Programm wird von der Liste gelöscht, der Fokus geht auf das nächste der Liste über.

Info

Markieren Sie ein Programm in der CNC-Programmliste und wählen Sie den Befehl '**Info**' im Menü 'CNC-Programm' oder im Kontextmenü. Es öffnet sich ein Dialog, der Informationen über das gewählte CNC-Programm enthält.

Queue-Größe festlegen

Hier können Sie die Größe des OutQueue-Datenpuffers definieren. Dies ist vor allem dann interessant, wenn Sie dem NC-Funktionsbausteinen im IEC-Programm nur einen begrenzten Speicher zur Verfügung stellen können, sodass nicht alle GeoInfo-Objekte darin Platz finden und die Ringpuffer-Funktionalität benutzt wird. Dadurch können Spezialeffekte auftreten (z.B. Abbremsen an knickfreier Stelle), die Sie mit dieser Funktion simulieren und nachvollziehen können.

Mögliche Werte: 5000 – 100 000 Bytes. Mit OK werden die Werte übernommen.

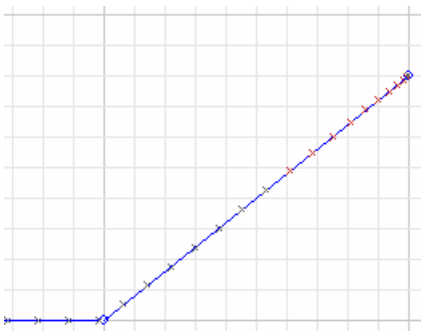
CNC-Programm Startposition festlegen

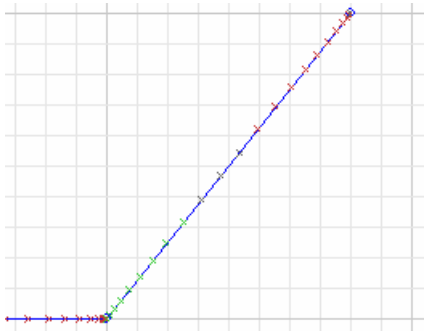
Im Dialog **Startpunkt** können zu Simulationszwecken vom Standardwert 0 abweichende Startposition für die Bahn eingegeben werden. Für folgenden Achsen sind Eingaben möglich: X, Y, Z, P, Q, U, V, W.

CNC-Programm Winkeltoleranz für Stop festlegen

Im Dialog **Winkeltoleranz für Stops** kann die Empfindlichkeit für einen Bahnknick eingestellt werden. Die Eingabe legt den Grenzwinkel für die Tangenten zweier Wegobjekte fest, ab der gestoppt wird:

Beispiel: Winkeltoleranz 45°:





Programm verschieben

Wählen Sie diesen Menüpunkt, und geben Sie im Dialog **Bewegungsvektor** einen Verschiebungsvektor ein. Das aktuelle CNC-Programm wird um diesen Vektor verschoben. Für folgenden Achsen sind Eingaben möglich: X, Y, Z, P, Q, U, V, W.

Programm drehen

Um das aktuelle Programm zu drehen, wählen Sie diesen Menüpunkt, und geben Sie im Dialog **Rotationswinkel** den Rotationswinkel ein. Das NC-Programm wird um den angegebenen Winkel gegen den Uhrzeigersinn um den Nullpunkt gedreht.

Programm strecken

Geben Sie im Dialog **Streckungsfaktor** den Streckungsfaktor ein. Das NC-Programm wird um den angegebenen Faktor gestreckt.

Richtung umdrehen

Wenn Sie diesen Menüpunkt anwählen, ändert sich die Bahn dahingehend, dass sie in der entgegengesetzten Richtung abgefahren wird. Die Hilfsmarken bleiben dabei an der selben Stelle

Objekt teilen

Haben Sie ein Wegobjekt angewählt (rot gezeichnet), können Sie eine Teilungs-Position (0 Anfang, 1 Ende) im Objekt angeben. Dieses Objekt wird dann an der spezifizierten Stelle in zwei Wegobjekte geteilt.

Beispiel: Fahrbefehl N10 wird mit Angabe einer Teilposition von 0.5 geteilt:

```
...
N0 G01 X123.000000 Y73.550000
N10 G01 X40.0 Y50.0
...
```

Ergebnis: neue Teilposition bei X=20

```
...
N0 G01 X20.000000 Y40.000000
N10 G01 X40.000000 Y50.000000
N20 G01 X123.000000 Y73.550000
...
```

CNC-Programm aus Datei lesen

Mit diesem Menüpunkt können Sie ein in einer ASCII-Datei abgelegtes NC-Programm laden. Im Standarddialog zum Öffnen einer Datei können Sie die gewünschte TXT-Datei auswählen. Im darauf folgenden Dialog geben Sie einen Programmnamen an.

CNC-Programm in Datei schreiben

Dieser Menüpunkt gibt Ihnen die Möglichkeit, das aktuelle CNC-Programm in eine beliebige ASCII-Datei zu schreiben. Existiert die Datei bereits, fragt CoDeSys nach einer Bestätigung.

DXF-Datei importieren

Mit diesem Menüpunkt können Sie eine DXF-Datei importieren. Im Standarddialog zum Öffnen einer Datei können Sie die gewünschte DXF-Datei auswählen. Daraufhin erscheint der Dialog DXF-Import Optionen, in dem Sie eine der folgenden Optionen einstellen:

- Ein NC-Programm für gesamte DXF-Datei: alle in der DXF-Datei enthaltenen Bahnen werden in ein CNC-Programm geschrieben.
- Ein NC-Programm pro DXF-Layer: je DXF-Layer wird ein eigenes CNC-Programm erzeugt.
- Ein NC-Programm pro zusammenhängendem Abschnitt: für jeden zusammenhängenden Bahnabschnitt wird ein eigenes CNC-Programm generiert. Da in einer DXF-Datei die einzelnen Wegobjekte ohne Reihenfolge gespeichert sind, versucht CoDeSys, die Objekte miteinander zu verbinden, um daraus eine zusammenhängende Bahn zu erzeugen.

OutQueue in Datei schreiben

Mit dieser Funktion können Sie das gesamte CNC-Programm in eine OutQueue, also eine Liste von GEOINFO-Strukturobjekten, verwandeln und in eine Datei speichern, die Sie ins Dateisystem der Steuerung laden und dort zur Laufzeit einlesen können (c)). Gehen Sie so vor, wenn es sich um große CNC-Programme handelt, die nicht in den globalen Datenspeicher der Steuerung passen, oder ohne das Verändern des CoDeSys-Projekts getauscht werden müssen.

3.4 CNC Texteditor

Der Texteditor befindet sich im rechten oberen Fensterteil der CNC-Programmliste. Hier kann ein CNC-Programm nach DIN66025 eingegeben und verändert werden. Sehen Sie hierzu Kapitel 3.2 , Unterstützte und erweiterte Elemente der CNC-Sprache DIN66025). Das Programm wird entsprechend im graphischen Editor (siehe Kapitel 3.5) dargestellt. Wird es dort modifiziert, wird es auch im Textteil automatisch aktualisiert.

Zur Bearbeitung stehen die Befehle der Menüs CNC-Programm und Extras zur Verfügung, sehen Sie hierzu 3.5, Graphischer Editor.

(Sehen Sie hierzu auch die Programmierbeispiele in Kapitel 1)

Durch Drücken von F2 öffnet die Eingabehilfe und gibt die Möglichkeit, **globale Variablen** in das NC-Programm einzugeben. Für die Darstellung im grafischen Editor wird – wenn vorhanden - der Initialwert der Variable verwendet.

Hinweis:	Man beachte, dass Verweise auf globale Variablen im Decoder-Baustein als Reaktion auf eine steigende Flanke im Execute-Eingang ausgewertet werden.

Wählt man die Option, dass CoDeSys beim Übersetzen eine fertige OutQueue-Struktur erstellt (Variante b)), so wird der Verweis auf die globale Variable natürlich bereits beim Übersetzen mit dem Initialwert ersetzt, und macht so die Verwendung globaler Variablen in diesem Zusammenhang unsinnig.

3.5 CNC Graphischer Editor

Der graphische Editor befindet sich im rechten unteren Fensterteil der CNC-Programmliste. Er visualisiert zum einen das im Texteditor erstellte CNC-Programm, zum anderen können mit Hilfe der Maus Änderungen an diesem Programm vorgenommen werden, die unmittelbar automatisch in den Programmtext im Texteditor nachgetragen werden.

Darstellung:

Es wird ein **Koordinatensystem** gezeichnet, in dem in gewissen Abständen ein Markierungsstrich gezeichnet wird. Ergänzend wird ein hellgraues **Gitternetz** daruntergelegt, welches über den Menüpunkt ‚Raster anzeigen' unter ‚Extras' ein- und ausblenden lässt.

Durch gedrückt Halten der linken Maustaste kann die Darstellung des NC-Programms beliebig verschoben werden. Mit dem Mausrad und der <Strg>-Taste wird der Zoomfaktor verändert.

Positionierungen (G00) werden grün dargestellt, alle übrigen **Elemente** in blau. Das **aktuelle Objekt**, auf dessen Code-Zeile im Texteditor der Cursor steht, wird rot gezeichnet.

Für **Spline-Abschnitte** (G05) wird die konvexe Hülle der kubischen Polynome in hellgrau gezeichnet.

Hinweis:	Eine Schaltpunktposition kann nur in CNC-Texteditor eingefügt werden ! (siehe Kapitel 3.2). Sie erscheint im grafischen Editor als grüner Punkt auf der Kurve.

(Sehen Sie hierzu auch die Programmierbeispiele Kapitel 1)

3.6 Kommandos und Optionen im CNC-Editor

Das **Menü 'Extras'** hält Befehle und Einstellmöglichkeiten für das Arbeiten und die Anzeige im grafischen Editor bereit. (Es kann natürlich auch durch Eingeben der entsprechenden Kommandos im Texteditor programmiert werden.) In der Werkzeugleiste sind die entsprechenden Schaltflächen verfügbar. Eine aktivierte Option ist im Extras Menü mit einem Haken versehen und die Schaltfläche erscheint gedrückt.

Zwischen den folgenden fünf Bearbeitungsmodi kann gewechselt werden:

 Selektiermodus

 Linien-Einfügemodus

 Kreis-Rechts-Einfügemodus

 Kreis-Links-Einfügemodus

 Spline-Einfügemodus

Zur Veränderung der Darstellung und der Wahl einer Kurvenkorrektur stehen die folgenden Befehle und Optionen bereit:

 Ausgabegröße anpassen

Programm neu nummerieren

Raster anzeigen

Splines/Ellipsen durch Geraden ersetzen

 Werkzeugradiuskorrektur

 Eckverrundung

 Eckverschleifung

 Schleifenvermeidung

 Satzunterdrückung

 Interpolation zeigen

Epsilon-Werte für Null ändern

Hinweis:	Es ist durchaus möglich, mehrere der Optionen Werkzeugradiuskorrektur, Eckverrundung, Eckverschleifung und Schleifenvermeidung zu wählen. Dadurch simuliert man die Wirkung hintereinander geschalteter Bahnvorverarbeitungsbausteine. Nur die Bahnvorverarbeitungen ‚Werkzeugradiuskorrektur' und ‚Eckverrundung' bzw. ‚Eckverschleifung' können nicht gleichzeitig gewählt werden.

'Extras' 'Selektiermodus'

 Wenn diese Option aktiviert ist, wird durch Klicken auf ein graphisches Element im CNC-Editor dieses selektiert (rote Farbe) und im Texteditor die entsprechende Zeile markiert. Durch Klicken auf den Endpunkt eines Elements kann dieser beliebig verschoben werden.

'Extras' 'Linien-Einfügemodus'

Wenn diese Option aktiviert ist, wird bewirkt ein Mausklick im Editorfeld das Einfügen eines G01-Elements in Form einer Linie. Das neue Element wird hinter das aktuell selektierte (rot) platziert. Die Mausposition legt den Zielpunkt der Linie fest.

'Extras' 'Kreis-Rechts-Einfügemodus'

Wenn diese Option aktiviert ist, bewirkt ein Mausklick im Editorfeld das Einfügen eines G02-Elements in Form eines Kreises für eine Bewegung im Uhrzeigersinn. Das neue Element wird hinter das selektierte (rot) platziert. Die Mausposition legt den Zielpunkt fest. Der Kreis-Radius wird standardmäßig auf 100 gesetzt und muss ggf. im Texteditor verändert werden.

'Extras' 'Kreis-Links-Einfügemodus'

Wenn diese Option aktiviert ist, bewirkt ein Mausklick im Editorfeld das Einfügen eines G03-Elements in Form eines Kreises für eine Bewegung gegen den Uhrzeigersinn. Das neue Element wird hinter das selektierte (rot) platziert. Die Mausposition legt den Zielpunkt fest. Der Kreis-Radius wird standardmäßig auf 100 gesetzt und muss ggf. im Texteditor verändert werden.

'Extras' 'Spline-Einfügemodus'

Wenn diese Option aktiviert ist, bewirkt ein Mausklick im Editorfeld das Einfügen eines Spline-Punktes. Dieser wird hinter das markierte, rote Element gestellt. Die Mausposition legt den Zielpunkt fest.

'Extras' 'Ausgabegröße anpassen'

 Betätigt man diesen Menüpunkt, wird der sichtbare Bildschirmbereich so gelegt, dass das gesamte NC-Programm gesehen werden kann.

'Extras' 'Programm neu nummerieren'

 Mit Hilfe dieses Befehls können die Zeilennummern (N<nummer>) automatisch fortlaufend auf Zehnerschritte angepasst werden.

'Extras' 'Splines/Ellipsen durch Geraden ersetzen'

Splines und Ellipsen benötigen bei der Interpolation vergleichsweise viel Rechenzeit. Um dies zu vermeiden, können alle Splines und Ellipsen eines NC-Programms mit diesem Menüpunkt durch eine Anzahl von Geraden angenähert werden. So können zum Design der Bahn Splines und Ellipsen verwendet, ohne dass diese zur Laufzeit gerechnet werden.

Wird der Befehl gewählt, stehen über einen Dialog zwei Optionen zur Konvertierung zur Verfügung:

a) **längenabhängig**, d.h. es wird pro x Längeneinheiten (die Anzahl x kann im Dialog eingegeben werden) des Splines/der Ellipse eine Gerade erstellt, oder

b) **winkelabhängig**, d.h. das Ursprungsobjekt wird so geteilt, dass die neuen Geraden einen Winkel kleiner x (im Dialog in Winkelgraden einzugeben) einschließen.

Tipp: Mit den Standardeinstellungen wird bei der Interpolation der Geraden – im Gegensatz zu Spline/Ellipse - nach jedem neuen Geradenstück auf die Geschwindigkeit 0 gebremst. Sie vermeiden dies, indem Sie die Winkeltoleranz entsprechend erhöhen.

'Extras' 'Werkzeugradiuskorrektur'

 Wenn diese Option aktiviert ist und im CNC-Programm Start (G41/G42) und Ende (G40) der Werkzeugradiuskorrektur und ein Werkzeugradius (D<radius>) programmiert sind, wird die entsprechend versetzte Bahn dargestellt. Dieser Menüpunkt entspricht dem Baustein SMC_ToolCorr der SoftMotion-Bibliothek. Die ursprüngliche Bahn wird als Referenz dazu hellgrau gezeichnet. Positionierungen in der korrigierten Bahn sind Blind-Positionierungen und werden dunkelgelb gezeichnet.

'Extras' 'Eckverrundung'

 Wenn diese Option aktiviert ist , wird die eckverrundete Bahn dargestellt und zeigt den Effekt an, den der Baustein SMC_RoundPath der SoftMotion-Bibliothek auf die programmierte Bahn hat. Voraussetzung hierfür ist, dass im CNC-Programm Start (G52), Ende (G50) und Verrundungsradius (D<radius>) gesetzt sind. Die ursprüngliche Bahn wird als Referenz dazu hellgrau gezeichnet.

'Extras' 'Eckverschleifung'

 Wenn diese Option aktiviert ist, wird die programmierte Bahn dargestellt und zeigt den Effekt an, den der Baustein SMC_SmoothPath der SoftMotion-Bibliothek SM_CNC.lib hat. Dieser legt verschleift ein Eck durch ein kubisches Polynom (Splinestück). Voraussetzung hierfür ist, dass im CNC-Programm Start (G51), Ende (G50) und Verrundungsradius (D<radius>) gesetzt sind. Die ursprüngliche Bahn wird als Referenz dazu hellgrau gezeichnet.

'Extras' 'Schleifenvermeidung'

 Wenn diese Option aktiviert ist, wird diejenige Bahn dargestellt, die dadurch entsteht, dass bei einem Schnittpunkt der Bahn mit sich selbst, die Bahn an dieser Stelle abgekürzt wird. Dadurch werden Bahnschleifen vermieden. Dieser Menüpunkt wirkt wie der Baustein SMC_AvoidLoop der SoftMotion-Bibliothek.

Voraussetzung hierfür ist, dass im CNC-Programm Start (G61) und Ende (G60) gesetzt sind. Die ursprüngliche Bahn wird als Referenz dazu hellgrau gezeichnet.

'Extras' 'Satzunterdrückung'

 Wenn diese Option aktiviert ist, werden sämtliche Zeilen des Texteditors, die mit "/" beginnen, ignoriert.

'Extras' 'Interpolation zeigen'

 Wenn diese Option aktiviert ist, werden Interpolationspunkte im 100 ms-Takt eingezeichnet, d.h. die Werkzeug-Position wird alle 100 ms durch ein kleines graues Kreuz markiert. Damit kann in etwa das Geschwindigkeitsverhalten (schnell = weite Abstände, langsam = enge Abstände) eingesehen werden.

'Extras' 'Epsilonwerte für Null ändern'

Die interne Prüfung eines Wertes x auf Null muss aufgrund der Ungenauigkeit einer Fließkommarechnung durch Überprüfung, ob $x < \varepsilon$ ist, ersetzt werden. Die Größe des ε-Wertes muss ggfs. (z.B. bei Verwendung von 32 Bit anstelle von 64 Bit Fließkommawerten, oder beim Import eines CNC-Programm mit begrenzter Genauigkeit) angepasst werden, wozu der Dialog **Null-Toleranzwerte** dient, der über den hier beschriebenen Befehl geöffnet werden kann.

Hinweis: Für die Standardbenutzung sollte die Änderung dieser Werte nicht nötig sein und vermieden werden!

3.7 Automatische Strukturbefüllung im CNC-Editor

Bei der Übersetzung des IEC-Programms wird automatisch ein Verzeichnis globaler Variablen "**CNC Data**" erzeugt. Dort werden die NC-Programme in gleichnamigen Datenstrukturen abgelegt.

Dabei existieren gemäß 3.1 drei mögliche Varianten a-c, die separat für jedes NC-Programm im NC-Editor im Menü ‚CNC-Programm' eingestellt werden:

1) 'beim Übersetzen Programmvariable erzeugen':

Diese Option entspricht Fall a). Das CNC-Programm wird in einer Struktur SMC_CNC_REF gespeichert, die in der Bibliothek SM_CNC.lib enthalten ist. Diese Strukturvariable muss von den Decoder- und Bahnvorverarbeitungsbausteinen verarbeitet werden, und kann nicht direkt in den Interpolator-Baustein eingegeben werden.

Obwohl diese Variante erhöhten online-Rechenbedarf hat, bietet sie die Möglichkeit, Variablen im NC-Programm zu verwenden, bzw. Modulationen des NC-Programms durch das SPS-Programm vorzunehmen.

2) 'beim Übersetzen OutQueue erzeugen':

Diese Option entspricht Fall b). Das CNC-Programm wird in einer Struktur SMC_OUTQUEUE gespeichert, die ebenfalls in der Bibliothek SM_CNC.lib enthalten ist. Diese Strukturvariable kann direkt in den Interpolator-Baustein eingegeben werden.

Obwohl so keine Variablen in der Bahn verwendet werden können, hat diese Methode den Vorteil, dass sie den online-Ressourcenbedarf minimiert.

3) 'nicht übersetzen':

Diese Option entspricht Fall c). Sie haben das Programm als ASCII-File oder OutQueue-File im Dateisystem Ihrer Steuerung abgelegt, und wollen es durch einen der Bausteine aus 10.2 zur Laufzeit einlesen. Deshalb soll es nicht in die IEC-Daten übernommen werden.

4 Der CAM-Editor in CoDeSys

4.1 Überblick

Der SoftMotion Kurvenscheiben-Editor (CAM-Editor) ist in die Programmieroberfläche von CoDeSys integriert. Hier können graphisch und tabellarisch elektronische Kurvenscheiben und Nockenschaltwerke programmiert werden, aus denen CoDeSys beim Übersetzen des Projekts automatisch globale Datenstrukturen (CAM Data) erzeugt. Auf diese Strukturen kann das IEC-Programm zugreifen.

Für die Umsetzung der Kurvenscheibe im IEC-Programm werden die von PLCopen genormten Bausteine verwendet (Bibliothek SM_PLCOpen.lib).

(Sehen Sie auch die Programmierbeispiele in Kapitel 1)

4.2 Definition einer Kurvenscheibe für SoftMotion

Eine Kurvenscheibe beschreibt vereinfacht gesehen die funktionale Abhängigkeit einer Größe (Slave) von einer anderen (Master).

Um diese Abhängigkeit zu beschreiben, wird die **Master-Achse** in verschiedene Intervalle geteilt. Für jedes Intervall [a,b] stellt CoDeSys zwei Möglichkeiten zur Verfügung, eine Funktionsabbildung der Master-Achse auf die **Slave-Achse** nachzubilden:

- **Gerade**: Die Abhängigkeit wird durch eine lineare Abbildung beschrieben. In diesem Intervall ist die erste Ableitung (Geschwindigkeit) konstant gemäß der Geradensteigung, die zweite Ableitung (Beschleunigung) gleich 0.

- **Polynom fünften Grades**: In diesem Intervallabschnitt wird die Abhängigkeit durch ein Polynom fünften Grades beschrieben. Erste und zweite Ableitung werden dadurch zu Polynomen vierten und dritten Grades.

Die Funktionen auf diesen Intervallen müssen so aneinander anschließen, dass an den Übergängen sowohl der Funktionswert, wie auch mindestens dessen erste und zweite Ableitungen stetig sind.

Im CAM-Editor können einzelne Stützpunkte und Geraden gesetzt werden. Dazwischen liegende Intervalle werden vom Editor durch Polynome fünften Grades automatisch verbunden, wobei die Stetigkeits- und Differenzierbarkeitsvoraussetzungen erfüllt werden.

Längs der Geraden sind Funktionswert, erste Ableitung (Geschwindigkeit, in diesem Fall konstant), sowie zweite Ableitung (Beschleunigung, in diesem Fall immer 0) festgelegt. Ein Punkt kann mit beliebiger erster und zweiter Ableitung definiert werden.

Zusätzlich hat der Anwender die Möglichkeit, Nocken, also binäre Positionsschalter, auf der Kurvenscheibe zu platzieren.

4.3 Starten des CAM-Editor und Erstellen einer Kurvenscheibe

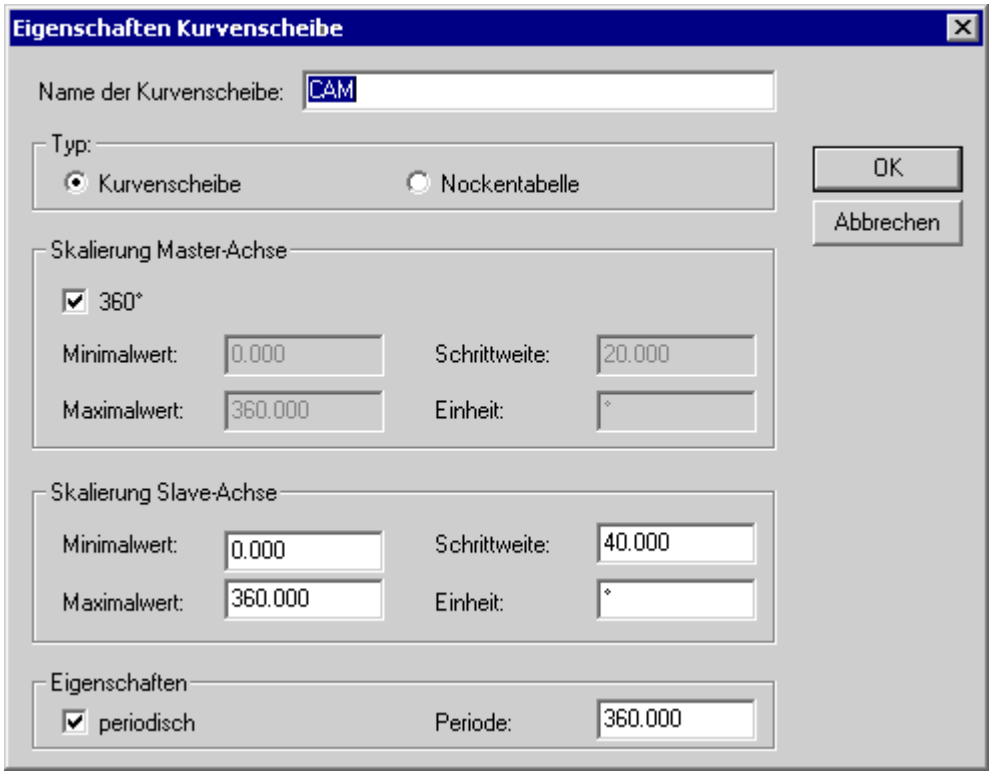
Start

Der **Kurvenscheiben**-Editor (CAM-Editor) wird aus der Registerkarte 'Ressourcen' gestartet. Es erscheint ein dreigeteiltes Fenster. Solange keine Kurvenscheibe angelegt ist, ist das Fenster leer. Ansonsten findet sich in der linken Spalte ein Baum mit allen programmierten Kurvenscheiben und in der oberen Fensterhälfte wird die Funktionsabhängigkeit Master-Slave-Achse graphisch dargestellt. Für die untere Fensterhälfte kann zwischen drei Darstellungen gewählt werden. Die Befehle dafür befinden sich im Menü 'Extras' (siehe Kapitel 4.4.1, Allgemeine Editor-Einstellungen) : Darstellung der

ersten (blaue Kurve, Geschwindigkeit) und zweiten (grüne Kurve, Beschleunigung) Ableitung, Darstellung einer Tabelle mit allen Kurvenscheibenelementen (Punkte/Geraden) oder Darstellung einer Tabelle mit allen Nocken. Die Tabellen können editiert werden.

'Extras' 'Neue Kurvenscheibe erstellen'

Wählen Sie den Befehl '**Neue Kurvenscheibe**' aus dem Menü 'Einfügen' oder aus dem Kontextmenü und nehmen Sie im sich öffnenden Dialog '**Eigenschaften Kurvenscheibe**' die gewünschten Einstellungen vor.



Name der Kuvenscheibe: Eine Bezeichnung für die neue Kurvenscheibe

Typ: Eine **Kurvenscheibe** enthält sowohl Kurvenscheiben-Elemente als auch Nocken; ein Nockenschaltwerk (**Nockentabelle**) enthält nur Nocken

Skalierung Master-Achse: Hier wird die Skalierung der Master-Achse definiert. Wenn Sie die Option **360°** aktivieren, werden die Einstellungen von **Minimalwert**, **Maximalwert**, **Schrittweite** und **Einheit** automatisch gesetzt (0, 360, 20, °), ansonsten können Sie diese auch individuell definieren.

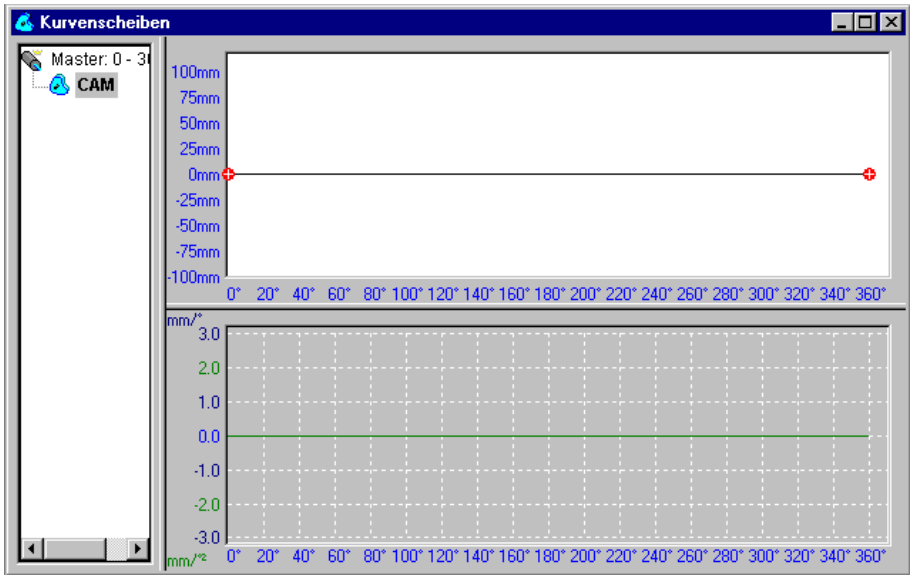
Skalierung Slave-Achse: Hier wird die Skalierung der Slave-Achse definiert. Die Default-Einstellungen sehen Sie in der oben gezeigten Abbildung des Dialogs.

Eigenschaften: Wenn die Option **periodisch** aktiviert ist, wird sichergestellt, dass am Anfangs- und Endpunkt der Kurvenscheibe Funktionswerte sowie erste und zweite Ableitungen übereinstimmen. Änderungen, die beim Editieren der Kurve am Endpunkt vorgenommen werden, werden dann negiert.

Schließen Sie den Dialog mit **OK**, um die Einstellungen zu bestätigen.

Daraufhin erscheint in der linken Spalte (Kurvenscheibenliste) des CAM-Editorfensters der Name der neu erstellten Kurvenscheibe. Solange dieser Eintrag markiert ist, wird die Kurvenscheibe im Editor dargestellt und kann bearbeitet werden (siehe Kapitel 4.4, CAM editieren). In den beiden rechten Fensterteilen wird die neu angelegte Kurvenscheibe dargestellt. Sie sehen nun die waagrechten blauen Masterachsen, die blaue Positionsachse (Slave) im oberen Fenster und die Geschwindigkeits-

(dunkelblau) und Beschleunigungsskala (grün) im unteren Fenster. Die folgende Darstellung entspricht den Defaulteinstellungen im Eigenschaften-Dialog:



Zur Bearbeitung der Einstellungen kann der Eigenschaften-Dialog für die aktuell markierte Kurvenscheibe jederzeit über den Befehl 'Einstellungen' wieder geöffnet werden, der im Menü 'Extras' oder im Kontextmenü verfügbar ist. bzw. mit einem Doppelklick auf den Eintrag in der Kurvenscheibenliste.

Kurvenscheiben-Baum

Der in der linken Fensterhälfte dargestellte Kurvenscheibenbaum enthält alle Kurvenscheiben und Nockenschaltwerke. Diese Elemente werden stets so sortiert, dass alle Elemente, die dieselbe Master-Skalierung haben, d.h. sich potentiell auf dieselbe Achse beziehen, denselben "Vater" haben.

4.4 Editieren einer Kurvenscheibe

In der linken Spalte des Editors wählt man die Kurvenscheibe aus, an der man arbeiten will. Dazu führt man einen Mausklick auf den Eintrag aus, so dass dieser markiert dargestellt (blau hinterlegt) wird und in den Editorfenstern angezeigt wird.

Durch gleichzeitiges Drücken der <Strg>-Taste und Mausklick auf eine oder mehrere weitere Kurvenscheiben mit demselben 'Vater' (siehe Kapitel 4.3, 'Kurvenscheibenbaum') werden diese Kurvenscheiben ebenfalls mit angezeigt.

4.4.1 Allgemeine Editor-Einstellungen

Der Editiermodus kann im Menü "**Einfügen**" oder über die entsprechende Schaltfläche in der Werkzeugleiste ausgewählt werden:

- Element selektieren
- Punkt einfügen
- Gerade einfügen
- Nocke einfügen

Bearbeitung der allgemeinen Einstellungen der Kurvenscheibe: Um die bereits beim Einfügen einer neuen Scheibe im Dialog '**Kurvenscheiben Eigenschaften**' vorgenommenen Einstellungen zu verändern, wählt man den Befehl Einstellungen im Menü '**Extras**' (siehe Kapitel 4.4.3).

Verändern der Anzeige der Kurve: Dazu stehen die Befehle "Kurve komplettieren" und „Extrema anzeigen“ im Menü "Extras" zur Verfügung.

Der Anzeigemodus des unteren Fensters kann im Menü "**Extras**" oder über die entsprechende Schaltfläche in der Werkzeugleiste ausgewählt werden:

 **Geschwindigkeit/Beschleunigung anzeigen**: Im unteren Fenster werden erste (blau) und zweite (grün) Ableitung der Kurvenscheibe visualisiert.

 **Kurve als Tabelle**: Das untere Fenster zeigt die einzelnen Elemente (Punkte/Geraden), aus denen die Kurvenscheibe besteht und ihre Eigenschaften in Form einer editierbaren Tabelle.

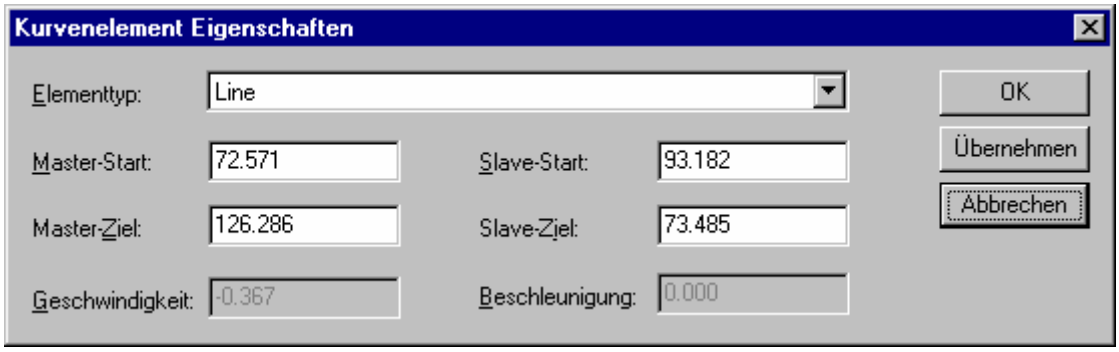
 **Nocken als Tabelle**: Das untere Fenster zeigt die einzelnen Nocken und ihre Eigenschaften in Form einer editierbaren Tabelle.

4.4.2 Bearbeiten der Eigenschaften einzelner Kurvenelemente:

Die Attribute einzelner Elemente können im Eigenschaften-Dialog oder durch Selektieren und Verschieben im Editorfenster verändert werden:

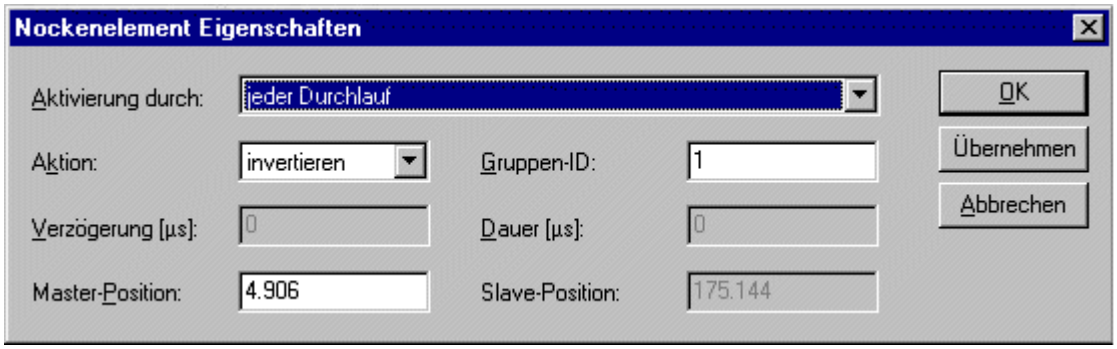
1. *Im Eigenschaften-Dialog:*

Punkt, Gerade: Über Doppelklick auf das Element wird der Dialog '**Kurvenelement Eigenschaften**' geöffnet, in dem man folgende Eigenschaften eines Elements durch numerische Eingaben verändern kann:



Elementtyp:	Line (Gerade) bzw. Point (Punkt); der Typ kann hier verändert werden; wird aus einem Punkt eine Gerade gemacht, wird automatisch eine bestimmte Länge vorgegeben; wird aus einer Geraden ein Punkt gemacht, werden automatisch die Koordinaten des Startpunkts der Geraden übernommen
Master-Start, Master-Ziel:	Start- und Endwerte (nur für Geraden-Elemente) bzgl. der X-Achse (Einheit siehe Dialog 'Eigenschaften Kurvenscheibe' ('Extras' 'Einstellungen'))
Slave-Start, Slave-Ziel:	Start- und Endwerte (nur für Geraden-Elemente) bzgl. der X-Achse (Einheit siehe 'Extras' 'Einstellungen')
Geschwindigkeit:	(nur für Punkt-Elemente)
Beschleunigung:	(nur für Punkt-Elemente)

Nocke: Über Doppelklick auf das Element wird der Dialog **Nockenelement Eigenschaften** geöffnet, in dem folgende Werte eingestellt werden können:



Aktivierung durch: Die Nocke wird aktiviert, d.h. die mit der Nocken-Gruppen-ID (s.u.) verknüpfte BOOLsche Variable im Programm (Nocken-Bit) wird auf TRUE gesetzt, wenn die Kurve durchlaufen wird; dafür kann eine der folgenden drei Optionen eingestellt werden:

positiver Durchlauf: nur wenn die Kurve von links nach rechts durchlaufen wird; nach 'Übernehmen' zeigt der grüne Pfeil über der Nocke nach rechts

negativer Durchlauf: nur wenn die Kurve von rechts nach links durchlaufen wird; nur wenn die Kurve von links nach rechts durchlaufen wird; nach 'Übernehmen' zeigt der grüne Pfeil über der Nocke nach links

jeder Durchlauf: bei jedem Durchlaufen der Kurve; nach 'Übernehmen' zeigt der grüne Pfeil über der Nocke nach links und rechts

Aktion: Eine der folgenden Optionen kann eingestellt werden um festzulegen, welche Auswirkung die Aktivierung der Nocke auf die Aktion, die mit ihr im Projekt verknüpft ist, haben soll:

ein: Die Aktion wird gestartet (das "Nocken-Bit" wird auf TRUE gesetzt); Das Nocken-Quadrat wird grün gefüllt

aus: Die Aktion wird gestartet (das "Nocken-Bit" wird auf FALSE gesetzt); Das Nocken-Quadrat wird rot gefüllt

invertieren: Wenn die Aktion gerade aktiv ist, wird sie gestoppt, wenn sie inaktiv ist, wird sie gestartet (das Nocken-Bit wird invertiert); Das Nocken-Quadrat wird gelb gefüllt

zeitgesteuert ein: Die Aktion wird mit den Werten, die in den Feldern Verzögerung und Dauer eingetragen sind, gestartet; Das Nocken-Quadrat wird cyan gefüllt

ID: Kennnummer der Nocke zur Referenzierung im Projekt (INT); mehrere Nocken können dieselbe Gruppen-ID erhalten und dadurch in dem Sinne "gruppiert" werden, dass die ihnen zugeordnete Aktion denselben digitalen Schalter bedient.

Verzögerung [µs]: Zeitspanne, nach der nach Durchlauf durch das Nockenelement die Aktion, mit der sie im Projekt verknüpft ist, gestartet wird (nach der das Nocken-Bit auf TRUE gesetzt werden soll). (nur, wenn Aktion = zeitgesteuert ein).

Dauer [µs]: Angabe, wie lange die Aktion, mit der die Nocke im Projekt verknüpft ist, nach ihrem Start aktiv sein soll (wie lange das Nocken-Bit TRUE sein soll). (nur, wenn Aktion = zeitgesteuert ein).

Master-Position: X-Position der Nocke

Slave-Position: Y-Position der Nocke, nicht editierbar, da durch den Kurvenverlauf vorgegeben

Die in den Eigenschaften-Dialogen eingetragenen Werte können mit **OK** oder **Übernehmen** bestätigt werden, worauf die Kurve im Editor entsprechend angepasst wird. OK schließt gleichzeitig den Dialog, während er bei Übernehmen geöffnet bleibt.

(Bearbeiten der Eigenschaften einzelner Kurvenelemente)

2. Durch Selektieren und Verschieben im Editorfenster:

Ein Element kann durch Mausklick selektiert und bei gedrückt gehaltenem Mauszeiger verschoben werden. Dies hat rückkoppelnd eine Veränderung der entsprechenden Werte im jeweiligen 'Eigenschaften'-Dialog (s.o.) zur Folge:

Punkt: Wenn ein Punkt angeklickt wird, erscheint ein kleines rotes Quadrat, das die Steigung (Geschwindigkeit) wiedergibt. Über ein Verschieben dieses Quadrats kann die Steigung des Punktes verändert werden. Diese wird dabei durch eine Hilfstangente angezeigt. Ebenso kann der Punkt selbst verschoben werden.

Gerade: Wenn eine Gerade angeklickt wird, erscheinen an den Endpunkten kleine rote Quadrate. Ein weiterer Mausklick auf einen der Endpunkte und ein Verschieben desselben ermöglicht ein Verändern der Steigung; wird dagegen auf die Geradenlinie geklickt, können Sie diese ohne Veränderung der Steigung verschieben.

Nocke: Wenn eine Nocke angeklickt wird, wird der Rand des Nockenquadrats rot. Die Nocke kann dann durch Bewegen des Mauszeigers auf der Kurvenbahn verschoben werden.

4.4.3 Befehle der Menüs 'Extras' und 'Einfügen'

'Extras' 'Einstellungen'

 Dieser Befehl öffnet die Dialogbox 'Eigenschaften Kurvenscheibe', die auch bei der Erstellung der Kurvenscheibe zu sehen war. (Sehen Sie die Abbildung in Kapitel 4.3.) Hier können Skalierung und Einheiten geändert werden.

'Extras' 'Kurve komplettieren'

Wenn diese Option aktiviert ist (Haken vor Menüpunkt bzw. Schaltfläche in Werkzeugleiste "gedrückt"), werden in der Kurve zusätzlich die die Elemente (Punkte, Geraden, Nocken) verbindenden Polynome fünften Grades angezeigt. Ansonsten werden nur die einzelnen Elemente dargestellt.

'Extras' 'Extrema anzeigen'

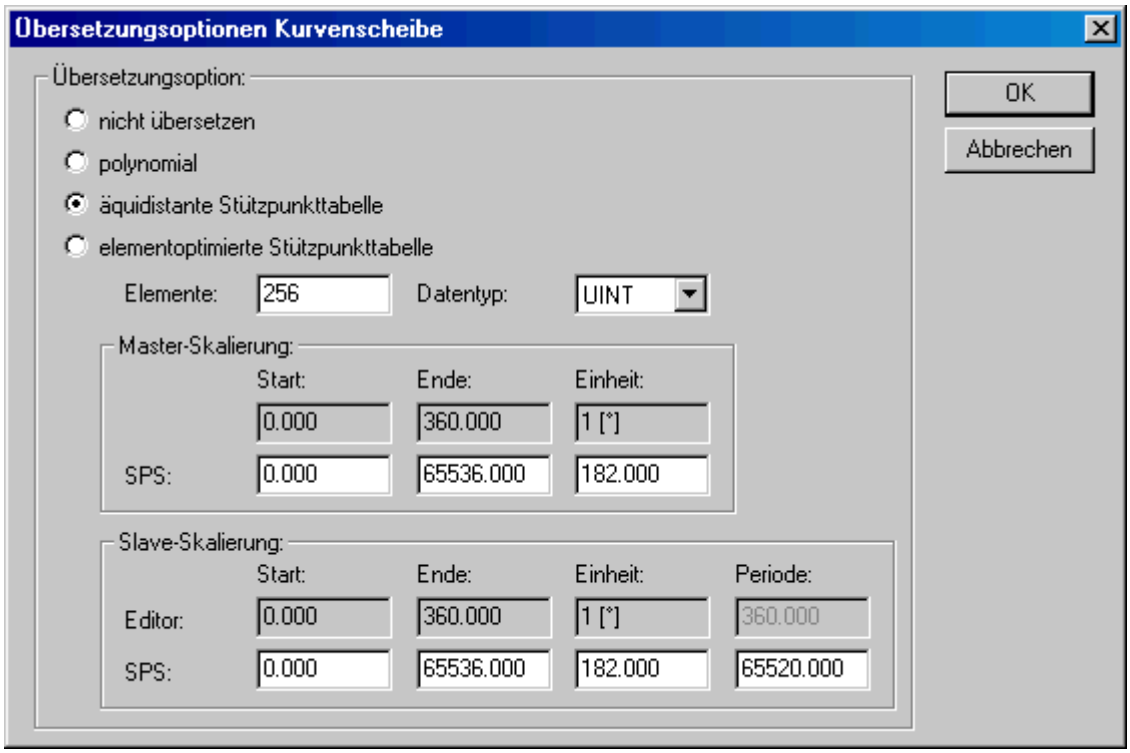
Ist diese Option aktiviert (Haken vor Menüpunkt bzw. Schaltfläche in Werkzeugleiste "gedrückt"), werden zusätzlich zur Kurvenscheibe und ggfs. ihren Ableitungen deren Extremwerte (Maximum/Minimum) angezeigt.

'Extras' 'Übersetzungsoptionen'

Dieser Menüpunkt öffnet einen Dialog, mit dessen Hilfe die Übersetzung der Kurvenscheibe (siehe auch 4.1) eingestellt werden kann.

Es gibt grundsätzlich drei Übersetzungsmodi:

1. **polynomial**: Beim Übersetzen werden Variablen der Struktur MC_CAM_REF angelegt. Diese enthalten für jeden Abschnitt die Beschreibung des Polynoms fünften Grades, welches die Kurvenscheibe beschreibt. Strukturen dieser Art werden als Eingang des MC_CamIn-Bausteins verwendet. Die Struktur ist Bestandteil der Bibliothek SM_DriveBasic.lib.



2. **äquidistant:** Entsprechend den Einstellungen im unteren Teil des Dialogs wird eine Stützpunkttabelle des Typs SMC_CAMTable_<Datentyp>_<Elementzahl>_1 erzeugt. Das Positions-Array dieses Dialogs enthält die Slave-Werte der Kurvenscheibe bezüglich Master-Werte, die gleichmäßig auf den Definitionsbereich der Master-Achse verteilt sind. Dabei bezieht sich der erste Wert der Tabelle auf die Slave-Position am Master-Minimum der Kurvenscheibe. Der letzte bezieht sich bei nicht-periodischen Kurvenscheiben auf die Slave-Position am Master-Maximum. Bei periodischen Kurvenscheiben muss dieser Wert nicht erneut beschrieben werden, da er mit dem Wert am Master-Minimum übereinstimmt; deshalb werden die Intervalle etwas enger gemacht und der letzte Tabellenwert beschreibt die Slave-Position bei Master.Ende - (Master.Ende-Master.Start)/Elementzahl.
3. **elementoptimiert**: Entsprechend den Einstellungen im unteren Bereich des Dialogs wird eine zweidimensionale (typischerweise nicht äquidistante) Stützpunkttabelle des Typs SMC_CAMTable_<Datentyp>_<Elementzahl>_2 erzeugt. Die darin enthaltene Tabelle enthält Paare von Master- und zugehöriger Slave-Positionen. Die Unterteilung wird dabei so durchgeführt, dass Elemente mit konstanter Geschwindigkeit (Linien) je nur einen Stützpunkt am Anfang und einen am Ende erhalten. Die restlichen Stützpunkte werden möglichst gleichmäßig auf den Rest der Kurvenscheibe verteilt.
4. **nicht übersetzen**: Für die Kurvenscheibe werden keine globalen Variablen erzeugt. Diese Option wird hauptsächlich dann verwendet, wenn die Kurvenscheibe zur Laufzeit aus dem Dateisystem geladen werden soll (siehe 10.3), z.B. weil sie – ohne, dass das laufende CoDeSys-Projekt berührt wird – geändert werden muss.

Die **Skalierung** ist nur für die Stützpunkttabellen von Belang. Hier kann eine Skalierung sowohl der Master- als auch der Slave-Achse entweder über Anfangs- und Endwertzuordnung, oder über Anfangswert- und Einheitenzuordnung definiert werden.

'Extras' 'Kurvenscheibe in Datei schreiben'

Dieser Menüpunkt öffnet einen Dateiauswahl-Dialog, in dem eine *.CAM-Datei angegeben werden kann, in welche die aktuelle Kurvenscheibe geschrieben wird. Diese Datei kann zur Laufzeit vom Funktionsblock SMC_ReadCAM (siehe 10.3) gelesen und von diesem zu einer Standard-Datenstruktur verwandelt werden. In Abhängigkeit von der eingestellten Übersetzungsoption wird die Kurvenscheibe in polynomialer, äquidistanter oder elementoptimierter Form abgespeichert.

'Extras' 'Kurvenscheibe aus Datei lesen'

Mit diesem Menüpunkt kann eine *.CAM-Datei wieder in CoDeSys eingelesen werden. Nach Auswahl der entsprechenden Datei wird der Dialog ‚Eigenschaften Kurvenscheibe' geöffnet, in welchem ein Name für die Kurvenscheibe und die Slave-Skala angepasst werden muss.

Da beim Erzeugen der Kurvenscheibe nur die Informationen ausgegeben werden, die für die Ausführung der Kurvenscheibe benötigt werden, kann sich eine gelesene Kurvenscheibe von der ursprünglichen unterscheiden.

'Extras' 'Kurvenscheibe als ASCII-Tabelle exportieren'

Hier können Sie die aktuelle Kurvenscheibe als ASCII-Textfile exportieren. Dabei kann die Anzahl an Punkten spezifiziert werden. Der Start- und Endpunkt ist stets als erster bzw. letzter Punkt enthalten. Daraufhin wird ein Textfile mit folgendem Aufbau erzeugt:

<Master-Position>;<Slave-Position><CR><LF>

Das so erzeugte Textfile kann beispielsweise in andere Programme importiert und zur Antriebsauslegung verwendet werden.

'Extras' CAM Kurvenscheibe aus ASCII-Tabelle importieren'

Mit dieser Funktion können Kurvenscheibentabellen, die als ASCII-Datei in folgendem Format abgelegt sind, importiert werden:

<Master-Position>;<Slave-Position>

Dabei ist auch zu beachten, dass die Punkte bezüglich der Master-Position aufsteigend sortiert sind.

Konnte die Kurvenscheibe eingelesen werden, kann der Anwender im Eigenschaften-Dialog den Namen, das Intervall und die Skaleneinteilung festlegen. Anschließend hat er die Möglichkeit, die Anzahl Stützstellen der Kurvenscheibe zu reduzieren.

'Einfügen' 'Element selektieren'

 Dieser Menüpunkt dient dem Ein- und Ausschalten des Selektiermodus. Solange er aktiviert ist (Haken vor Menüpunkt, Schaltfläche in Werkzeugleiste erscheint gedrückt), kann mit einem Druck der linken Maustaste das Kurvenelement (Punkt, Gerade, Nocke) selektiert werden, auf das der Mauszeiger gerichtet ist. Daraufhin werden die entsprechenden Markierungspunkte (kleine rote Quadrate) angezeigt und das Element kann editiert werden.

'Einfügen' 'Punkt einfügen'

 Wenn dieser Befehl im Menü 'Einfügen' oder in der Werkzeugleiste aktiviert wird, wird folgendermassen ein neuer Punkt in die Kurve eingefügt: Man bewegt den Mauszeiger an die Position des neuen Stützpunktes. Drückt man nun die linke Maustaste, wird der Punkt eingefügt und erhält eine waagrechte Tangente. Durch Loslassen der linken Maustaste wird der Punkt übernommen (Darstellung als roter gefüllter Kreis mit Fadenkreuz) und automatisch in den Modus "Element selektieren" gewechselt.

Hält man während des Einfügens die linke Maustaste gedrückt, kann die Steigung der Tangente (Geschwindigkeit) unmittelbar durch Bewegen der Maus verändert werden.

'Einfügen' 'Gerade einfügen'

 Wenn dieser Befehl im Menü 'Einfügen' oder in der Menüleiste aktiviert wird, kann folgendermassen eine Gerade in die Kurve eingefügt werden: Man bewegt den Mauszeiger an die Position des Anfangspunktes der Gerade und hält die linke Maustaste gedrückt. Der Anfangspunkt wird durch ein kleines rotes Quadrat markiert und durch Bewegen der Maus auf den gewünschten Endpunkt (ebenfalls ein rotes Quadrat) kann man die Gerade erzeugen. Der Endpunkt der Geraden kann dabei nicht links vom Anfangspunkt und nicht rechts vom nächsten definierten Punkt platziert

werden. Sobald die linke Maustaste losgelassen wird, wird der Endpunkt übernommen und automatisch in den Modus "Element selektieren " gewechselt.

'Einfügen' 'Nocke einfügen'

 Wenn dieser Befehl im Menü 'Einfügen' oder in der Werkzeugleiste aktiviert wird, kann folgendermassen eine Nocke in die Kurve eingefügt werden: Man bewegt den Mauszeiger an die gewünschte Position für die Nocke und drückt die linke Maustaste. Die Nocke wird auf der Kurve entsprechend der mit dem Mauszeiger gewählten X-Position eingehängt. Solange die linke Maustaste noch gedrückt ist, kann sie durch Bewegen der Maus auf der Kurve verschoben werden. Durch Loslassen der linken Maustaste wird die Position übernommen und automatisch in den Modus "Element selektieren" gewechselt.

4.5 Verwendung von Kurvenscheiben

Zur Ausführung der Kurvenscheiben stehen in der Bibliothek SM_PLCopen.lib Bausteine zur Verfügung, die im Kapitel 5.4 beschrieben werden.

Dieses Kapitel soll einige Erklärungen zur Anwendung der Kurvenscheiben, speziell zur Aneinanderreihung von Kurvenscheiben und genaueren Bedeutung der verschiedenen Parameter (Periodizität, Offset, etc.) liefern:

4.5.1 Auswirkungen von Baustein-Parametern

Editor-Kurvenscheibeneigenschaften

- **Periodisch**, - **Periode**

Diese beiden Einstallungen sind lediglich für das Erstellen der Kurve relevant; auf die Abarbeitung der Kurve in der Applikation wirken sie nicht.

Periodisch entscheidet, ob die Kurve sprungfrei aneinander gehängt, d.h. periodisch ausgeführt werden kann. Ist diese Option angewählt, haben Anfangs- und Endpunkt dieselbe Slave-Position oder die Differenz zwischen Ihnen ist ein Vielfaches der Periode.

Auch wenn die „periodisch“-Option nicht angewählt ist, können die Kurvenscheiben trotzdem hintereinander ausgeführt werden; dann muss sich allerdings der Anwender darum kümmern, dass die Übergänge im gewünschten Masse stetig sind.

CAMTableSelect

- **Periodisch**

Dieser Eingang entscheidet, ob eine Kurvenscheibe, wenn die Masterposition den Kurvenscheibenbereich verlässt, erneut ausgeführt wird.

Ist dieser Eingang FALSE, so wird am Ende der Kurvenscheibe der Ausgang EndOfProfile am CamIn-Baustein gesetzt, und der Slave wird an der letzten durch die Kurvenscheibe programmierten Position gehalten. Man beachte, dass wenn der Master wieder in den zulässigen Bereich eintritt, der Slave immer noch gemäß der Kurvenscheibe gesteuert wird; die Kurvenscheibenaktion wird also durch das Verlassen des beschriebenen Masterbereichs nicht beendet.

Ist der Eingang TRUE, wird die Kurvenscheibe aneinandergereiht, d.h. die Kurvenscheibe wird in Masterrichtung an ihr bisheriges Ende verschoben.

Dadurch steht also die Masterperiode (Breite) einer Kurvenscheibe in keinem Verhältnis zur Positionsperiode des Masterantriebs. Man beachte deshalb, dass die Zuordnung Slaveposition = Kurvenscheibe(Masterposition), also die Bestimmung der Slaveposition durch die Masterposition über

die Kurvenscheibe, nur im ersten Kurvenscheiben-Zyklus zutrifft, und dann nicht mehr, wenn sich die Breite der Kurvenscheibe von der Periode des Masters unterscheidet.

- **MasterAbsolute**

Steht dieser Eingang auf TRUE, wird die Kurvenscheibe beim Start an der aktuellen Masterposition begonnen. Dieser Punkt kann also auch in der Mitte der Kurvenscheibe liegen. Liegt dieser Punkt außerhalb des durch die Kurvenscheibe spezifizierten Bereichs, wird ein Fehler gemeldet.

Ist der Eingang FALSE, wird die Kurvenscheibe zur aktuellen Position hin verschoben, d.h. der Nullpunkt der Kurvenscheibe ist dann die aktuelle Masterposition. Kurvenscheiben, deren Masterbereich nicht die 0 enthält, können nicht mit diesem Modus verwendet werden, da sonst eine Fehler gemeldet wird (Master verlässt spezifizierten Bereich).

- **SlaveAbsolute**

Dieser Eingang steht in Zusammenhang mit dem Eingang StartMode von MC_CamIn. Folgende Tabelle beschreibt seine Auswirkungen auf den Start-Modus:

CamIn.StartMode	absolute	relative	ramp_in	ramp_in_pos	ramp_in_neg
CamTableSelect.SlaveAbsolute					
TRUE	absolut	relativ	ramp_in absolut	ramp_in_pos absolut	ramp_in_neg absolut
FALSE	relativ	relativ	ramp_in relativ	ramp_in_pos relativ	ramp_in_neg relativ

Für eine genauere Beschreibung der Modi siehe CamIn.StartMode

CamIn Baustein

- **MasterOffset**, - **MasterScaling**

Diese beiden Einstellungen transformieren die Masterposition gemäß folgender Formel und verwenden die transformierte Position zur Auswertung der Kurvenscheibe:

```
x = Master-Position*MasterScaling + MasterOffset
```

MasterScaling>1 bewirkt also, dass die Kurve schneller abgearbeitet (komprimiert wird), bei <1 wird sie gestreckt.

- **SlaveOffset**, - **SlaveScaling**

Diese beiden Einstellungen verschieben und Strecken eine Kurve in Slave-Richtung, und zwar wird die Kurve zunächst gestreckt und anschließend verschoben.

SlaveOffset>1 bewirkt, dass der Slave eine größere Bewegung macht (Kurve wird gestreckt); bei <1 wird sie komprimiert.

- **StartMode : absolute/relative/ramp_in/ramp_in_pos/ramp_in_neg**

Relativ bedeutet, dass die Kurvenscheibe um die aktuelle Slave-Position beim Start der Kurvenscheibe verschoben wird. Dies ergibt nur dann richtig Sinn, wenn die Slavesollposition gemäß der Kurvenscheibe am Startpunkt 0 ist, andernfalls entsteht ein Sprung.

Absolut bedeutet, dass die Kurvenscheibe unabhängig von der aktuellen Slave-Position gewertet wird. Steht der Slave auf einer anderen Position als sich aus der Kurvenscheibe und der aktuellen Master-Position ergibt, entsteht ein Sprung.

Die ramp_in-Option bewirkt, dass etwa entstehende Sprünge (s.o.) beim Start der Kurvenscheibe durch eine Augleichsbewegung (gemäß den eingestellten Begrenzungen VelocityDiff, Acceleration, Deceleration) kompensiert werden. Dabei würde ramp_in_pos (sofern es sich um einen rotatorischen Slave handelt) nur in positive Richtung ausgleichen; ramp_in_neg in negative. Bei linearen Slaves ergibt sich die Ausgleichrichtung automatisch, d.h. ramp_in_pos und ramp_in_neg werden wie ramp_in behandelt.

4.5.2 Umschalten zwischen Kurvenscheiben

Grundsätzlich kann zu jedem Zeitpunkt zwischen zwei Kurvenscheiben umgeschaltet werden. Man sollte dabei folgendes beachten: Bei einer steigenden Flanke bestimmt sich die Slave-Sollposition strikt nach der aktuellen Masterposition (verrechnet mit Masteroffset und Masterscaling) und der verwendeten Kurvenscheibe und wird lediglich durch Slaveoffset und –scaling verändert. Dies führt dazu, dass – sollte die Master- oder Slaveperiode der Kurvenscheibe und der tatsächlichen Positionsperiode der Antriebe unterschiedlich sein – sich pro Kurvenscheibenzyklus bzw. Antriebsumdrehung weitere Verschiebungen der Master- und Slave-Position ergeben, sodass die einfache Funktionsgleichung

```
Slaveposition = CAM(Masterposition)
```

nicht mehr zutrifft, sondern vielmehr gilt :

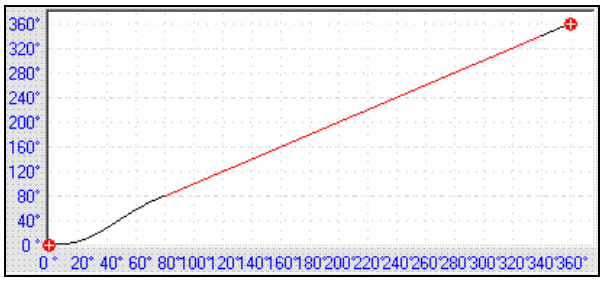
```
Slaveposition = CAM(Masterposition + Offset1) + Offset2
```

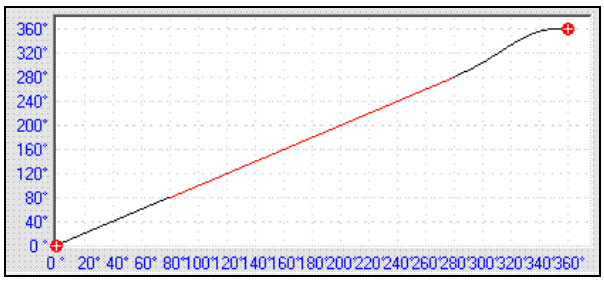
Wobei sich Offset1 und Offset2 am Ende jeder Kurvenscheibe ändern können.

Daraus resultiert, dass, wenn man den Kurvenscheibenbaustein MC_CamIn neu startet (steigende Flanke bei bExecute), dieser, da für ihn eine komplett neue Aktion gestartet wird, sein Gedächtnis und damit auch Offset1 und Offset2 löscht, und wieder die ursprüngliche Gleichung heranzieht, die einen ganz anderen Slavesollwert zur Folge haben kann. Deshalb startet man den MC_CamIn-FB nur dann neu, wenn man eine andere Kurvenscheibe verfahren will.

Beim Umschalten zwischen zwei Kurven hat man folgendes zu beachten:

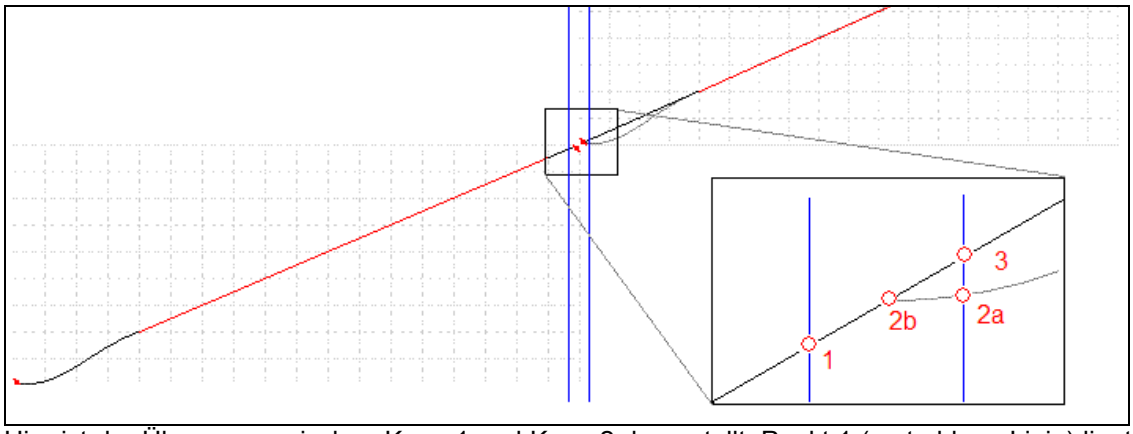
Beispielsweise soll von Kurve1 auf Kurve2 umgeschaltet werden:





(1) Die Geschwindigkeiten und Beschleunigungen des Endpunkts der letzten Kurve und der neuen Kurve sollten übereinstimmen, um Sprünge und Rucks zu vermeiden. Das ist im Beispiel der Fall: der Endpunkt der Kurve1 und der Startpunkt von Kurve2 haben identische Geschwindigkeits- und Beschleunigungswerte.

(2a) Ist die neue Kurve so konfiguriert, dass sie die Slaveposition bei 0 anfängt, dann kann man die Kurve relativ starten, muss dann aber auch die vorige Kurve als nicht-periodisch (MC_CamTableSelect) starten. Begründung: würde man die Kurve als periodisch starten, könnten folgende Stützstellen berechnet werden:



Hier ist der Übergang zwischen Kurve1 und Kurve2 dargestellt. Punkt 1 (erste blaue Linie) liegt noch auf Kurve1 und wird ganz normal behandelt. Beim nächsten Aufruf des Bausteins ist der Master (zweite blaue Linie) über das Ende der Kurvenscheibe1 hinausgewandert. Jetzt ist es

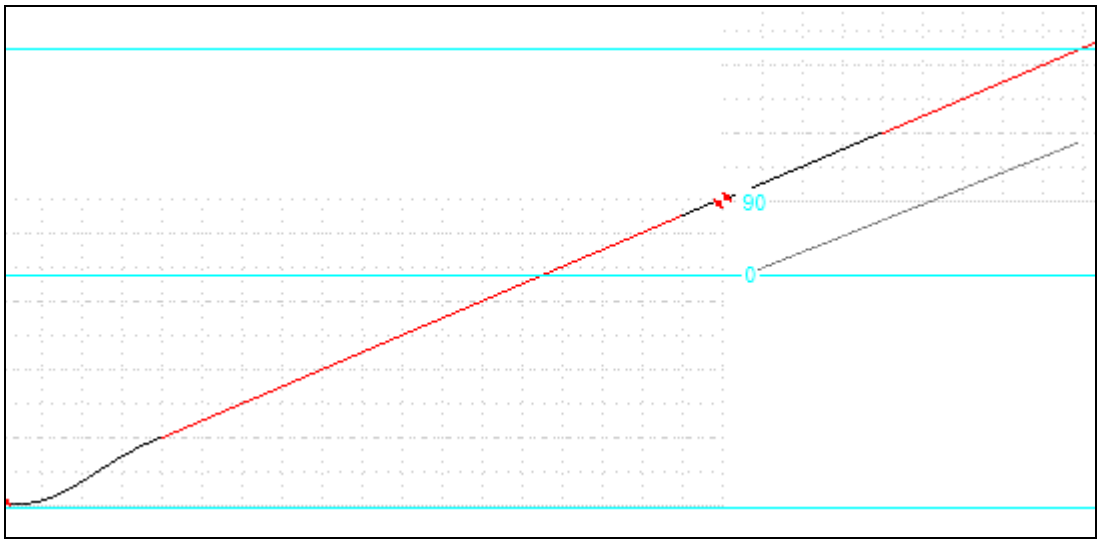
wichtig, dass man die vorige Kurvenscheibe nicht-periodisch ausgeführt hat. Ist dies der Fall, so wird nämlich die Slaveposition 2b berechnet; wurde sie periodisch ausgeführt, wird die Position 2a berechnet, die ja entstehen würde, wenn man Kurve1 wiederholen wollte. Gleichzeitig wird – unabhängig ob periodisch oder nicht – EndOfProfile gesetzt.

Dieser Ausgang kann nun dazu benutzt werden, um Kurve2 zu starten. Dazu wird die MC_CamIn-Instanz mit Execute=FALSE aufgerufen, und im selben Zyklus noch mal mit Execute=TRUE und der neuen Kurvenscheiben-CamTableID, was die Ausgabe von „3“ auf den Slave-Drive bewirkt.

(2b) Will man die Kurve absolut starten, so ist zu beachten, dass der Slave beim Start der Kurvenscheibe auch an der entsprechenden Position steht, da sonst ein Sprung entsteht. Dies ist beim Umschalten von Kurvenscheiben i.d.R. dann der Fall, wenn die Periode der Kurvenscheibe auch der Slaveperiode entspricht.

Im obigen Beispiel könnte man Kurve2 im Modus absolut starten, wenn die Positionsperiode von Master und Slave in Übereinstimmung mit dem Bereich der Kurvenscheibe jeweils auch 360° betragen. Dann ist es auch egal, ob die Kurve periodisch oder nicht-periodisch gestartet wird.

Nicht möglich wäre dies, wenn die Slave-Periode z.B. 270° (durch hellblaue Striche gekennzeichnet) betragen würde:



Dann stünde beim Übergang von Kurve1 und Kurve2 der Slave auf Position 90. Würde man jetzt Kurve2 im Modus absolute starten, würde sich die graue Kurve und damit ein Positionssprung ergeben. Man könnte sich aber z.B. dadurch behelfen, dass man einen entsprechenden Slave-Offset von 90 programmiert.

4.6 Kurvenscheiben-Datenstrukturen

Beim Übersetzen des Projekts wird aus den im CAM-Editor programmierten Kurven eine Globale Variablenliste (Register 'Ressourcen' in CoDeSys) mit dem Namen **CAM Data** generiert. Hier wird die Beschreibung jeder Kurvenscheibe in jeweils einer Struktur entsprechend den Einstellungen im Dialog ‚Übersetzungsoptionen’ abgelegt und so dem IEC-Programm, bzw. den Kurvenscheiben-Bausteinen zur Verfügung gestellt. Für ein fehlerfreies Übersetzen müssen die entsprechenden Strukturdefinitionen im IEC-Programm vorhanden sein.

(Sehen Sie hierzu auch die Programmierbeispiele in Kapitel 11.)

Zudem wird beim Übersetzen eine Datenstruktur _SMC_CAM_LIST (ARRAY OF POINTER TO MC_CAM_REF) angelegt, die auf die einzelnen Kurvenscheiben über Pointer verweist.

Natürlich können die entsprechenden Datenstrukturen auch aus dem IEC-Programm zur Laufzeit erstellt bzw. befüllt werden. Aus diesem Grund werden sie im folgenden genauer beschrieben. Nicht aufgeführte Variablen der Struktur sind nur von interner Bedeutung.

MC_CAM_REF

Diese Datenstruktur repräsentiert eine allgemeine Kurvenscheibe und enthält folgende Elemente:

wCAMStructID: WORD

Mithilfe dieser Variable, die stets einen festen Wert hat, überprüfen die Bausteine, ob es sich bei der eingegebenen Datenstruktur um eine MC_CAM_REF handelt.

xStart, xEnd: LREAL

Definitionsbereich der Kurvenscheibe. Start- und Endposition des Masters.

byType:BYTE

Diese Variable beschreibt den Kurvenscheibentyp, also die Art und Weise, wie die Kurvenscheibe repräsentiert wird.

1: äquidistante, 1-dimensionale Tabelle aus Slave-Positionen

2: nicht-äquidistante, 2-dimensionale Tabelle aus Master/Slave-Punktepaaren

3: polynomiale Beschreibung über einzelne Punkte bestehend aus Masterposition, Slave-Position, -geschwindigkeit und –Beschleunigung (XYVA).

byVarType:BYTE (nur für byType=1 oder byType=2)

Variablentyp, aus welcher die Kurventabelle besteht:

0: INT
1: UINT
2: DINT
3: UDINT
4: REAL
5: LREAL

nElements:INT

Anzahl an Elementen, also je nach Typ SlavePositionen, Master/Slave-Positionen oder XYVA-Punkten.

byInterpolationQuality:BYTE (nur für byType=1 oder byType=2)

Feininterpolationsgrad: 1: linear (default), 3: kubisch

pce: POINTER TO BYTE

Zeiger auf das eigentliche Datenelement; abhängig vom Typ:

Typ	
1 (äquidistant)	SMC_CAMTable_<VarType>_<nElements>_1
2 (nicht-äquidistant)	SMC_CAMTable_<VarType>_<nElements>_2
3 (XYVA)	ARRAY OF SMC_CAMXYVA

nTappets: INT

Anzahl Nockenschaltaktionen.

pt: POINTER TO SMC_CAMTappet

Zeiger auf ein ARRAY OF SMC_CAMTappet

strCAMName:STRING

Name der Kurvenscheibe

SMC_CAMXYVA

Eine XYVA-Kurvenscheibe besteht aus einem Array of SMC_CAMXYVA. Jede Variable dieses Arrays beschreibt einen Punkt der Kurvenscheibe über **dX** (Masterposition), **dY** (Slaveposition), **dV** (erste Ableitung dY/dX; entspricht der Slave-Geschwindigkeit bei konstanter Mastergeschwindigkeit 1) und **dA** (zweite Ableitung d^2Y/dX^2; entspricht der Slave-Beschleunigung bei konstanter Mastergeschwindigkeit 1). Der Start- und Endpunkt der Kurvenscheibe muss in jedem Fall enthalten sein.

SMC_CAMTable_<Variablen-Typ>_<Anzahl Elemente>_1

In dieser Datenstruktur wird eine äquidistante Kurventabelle abgelegt. Die einzelnen Slave-Positionen werden in **Table: ARRAY[0..<Anzahl Elemente>-1] OF <Variablen-Typ>** gespeichert. Der Start- und Endpunkt der Kurvenscheibe muss in jedem Fall enthalten sein.

Die Variablen **fEditorMasterMin**, **fEditorMasterMax**, **fTableMasterMin**, **fTableMasterMax** beschreiben eine zusätzliche Skalierung der Tabellen, indem der Definitions-/Wertebereich in SoftMotion-Einheiten (fEditorMaster, fEditorSlave) und skaliert auf Tabelleneinheiten (fTableMaster, fTableSlave) gespeichert wird.

SMC_CAMTable_<Variablen-Typ>_<Anzahl Elemente>_2

Eine nicht-äquidistante Kurventabelle wird in **Table: ARRAY[0..<Anzahl Elemente>-1] OF ARRAY[0..1] OF <Variablen-Typ>** abgelegt. Im Gegensatz zur äquidistanten Form ist das erste Element die Master-Position, das zweite die Slave-Position.

4.6.1 Beispiel für manuell erzeugte Kurvenscheibe

Dieses Beispiel zeigt, wie eine Kurvenscheibe im IEC-Programm, ohne Verwendung des Editors erzeugt werden kann:

Variablen-Deklaration:

```
CAM: MC_CAM_REF:=(
     byType:=2, (* non-equidistant )
     byVarType:=2, (* UINT *)
     nElements:=128,
     xStart:=0,
     xEnd:=360);

Table: SMC_CAMTable_UINT_128_1:=(
     fEditorMasterMin := 0, fEditorMasterMax := 360,
     fTableMasterMin := 0, fTableMasterMax := 65536,
     fEditorSlaveMin := 0, fEditorSlaveMax := 360,
     fTableSlaveMin := 0, fTableSlaveMax := 65536);
```

Programmteil:

```
(* Kurvenscheibe erzeugen (als Beispiel eine Gerade); einmalig *)
FOR i:=0 TO 127 DO
     Table.Table[i][0]:=Table.Table[i][1]:=REAL_TO_UINT(i / 127.0 * 65536);
END_FOR

(* Zeiger verknüpfen; muss jeden Zyklus erfolgen !!! *)
CAM.pce := ADR(Table);
```

Die so erzeugte Kurvenscheibe kann nun in den Baustein MC_CamTableSelect eingegeben, und dessen Ausgang wieder für MC_CamIn verwendet werden.

5 Die Bibliothek SM_PLCopen.lib

5.1 Überblick

Die Bausteine der Bibliothek "SM_PLCopen.lib" beruhen auf der Spezifikation von PLCopen: "Function blocks for motion control, Version 1.0".

Die vorliegende Beschreibung baut auf dieser Spezifikation auf. Bausteine, die dem Standard von PLCopen folgen, beginnen mit MC_<Bausteinname>, während 3S-spezifische Bausteine als SMC_<Bausteinname> benannt sind.

Die gänzlich in IEC1131-3 programmierten Funktionsblöcke lassen sich in drei Kategorien einteilen:

- Bausteine zur allgemeinen Bedienung, Überwachung und Parametrierung einzelner Antriebe. Eine Beschreibung sehen Sie in Kapitel 5.3.

- Bausteine zur unabhängigen Bewegungssteuerung einzelner Antriebe. Mit Hilfe dieser Bausteine können einzelne Achsen autonom auf verschiedene Arten bewegt werden. Eine Beschreibung sehen Sie in Kapitel 5.3.

- Bausteine zur Bewegungssteuerung eines Antriebs (Slave) in Abhängigkeit eines weiteren (Master). Diese Bausteine bieten die Möglichkeit Kurvenscheiben, elektronische Getriebe und Phasenverschiebungen zu realisieren. Eine Beschreibung sehen Sie in Kapitel 5.4.

- Zusatzbausteine siehe Kapitel 5.5.

- Außerdem sind für alle wichtigen Bausteine Visualisierungs-Templates in der Bibliothek enthalten, die mit einer Instanz des zugehörigen Bausteins verknüpft dessen Ein- und Ausgänge darstellen und v.a. für die Applikations-Erstellungs- und Testphase von großem Nutzen sind.

Voraussetzungen:

Diese Bibliothek baut auf der Bibliothek "SM_DriveBasic.lib" (siehe Kapitel 2.3) auf, in welcher u.a. die Struktur AXIS_REF definiert ist.

5.2 PLCopen-Spezifikation "Function blocks for motion control, Version 1.0"

Es wird empfohlen, neben dieser Beschreibung die PLCopen-Spezifikation "Function blocks for motion control, Version 1.0" zu lesen. Die wichtigsten Dinge werden hier kurz zusammengefasst:

Bausteine werden auf zweierlei Arten aktiviert:

a) **Enable-Eingang**: Verfügt der Baustein über einen Enable-Eingang (wie z.B. MC_ReadParameter), ist er genau solange aktiv, wie Enable TRUE ist.

b) **Execute-Eingang**: Der Baustein wird durch eine steigende Flanke (Übergang von FALSE nach TRUE) des Execute-Eingangs aktiviert und wird erst dann wieder inaktiv, falls er die Bewegung abgeschlossen hat, die Kontrolle über die Achse (AXIS_REF) ihm durch einen anderen Baustein entzogen wird, oder er eine neue steigende Flanke am Execute-Eingang erhält und die Bewegung damit neu startet. Man beachte auch, dass alle Eingangs-Variablen nur bei steigender Flanke übernommen werden.

Die Bausteine zeigen über den **Done-Ausgang** (oder einen sinngemäßen Ausgang) entweder die Gültigkeit der Ausgänge (z.B. MC_ReadStatus) oder den Abschluss der Bewegung (z.B. MC_MoveAbsolute) an.

Ein bewegungserzeugender Baustein, der von einem anderen unterbrochen wird, setzt seinen CommandAborted-Ausgang, um dies anzuzeigen.

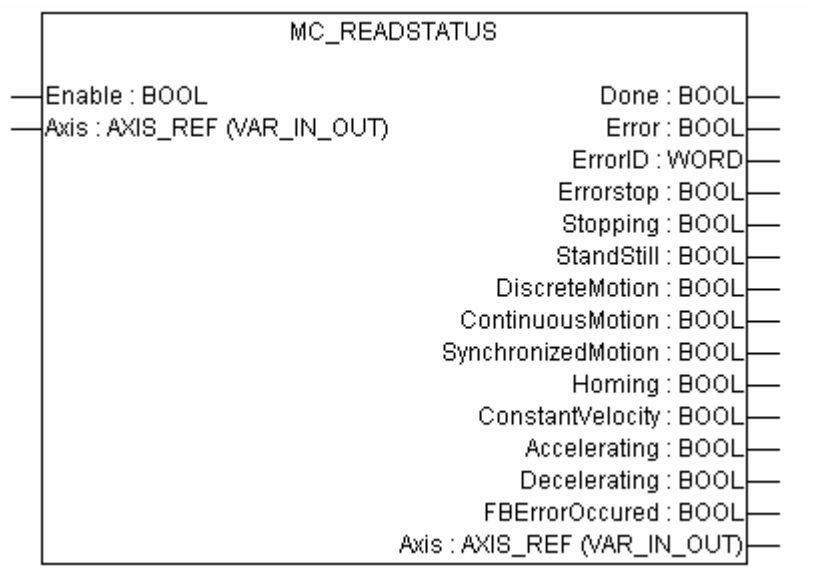
Die Ausgänge der Execute-gestarteten Bausteine bleiben – nach dem Setzen ihres Done-Ausgangs - so lange ungeändert, wie der Execute-Eingang gesetzt ist. Durch eine fallende Flanke werden sie gelöscht. Wurde noch vor Beendigung eine fallende Flanke detektiert, so werden die Ausgänge für einen Zyklus gesetzt und im nächsten gelöscht.

Alle bewegungserzeugenden Bausteine verlangen, dass in der entsprechenden Achse die Reglerfreigabe erteilt, die Bremse gelöst ist. Ansonsten melden Sie einen Fehler.

5.3 Bausteine zur Bewegungssteuerung einzelner Achsen

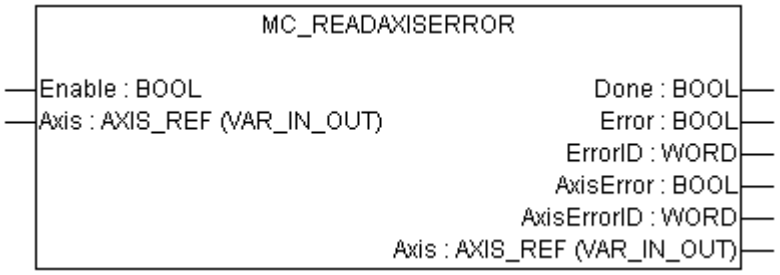
MC_ReadStatus

Dieser Baustein der SM_PLCopen.lib liefert einige ausgewählte Zustände einer Achse.



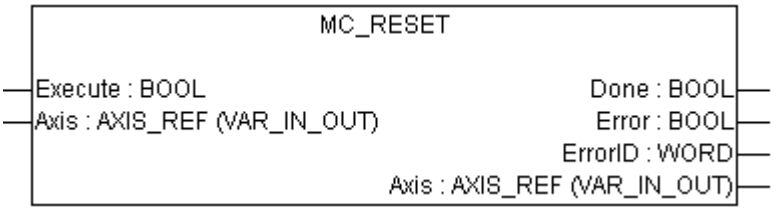
MC_ReadAxisError

Dieser Baustein der SM_PLCopen.lib gibt allgemeine Fehler, die am Antrieb aufgetreten sind, zurück.



MC_Reset

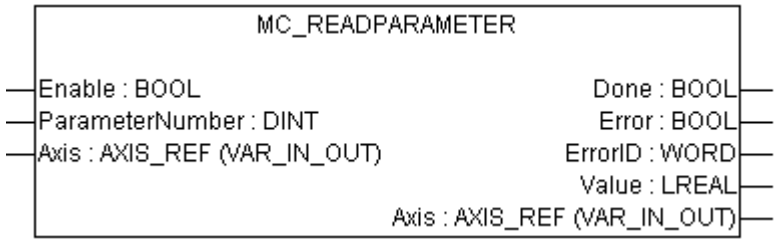
Dieser Baustein der SM_PLCopen.lib setzt den Achs-Status (SMC_AXIS_STATE) von error_stop zurürck auf standstill.

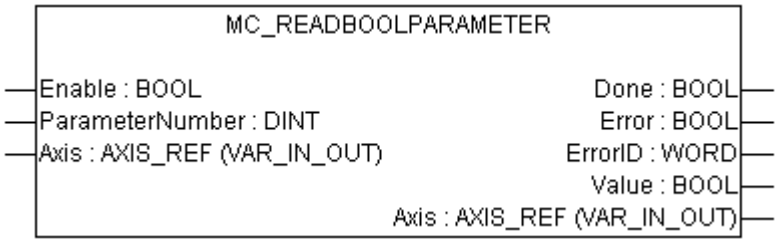


MC_ReadParameter,
MC_ReadBoolParameter

Mit diesen Bausteinen der SM_PLCopen.lib lassen sich einige Standardparameter aus der Antriebsstruktur lesen. Deren Nummern sind teilweise durch PLCopen spezifiziert, teilweise von 3S – Smart Software Solutions GmbH festgelegt und im Softmotion Drive Interface beschrieben.

Um antriebsherstellerspezifische Daten aus dem Antrieb heraus zu lesen, kann man diese Bausteine ebenso verwenden. Ein zur jeweiligen Antriebsbibliothek (XXXDrive.lib) gehörendes Dokument beschreibt die Kodierung der antriebsspezifischen Parameternummern.

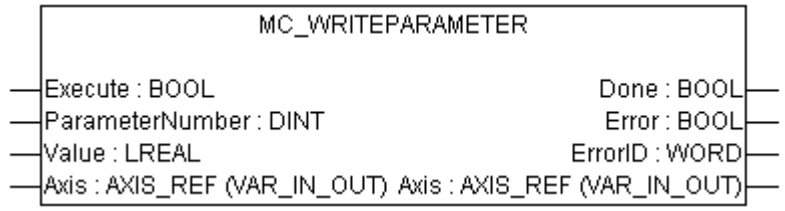


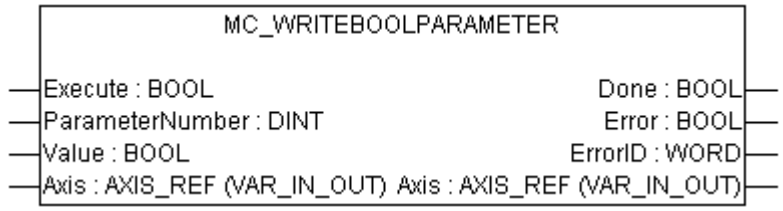


MC_WriteParameter,
MC_WriteBoolParameter

Mit diesen Bausteinen lassen sich einige Standardparameter der Antriebsstruktur setzen. Deren Nummern sind teilweise durch PLCopen spezifiziert, teilweise von 3S – Smart Software Solutions GmbH festgelegt und im Softmotion Drive Interface beschrieben

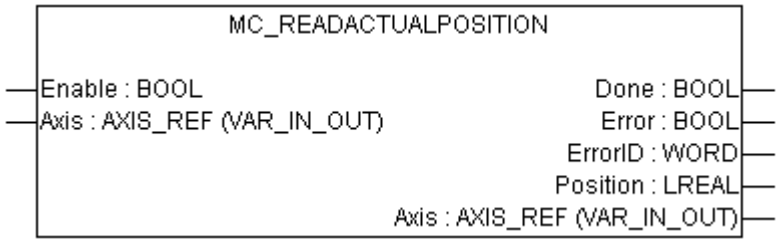
Um antriebsherstellerspezifische Daten an den Antrieb zu senden, kann man diese Bausteine ebenso verwenden, Ein zur jeweiligen Antriebsbibliothek (XXXDrive.lib) gehörendes Dokument beschreibt die Kodierung der antriebsspezifischen Parameternummern.





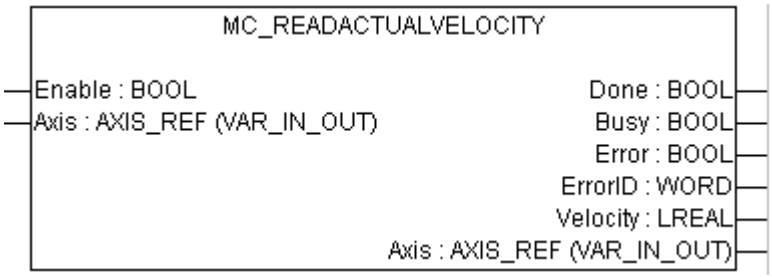
MC_ReadActualPosition

Dieser Baustein liefert die momentane Position des Antriebs zurück.



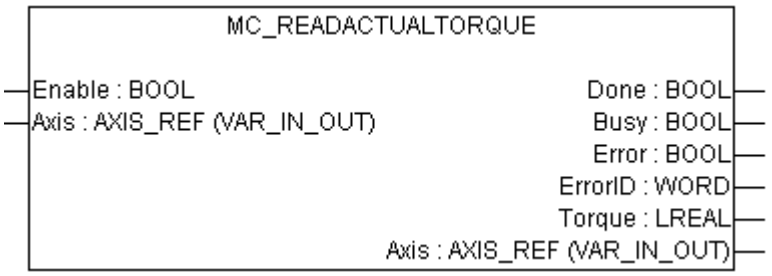
MC_ReadActualVelocity

Dieser Baustein liefert die momentane Geschwindigkeit des Antriebs zurück.



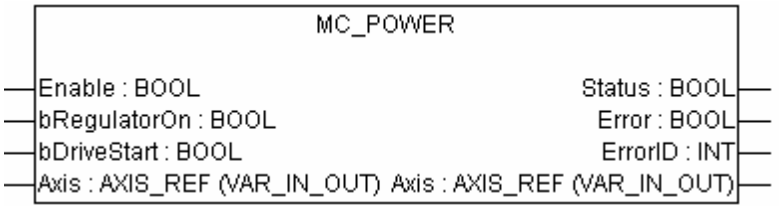
MC_ReadActualTorque

Dieser Baustein liefert das momentane Drehmoment bzw. die momentane Kraft des Antriebs zurück.



MC_Power

Dieser Baustein der SM_PLCopen.lib kontrolliert die Leistungszuschaltung und den Bremsenzustand des Antriebs. Wurde ein Antrieb nicht auf diesem Weg eingeschaltet, keine Reglerfreigabe gesetzt oder die Bremse nicht gelöst, kann keine Bewegungssteuerung auf ihm erfolgen.



Eingänge des Bausteins:

Enable : BOOL (Default: FALSE)

Solange diese Variable TRUE ist, ist der Antrieb eingeschaltet.

bRegulatorOn : BOOL (Default: FALSE)

Schaltet die Regelung ein/aus.

bDriveStart : BOOL (Default: FALSE)

Setzt bzw. löst die Bremse im Antrieb.

Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Ausgänge des Bausteins:

Status : BOOL (Default: FALSE)

Zeigt an, ob der Antrieb augenblicklich in (TRUE) oder ohne Regelung (FALSE) ist.

Error : BOOL (Default: FALSE)

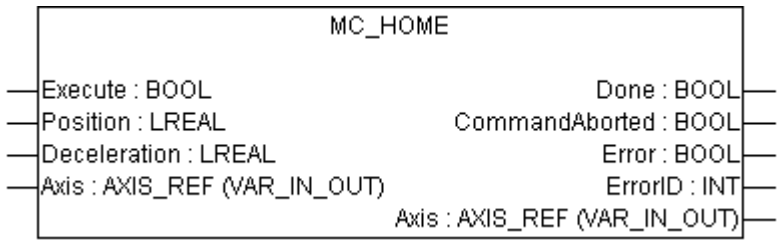
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT);

Fehlernummer

MC_Home

Dieser Baustein der SM_PLCopen.lib stößt eine Referenzfahrt im Antrieb an. Diese ist antriebsherstellerspezifisch und wird lediglich über das DriveInterface angestossen. Meldet der Antrieb, dass sie erfolgreich beendet wurde, wird der Ausgang 'Done' gesetzt.



Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Die Referenzfahrt des Antriebs soll bei steigender Flanke gestartet werden.
Position : REAL

Angabe der absoluten Position zum Zeitpunkt des Referenzsignals.

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

Bei TRUE ist die Referenzfahrt abgeschlossen und der Antrieb wieder im Stillstand.

CommandAborted : BOOL (Default: FALSE)

Bei TRUE wurde der Befehl durch einen anderen abgebrochen.

Error : BOOL (Default: FALSE)

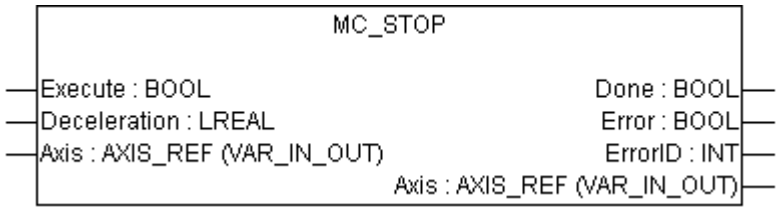
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

MC_Stop

Dieser Baustein der SM_PLCopen.lib bremst die Achse auf Geschwindigkeit 0. Dieser Baustein kann nicht unterbrochen werden und blockiert die Achse so lange wie der Eingang Execute gesetzt und die Achse noch nicht vollständig abgebremst ist.



Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv, d.h. beginnt den Bremsvorgang.

Deceleration : REAL

Bremswert [u/s^2]

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass der Antrieb steht

Error : BOOL (Default: FALSE)

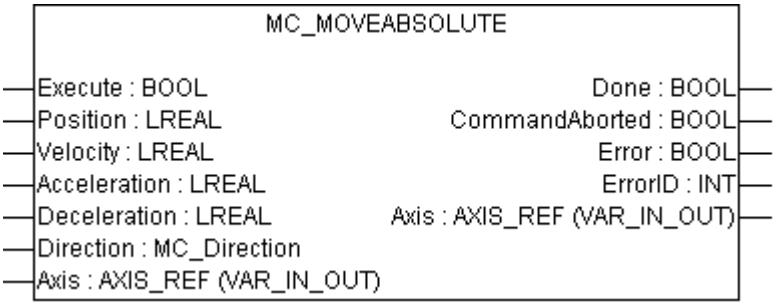
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

MC_MoveAbsolute

Dieser Baustein der SM_PLCopen.lib bewegt die Achse an eine absolute Position entsprechend der vorgegebenen Werte von Geschwindigkeit, Bremsung und Beschleunigung. Bei einer Linearachse hat die Angabe der Richtung keine Bedeutung; bei Rotationsachsen bestimmt sie den Drehsinn.



Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv

Position : REAL

Zielposition für die Bewegung (techn. Einheit [u])

Velocity : REAL

Wert der Soll-Geschwindigkeit, die nicht zwingend erreicht werden muss [u/s].

Acceleration : REAL

Gewünschte Beschleuningung [u/s^2]

Deceleration : REAL

Gewünschter Bremswert [u/s^2]

nDirection : MC_Direction (Default: shortest)

Dieser Enumerationswert gibt die gewünschte Richtung an; nur relevant für Rotationsachsen; siehe SM_DriveBasic.lib. Zulässige Werte sind: current (momentane Bewegungsrichtung), positive, negative, shortest (von der aktuellen Position aus gesehene kürzeste Entfernung), fastest (Richtung, so dass Bewegung schnellstmöglich beendet wird).

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass die gewünschte Position erreicht wurde.

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen; Ausnahme: MoveSuperImposed.

Error : **BOOL** (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

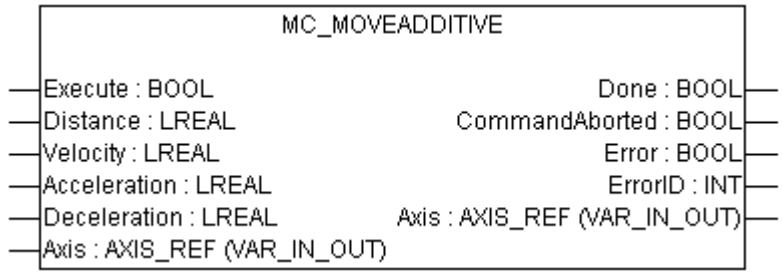
ErrorID : SMC_Error (INT)

Fehlernummer

MC_MoveAdditive

Dieser Baustein der SM_PLCopen.lib hat zweierlei Wirkungsweisen abhängig von dem Status, in welchem sich die Achse befindet:

1. discrete_motion:
   Zur Zielposition des gerade auf der Achse arbeitenden Bausteins wird die Distanz aufaddiert, und die neue Zielposition entsprechend der Vorgaben angefahren.
2. continuous_motion oder standstill:
   Die Distanz wird gemessen von der momentanen Position entsprechend der Vorgaben zurückgelegt.



Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv.

Distance : REAL

Relative Strecke für die Bewegung (techn. Einheit [u]).

Velocity : REAL

Wert der Soll-Geschwindigkeit, die nicht zwingend erreicht werden muss, [u/s].

Acceleration : REAL

Gewünschte Beschleuningung [u/s^2]

Deceleration : REAL

Gewünschter Bremswert [u/s^2]

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass die gewünschte Distanz erreicht wurde.

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen; Ausnahme: MoveSuperImposed .

Error : BOOL (Default: FALSE)

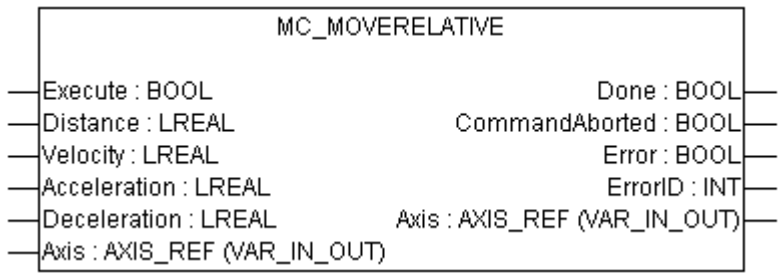
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

MC_MoveRelative

Dieser Baustein der SM_PLCopen.lib bewegt die Achse um eine relative Distanz entsprechend der vorgegebenen Werte von Geschwindigkeit, Beschleunigung und Bremsung. Die Distanz kann dabei positiv oder negativ sein.



Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv.

Distance : REAL

Relative Strecke für die Bewegung (techn. Einheit [u]).

Velocity : REAL

Wert der Soll-Geschwindigkeit, die nicht zwingend erreicht werden muss, [u/s].

Acceleration : REAL

Gewünschte Beschleunigung [u/s^2].

Deceleration : REAL

Gewünschter Bremswert [u/s^2].

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass die gewünschte Distanz erreicht wurde.

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen; Ausnahme: MoveSuperImposed.

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

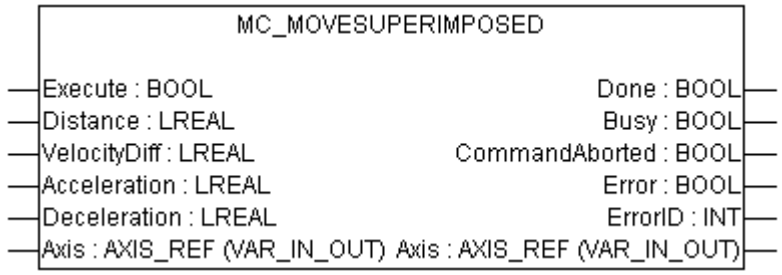
Fehlernummer

MC_MoveSuperImposed

Dieser Baustein der SM_PLCopen.lib führt ggfs. zusätzlich zur gerade stattfindenden eine Bewegung aus, die die Achse eine definierte Distanz zurücklegen lässt. Die angegebenen Werte von Geschwindigkeit, Beschleunigung und Bremsung sind dabei relativ zu sehen, also unabhängig von der darunter liegenden Bewegung.

Der ursprünglich aktive Baustein wird von MC_MoveSuperImposed nicht unterbrochen. Wird der ursprünglich aktive Baustein von einem weiteren unterbrochen, während MC_MoveSuperImposed aktiv ist, wird die Bewegung von MC_MoveSuperImposed abgebrochen.

Man beachte, dass der Aufruf von MC_MoveSuperImposed nach dem des Bausteins, der die darunter liegende Bewegung erzeugt, erfolgen muss!



Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv.

Distance : REAL

Relative Strecke für die Bewegung (techn. Einheit [u]).

VelocityDiff : REAL

Maximale Differenz der Geschwindigkeit zu der momentanen Bewegung, die aber nicht zwingend erreicht werden muss, [u/s].

Acceleration : REAL

Gewünschte Beschleunigung, [u/s^2].

Deceleration : REAL

Gewünschter Bremswert [u/s^2].

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass die gewünschte Distanz zurückgelegt wurde.

Busy : BOOL (Default: FALSE)

TRUE zeigt an, dass die zusätzlich gestartete Bewegung augenblicklich ausgeführt wird.

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen.

Error : **BOOL** (Default: FALSE)

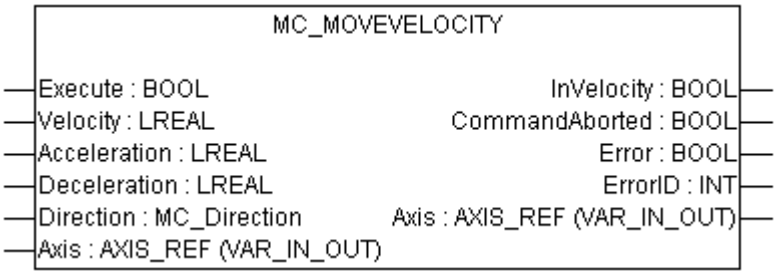
TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

MC_MoveVelocity

Dieser Baustein der SM_PLCopen.lib bewirkt eine endlose Bewegung der Achse mit vorgegebener Geschwindigkeit. Um diese Geschwindigkeit zu erreichen, hält sich MC_MoveVelocity an die programmierten Werte von Beschleunigung und Bremsung. Die Sollgeschwindigkeit ist immer positiv. Die Eingangsvariable Direction legt die Richtung fest.



Ein-/Ausgang (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv.

Velocity : REAL

Wert der maximalen Geschwindigkeit, die jedoch nicht zwingend erreicht werden muss, [u/s].

Acceleration : REAL

Gewünschte Beschleunigung, [u/s^2].

Deceleration : REAL

Gewünschter Bremswert [u/s^2].

Direction : MC_Direction (Default: current)

Dieser Enumerationswert gibt die gewünschte Richtung an; siehe SM_DriveBasic.lib. Zulässige Werte: current, positive, negative.

Ausgänge des Bausteins:

InVelocity : BOOL (Default: FALSE)

TRUE zeigt an, dass die vorgegebene Geschwindigkeit erreicht ist.

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen; Ausnahme: MoveSuperImposed.

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

MC_PositionProfile

Dieser Baustein der SM_PLCopen.lib fährt ein vorgegebenes Positionsprofil ab. Dazu muss eine Variable der Struktur MC_TP_REF definiert und befüllt werden.

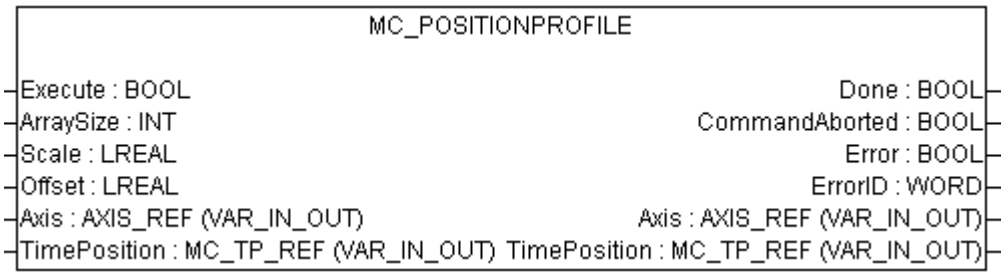
MC_TP_REF enthält folgende Variablen:

Variable	Typ	Initwert	Beschreibung
Number_of_pairs	INT	0	Anzahl Profil-Punkte
IsAbsolute	BOOL	TRUE	Positionen absolut oder relativ
MC_TP_Array	ARRAY[1..100] OF SMC_TP		Punkte

SMC_TP enthält folgende Variablen:

Variable	Typ	Initwert	Beschreibung
delta_time	TIME	t#0s	Zeit zwischen Erreichen des vorigen und dieses Punktes
position	REAL	0	(Absolut-/Relativ-)Position des Punktes

Der Baustein legt durch die vorgegebenen Positionen eine zweifach stetig differenzierbare Kurve aus kubischen Polynomen.



Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Axis : AXIS_REF

Hier wird die Struktur übergeben, die im Drive Interface (SM_DriveBasic.lib) mit den Daten der Achse gefüllt wurde.

TimePosition : MC_TP_REF

Angaben zu Zeit- und Positionswerten, s.o.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei einer steigenden Flanke wird der Baustein aktiv

ArraySize : INT

Grösse des Arrays (max. 1..100)

Scale : REAL (Default: 1)

Allgemeiner Skalierungsfaktor für das Profil

Offset : REAL

Allgemeiner Offset für das Profil [u]

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass die gewünschte Distanz zurückgelegt wurde

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen; Ausnahme: MoveSuperImposed

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

MC_VelocityProfile

Dieser Baustein der SM_PLCopen.lib funktioniert analog zum Baustein MC_PositionProfile. Allerdings werden in der Eingangsvariablen der Struktur MC_TV_REF hier die Punkte über ihre Geschwindigkeiten festgelegt.

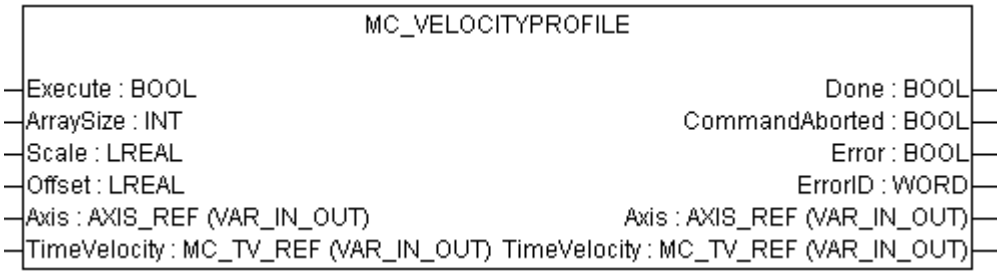
MC_TV_REF enthält folgende Variablen:

Variable	Typ	Initwert	Beschreibung
Number_of_pairs	INT	0	Anzahl Profil-Punkte
IsAbsolute	BOOL	TRUE	Positionen absolut oder relativ
MC_TV_REF	ARRAY[1..100] OF MC_TV_REF		Punkte

SMC_TV enthält folgende Variablen:

Variable	Typ	Initwert	Beschreibung
delta_time	TIME	t#0s	Zeit zwischen Erreichen des vorigen und dieses Punktes
velocity	REAL	0	(Absolut-/Relativ-)Geschwindigkeit des Punktes

Der Baustein legt durch die vorgegebenen Punkte eine stetig differenzierbare Kurve aus Parabeln. Die Position der Achse berechnet sich aus der Position am Start und dem Integral über die Geschwindigkeit.



MC_AccelerationProfile

Dieser Baustein der SM_PLCopen.lib funktioniert analog zum Baustein MC_PositionProfile. Allerdings werden in der Eingangsvariablen der Struktur MC_TA_REF hier die Punkte über ihre Beschleunigungen festgelegt.

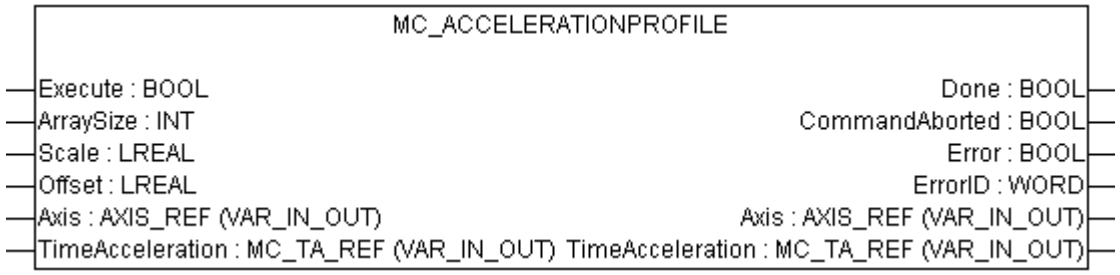
MC_TA_REF enthält folgende Variablen:

Variable	Typ	Initwert	Beschreibung
Number_of_pairs	INT	0	Anzahl Profil-Punkte
IsAbsolute	BOOL	TRUE	Positionen absolut oder relativ
MC_TA_Array	ARRAY[1..100] OF SMC_TA		Punkte

SMC_TA enthält folgende Variablen:

Variable	Typ	Initwert	Beschreibung
delta_time	TIME	t#0s	Zeit zwischen Erreichen des vorigen und dieses Punktes
acceleration	REAL	0	(Absolut-/Relativ-)Beschleunigung des Punktes

Der Baustein legt durch die vorgegebenen Punkte eine stetige Kurve aus Geraden. Die Geschwindigkeit der Kurve ergibt sich aus der Geschwindigkeit am Start des Profils und dem Integral über die Beschleunigung. Die Position der Achse berechnet sich aus der Position am Start und dem Integral über die Geschwindigkeit.

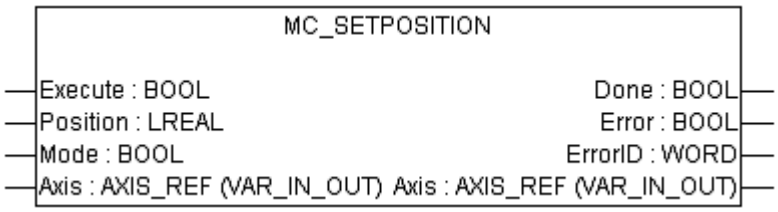


MC_SetPosition

Dieser Baustein verschiebt den Nullpunkt der Achse, so dass:

- im Absolut-Modus (Mode = FALSE; Default) der über den Eingang *Position* vorgegebene Wert die momentane Soll-Position wird, bzw.

- im Relativ-Modus (Mode = TRUE) die aktuelle Soll-Position um den Wert *Position* verschoben wird.



Grundsätzlich kann der Baustein zu jedem Zeitpunkt aufgerufen werden. Man beachte aber, dass bei einer bahngesteuerten Bewegung, wenn die Sollpositionen dem Baustein direkt z.B. über SMC_ControlAxisByPos zugeführt werden, ein Sollpositions-Sprung entstehen kann.

MC_TouchProbe

Dieser Baustein dient dazu, über einen schnellen Eingang die Position des Antriebs hochgenau zu erfassen. Da dies i.d.R. schneller als im normalen SPS-Zyklus geschehen muss, wird häufig entweder

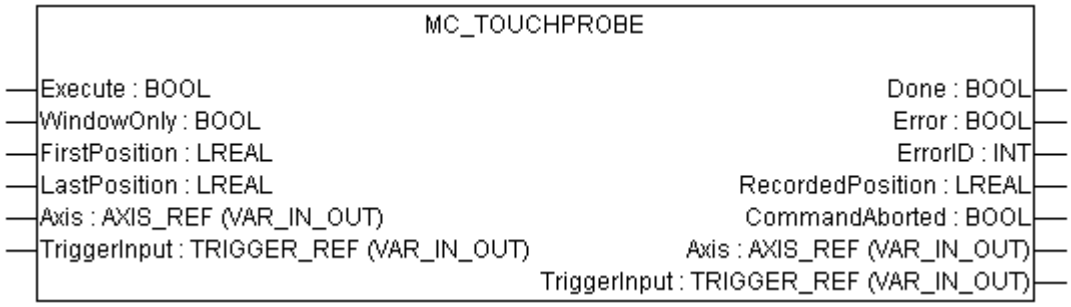
der Antrieb mit dieser Funktion beaufschlagt, oder sie wird – vom SPS-Takt unabhängig - über Interrupts o.ä. durchgeführt.

Der Eingang TriggerInput ist vom Typ TRIGGER_REF und beschreibt den Trigger-Eingang näher:

Variable	Typ	Initwert	Beschreibung
bFastLatching	BOOL	TRUE	schnelles Latchen über DriveInterface (TRUE) oder Latchen im SPS-Takt (FALSE)
iTriggerNumber	INT	-1	nur für bFastLatching = TRUE: Triggernummer; abhängig von DriveInterface
bInput	BOOL	FALSE	nur für bFastLatching = TRUE: Eingangssignal; TRUE löst Latching aus.
bActive	BOOL	FALSE	interne Variable

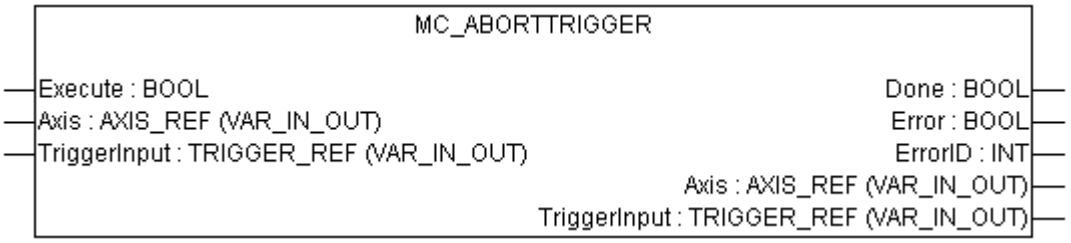
Die Fenster-Funktion, die über WindowOnly, FirstPosition, LastPosition aktiviert und definiert, ist abhängig davon, ob sie vom DriveInterface unterstützt wird und liefert im negativen Fall einen Fehler.

Der Baustein ist unabhängig vom Zustand der Achse und so lange aktiv, bis eine Position gelatcht, bzw. der Vorgang durch MC_AbortTrigger abgebrochen wurde.



MC_AbortTrigger

Dieser Baustein bricht ein auf dem TriggerInput stattfindendes Latching ab.



5.4 Bausteine zur synchronisierten Bewegungssteuerung (multi-axis)

MC_CamTableSelect

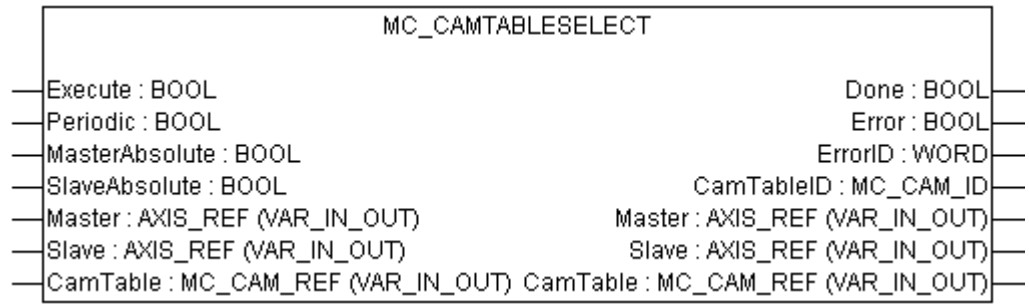
Mit diesem Baustein der SM_PLCopen.lib wird eine Kurvenscheibe ausgesucht, Master- und Slaveachse bestimmt und einige Voreinstellungen getroffen. Das als Ausgang verfügbare Objekt CamTableID soll später dem Kurvenscheiben-Baustein MC_CamIn zugeführt werden.

Die Masterachse kann virtuell sein, d.h. sie muss nicht physikalisch vorhanden sein. Von ihr werden, steht die Variable bRegulatorOn auf TRUE die Sollwerte, ansonsten die Istwerte verwendet.

Die zu fahrende Kurvenscheibe kann entweder von Hand in ein Objekt der Struktur MC_CAM_REF programmiert oder mit Hilfe des in CoDeSys integrierten Kurvenscheibeneditors (siehe Dokument "SoftMotion CAM-Editor") erstellt werden.

Mit der Variable Periodic wird entschieden, ob die Kurvenscheibe am Ende wieder von vorn begonnen werden soll, oder nicht.

Mit MasterAbsolute und SlaveAbsolute entscheidet man, ob die Kurvenscheiben-Abbildung der Masterachse auf die Slaveachse sich auf Absolutwerte oder Inkremente beziehen soll.



Man beachte auch Kapitel 4.5.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Master : AXIS_REF

Masterachse.

Slave : AXIS_REF

Slaveachse.

CamTable : MC_CAM_REF

Beschreibung der Kurvenscheibe.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei steigender Flanke wählt der Baustein eine neue Kurvenscheibe.

Periodic : BOOL (Default: TRUE)

periodische/ nicht-periodische Kurvenscheibe.

MasterAbsolute : BOOL (Default: TRUE)

Kurvenscheibe bezieht sich auf absolute/ relative (bzgl. Position bei steigender Flanke im Execute von CAMIn) Master-Position.

SlaveAbsolute : BOOL (Default: TRUE)

Kurvenscheibe bezieht sich auf absolute/ relative (bzgl. Position bei steigender Flanke im Execute von CAMIn) Slave-Position.

Ausgänge des Bausteins:

Done : BOOL (Default: FALSE)

TRUE zeigt an, dass die gewünschte Distanz zurückgelegt wurde

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

CAMTableID : MC_CAM_ID

Ausgang, der die Kurvenscheibe beschreibt. Dient als Eingang für gleichnamigen Eingang in MC_CamIn.

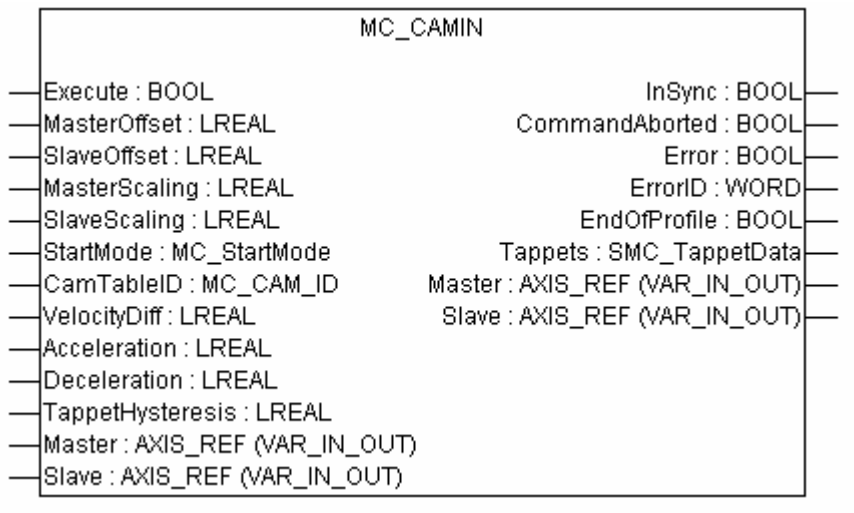
MC_CamIn

Dieser Baustein der SM_PLCopen.lib setzt eine in MC_CAMTABLESELECT gewählte Kurvenscheibe um.

Zu den Offsets und Skalierungen kann noch der Startmodus definiert werden. Man beachte, dass die Modi ramp_in, ramp_in_pos und ramp_in_neg, welche bewirken, dass beim Start, falls Istwert des Slave und Kurvenscheibenstartwert nicht übereinstimmen, der Sollwert des Slave stetig auf die Kurvenscheibe zugeführt wird, die Festlegung der Parameter VelocityDiff, Acceleration und Deceleration voraussetzen.

Man beachte auch Kapitel 4.5.

Dieser Baustein enthält eine zusätzliche Funktionalität. Er detektiert Nocken und kann diese über den Ausgang Tappets an einen oder mehrere SMC_GetTappetValue-Funktionsbausteine übergeben (siehe SMC_GetTappetValue). Man beachte, dass der CamIn-Baustein pro Zyklus höchstens drei Nocken gleichzeitig erfassen kann. Der Baustein SMC_CAMRegister arbeitet ohne diese Beschränkung.



Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Master : AXIS_REF

Masterachse.

Slave : AXIS_REF

Slaveachse.

Eingänge des Bausteins:

Execute : BOOL (Default: FALSE)

Bei steigender Flanke beginnt der Baustein mit der Bewegung.

MasterOffset : LREAL (Default: 0)

zusätzlicher Offset auf Master-Position.

SlaveOffset : LREAL (Default: 0)

zusätzlicher Offset auf Slave-Position.

StartMode : MC_StartMode (absolute/relative/ramp_in/ramp_in_pos/ramp_in_neg) (Default: absolute)

Kurvenscheibe wird endweder relativ (relative) zur Ist-Position bezogen, oder absolut dazu ohne (absolute) oder mit langsamem Einrampen auf kürzestem Weg (ramp_in), in positive (ramp_in_pos) oder negative (ramp_in_neg) Richtung.

CamTableID : MC_CAM_ID

Ausgang von MC_CamTableSelect

Velocity, Acceleration, Deceleration: LREAL (Default: 0)

zusätzliche Geschwindigkeit, Beschleunigung, Bremsung für ramp_in-Modus.

TappetHysteresis: LREAL (Default: 0)

Breite des Hysteresebands um die Nocken.

Ausgänge des Bausteins:

InSync : BOOL (Default: FALSE)

TRUE zeigt an, dass die Bewegung auf der Kurvenscheibe liegt.

CommandAborted : BOOL (Default: FALSE)

Die gestartete Bewegung wurde durch einen anderen Funktionsblock, der auf dieselbe Achse wirkt, unterbrochen; Ausnahme: MoveSuperImposed

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

EndOfProfile : BOOL

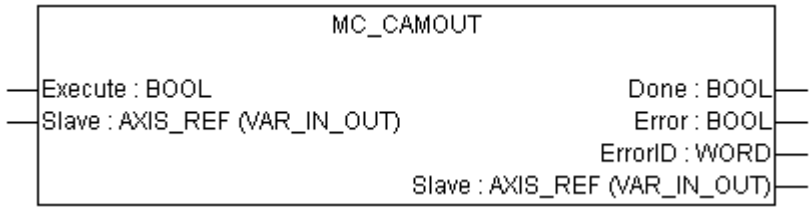
Zeigt das Ende einer Kurvenscheibe an. Bei periodischen Kurvenscheiben wird dieser Ausgang gepulst.

Tappets : SMC_TappetData

Nocken-Ausgang. Die einzelnen Nockenschalt-Stellungen werden endgültig vom Baustein SMC_GetTappetValue ausgewertet.

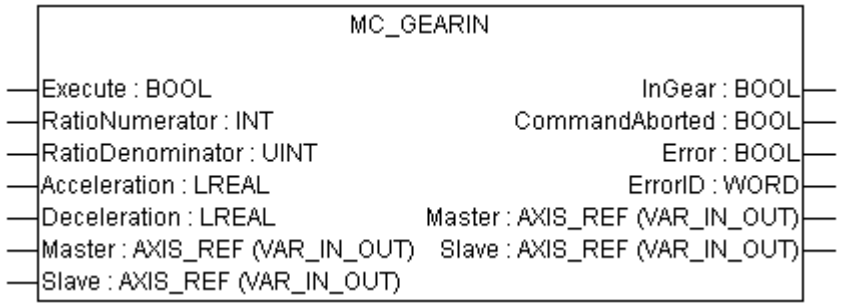
MC_CamOut

Dieser Baustein der SM_PLCopen.lib entkoppelt den Slave-Antrieb vom Master, und fährt den Slave mit der momentanen Geschwindigkeit weiter.



MC_GearIn

Dieser Baustein der SM_PLCopen.lib koppelt die Slave-Achse an die Master-Achse. Die Geschwindigkeit der Slave-Achse ist dabei ein f-faches der Master-Achse. Dieser Faktor f berechnet sich aus dem Quotienten der Eingabeparameter RatioNumerator und RatioDenominator.

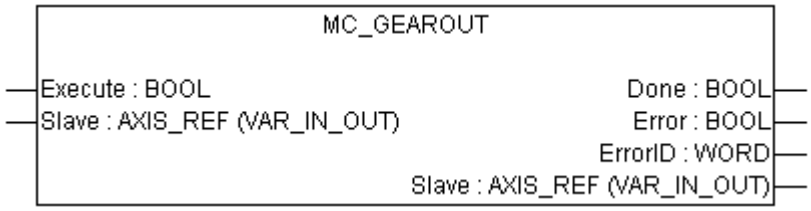


Der Baustein beschleunigt bzw. bremst die Slave-Achse so lange, bis ihre Geschwindigkeit zu der der Master-Achse das gewünschte Verhältnis hat, wobei er die Werte Acceleration und Deceleration benutzt. Ist dies erledigt, leitet sich die Geschwindigkeit der Slave-Achse von der Master-Achse ab.

Steht die Variable bRegulatorOn der Master-Achse auf TRUE werden ihre Geschwindigkeits-Sollwerte, ansonsten die Istwerte verwendet.

MC_GearOut

Dieser Baustein der SM_PLCopen.lib entkoppelt den Slave-Antrieb vom Master, und fährt den Slave mit der momentanen Geschwindigkeit weiter.

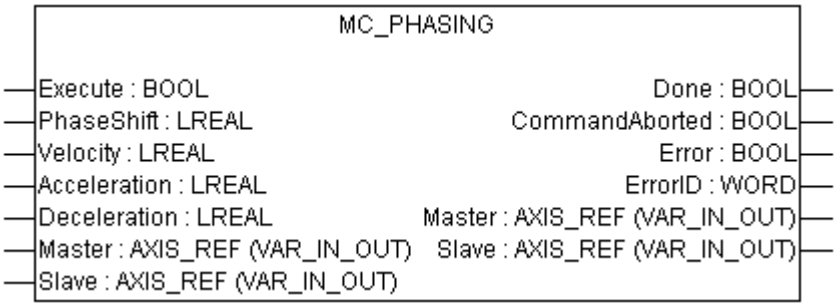


MC_Phasing

Dieser Baustein der SM_PLCopen.lib sorgt für eine konstanten Abstand zwischen der Master- und Slave-Achse. In diesem Fall haben Master und Slave natürlich gleiche Geschwindigkeit und Beschleunigung.

Um dies zu erreichen, erhält die Slave-Achse mittels Beschleunigung oder Bremsung die gleiche Geschwindigkeit wie die Master-Achse. Ist dies erreicht, wird auf der Master-Achse eine zusätzliche Bewegung (ähnlich MC_MoveSuperImposed) ausgeführt, die die gewünschte Phasenverschiebung bewirkt.

Der Baustein MC_Phasing bleibt so lange aktiv bis er von einem anderen Baustein unterbrochen wird.



5.5 Zusatzbausteine

SMC_GetCamSlaveSetPosition

Dieser Baustein berechnet die aktuelle Sollposition einer Achse (Slave), wenn diese mit einer Kurvenscheibe an die Bewegung einer anderen Achse (Master) gekoppelt wäre. Beide Achsen werden dabei aber nicht bewegt oder beeinflusst.

Zur Anwendung kommt dieser Baustein, wenn eine Slave-Achse vor dem Einkuppeln in eine Kurvenscheibe auf die dadurch entstehende Sollposition bewegt werden soll.

Der Baustein errechnet den entsprechenden Wert innerhalb eines Zyklus, weshalb ein Done-Ausgnag nicht nötig ist.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

CamTableID : MC_CAM_ID

Kurvenscheibe; Ausgang von MC_CamTableSelect.

Master : AXIS_REF

Master-Achse.

Slave : AXIS_REF

Achse, für die die Kurvenscheiben-Sollposition ausgerechnet wird.

Eingänge (VAR_IN) des Bausteins (nicht aufgeführte entsprechen denen bei MC_CamIn):

Enable : BOOL

Aktiviert den Baustein.

Ausgänge des Bausteins:

fStartPosition : LREAL

Berechnete Sollposition für den Slave.

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

SMC_CAMBounds

Dieser Baustein berechnet die maximalen Sollwerte von Position, Geschwindigkeit und Beschleunigung/Verzögerung des Slaves, wenn dieser über eine gegebene Kurvenscheibe absolut an einen Master gebunden ist, der mit vorgegebener maximaler Geschwindigkeit und Beschleunigung/Verzögerung bewegt wird.

Dieser Baustein ist u.a. dann besonders hilfreich, wenn eine Kurvenscheibe online erstellt bzw. verändert wird und man die Einhaltung von Maximalwerten vorab überprüfen will.

Eingänge (VAR_IN) des Bausteins:

bExecute: BOOL

Aktiviert die Berechnung.

dMasterVelMax : LREAL (Default: 1)

Maximale Geschwindigkeit (Absolutwert) des Masters [u/s].

dMasterAccMax : LREAL (Default: 0)

Maximale Beschleunigung/Verzögerung (Absolutwert) des Masters [u/s²].

dMasterScaling, dSlaveScaling : LREAL (Default: 1)

Master- bzw. Slave-Skalierungsfaktor, der bei der Kurvenscheibe verwendet wird.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

CAM: MC_CAM_REF

Kurvenscheibe

Ausgänge des Bausteins:

bDone : BOOL

TRUE zeigt an, dass die Berechnung abgeschlossen ist.

bError : BOOL

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

nErrorID : SMC_Error (INT)

Fehlernummer.

dMaxPos, dMinPos : LREAL

maximale bzw. minimaler Slave-Positionswert [u].

dMaxVel, dMinVel : LREAL

Maximaler bzw. minimaler (auch negativ) Slave-Geschwindigkeitswert [u/s].

dMaxAccDec, dMinAccDec : LREAL

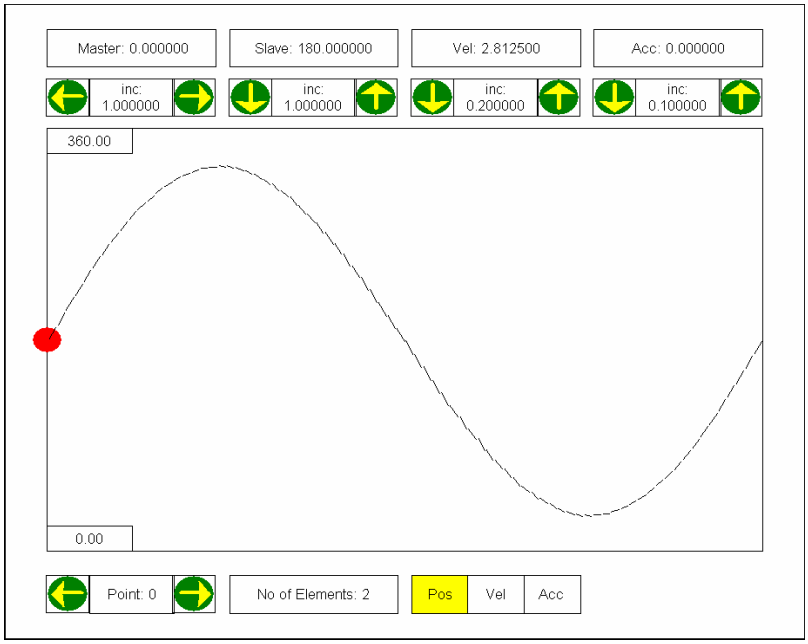
Maximaler bzw. minimaler (auch negativ) Slave-Beschleunigungs- bzw. Bremswert [u/s²].

SMC_CAMEditor,
SMC_CAMVisu

Mit diesen Bausteinen kann ein Online-Kurvenscheiben-Editor erzeugt werden.

SMC_CAMEditor muss in der SoftMotion-Task, während SMC_CAMVisu in einer langsameren, niedrigerprioren Task aufgerufen werden sollte.

Beide Bausteine sollten mit dem entsprechenden Visualisierungs-Template (SMC_CAMEditor) verknüpft sein, welches die eingegebene Kurvenscheibe darstellt und dem Anwender erlaubt, diese – auch bei laufender Kurvenscheibe - zu ändern.



Der rote Kreis markiert den aktuellen Kurvenscheibenpunkt. Dieser kann über die Pfeile unten links gewechselt werden. Über die Leiste unten rechts kann ausgewählt werden, ob Position, Geschwindigkeit oder Beschleunigung angezeigt werden soll. Mit den Pfeilen oben können die Master-, Slave-Position, Slave-Geschwindigkeit und –Beschleunigung um das eingestellte Inkrement verschoben werden.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins SMC_CAMEditor:

CAM : MC_CAM_REF

Kurvenscheibe, die dargestellt und verändert werden soll.

Eingänge des Bausteins SMC_CAMEditor:

Enable : BOOL (Default: FALSE)

Bei TRUE arbeitet der Baustein.

dYPeriod : LREAL

Slave-Periode (für periodische Kurvenscheiben).

bPeriodic : BOOL (Default: TRUE)

TRUE für periodische Kurvenscheibe, sonst FALSE.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins SMC_CAMEditor:

ce:SMC_CAMEditor

SMC_CAMEditor-Instanz.

SMC_CAMRegister

Dieser Baustein stellt ein Nockenschaltwerk dar. Er arbeitet – wie MC_CamIn – auf einer MC_CAM_REF-Struktur, wobei er die eigentliche Kurvenscheibeninformation negiert und daraus lediglich die Nocken-Information liest.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Master : AXIS_REF

Hier wird die Achs-Struktur übergeben, die die Nocken schalten soll

CamTable : MC_CAM_REF

Beschreibung einer (auch leeren) Kurvenscheibe, die die Beschreibung der Nocken enthält.

bTappet : ARRAY [1..MAX_NUM_TAPPETS] OF BOOL

Nocken-Bits.

Eingänge des Bausteins:

Enable : BOOL (Default: FALSE)

Bei TRUE beginnt der Baustein, die Nocken zu schalten

MasterOffset : REAL

Offset auf die Masterposition

MasterScaling : REAL (Default: 1)

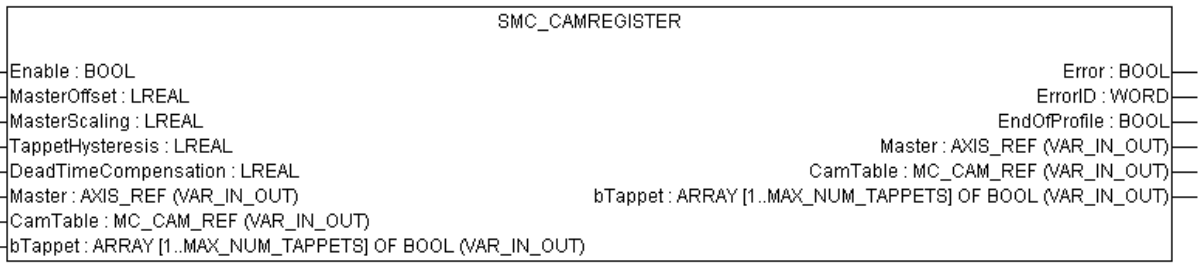
Allgemeiner Skalierungsfaktor für die Masterachse

TappetHysteresis : REAL

Hystereseumgebung um Nocken.

DeadTimeCompensation : REAL

Totzeit in sec. Durch lineare Extrapolation wird die zu erwartende Position der Masterachse vorausberechnet.



Ausgänge des Bausteins:

Error : BOOL (Default: FALSE)

TRUE zeigt an, dass im Funktionsblock ein Fehler aufgetreten ist.

ErrorID : SMC_Error (INT)

Fehlernummer

EndOfProfile : BOOL

Beim Übergang vom Profilende zum –anfang wird dieser Ausgang für einen Zyklus TRUE

SMC_GetTappetValue

Dieser Baustein wertet den Tappets-Ausgang des Bausteins MC_CamIn aus und gibt den aktuellen Status einer Nocke aus.

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Tappets : SMC_TappetData

Eingänge des Bausteins:

iID : INT

Gruppen-ID der auszuwertenden Nocke

bInitValue : BOOL

Initialwert der Nocke. Wird beim ersten Aufruf zugewiesen.

bSetInitValueAtReset : BOOL

TRUE: beim Neustart des CamIn-FBs wird der Wert der Nocke auf bInitValue gesetzt

FALSE: beim Neustart des CamIn-FBs bleibt der Nockenwert erhalten

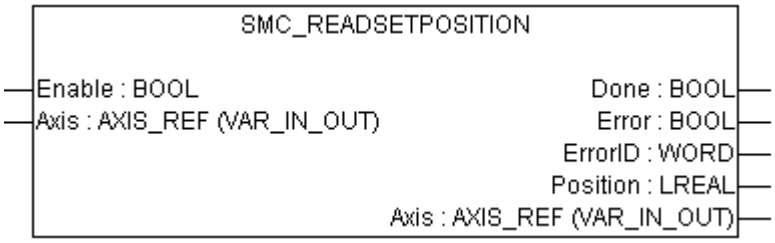
Ausgänge des Bausteins:

bTappet : BOOL (Default: FALSE)

Wert der Nocke

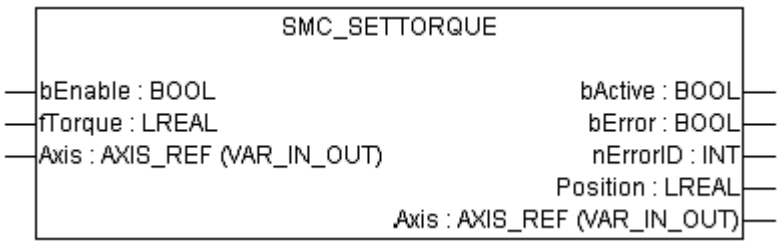
SMC_ReadSetPosition

Dieser Baustein liest die gegenwärtige Soll-Position des Antriebs aus.



SMC_SetTorque

Mit diesem Baustein kann, sofern der Antrieb im Regelungsmodus „Drehmoment“ steht, ein Solldrehmoment erzeugt werden



6 Die Bibliothek SM_CNC.lib

6.1 Überblick

Diese **Bausteinbibliothek** stellt Bausteine und Funktionen für ein mit SoftMotion realisiertes IEC-Programm zur Verfügung. Sie enthält die Bewegungsfunktionalität und wird in das IEC-Programm eingebunden. Sie umfasst unter anderem die unten aufgelisteten Bausteine, die sowohl die in den Editoren offline entworfenen Bewegungen ausführen, als auch die im IEC-Programm online geplante Bahnen realisieren können. Dazu greifen sie auf interne Strukturen zu, die die einzelnen Bahnobjekte beschreiben.

- SMC_NCDecoder — Entschlüsselung der im CNC-Editor programmierten Bahn zu Strukturobjekten
- SMC_ToolCorr — Bahnvorverarbeitung: Werkzeugradiuskorrektur
- SMC_AvoidLoop — Bahnvorverarbeitung: Schleifenvermeidung
- SMC_SmoothPath — Bahnvorverarbeitung: Eckverschleifung mit Splines
- SMC_RoundPath — Bahnvorverarbeitung: Eckverrundung mit Kreisbögen
- SMC_CheckVelocities — Überprüfung der Endgeschwindigkeiten der Segmente
- SMC_Interpolator — Umwandlung der entschlüsselten, ggfs. vorverarbeiteten Bahnobjekte in diskrete Punkte
- Hilfsfunktionen — zum Verschieben und Rotieren einer erstellten Bahn
- globale Variablen — Setzen einiger interner Parameter
- Strukturen SMC_POSINFO, SMC_GEOINFO, SMC_VECTOR3D und SMC_VECTOR6D zur Speicherung von Positionen, Bahnabschnitten und Vektoren
- Struktur SMC_OUTQUEUE zur Verwaltung von GEOINFO-Objekten in einer Liste definierter Größe

Die Bausteine und Strukturen werden in den folgenden Kapiteln beschrieben. Sehen Sie auch die Beispielprogramme in Kapitel 11.

Hinweis:	Man beachte, dass der CNC-Editor ein CNC-Programm auf zweierlei Arten übersetzen kann: als Programmvariable (SMC_CNC_REF), die den Decoder- und ggfs. Bahnvorverarbeitungsbausteine durchlaufen muss, und als OUTQUEUE, die direkt an den Interpolator übergeben werden kann.

6.2 Bausteine

6.2.1 SMC_NCDecoder

Der SMC_NCDecoder-Baustein (SM_CNC.lib) wandelt ein im CNC-Editor erstelltes CNC-Programm in eine Liste von SoftMotion-GEOINFO-Strukturobjekten um. Pro Zyklus wird eine Programmzeile entschlüsselt.

Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt die Entschlüsselung, wenn dieser Eingang eine steigende Flanke aufweist.

bAppend: BOOL

Steht dieser Eingang auf FALSE, so wird bei jedem Reset die Ausgabe-Datenqueue DataOut geleert. Bei TRUE werden die neuen Daten ans Ende der DataOut-Queue geschrieben.

bStepSuppress: BOOL

Steht dieser Eingang auf TRUE, so werden Zeilen des CNC-Programms, die mit ‚/' beginnen, ignoriert. Bei FALSE (default) werden diese trotzdem verarbeitet.

piStartPosition

Position, die der zu bewegende Punkt am Anfang der Bahn hat.

nSizeOutQueue: UDINT

Hier wird dem Baustein die Größe des Datenpuffers mitgeteilt, in den die Liste von GEOINFO-Strukturobjekten geschrieben wird. Dieser muss mindestens fünf mal so groß wie eine GEOINFO-Struktur. Ist dies nicht der Fall, so führt der SMC_NCDecoder keine Aktionen durch. Der Wert darf gesetzt und hernach nur während eines Resets verändert werden. Es empfiehlt sich, den eigentlichen Buffer wie unten beschrieben z.B. als `Example: Array of SMC_GeoInfo` anzulegen. Die entsprechende Buffergröße ergibt sich dann aus `sizeof(Example)`.

pbyBufferOutQueue: POINTER TO BYTE

Dieser Eingang muss auf das erste Byte des für die OUTQUEUE-Struktur angelegten Speicherbereichs zeigen. Dieser Speicherbereich muss mindestens so groß sein, wie in nSizeOutQueue angegeben. Typischerweise erfolgt das Belegen dieses Speichers im Deklarationsteil des IEC-Programms mittels eines Arrays of SMC_GeoInfo (z.B. `BUF: ARRAY[1..50] OF SMC_GEOINFO;` für einen Buffer, der 50 Bahnelemente aufnehmen kann). Auch dieser Wert darf gesetzt und später nur während eines Resets verändert werden.

Ein/Ausgang (VAR_IN_OUT) des Bausteins:

ncprog: SMC_CNC_REF

In dieser IN_OUT Variablen wird das CNC-Programm (Struktur SMC_CNC_REF der SM_DriveBasic.lib) übergeben. Dieses kann entweder vom IEC-Programm erzeugt, oder im CNC-Editor programmiert worden sein.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Abarbeitung des Programms abgeschlossen ist. Hernach führt der SMC_NCDecoder bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ ERROR (INT)

Dieser Enum-Ausgang beschreibt ggf. einen Fehler, der beim Dekodieren aufgefallen ist. Nach einem Fehler wird die Abarbeitung bis zu einem Reset gestoppt.

poqDataOut: POINTER TO SMC_OUTQUEUE

Dieser Ausgang zeigt auf eine SMC_OUTQUEUE-Struktur, die die entschlüsselten SMC_GEOINFO-Objekte verwaltet.

iStatus: SMC_DEC_STATUS (INT)

In dieser Enum-Variable wird der aktuelle Status des Bausteins ausgegeben. Dieser kann sein:

WAIT_PROG	0	Programm noch nicht gefunden
READ_WORD	1	Wort gelesen.
PROG_READ	2	Programmende erreicht.

iLineNumberDecoded: INT

In dieser Variablen steht die Zeilennummer (nicht die Satznummer) der zuletzt bearbeiteten Zeile.

GCodeText: SMC_GCODE_TEXT

Über diese Datenstruktur kommuniziert der Baustein zu einer Instanz von SMC_GCodeViewer und übergibt dieser eine textuelle Represäntation (G-Code) der aktuellen Zeile.

6.2.2 SMC_GCodeViewer

Dieser Baustein kann in Kombination mit SMC_NCDecoder verwendet werden. Er sammelt und speichert die textuelle Repräsentation (G-Code) der einzelnen GeoInfo-Objekte und gibt diese in Form eines ARRAY OF STRING wieder aus, welches z.B. über eine Tabelle in der Visualisierung ausgegeben werden kann. Über den Eingang, der typischerweise mit dem Interpolator-Ausgang *iActObjectSourceNo* verbunden ist, erhält der Baustein Information darüber, welche Zeile bereits abgearbeitet wurde und löscht diese. So kann ein Anzeigeelement des aktuellen G-Codes erzeugt werden.

Man beachte, dass die Ausgabe dieses Bausteins nicht mit dem G-Code übereinstimmt, der im Editor oder Textfile gespeichert ist, da hier beispielsweise keine Leerzeilen oder Kommentare übernommen werden. Evtl. verwendete Variablen sind ebenso bereits ersetzt.

Der Baustein wird im Task-Kontext des SMC_NCDecoder-Bausteins aufgerufen.

Ihm muss ein ausreichend großer Buffer übergeben werden. Die Anzahl der SMC_GCodeViewer_DATA-Objekte muss mindestens so groß sein, wie die Summe aller SMC_GeoInfo-Objekte, die in den Buffern der Bausteine SMC_NCDecoder und der Bahnvorverarbeitungs-Bausteine angelegt sind.

Eingänge des Bausteins:

bEnable: BOOL

Der Baustein ist aktiv, wenn dieser Eingang auf TRUE steht.

iActObjectSourceNo: INT

Zeilennummer, die der Interpolator aktuell bearbeitet.

uiNumberOfBufferEntries: UINT

Größe des in pBuffer übergebenen Arrays.

pBuffer: POINTER TO SMC_GCODEVIEWER_DATA;

Adresse des Arrays, welches als Buffer für den Baustein reserviert wurde.

Ein/Ausgang (VAR_IN_OUT) des Bausteins:

GCodeText: SMC_GCODE_TEXT

Datenquelle. Wird mit dem gleichnamigen Ausgang des SMC_NCDecoder-Bausteins verknüpft.

Ausgänge des Bausteins:

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

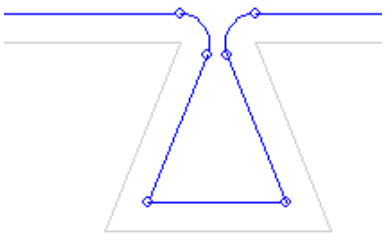
wErrorID: SMC_ ERROR (INT)

Dieser Enum-Ausgang beschreibt ggf. einen Fehler, der aufgetreten ist. Nach einem Fehler wird die Aufbereitung der Daten gestoppt, bis der Baustein durch ein neues bEnable frisch gestartet wird.

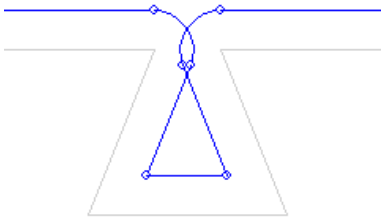
asGCode: ARRAY[0..c_uiLines] OF STRING

Textzeilen, die das aktuelle Segment des G-Codes enthalten. Im ersten Element steht die Zeile, die gerade vom Interpolator verfahren wird.

6.2.3 SMC_ToolCorr



Der SMC_ToolCorr-Baustein (SM_CNC.lib) dient der **Bahnvorverarbeitung**: Er erzeugt aus einer vorgegebenen Bahn eine versetzte Bahn, bei der jeder Punkt jedes Wegobjekts zu seinem Original und dessen direkten Nachbarn einen vorgebbaren Abstand hat (Werkzeugradiuskorrektur) . Die so erzeugte Bahn gewährleistet also, dass jeder seiner Punkte zur Originalbahn einen festen Abstand hat. Eine typische Anwendung ist das Fräsen einer programmierten Kontur mit einem Fräsbohrer bestimmter Dicke. Der Fräsbohrer muss dann eine entsprechend versetzte Bahn fahren – die mit Hilfe des Bausteins SMC_ToolCorr erzeugt werden kann - um den Radius des Bohrers zu kompensieren:



Folgende Einschränkung sei dabei beachtet: Ist die Kontur und der Bohrradius so gewählt, dass innerhalb der versetzten Bahn ein Schnitt mit sich selbst stattfindet – d.h. die gewollte Kontur wird beim Befahren der versetzten Bahn zerstört -, so wird darauf keine Rücksicht genommen (siehe Skizze rechts). Um dies zu vermeiden, verwende man den Baustein SMC_AvoidLoop.

Der Baustein *SMC_ToolCorr* arbeitet folgendermaßen:

Alle in der Eingangs-OUTQUEUE-Struktur befindlichen SMC_GEOINFO-Objekte werden nacheinander abgegangen. Wenn in einem dieser Objekte Bit1 (Bit2) der Variablen Intern_Mark gesetzt ist, so wird ab dort begonnen, die Bahn in Fahrrichtung nach links (rechts) um den aktuellen Werkzeugradius zu versetzen. Damit eine stetige Bahn entsteht, wird ein Positionierungsobjekt (MoveType = 100) eingefügt, bzw. wenn dem Objekt ein solches Positionierungsobjekt vorausgeht, dieses direkt auf den Startpunkt der versetzten Bahn verschoben. Jedes weitere Objekt wird dann ebenfalls versetzt, solange bis Bit0 von Intern_Mark gesetzt ist, was zum Abbruch der Werkzeugradiuskorrektur führt. Auch hier wird für eine stetige Fortsetzung der Bahn durch ein Positionierobjekt gesorgt. Eine begonnene Werkzeugradiuskorrektur muss erst durch Setzen des Bits0 beendet werden, bevor mit der Versetzung in die jeweils andere Richtung begonnen werden kann. Der SMC_NCDecoder setzt diese Bits als Reaktion auf die Befehle G41/G42/G40. Anders ausgedrückt: Die Werkzeugradiuskorrektur wird für alle Objekte durchgeführt, die innerhalb der Befehle G41 und G40 bzw. G42 und G40 stehen.

Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Werkzeugradiuskorrektur, wenn dieser Eingang eine steigende Flanke aufweist.

bAppend: BOOL

Steht dieser Eingang auf FALSE, so wird bei jedem Reset die Ausgabe-Datenqueue DataOut geleert. Bei TRUE werden die neuen Daten ans Ende der DataOut-Queue geschrieben.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der zu versetzenden Bahn enthält, typischerweise auf den Ausgang DataOut des Vorgängerbausteins (z.B. des SMC_NCDecoders).

dToolRadius: LREAL

In dieser Eingangsvariablen steht der Wert, der addiert auf den jeweiligen ToolRadius-Wert des SMC_GEOINFO-Objekts, den Werkzeugradius angibt, um den die Bahn versetzt werden soll (s.o.). Er darf online verändert werden. Somit hat man die Möglichkeit diesen Wert offline vorzugeben (durch die SMC_GEOINFO-Struktur) und ihn online zu modulieren. Man beachte dabei aber, dass eine Werkzeugradiusänderung während der Bearbeitung eines zu versetzenden Blocks zum Abbruch der Werkzeugbahnkorrektur führt und deshalb zu vermeiden ist. Durchaus möglich ist dies aber während eines Resets oder, wenn gewährleistet ist, dass der Baustein gerade nicht mitten in der Versetzung eines Blockes ist (`Status = TC_ORIG`). Default-Wert: 0.

nSizeOutQueue: UDINT

Hier wird dem Baustein die Größe des Datenpuffers mitgeteilt, in den die Liste von GEOINFO-Strukturobjekten geschrieben wird. Dieser muss mindestens fünf mal so groß wie eine GEOINFO-Struktur. Ist dies nicht der Fall, so führt der SMC_NCDecoder keine Aktionen durch. Der Wert darf gesetzt und hernach nur während eines Resets verändert werden. Es empfiehlt sich, den eigentlichen Buffer wie unten beschrieben z.B. als `Example: Array of SMC_GeoInfo` anzulegen. Die entsprechende Buffergröße ergibt sich dann aus `sizeof(Example)`.

pbyBufferOutQueue: POINTER TO BYTE

Dieser Eingang muss auf das erste Byte des für die OUTQUEUE-Struktur angelegten Speicherbereichs zeigen. Dieser Speicherbereich muss mindestens so groß sein, wie in nSizeOutQueue angegeben. Typischerweise erfolgt das Belegen dieses Speichers im Deklarationsteil des IEC-Programms mittels eines Arrays of SMC_GeoInfo (z.B. `BUF: ARRAY[1..50] OF SMC_GEOINFO;` für einen Buffer, der 50 Bahnelemente aufnehmen kann). Auch dieser Wert darf gesetzt und später nur während eines Resets verändert werden.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

Sollte ein Fehler auftreten, enthält diese Variable die Fehlernummer.

poqDataOut: POINTER TO SMC_OUTQUEUE

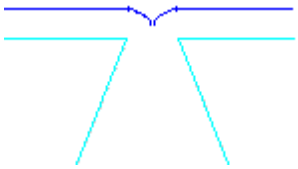
Dieser Ausgang zeigt auf eine SMC_OUTQUEUE-Struktur, die die versetzte Bahn als Liste von SMC_GEOINFO-Objekten enthält.

iStatus: SMC_TC_STATUS (INT)

In dieser Enum-Variable wird der aktuelle Status des Bausteins ausgegeben. Dieser kann sein:

TC_ORIG	0	Keine Werkzeugradiuskorrektur am aktuellen Objekt.
TC_RIGHT	1	Versetze Objekte nach rechts.
TC_LEFT	2	Versetze Objekte nach links.
TC_END	4	Abarbeitung der Objekte abgeschlossen.

6.2.4 SMC_AvoidLoop



Der Baustein *SMC_AvoidLoop* (SM_CNC.lib) dient der Bahnvorverarbeitung: Er erzeugt aus einer vorgegebenen Bahn eine schleifenfreie **Kopie** dieser Bahn. Existiert also in der Originalbahn ein Punkt, bei dem sie sich selbst schneidet, wird die Bahn an diesem Punkt abgeschnitten, die Schleife herausgelassen, und mit dem Rest fortgesetzt. Dadurch entsteht eine schleifenfreie, stetige Bahn.

Anwendung findet dies beispielsweise bei dem unter Der Baustein SMC_ToolCorr beschriebenem Problem.

Der Baustein *SMC_AvoidLoop* arbeitet folgendermaßen:

Der Baustein durchläuft dabei alle sich in der Eingangs-SMC_OUTQUEUE-Struktur befindlichen SMC_GEOINFO-Objekte. Wenn in einem dieser Objekte Bit7 der Variablen Intern_Mark gesetzt ist, wird die Schleifenvermeidung eingeschaltet. Ab jetzt prüft der Baustein, ob ein Schnittpunkt des aktuellen Objekts mit den folgenden SMC_GEOINFO-Objekten vorliegt, die vor einem SMC_GEOINFO-Objekt liegen, bei dessen Variable Intern_Mark Bit6 gesetzt und damit die Schleifenvermeidung ausgeschaltet wird. Im negativen Fall, wird das Objekt unverändert in die Ausgangs-SMC_OUTQUEUE kopiert, andernfalls das erste der beiden sich schneidenden Objekte am Schnittpunkt abgeschnitten, die zwischen den Schnittobjekten liegenden SMC_GEOINFO-Objekte verworfen, und die neue Bahn mit dem zweiten Schnittobjekt vom Schnittpunkt aus fortgesetzt. Der SMC_NCDecoder setzt die Bits 6 und 7 von Intern_Mark als Reaktion auf die Befehle G61/G60.

Entscheidend ist hierbei, wie viele Objekte sich in der Eingangs-SMC_OUTQUEUE befinden, sprich wie groß die Eingangs-SMC_OUTQUEUE dimensioniert ist. Eine kleine Eingangs-SMC_OUTQUEUE hat zur Folge, dass eine Schleife, die mehr Objekte umfasst als in die SMC_OUTQUEUE passen, nicht detektiert werden kann.

Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Werkzeugradiuskorrektur, wenn dieser Eingang eine steigende Flanke aufweist.

bAppend: BOOL

Steht dieser Eingang auf FALSE, so wird bei jedem Reset die Ausgabe-Datenqueue DataOut geleert. Bei TRUE werden die neuen Daten ans Ende der DataOut-Queue geschrieben.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der Bahn enthält, typischerweise auf den Ausgang DataOut des Vorgängerbausteins (z.B. des SMC_NCDecoders). Sie ist entsprechend groß zu dimensionieren (s.o.).

nSizeOutQueue: UDINT

Hier wird dem Baustein die Größe des Datenpuffers mitgeteilt, in den die Liste von GEOINFO-Strukturobjekten geschrieben wird. Dieser muss mindestens fünf mal so groß wie eine GEOINFO-Struktur. Ist dies nicht der Fall, so führt der SMC_NCDecoder keine Aktionen durch. Der Wert darf gesetzt und hernach nur während eines Resets verändert werden. Es empfiehlt sich, den eigentlichen Buffer wie unten beschrieben z.B. als `Example: Array of SMC_GeoInfo` anzulegen. Die entsprechende Buffergröße ergibt sich dann aus `sizeof(Example)`.

pbyBufferOutQueue: POINTER TO BYTE

Dieser Eingang muss auf das erste Byte des für die OUTQUEUE-Struktur angelegten Speicherbereichs zeigen. Dieser Speicherbereich muss mindestens so groß sein, wie in nSizeOutQueue angegeben. Typischerweise erfolgt das Belegen dieses Speichers im Deklarationsteil des IEC-Programms mittels eines Arrays of SMC_GeoInfo (z.B. `BUF: ARRAY[1..50] OF SMC_GEOINFO;` für einen Buffer, der 50 Bahnelemente aufnehmen kann). Auch dieser Wert darf gesetzt und später nur während eines Resets verändert werden.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

Sollte ein Fehler auftreten, enthält diese Variable die Fehlernummer.

poqDataOut: POINTER TO SMC_OUTQUEUE

Dieser Ausgang zeigt auf eine SMC_OUTQUEUE-Struktur, die die SMC_GEOINFO-Objekte der schleifenfreien Bahn verwaltet.

iStatus: SMC_AL_STATUS (INT)

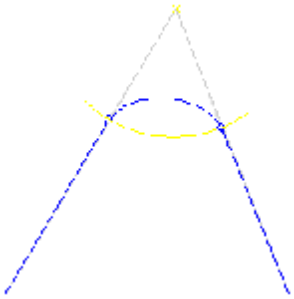
In dieser Enum-Variable wird der aktuelle Status des Bausteins ausgegeben. Dieser kann sein:

AL_OFF	0	Schleifenvermeidung ausgeschaltet.
AL_ON	1	Schleifenvermeidung eingeschaltet.
AL_END	2	Abarbeitung der Objekte abgeschlossen.

6.2.5 SMC_SmoothPath

Der *SMC_SmoothPath*-Baustein (SM_CNC.lib) ist ein möglicher Teil der Bahnvorverarbeitung. Er verrundet Bahnecken, so dass sich eine glatte Bahn ergibt (Eckverschleifung). Der Sinn besteht darin, dass, wenn die Fahrgenauigkeit im Verhältnis zur Geschwindigkeit eine untergeordnete Rolle spielt, Ecken, in denen naturgemäß auf die Geschwindigkeit 0 abgebremst werden muss, vermieden werden.

Dazu wird die Bahn im vorgegebenem Abstand zur Ecke abgeschnitten und ein **Splinestück eingefügt**. Die Größe dieses Abstands erhält der Baustein zum einen aus dem SMC_GEOINFO-Strukturobjekt des ersten der zu verschleifenden Objekte, zum anderen aus einem seiner Eingänge. Diese beiden Werte werden addiert und ergeben den Radius des Kreises mit Zentrum in der Ecke, der mit den umgebenden Objekten zum Schnitt gebracht wird.



Der Baustein *SMC_SmoothPath* geht dabei folgendermaßen vor:

Alle in der Eingangs-OUTQUEUE-Struktur befindlichen SMC_GEOINFO-Objekte werden nacheinander abgegangen. Wenn in einem dieser Objekte Bit4 der Variablen Intern_Mark gesetzt ist, wird ab dort begonnen, Ecken zu verschleifen, solange bis Bit3 von Intern_Mark einer der folgenden Objekte gesetzt ist. Der SMC_NCDecoder setzt diese Bits als Reaktion auf die Befehle G51/G50. Anders ausgedrückt: **Die Eckverschleifung wird für alle Objekte durchgeführt, die innerhalb der Befehle G51 und G50 stehen.**

Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Werkzeugradiuskorrektur, wenn dieser Eingang eine steigende Flanke aufweist.

bAppend: BOOL

Steht dieser Eingang auf FALSE, so wird bei jedem Reset die Ausgabe-Datenqueue DataOut geleert. Bei TRUE werden die neuen Daten ans Ende der DataOut-Queue geschrieben.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der unverrundeten Bahn enthält, typischerweise auf den Ausgang DataOut des Vorgängerbausteins (z.B. des SMC_NCDecoders).

dEdgeDistance: LREAL

In dieser Eingangsvariablen steht der Wert, der addiert auf den jeweiligen ToolRadius-Wert des SMC_GEOINFO-Objekts, den Abstand zu einer Ecke angibt, ab dem die jeweiligen Objekte abgeschnitten und durch eine Spline ersetzt werden (s.o.). Er darf online verändert werden. Somit hat man die Möglichkeit diesen Wert offline vorzugeben (durch die SMC_GEOINFO-Struktur) und ihn online zu modulieren. Default-Wert: 0.

dAngleTol: REAL

Dieser Eingang beschreibt den Winkeltoleranz-Wert, bis zu welchem ein Bahnknick nicht verschliffen werden soll (vgl. 3.3).

nSizeOutQueue: UDINT

Hier wird dem Baustein die Größe des Datenpuffers mitgeteilt, in den die Liste der verrundeten GEOINFO-Strukturobjekten geschrieben wird. Dieser muss mindestens fünf mal so groß wie eine SMC_GEOINFO-Struktur, also etwa 2KB sein. Ist dies nicht der Fall, so führt der *SMC_SmoothPath*-Baustein keinerlei Aktionen durch. Der Wert darf gesetzt und hernach nur während eines Resets verändert werden.

poqBufferOutQueue: POINTER TO BYTE

Dieser Eingang muss auf das erste Byte des für die OUTQUEUE-Struktur angelegten Speicherbereichs zeigen. Dieser Speicherbereich muss mindestens so groß sein, wie in nSizeOutQueue angegeben. Typischerweise erfolgt das Belegen dieses Speichers im Deklarationsteil des IEC-Programms mittels eines Byte-Arrays (z.B. `BUF: ARRAY[1..10000] OF BYTE;` für einen 10000 Byte großen Speicherbereich). Auch dieser Wert darf gesetzt und hernach nur während eines Resets verändert werden.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

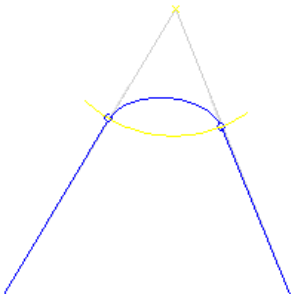
Sollte ein Fehler auftreten, enthält diese Variable die Fehlernummer.

poqDataOut: POINTER TO SMC_OUTQUEUE

Dieser Ausgang zeigt auf eine SMC_OUTQUEUE-Struktur, die die verrundeten SMC_GEOINFO-Objekte verwaltet.

6.2.6 SMC_RoundPath

Der *SMC_RoundPath*-Baustein (SM_CNC.lib) ist dem Baustein SMC_SmoothPath sehr ähnlich. Er verrundet Ecken, die beim Übergang zweier Geraden entstehen, durch **Kreisbögen**.



Dazu wird die Bahn im Abstand r zur Ecke abgeschnitten und ein Splinestück eingefügt. Die Größe dieses Abstands erhält der Baustein zum einen aus dem SMC_GEOINFO-Strukturobjekt des ersten der zu verrundenden Objekte, zum anderen aus dem Eingand dRadius. Letzterer ist dominant, d.h. nur, wenn er 0 ist, wird der des Objekts herangezogen. Sollte der eingestellte Wert größer sein als die halbe Länge eines der beiden SMC_GEOINFO-Objekte, wird stattdessen diese halbe Länge verwendet.

Der Baustein *SMC_RoundPath* geht dabei folgendermaßen vor:

Alle in der Eingangs-SMC_OUTQUEUE-Struktur befindlichen SMC_GEOINFO-Objekte werden nacheinander abgegangen. Wenn in einem dieser Objekte Bit5 der Variablen Intern_Mark gesetzt ist, wird ab dort begonnen, Ecken zu verrunden, solange bis Bit3 von Intern_Mark einer der folgenden Objekte gesetzt ist. Der SMC_NCDecoder setzt diese Bits als Reaktion auf die Befehle G52 und G50: **Die Eckverrundung wird für alle Objekte durchgeführt, die innerhalb der Befehle G52 und G50 stehen**.

Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Werkzeugradiuskorrektur, wenn dieser Eingang eine steigende Flanke aufweist.

bAppend: BOOL

Steht dieser Eingang auf FALSE, so wird bei jedem Reset die Ausgabe-Datenqueue DataOut geleert. Bei TRUE werden die neuen Daten ans Ende der DataOut-Queue geschrieben.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der unverrundeten Bahn enthält, typischerweise auf den Ausgang DataOut des Vorgängerbausteins (z.B. des SMC_NCDecoders).

dRadius: LREAL

In dieser Eingangsvariablen steht der Wert, der den Abstand zu einer Ecke angibt, ab dem die jeweiligen Objekte abgeschnitten und durch einen Kreisbogen ersetzt werden (s.o.). Er darf online verändert werden. Somit hat man die Möglichkeit, diesen Wert an die Gegebenheiten der Bahn anzupassen. Default-Wert: 0.

dAngleTol: REAL

Dieser Eingang beschreibt den Winkeltoleranz-Wert, bis zu welchem ein Bahnknick nicht verschliffen werden soll (vgl. 3.3).

nSizeOutQueue: UDINT

Hier wird dem Baustein die Größe des Datenpuffers mitgeteilt, in den die Liste von GEOINFO-Strukturobjekten geschrieben wird. Dieser muss mindestens fünf mal so groß wie eine GEOINFO-Struktur. Ist dies nicht der Fall, so führt der SMC_NCDecoder keine Aktionen durch. Der Wert darf gesetzt und hernach nur während eines Resets verändert werden. Es empfiehlt sich, den eigentlichen Buffer wie unten beschrieben z.B. als `Example: Array of SMC_GeoInfo` anzulegen. Die entsprechende Buffergröße ergibt sich dann aus `sizeof(Example)`.

pbyBufferOutQueue: POINTER TO BYTE

Dieser Eingang muss auf das erste Byte des für die OUTQUEUE-Struktur angelegten Speicherbereichs zeigen. Dieser Speicherbereich muss mindestens so groß sein, wie in nSizeOutQueue angegeben. Typischerweise erfolgt das Belegen dieses Speichers im Deklarationsteil

des IEC-Programms mittels eines Arrays of SMC_GeoInfo (z.B. `BUF: ARRAY[1..50] OF SMC_GEOINFO;` für einen Buffer, der 50 Bahnelemente aufnehmen kann). Auch dieser Wert darf gesetzt und später nur während eines Resets verändert werden.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

Sollte ein Fehler auftreten, enthält diese Variable die Fehlernummer.

poqDataOut: POINTER TO SMC_OUTQUEUE

Dieser Ausgang zeigt auf eine SMC_OUTQUEUE-Struktur, die die verrundeten SMC_GEOINFO-Objekte verwaltet.

6.2.7 SMC_CheckVelocities

Dieser Baustein überprüft die Geschwindigkeiten der einzelnen Bahnsegmente. Er muss für den Fall, dass die OutQueue nicht vom Editor, sondern im IEC-Programm (z.B. durch SMC_NCDecoder) erzeugt wurde, stets direkt vor dem Interpolator aufgerufen werden.

Hauptaufgabe dieser Funktion ist es, die Bahn auf Knicke zu untersuchen und dort auf Geschwindigkeit 0 zu reduzieren.

Eingänge des Bausteins:

bExecute: BOOL

Bei steigender Flanke wird die Überprüfung begonnen.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der Bahn enthält, typischerweise auf den Ausgang *poqDataOut* des Vorgängerbausteins (z.B. SMC_NCDecoder/SMC_SmoothPath).

dAngleTol: REAL

Dieser Eingang beschreibt den Winkeltoleranz-Wert, bis zu welchem an einem Bahnknick kein Stop durchgeführt werden soll (vgl. 3.3).

Ausgänge des Bausteins:

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

Dieser Enum-Ausgang beschreibt ggf. einen Fehler.

poqDataOut: POINTER TO SMC_OUTQUEUE

Dieser Ausgang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, welches die Bahn mit zulässigen Geschwindigkeitswerten enthält und nun direkt in den Interpolator fließen soll.

6.2.8 SMC_LimitCircularVelocities

Dieser Baustein überprüft die einzelnen Elemente der OutQueue und begrenzt die Bahn-Geschwindigkeiten von kreisförmigen Elementen in Abhängigkeit von deren Radien.

Die Bahn-Beschleunigung, wenn man mit konstanter Geschwindigkeit (v) über einen Übergang von Gerade zu Kreis mit Radius r fährt, springt betragsmäßig von 0 auf den Wert { EQ }{ EQ \r(v²;r) }. Um diesen Beschleunigungssprung auf den Wert a zu begrenzen, darf die Geschwindigkeit des Kreisbogen dementsprechend beim Übergang v = { EQ \r(a * r) } nicht überschreiten.

Dieser Baustein überprüft die Übergänge zweier Elemente (Gerade auf Kreisbogen, Kreisbogen auf Gerade und Kreisbogen auf Kreisbogen), und passt die Endgeschwindigkeit des ersten Elements an, sodass der Beschleunigungssprung den Wert **dMaxAccJump** nicht überschreitet.

Außerdem begrenzt der Baustein die Bahnbeschleunigung auf Kreisen auf den Wert **dMaxAcc**, indem er die Bahn-Geschwindigkeit des Kreises entsprechend verringert.

Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Werkzeugradiuskorrektur, wenn dieser Eingang eine steigende Flanke aufweist.

bAppend: BOOL

Steht dieser Eingang auf FALSE, so wird bei jedem Reset die Ausgabe-Datenqueue DataOut geleert. Bei TRUE werden die neuen Daten ans Ende der DataOut-Queue geschrieben.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der zu verändernden Bahn enthält, typischerweise auf den Ausgang DataOut des Vorgängerbausteins (z.B. des SMC_NCDecoders).

dMaxAcc: LREAL

In dieser Eingangsvariablen steht der maximale Beschleunigungswert, der für Kreisbögen zulässig ist. Ein Wert gleich 0 hat zur Folge, dass diese Überprüfung nicht durchgeführt wird.

dMaxAccJump: LREAL

In dieser Eingangsvariablen steht der maximale Beschleunigungssprung (a) für einen Übergang zweier Objekte. Ein Wert gleich 0 bedeutet, dass diese Überprüfung nicht durchgeführt wird.

nSizeOutQueue: UDINT

Hier wird dem Baustein die Größe des Datenpuffers mitgeteilt, in den die Liste von GEOINFO-Strukturobjekten geschrieben wird. Dieser muss mindestens fünf mal so groß wie eine GEOINFO-Struktur. Ist dies nicht der Fall, so führt der SMC_NCDecoder keine Aktionen durch. Der Wert darf gesetzt und hernach nur während eines Resets verändert werden. Es empfiehlt sich, den eigentlichen Buffer wie unten beschrieben z.B. als `Example: Array of SMC_GeoInfo` anzulegen. Die entsprechende Buffergröße ergibt sich dann aus `sizeof(Example)`.

pbyBufferOutQueue: POINTER TO BYTE

Dieser Eingang muss auf das erste Byte des für die OUTQUEUE-Struktur angelegten Speicherbereichs zeigen. Dieser Speicherbereich muss mindestens so groß sein, wie in nSizeOutQueue angegeben. Typischerweise erfolgt das Belegen dieses Speichers im Deklarationsteil des IEC-Programms mittels eines Arrays of SMC_GeoInfo (z.B. `BUF: ARRAY[1..50] OF SMC_GEOINFO;` für einen Buffer, der 50 Bahnelemente aufnehmen kann). Auch dieser Wert darf gesetzt und später nur während eines Resets verändert werden.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

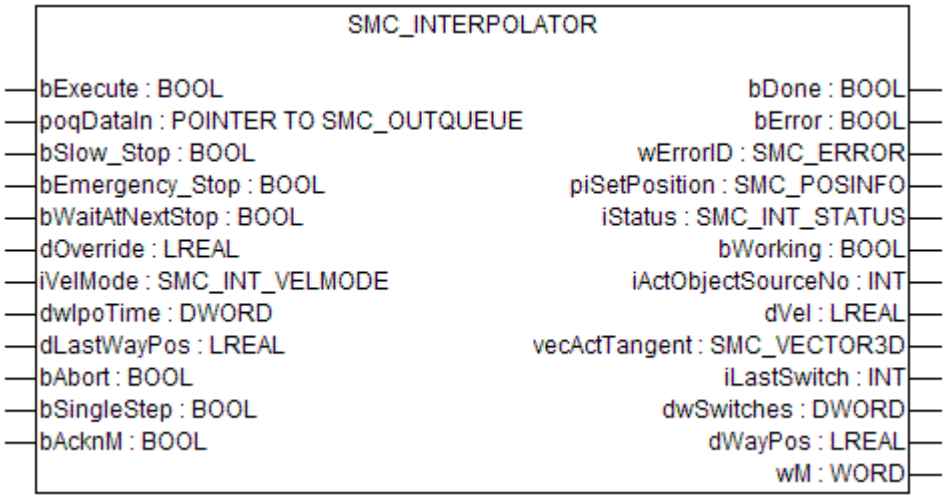
Sollte ein Fehler auftreten, enthält diese Variable die Fehlernummer.

poqDataOut: POINTER TO SMC_OUTQUEUE

Dieser Ausgang zeigt auf eine SMC_OUTQUEUE-Struktur, die die neuen SMC_GEOINFO-Objekte verwaltet.

6.2.9 SMC_Interpolator

Der SMC_Interpolator-Baustein (SM_CNC.lib) hat die Aufgabe, eine vorliegende, durch GEOINFO-Objekte beschriebene, stetige Bahn unter Rücksichtnahme auf vorgegebenes Geschwindigkeitsprofil und Zeitraster in **diskrete Bahnpunkte** zu überführen. Diese Positionsvorgaben werden üblicherweise dann vom IEC-Programm (z.B. auf die Antriebs-Achsenstellungen) transformiert und über das Drive Interface an die Antriebe gesendet.



Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Interpolation, wenn dieser Eingang eine steigende Flanke aufweist.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der Bahn enthält, die interpoliert werden soll, typischerweise auf den Ausgang *poqDataOut* des Vorgängerbausteins SMC_CheckVelocities.

bSlow_Stop: BOOL

Steht dieser Eingang auf FALSE (default), so wird die Bahn ohne Unterbrechung befahren. Mit TRUE bringt man den SMC_Interpolator dazu, entsprechend dem eingestellten Geschwindigkeitsprofil (*byVelMode* s.u.) und der maximalen Verzögerung des aktuellen GEOINFO-Objekts (d*Decel* s.u.) die Geschwindigkeit auf 0 zu vermindern und solange zu warten, bis Slow_Stop wieder auf FALSE gesetzt wird.

bEmergency_Stop: BOOL

Dieser Eingang steht standardmäßig auf FALSE. Wird er TRUE, dann führt der SMC_Interpolator einen sofortigen Stopp aus, d.h. die Stell-Position wird beibehalten. Die Geschwindigkeit wird somit also unmittelbar auf 0 gesetzt.

bWaitAtNextStop: BOOL

Steht dieser Eingang auf FALSE (default), so wird die Bahn ohne Unterbrechung befahren. Mit TRUE bringt man den SMC_Interpolator dazu, beim nächsten natürlichen Stopp, also an Punkten, wo die Geschwindigkeit 0 beträgt – typischerweise an Bahnecken -, die Stell-Position beizubehalten und solange zu verweilen, bis *bWaitAtNextStop* wieder auf FALSE gesetzt wird.

dOverride: LREAL

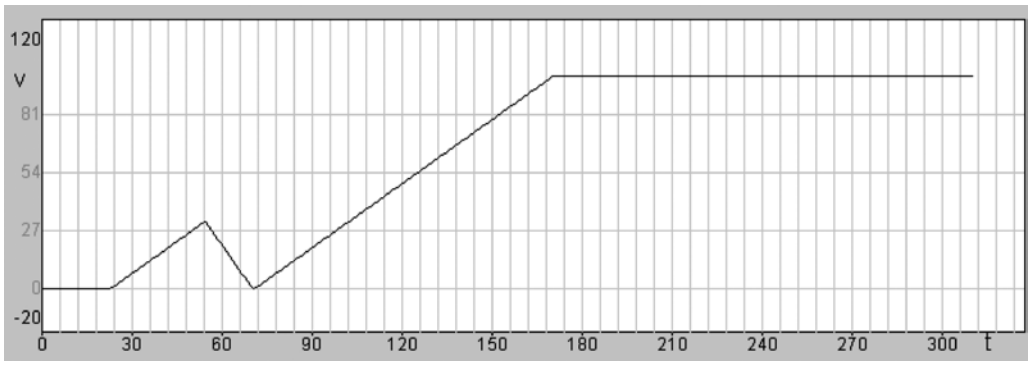
Mit Hilfe dieser Variablen kann man den Override bedienen. Zulässige Werte sind größer als 0.01. Der Override wird auf die Sollgeschwindigkeit der einzelnen Objekte multipliziert und ermöglicht es so, online die Sollgeschwindigkeit zu erhöhen bzw. zu vermindern. Ein Override von 1 (default) beispielsweise bewirkt, dass die programmierten Sollgeschwindigkeiten ausgeführt werden, während ein Override von 2 diese verdoppeln würde.

Auch wenn der Override zu jeder Zeit geändert werden darf, sei erwähnt, dass eine Änderung nur übernommen wird, wenn gerade nicht beschleunigt oder gebremst wird.

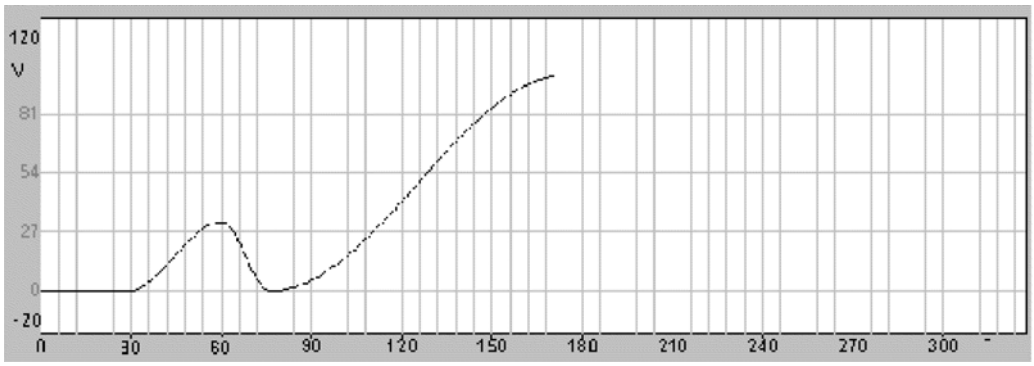
iVelMode: SMC_INT_VELMODE

Mit diesem Eingang wird das Geschwindigkeitsprofil festgelegt. Der Wert "TRAPEZOID" (default) bewirkt ein trapezförmiges Geschwindigkeitsprofil, "SIGMOID" ein S-förmiges :

Beispiel eines trapezförmigen Geschwindigkeitsprofils (Vel_Mode = 0):



Beispiel eines S-förmigen Geschwindigkeitsprofils (Vel_Mode = 1):

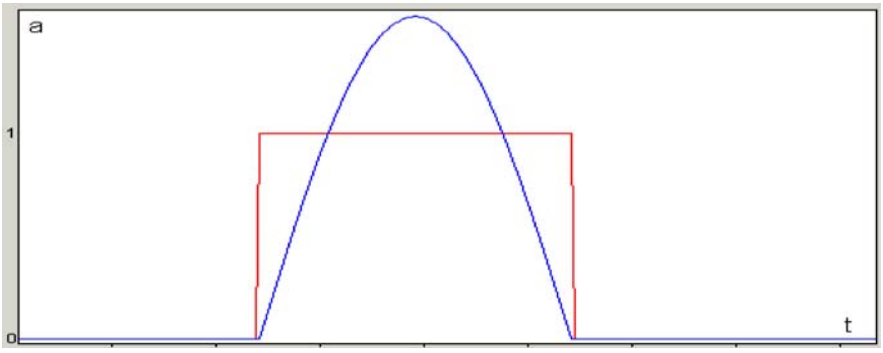


In obigen Beispielen wurde die maximale Beschleunigung (*Accel*) kleiner gewählt als die maximale Bremsung (*Decel*). Dies verursacht die unterschiedliche Steigung der Geschwindigkeit beim Beschleunigen und Bremsen.

Das S-förmige Geschwindigkeitsprofil hat den Vorteil, dass die dazugehörige Beschleunigung – im Gegensatz zur trapezförmigen - stetig ist, und somit gerade bei schweren Maschinen Entlastung bringt. Dies bezahlt man mit einer leicht erhöhten Rechenzeit.

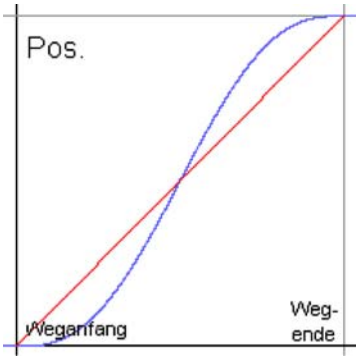
Da das S-förmige Geschwindigkeitsprofil (blau) so ausgelegt ist, dass ein Wechsel zum trapezförmigen Profil (rot) keine Änderung der für das Fahren der Bahn benötigten Zeitdauer zur Folge haben soll, muss das vorsichtige Steigern der Beschleunigung am Anfang und Ende durch eine höhere Beschleunigung in der Mitte ausgeglichen werden. Dabei gilt zu es beachten, dass die in den

Geo-Objekten programmierte maximale Beschleunigung bzw. Bremsung im Maximum um den Faktor π/2 überschritten werden:



Eine Online-Änderung wird – wie bei *Override* – erst übernommen, wenn eine Beschleunigungs- bzw. Bremsphase abgeschlossen ist.

Um die Zusatzachsen ebenso S-förmig (blau, siehe Skizze) statt linear (rot) zu interpolieren, müssen – unabhängig von dem Eingang byVelMode des SMC_Interpolators – entsprechende Bits in der Variable `wSProfile` von `piStartPos` des aktuellen Wegobjekts gesetzt sein. Dann werden die Zusatzachse nicht linear zur Weglänge im X-,Y-,Z-Raum interpoliert, sondern in polynomialer Abhängigkeit davon, sodass sich ein S-förmiges Profil für die Achsposition ergibt, die am Anfang und Ende eines Wegabschnitts Geschwindigkeit und Beschleunigung null hat.



dwIpoTime: DWORD

Dieser Eingang, der für **jeden Aufruf** gesetzt werden muss, beinhaltet die Zykluszeit in µsec.

dLastWayPos: LREAL

Dieser Eingang gibt dem Anwender die Möglichkeit, die Wegstrecke, die der Interpolator abfährt, zu messen. Der Ausgang dWayPos ist die Summe aus dLastWayPos und der in diesem Zyklus zurückgelegten Distanz. Ist dLastWayPos=0, so enthält dWayPos die Länge des aktuellen Wegabschnitts. Setzt man dLastWayPos gleich dem Ausgang dWayPos, so wird dWayPos stets um das aktuelle Wegstück inkrementiert, und man erhält die insgesamt gefahrene Wegstrecke. Dabei kann dLastWayPos jederzeit auf 0 oder einen anderen Wert (zurück)gesetzt werden.

bAbort: BOOL

Dieser Eingang bricht die Bearbeitung einer Kontur ab.

bSingleStep: BOOL

Dieser Eingang bewirkt, dass der Interpolator an jedem Übergang zwischen zwei Bahnobjekten (auch an Übergangen mit gleicher Tangente) für die Dauer eines Zyklus anhält. Wird bSingleStep während des Verfahrens auf TRUE gesetzt, hält der Interpolator am nächsten Objektende an, das er ohne ein Überschreiten der vorgegebenen Bremsung erreichen kann.

Wenn der Interpolator am nächsten möglichen Stop (also an Punkten, wo die Geschwindigkeit 0 beträgt) anhalten soll, muss bWaitAtNextStop verwendet werden.

bAcknM: BOOL

Mit diesem Eingang kann eine anstehende Zusatzfunktion (M-Funktion) quittiert werden. Ist der Eingang TRUE, wird diese gelöscht und die Bahn wird fortgesetzt.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

Dieser Enum-Ausgang beschreibt ggf. einen Fehler, der beim Interpolieren aufgefallen ist. Nach einem Fehler wird die Abarbeitung bis zu einem Reset gestoppt.

piSetPosition: SMC_POSINFO

In *Set_Position* steht die gemäß den Vorgaben berechnete Soll-Position. *Set_Position* ist eine SMC_POSINFO-Struktur und enthält nicht nur die kartesischen Koordinaten des anzufahrenden Bahnpunktes, sondern auch die Stellung der Zusatzachsen.

iStatus: SMC_INT_STATUS (INT)

In dieser Enum-Variable wird der aktuelle Status des Bausteins ausgegeben. Dieser kann sein:

IPO_UNKNOWN	0	Interner Zustand. Zustand darf nach vollständigem SMC_Interpolator-Durchlauf nicht auftreten.
IPO_INIT	1	Initialisierungszustand; DataIn ist und war noch nicht voll.
IPO_ACCEL	2	Baustein befindet sich in Beschleunigungsphase.
IPO_CONSTANT	3	Baustein fährt mit konstanter Geschwindigkeit.
IPO_DECEL	4	Baustein befindet sich in Bremsphase.
IPO_FINISHED	5	Abarbeitung der *GEOINFO*-Liste abgeschlossen. Nachträglich in *DataIn* ankommende GEOINFO-Objekte werden nicht mehr bearbeitet.
IPO_WAIT	6	Baustein wartet, da einer der folgenden Situationen eingetreten ist: *Emergency_Stop* = TRUE *Slow_Stop* = TRUE und *Vel* = 0 *Wait_At_Next_Stop* = TRUE und *Vel* = 0

bWorking: BOOL

Ausgang wird TRUE, sobald die Abarbeitung der Liste begonnen und noch nicht abgeschlossen wurde (IPO_ACCEL oder IPO_CONSTANT oder IPO_DECEL oder IPO_WAIT). Ansonsten steht *Working* auf FALSE.

iActObjectSourceNo: INT

Hier steht der Eintrag *SourceLine_Nr* des aktuell befahrenen GEOINFO-Objekts aus der *DataIn*-Queue. Arbeitet der SMC_Interpolator nicht (mehr) (*Working* = FALSE), steht hier –1.

dVel: LREAL

In dieser Variable steht die aktuelle Geschwindigkeit, die sich ergibt, wenn sich ein Objekt in der Zeit Ipo_Time von der vorigen Raumkoordinate nach *Set_Position* bewegt.

vecActTangent: SMC_VECTOR3D

In dieser Struktur befindet sich die Bahnrichtung im Punkt *Set_Position*. Für den Fall *Vel* = 0 befinden sich auch in *Act_Tangent* lauter Nullen.

iLastSwitch: INT

In dieser Variable steht die Nummer der zuletzt passierten Hilfsmarke. Sollten in einem Zyklus mehrere Hilfsmarken überlaufen werden, wird nur jeweils die letzte ausgegeben.

dwSwitches: DWORD

Dieses DWORD enthält den momentanen Schaltzustand aller der Hilfsmarken zwischen 1 und 32. Bit0 des DWORDs steht für Hilfsmarke1, Bit31 für Hilfsmarke 32. Anders als bei iLastHelpMark kann so ausgeschlossen werden, dass eine Hilfsmarke verloren geht.

dWayPos: LREAL

Beschreibung siehe Eingang dLastWAyPos.

Pro Aufruf wird vom SMC_Interpolator unter Rücksichtnahme auf die vorgegebenen Parameter, die Geschwindigkeitshistorie und die letzte Bahnposition der folgende Bahnpunkt berechnet und ausgegeben. Wurde das jeweils erste GEOINFO-Objekt abgearbeitet, wird es aus der poqDataIn-SMC_OUTQUEUE-Struktur gelöscht.

wM: WORD

Passiert der Interpolator ein M-Objekt, d.h. eine Zeile die eine Zusatzfunktion beschreibt, wird dieser Ausgang auf den entsprechenden Wert gesetzt und so lange gewartet, bis er durch den Eingang bAcknM quittiert wird.

Man beachte: Am Ende einer Kontur ist die SMC_OUTQUEUE-Variable leer. Will man dieselbe Kontur erneut verfahren, muss man entweder erneut das CNC-Programm via Decoder und Bahnvorverarbeitungsbausteine in eine SMC_OUTQUEUE-Struktur überführen, oder man wendet die Funktion SMC_RESTOREQUEUE (siehe 6.5) an. Letzteres ist nur dann möglich, wenn der OUTQUEUE-Buffer so groß gewählt wurde, dass die ganze Kontur darin Platz findet.

6.2.10 SMC_GetMParameters

Mit diesem Baustein kann man, während der Interpolator auf einer M-Funktion steht, auf Parameter, die man der M-Funktion mitgegeben hat (K, L, O, siehe 3.2.2), abfragen.

Eingänge des Bausteins:

bEnable: BOOL

Baustein ist aktiv, wenn dieser Eingang gesetzt ist

Ein-/Ausgänge (VAR_IN_OUT) des Bausteins:

Interpolator: SMC_Interpolator

Interpolator-Instant

Ausgänge des Bausteins:

bMActive: BOOL (Default: FALSE)

TRUE, wenn aktuell eine M-Funktion ansteht

dK, dL: LREAL (Default: 0)

M-Parameter, die über die Worte K und L festgelegt wurden

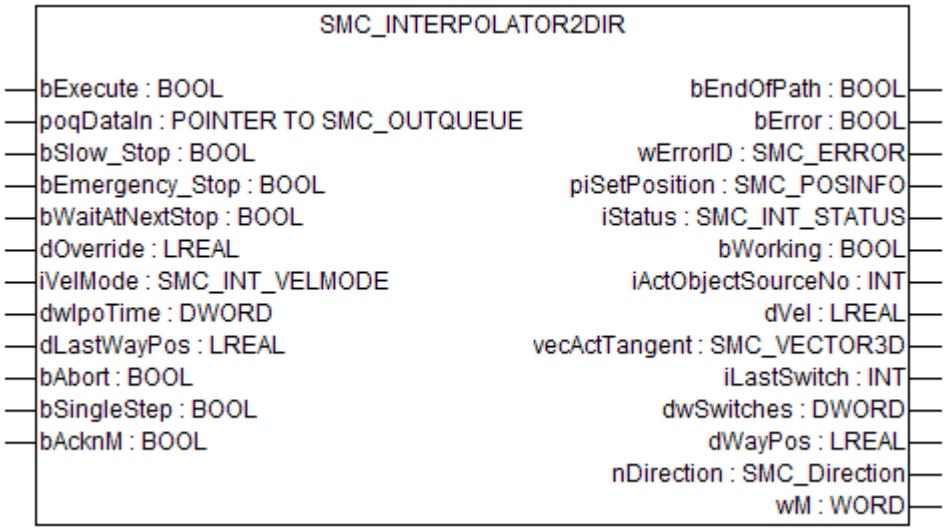
MParameters: SMC_M_PARAMETERS

M-Parameter, die über die globale Datenstruktur *gSMC_MParameters* bzw. über die durch O übergebene Variable festgelegt wurden.

6.2.11 SMC_Interpolator2Dir

Dieser Baustein entspricht in seiner Funktion und der Belegung seiner Ein- und Ausgänge dem Funktionsbaustein SMC_Interpolator, mit dem Unterschied, dass er eine Bahn auch rückwärts interpolieren kann.

Dazu wird dem Eingang **dOverride** ein negativer Wert zugewiesen, worauf SMC_Interpolator2Dir in negative Richtung interpoliert. Auf diesen Eingang kann z.B. der analoge Geschwindigkeitseingang eines Handrades liegen, so dass der Bediener mit beliebiger Geschwindigkeit vorwärts und rückwärts fahren kann.



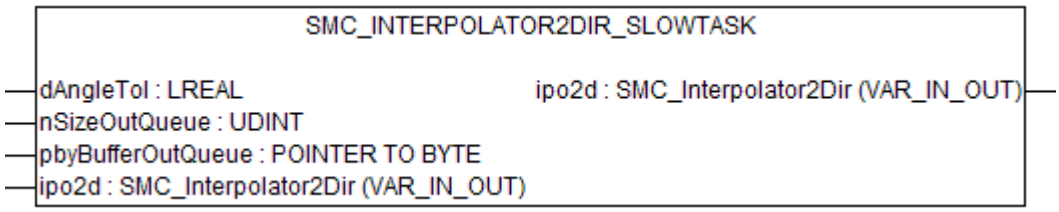
Zusätzliche Ein- und Ausgänge des Bausteins:

nDirection: SMC_Direction

Der Baustein gibt aus, in welche Richtung er aktuell verfährt. Mögliche Werte sind IPO_positive, IPO_negative und IPO_standstill.

Folgende Voraussetzungen müssen für das Verwenden dieses Bausteins gegeben sein:

1. Die Bahn muss vollständig in poqDataIn passen. Da der Baustein abwechselnd vorwärts und rückwärts zu fahren hat, muss die komplette Bahn im Speicher vorliegen.
2. Ein zusätzlicher Baustein, **SMC_Interpolator2Dir_SlowTask**, wird aufgerufen:



Dieser Baustein ist für das Erzeugen des Rückwärts-Pfades verantwortlich. Er wurde von SMC_Interpolator2Dir abgespalten, damit er auf stark ausgelasteten und weniger performanten Zielsystemen in eine nieder-priore Task ausgelagert werden kann.

Eingänge des Bausteins:

dAngleTol: LREAL

Winkeltoleranz für den Rückwärtspfad. Typischerweise identisch zur Winkeltoleranz des Ur-Pfades.

nSizeOutQueue: UDINT, pbyBufferOutQueue: POINTER TO BYTE

Größe und Zeiger auf Datenpuffer, in welchen die Rückwärtsbahn gespeichert werden soll. Muss mindestens so groß sein, dass die komplette Bahn darin Platz findet.

Ipo2d: SMC_Interpolator2Dir

SMC_Interpolator2Dir-Instanz, für welche der Rückwärts-Pfad erzeugt werden soll.

6.3 Hilfsfunktionen und –funktionsblöcke für Bahnrotationen, –translationen und -skalierungen

Die in der SM_CNC.lib enthaltenen Funktionsblöcke SMC_RotateQueue2D und SMC_TranslateQueue3D drehen bzw. verschieben die in einer SMC_OutQueue gespeicherte Bahn.

Über die Eingangsvariable ***DataIn*** übergibt man die zu rotierende bzw. zu verschiebende Bahn in Form eines Pointers auf das SMC_OUTQUEUE-Strukturobjekt.

Der Eingang ***bEnable***, der mit FALSE initialisiert ist, verhindert so lange das Rotieren bzw. Versetzen der Bahn, bis er auf TRUE gesetzt wird. Dann werden alle GEOINFO-Elemente von *poqDataIn* bearbeitet. Sobald *bEnable* FALSE wird, nehmen die Bausteine keine Veränderungen mehr vor.

Der Eingang ***bReset***, der ebenfalls mit FALSE initialisiert ist, bewirkt, daß die momentan in *poqDataIn* liegenden GEOINFO-Objekte nicht bearbeitet werden, sondern erst die ab jetzt eintreffenden rotiert bzw. verschwoben werden.

- **SMC_ROTATEQUEUE2D**
  Die in poqDataIn enthaltene Bahn wird um die Z-Achse um den in dPhi übergebenen Winkel in ° verschoben. Ein positiver Winkel bewirkt eine im mathematischen Sinn positive Drehung (gegen den Uhrzeigersinn).

- **SMC_TRANSLATEQUEUE3D**
  Die in poqDataIn enthaltene Bahn wird um den in vec übergebenen Vektor vom Strukturtyp SMC_VECTOR3D (siehe SMC_VECTOR3D) verschoben.

- **SMC_SCALEQUEUE3D**
  Die in poqDataIn enthaltene Bahn wird um den Faktor fScaleFaktor gestreckt.

Da eine plötzliche Änderung der charakteristischen Größen der Rotation/Translation (*dPhi*, *vec*) während der Abarbeitung einen Sprung in der Bahn zur Folge haben könnte, werden Änderungen in den Eingängen dieser Größen solange nicht beachtet bis die SMC_OUTQUEUE bei *poqDataIn* leer ist, oder ein *bReset* stattfindet.

Um eine Rotation in der (XY)-Ebene um einen anderen Punkt als (00), d.h. um den Punkt (XpYp) zu erreichen, verwende man hintereinander eine Translation um den Vektor –Xa-Yp°0), die Rotation mit dem gewünschten Winkel *dPhi* und eine weitere Translation um den Vektor (XyYp°0).

6.4 Einstellungen über globale Variablen

Unter „SoftMotion_CNC_Globals“ (siehe Resourcen unter Eintrag für Bibliothek SM_CNC.lib) sind einige interne Variablen und Konstanten definiert. Manche davon können geändert werden:

Prüfung auf Null (siehe 3.6, ‚Epsilon-Werte für Null ändern’):

- g_fSMC_CNC_EPS (Epsilon-Wert für exakte Nullprüfung)
- g_fSMC_CNC_EPS_RELUCTANT (Epsilon-Wert für tolerante Nullprüfung)

6.5 Strukturen der SM_CNC.lib

Im folgenden sehen Sie die Beschreibung einer Auswahl von Strukturen der SM_CNC.lib, die von den Bausteinen der SM_CNC.lib zur Speicherung von Positionen, Bahnabschnitten (Wegobjekten) und Vektoren verwendet werden: SMC_POSINFO, SMC_GEOINFO, SMC_VECTOR3D, SMC_VECTOR6D. Außerdem der SMC_OUTQUEUE-Struktur, die hilft, GEOINFO-Objekte in einer Liste definierter Größe zu verwalten.

SMC_POSINFO

In der in SMC_CNC.lib enthaltenen Struktur **SMC_POSINFO** werden Koordinaten und Stellung der Zusatzachsen einer Position gespeichert.

```
TYPE SMC_POSINFO:
STRUCT
       iFrame_Nr:INT;
       wAuxData:WORD;
       wSProfile:WORD;
       dX:LREAL;
       dY:LREAL;
       dZ:LREAL;
       A:LREAL;
       B:LREAL;
       dC:LREAL;
       dA1:LREAL;
       dA2:LREAL;
       dA3:LREAL;
       dA4:LREAL;
       dA5:LREAL;
       dA6:LREAL;
END_STRUCT
END_TYPE
```

Dabei enthalten die Variablen dX, dY und dZ die Position im Raum, dA1, ..., dA6 die Stellung der Zusatzachsen. iFrame_Nr gibt dem Benutzer die Möglichkeit, weitere Informationen abzulegen, die für die SoftMotion-Bausteine nicht von Belang sind. dA, dB und dC werden momentan nicht verwendet.

wAuxData gibt bitweise an, welche der Positionsachsen vom SMC_Interpolator berechnet werden sollen. Initialisiert wird wAuxData mit 2#111, d.h. X-, Y- und Z-Achse werden interpoliert. Ist das erste Bit gesetzt, wird die dX-Position berechnet, Bit 7 beispielsweise bewirkt eine Bearbeitung von dA2.

wSProfile gibt in gleicher Weise für die Zusatzachsen an (alle außer X,Y-Achse), ob sie vom Interpolator linear (FALSE) oder S-förmig (TRUE) interpoliert werden sollen. Dabei stehen Bit2 für die Z-Achse, Bit6 für P, Bit7 für Q, Bit8 für U, Bit9 für V und Bit10 für W.

SMC_GEOINFO

In der in SMC_CNC.lib enthaltenen Struktur **SMC_GEOINFO** wird je ein Wegobjekt gespeichert. Als ein Wegobjekt wird ein Teil der programmierten Bahn bezeichnet, das sich aufgrund seiner geometrischen Form vollständig in der folgenden Struktur abspeichern lässt:

```
TYPE SMC_GEOINFO:
STRUCT
       iObjNo:INT;
       iSourceLineNo:INT;
       iMoveType:MOVTYP;
       piStartPos:SMC_POSINFO;
       piDestPos:SMC_POSINFO;
       dP1:LREAL;
       dP2:LREAL;
       dP3:LREAL;
       dP4:LREAL;
       dP5:LREAL;
       dP6:LREAL;
       dP7:LREAL;
       dP8:LREAL;
       dT1:LREAL;
       dT2:LREAL;
       dToolRadius:LREAL;
       dVel:LREAL;
       dVelEnd:LREAL;
       dAccel:LREAL;
       dDecel:LREAL;
       dLength:LREAL;
       byInternMark:BYTE;
       dHelpPos: ARRAY[0..MAX_IPOSWITCHES] OF LREAL;
       iHelpID: ARRAY[0..MAX_IPOSWITCHES] OF INT;
```

```
END_STRUCT
END_TYPE
```

iObjNo: INT

In diesem Ganzzahlwert kann eine beliebige Objekt-Nummer gespeichert werden. Sie hat für die eigentliche Wegbeschreibung keine Bedeutung.

iSourceLineNo: INT

In diesem Ganzzahlwert wird typischerweise die Quellcode-Zeilennummer des CNC-Programms mitgeführt. Sie hat für die eigentliche Wegbeschreibung keine Bedeutung.

iMoveType: MOVTYP (INT)

Der Enum-Typ MOVTYP beinhaltet folgende zulässige Werte und beschreibt den Typ des Objekts:

LIN	1	gerade Bewegung (G01)
CLW	2	Kreis im Uhrzeigersinn (G02)
CCLW	3	Kreis gegen Uhrzeigersinn (G03)
SPLINE	5	Spline, Parabel (G05, G06)
ELLCLW	8	Ellipse im Uhrzeigersinn (G08)
ELLCCLW	9	Ellipse gegen den Uhrzeigersinn (G09)
LINPOS	100	gerade Positionierung (G00)
INITPOS	110	blinde Positionierung (Anfangspunkt noch nicht bekannt; wird vom SMC_Interpolator stetig ergänzt)
MCOMMAND	120	Zusatzfunktion, M-Funktion

piStartPos: SMC_POSINFO

Struktur SMC_POSINFO, die die exakte Anfangsposition beschreibt. (Wird im Fall Move_Type = INITPOS ignoriert).

piDestPos: SMC_POSINFO

Struktur SMC_POSINFO, die die exakte Endposition enthält.

dP1, ..., dP8: LREAL

In diesen Variablen werden, abhängig vom Move_Type (s.o.), weitere den Weg beschreibende Informationen gespeichert:

LIN LINPOS	Keine Bedeutung, da komplette Information bereits in Start_Pos und Dest_Pos enthalten.
CLW CCLW	P1: X-Koordinate des Kreismittelpunkts P2: Y-Koordinate des Kreismittelpunkts P3: Kreisradius
SPLINE	Spline-Parameter
ELLCLW, ELLCCLW	P1: X-Koordinate des Kreismittelpunkts P2: Y-Koordinate des Kreismittelpunkts P3: X-Komponente des Hauptachsen-1-Vektors P4: Y-Komponente des Hauptachsen-1-Vektors P5: Länge der Hauptachse P6: Länge der Nebenachse

	P7: Richtung der Hauptachse (°) P8: Verhältnis P6/P5
INITPOS	Keine Bedeutung

dT1, dT2: LREAL

In diesen Variablen werden Start und Ende des Laufparameters angegeben. Abhängig vom Move_Type bedeutet dies:

LIN LINPOS	Keine Bedeutung, da komplette Information bereits in Start_Pos und Dest_Pos enthalten.
CLW, ELLCLW	T1: Startwinkel im mathematischen Sinn in Grad: (0 = Osten, 90 = Nord, 180 = West, 360 = Süd) T2: Öffnungswinkel des Kreises, Kreisbogenlänge in Grad: (z.B.: 90=Viertelkreis, 180 = Halbkreis)
CCLW, ELLCCLW	T1: Startwinkel im mathematischen Sinn in Grad: (0 = Osten, 90 = Nord, 180 = West, 360 = Süd) T2: Negativer Öffnungswinkel des Kreises: (z.B.: -90=Viertelkreis, -180 = Halbkreis)
SPLINE	Start- und Endwert des Parameters t (siehe Beschreibung der Spline). Standardmäßig 0 und 1.
INITPOS	Keine Bedeutung.

dToolRadius: LREAL

In dieser Variablen wird für die Bahnvorverarbeitung nötige Information gespeichert (siehe Der Baustein SMC_ToolCorr, Der Baustein SMC_RoundPath). Der Eintrag hat keine Relevanz, wenn nicht entsprechende Bahnvorverarbeitungsbausteine (SMC_ToolCorr, SMC_RoundPath) durchlaufen werden.

dVel, dVelEnd: LREAL

Diese beiden Variablen, die in jedem Fall belegt sein müssen, enthalten Information für das Geschwindigkeitsprofil in diesem Objekt. dVel beschreibt die Soll-Geschwindigkeit, die erreicht werden soll, dVelEnd die Geschwindigkeit, die am Ende des Objekts gefahren werden muss (siehe Der Baustein SMC_Interpolator) jeweils in Wegeinheiten/sec.

dAccel, dDecel: LREAL

In dAccel und dDecel werden die maximal zulässige Beschleunigung und Verzögerung in Wegeinheiten/sec^2 gespeichert. beide Variablen sind mit dem Wert 100 vorbelegt.

dLength: LREAL

Diese Variable, die unbedingt belegt sein muss, enthält die Länge des Objekts in Wegeinheiten.

byIntern_Mark: BYTE

Hier sind Start- und Ende von Bahnvorverarbeitungen folgendermaßen abgelegt:

Bit 0 gesetzt	Ende der Werkzeugradiuskorrektur nach diesem Objekt
Bit 1 gesetzt	Beginn der Werkzeugradiuskorrektur links bei diesem Objekt
Bit 2 gesetzt	Beginn der Werkzeugradiuskorrektur rechts bei diesem Objekt
Bit 3 gesetzt	Ende der Bahnverrundung/-schleifung nach diesem Objekt

Bit 4 gesetzt	Beginn der Bahnverschleifung bei diesem Objekt
Bit 4 gesetzt	Beginn der Bahnverrundung bei diesem Objekt
Bit 6 gesetzt	Ende der Schleifenvermeidung nach diesem Objekt
Bit 7 gesetzt	Beginn der Schleifenvermeidung bei diesem Objekt

dHelpPos: ARRAY[0..MAX_IPOSWITCHES] OF LREAL,

iHelpID: ARRAY[0..MAX_IPOSWITCHES] OF INT:

In diesen Variablen steht die relative Position (0: Objektanfang, 1:Objektende; ähnlich G-Code: O) und die ID (vgl. G-Code: H) der Hilfsschalter.

Handelt es sich beim aktuellen Objekt um ein MCOMMAND, steht in iHelpID[0] die Nummer der M-Funktion.

SMC_VECTOR3D

In der in SMC_CNC.lib enthaltenen Struktur **SMC_VECTOR3D** wird ein 3-dimensionaler Vektor gespeichert:

```
TYPE SMC_VECTOR3D:
STRUCT
       dX:LREAL;
       dY:LREAL;
       dZ:LREAL;
END_STRUCT
END_TYPE
```

SMC_VECTOR6D

In der in SMC_CNC.lib enthaltenen Struktur **SMC_VECTOR6D** wird ein 6-dimensionaler Vektor gespeichert:

```
TYPE SMC_VECTOR6D:
STRUCT
       dX:LREAL;
       dY:LREAL;
       dZ:LREAL;
       dA:LREAL;
       dB:LREAL;
       dC:LREAL;
END_STRUCT
END_TYPE
```

SMC_OUTQUEUE und ihre Funktionen

Die in SMC_CNC.lib enthaltenen **SMC_OUTQUEUE**-Struktur hilft, *GEOINFO*-Objekte in einer Liste definierter Größe zu verwalten.

```
TYPE SMC_OUTQUEUE :
STRUCT
       wOUTQUEUEStructID: WORD;
       pbyBuffer: POINTER TO BYTE;
       nSize: UDINT;
       nReadPos: UDINT;
       nWritePos: UDINT;
       bFull: BOOL;
       bEndOfList: BOOL;
       byGenerator: BYTE;
END_STRUCT
END_TYPE
```

Mittels der Variable *wOUTQUEUEStructID*, die einen festen Wert hat, überprüfen die Bausteine intern, ob die übergebene Strukturvariable vom Typ SMC_OutQueue ist.
Die Variable *byGenerator* beschreibt den Erzeuger der Queue. Diese Information benutzt der Interpolator, um zu überprüfen, ob der SMC_CheckVelocities-Baustein wie vorgeschrieben als letztes durchlaufen wurde. Folgende Werte sind festgelegt:

Erzeuger	**Wert**
SMC_NCDecoder	1

SMC_AvoidLoop	10
SMC_LimitCircularVelocity	11
SMC_RoundPath	12
SMC_SmoothPath	13
SMC_ToolCorr	14
SMC_RotateQueue2D	30
SMC_ScaleQueue3D	31
SMC_TranslateQueue3D	32
SMC_CheckVelocities	254
CNC-Editor	255

DieSoftMotion-Bibliothek bietet folgende Funktionen für die Handhabung eines SMC_OUTQUEUE-Strukturobjekts:

BOOL **SMC_RESTOREQUEUE**(Enable: BOOL, POQ: POINTER TO SMC_OUTQUEUE)

Diese Funktion restauriert eine bereits interpolierte bzw. anderweitig abgearbeitete Struktur. Dies ist nur dann möglich, wenn die Liste bei POQ die komplette Bahn enthalten kann.

POINTER TO SMC_OUTQUEUE **SMC_APPENDOBJ**(POQ: POINTER TO SMC_OUTQUEUE, PGI: POINTER TO SMC_GEOINFO)

Diese Funktion hängt ans Ende der Liste bei *POQ* das GEOINFO-Objekt, das bei *PGI* steht, sofern *OQ* richtig initialisiert und nicht schon vollständig belegt ist. Im Erfolgsfall ist der Rückgabewert ein Zeiger auf das neue Listen-Element, ansonsten 0.

BOOL **SMC_DELETEOBJ**(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Diese Funktion löscht das N-te Objekt aus der Liste bei *POQ*, wobei bei 0 zu zählen begonnen wird. Ist N-1 größer als die Anzahl der gespeicherten GEOINFO-Objekte, geschieht nichts und der Rückgabewert ist FALSE, sonst TRUE.

UINT **SMC_GETCOUNT**(POQ: POINTER TO SMC_OUTQUEUE)

Der Rückgabewert dieser Funktion beträgt die Anzahl der bei *POQ* gespeicherten Objekte.

POINTER TO SMC_GEOINFO GETOBJ(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Diese Funktion liefert – vorausgesetzt *POQ* ist richtig initialisiert und es existiert ein N-tes Objekt – einen Zeiger auf das N-te *GEOINFO*-Objekt aus der Liste bei *POQ*.

POINTER TO SMC_GEOINFO GETOBJFROMEND(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Diese Funktion liefert – vorausgesetzt *POQ* ist richtig initialisiert und es existieren mindestens *N+1* Elemente – einen Zeiger auf das vom Ende gerechnete *N*-te *GEOINFO*-Objekt aus der Liste bei *POQ*; für *N=0* wird also das letzte Listenelement zurückgegeben.

Initialisierung der Struktur:

Die SoftMotion-Bausteine *SMC_NCDecoder*, *SMC_SmoothPath*, *SMC_RoundPath*, *SMC_AvoidLoop* und *SMC_ToolCorr*, die einen Zeiger auf eine intern verwaltete OutQueue-Struktur ausgeben, übernehmen selbst die Initialisierung der Struktur. Die Bausteine *SMC_SmoothPath*, *SMC_RoundPath*, *SMC_ToolCorr*, *SMC_AvoidLoop* und *SMC_Interpolator* erwarten als Eingang einen Zeiger auf eine korrekte OutQueue-Liste. Wird diese "von Hand" programmiert und befüllt, so muss die korrekte Initialisierung selbst vorgenommen werden. Dazu müssen die ersten beiden Variablen (Buffer, Size) gesetzt werden. Im Übrigen empfiehlt es sich dringend, bei der Arbeit mit einer SMC_OUTQUEUE die obigen Funktionen zu benützen, und – nach einer Initialisierung - eine Änderung der anderen Variablen möglichst zu vermeiden.

Komponenten der Struktur:

pbyBuffer: POINTER TO BYTE

Hier steht die Adresse eines zusammenhängenden Speicherbereichs, der für die Speicherung der GEOINFO-Objekte vorgesehen ist. Dieser Speicher muss im IEC-Programm belegt werden und dessen Adresse dann in diese Variable geschrieben werden. Das Belegen im Deklarationsteil kann

beispielsweise mittels eines Byte-Arrays erfolgen (BUF: ARRAY[1..10000] OF BYTE; für einen 10000 Byte großen Speicherbereich).

nSize: UDINT

Hier muss die Größe des bei *Buffer* reservierten Speicherbereichs stehen.

nReadPos: UDINT

In dieser Variable steht relativ zum ersten Byte des reservierten Speicherbereichs die Adresse des ersten Listenelements.

nWritePos: UDINT

In dieser Variable steht relativ zum ersten Byte des reservierten Speicherbereichs die Adresse des ersten freien Bytes hinter der Liste.

bFull: BOOL

Ist die Liste bis auf den Speicher für drei weitere GEOINFO-Strukturen (Sicherheitspuffer) voll, so wird diese Variable von der Funktion APPENDOBJ auf TRUE gesetzt. DELETEOBJ setzt sie wieder auf FALSE, wenn Elemente der Liste gelöscht werden.

bEndOfList: BOOL

Die SoftMotion-Bausteine, die als Eingang eine OutQueue-Struktur haben, warten mit der Abarbeitung derselben immer, bis diese voll ist, damit beispielsweise kein Data-Underrun (SMC_Interpolator) auftritt. Da bei den letzten SMC_GEOINFO-Objekte einer Bahn die OutQueue nicht mehr voll ist, muss nach dem Ablegen des letzten SMC_GEOINFO-Objekts diese Variable auf TRUE gesetzt werden, damit mit der Verarbeitung fortgefahren werden kann. Ist die Liste danach leer und soll aber neu befüllt werden, muss EndOfList wieder auf FALSE gesetzt werden.

SMC_CNC_REF

In dieser Datenstruktur werden geparste G-Code-Dateien verwaltet:

```
TYPE SMC_CNC_REF :
STRUCT
       wCNCREFStructID: WORD := 16#BA56;
       nElements: UDINT;
       diReadPos: UDINT := 0;
       udiBuffer: UDINT := 16#FFFFFFFF;
       pgc: POINTER TO SMC_GCODE_WORD := 0;
       piStartPosition: SMC_POSINFO;
       strProgramName:STRING := '';
       bRestart: BOOL;
END_STRUCT
END_TYPE
```

Mittels der Variablen *wCNCREFStructID*, die einen festen Wert hat, überprüfen die Bausteine intern, ob die übergebene Strukturvariable vom Typ SMC_CNC_REF ist.
Die Variable *pgc* zeigt auf das erste SMC_GCODE_WORD. *nElements* enthält die Anzahl an SMC_GCODE_WORD-Strukturen bei *pgc*.
Die Startposition des CNC-Programms ist in *piStartPosition* gespeichert, seine Bezeichnung in *strProgramName*.

Die Variablen *diReadPos* und *udiBuffer* werden intern genutzt.

Die Variable bRestart wird vom Baustein SMC_NCDecoder gesetzt, wenn Sprünge im CNC-Programm (G20) verwendet werden und teilt dem Erzeuger der Datenstruktur mit, dass alle Zeiger frisch initialisiert und die Datenstruktur von Neuem erzeugt werden soll.

SMC_GCODE_WORD

In dieser Datenstruktur werden G-Code-Worte gespeichert:

```
TYPE SMC_GCODE_WORD :
STRUCT
       byLetter:BYTE:=0;
       fValue:LREAL:=0;
       diValue:DINT:=0;
       pAdr:POINTER TO BYTE:=0;
       byVarType:BYTE:=0;
END_STRUCT
END_TYPE
```

byLetter enthält den ASCII_Code des Buchstaben des Wortes, *fValue* und *diValue* dessen Wert als Gleitkomma- und Integerzahl. Werden statt festen Werten Variablen verwendet, so steht in pADR ein Zeiger auf diese Variable und in *byVarType* deren Typ:

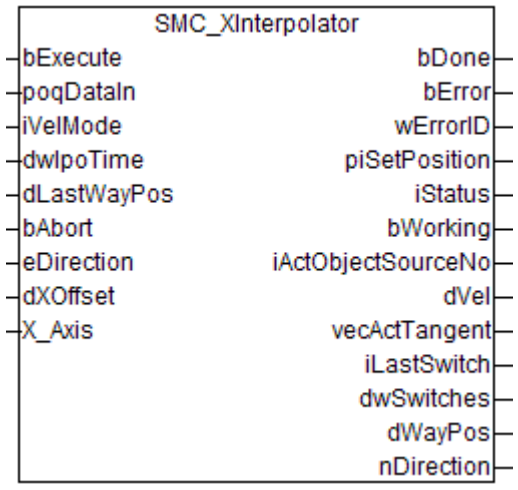
1	INT
2	BYTE
3	WORD
4	DINT
5	DWORD
6	REAL
14	SINT
15	USINT
16	UINT
17	UDINT
22	LREAL

6.6 Bahn-Kurvenscheiben mit SMC_XInterpolator

Der Baustein SMC_XInterpolator realisiert eine Mischung aus Kurvenscheibe und CNC. Man stelle sich vor, man möchte eine gewisse Form (beschrieben durch G-Code) aus einem Werkstück schneiden, wobei das Werkstück – von einem anderen Prozess – (z.B. entlang der X-Achse) bewegt wird und die anderen Achsen (Y, Z, etc.) gemäß der aktuellen Position des Werkstücks (X) und der Vorgabe aus der Bahnkontur gesteuert werden soll.

Die Bewegung des Werkstücks erfolgt stets in X-Richtung (andere Fälle können durch Drehung darauf abgebildet werden).

Der Baustein SMC_XInterpolator besitzt folgende Ein- und Ausgänge:



Eingänge des Bausteins:

bExecute: BOOL

Der Baustein führt einen Reset aus und beginnt mit der Interpolation, wenn dieser Eingang eine steigende Flanke aufweist.

poqDataIn: POINTER TO SMC_OUTQUEUE

Dieser Eingang zeigt auf das SMC_OUTQUEUE-Strukturobjekt, das die SMC_GEOINFO-Objekte der Bahn enthält, die interpoliert werden soll.

dLastWayPos: LREAL

Dieser Eingang gibt dem Anwender die Möglichkeit, die Wegstrecke, die der Interpolator abfährt, zu messen. Der Ausgang dWayPos ist die Summe aus dLastWayPos und der in diesem Zyklus zurückgelegten Distanz. Ist dLastWayPos=0, so enthält dWayPos die Länge des aktuellen Wegabschnitts. Setzt man dLastWayPos gleich dem Ausgang dWayPos, so wird dWayPos stets um das aktuelle Wegstück inkrementiert, und man erhält die insgesamt gefahrene Wegstrecke. Dabei kann dLastWayPos jederzeit auf 0 oder einen anderen Wert (zurück)gesetzt werden.

bAbort: BOOL

Dieser Eingang bricht die Bearbeitung einer Kontur ab.

eDirection: MC_Direction

Hier steht, ob das Werkstück entlang der X-Achse in positiver (positive) oder negativer (negative) Richtung bewegt wird. Andere Werte sind nicht zulässig.

dXOffset: LREAL

Offset zur Position der X-Achse.

X_Axis: AXIS_REF

X-Achse, Position des Werkstücks.

Ausgänge des Bausteins:

bDone: BOOL

Diese Variable wird auf TRUE gesetzt, wenn die Eingangsdaten aus DataIn vollständig verarbeitet sind. Hernach führt der Baustein bis zu einem Reset keine Aktionen mehr durch. Steht der bExecute-Eingang auf FALSE wird bDone wieder auf FALSE gesetzt.

bError: BOOL

Sollte ein Fehler auftreten wird dieser Wert TRUE.

wErrorID: SMC_ERROR (INT)

Dieser Enum-Ausgang beschreibt ggf. einen Fehler, der beim Interpolieren aufgefallen ist. Nach einem Fehler wird die Abarbeitung bis zu einem Reset gestoppt.

piSetPosition: SMC_POSINFO

In *Set_Position* steht die gemäß den Vorgaben berechnete Soll-Position. *Set_Position* ist eine SMC_POSINFO-Struktur und enthält nicht nur die kartesischen Koordinaten des anzufahrenden Bahnpunktes, sondern auch die Stellung der Zusatzachsen.

iStatus: SMC_INT_STATUS (INT)

In dieser Enum-Variable wird der aktuelle Status des Bausteins ausgegeben. Dieser kann sein:

IPO_UNKNOWN	0	Interner Zustand. Zustand darf nach vollständigem SMC_Interpolator-Durchlauf nicht auftreten.
IPO_ACCEL	2	Baustein befindet sich in Beschleunigungsphase.
IPO_CONSTANT	3	Baustein fährt mit konstanter Geschwindigkeit.
IPO_DECEL	4	Baustein befindet sich in Bremsphase.
IPO_FINISHED	5	Abarbeitung der *GEOINFO*-Liste abgeschlossen. Nachträglich in *DataIn* ankommende GEOINFO-Objekte werden nicht mehr bearbeitet.

bWorking: BOOL

Ausgang wird TRUE, sobald die Abarbeitung der Liste begonnen und noch nicht abgeschlossen wurde (IPO_ACCEL oder IPO_CONSTANT oder IPO_DECEL oder IPO_WAIT). Ansonsten steht *Working* auf FALSE.

iActObjectSourceNo: INT

Hier steht der Eintrag *SourceLine_Nr* des aktuell befahrenen GEOINFO-Objekts aus der *DataIn*-Queue. Arbeitet der SMC_Interpolator nicht (mehr) (*Working* = FALSE), steht hier –1.

dVel: LREAL

In dieser Variable steht die aktuelle Geschwindigkeit, die sich ergibt, wenn sich ein Objekt in der Zeit Ipo_Time von der vorigen Raumkoordinate nach *Set_Position* bewegt.

vecActTangent: SMC_VECTOR3D

In dieser Struktur befindet sich die Bahnrichtung im Punkt *Set_Position*. Für den Fall *Vel* = 0 befinden sich auch in *Act_Tangent* lauter Nullen.

iLastSwitch: INT

In dieser Variable steht die Nummer der zuletzt passierten Hilfsmarke. Sollten in einem Zyklus mehrere Hilfsmarken überlaufen werden, wird nur jeweils die letzte ausgegeben.

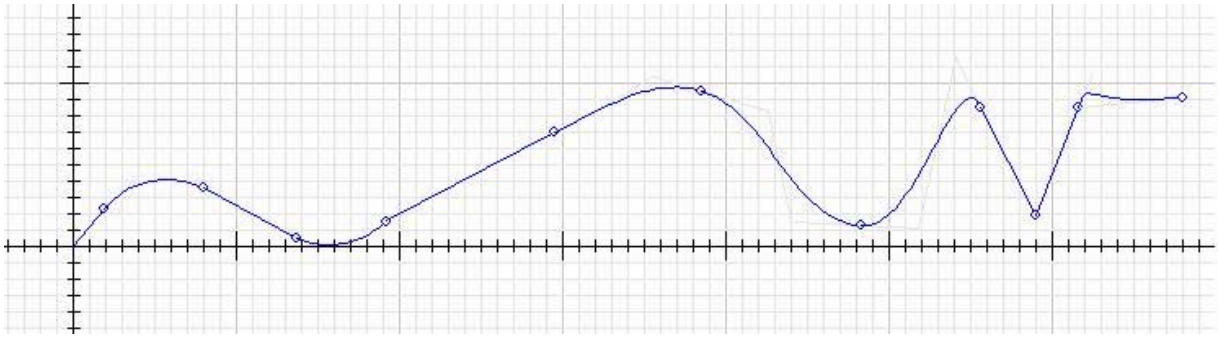
dwSwitches: DWORD

Dieses DWORD enthält den momentanen Schaltzustand aller der Hilfsmarken zwischen 1 und 32. Bit0 des DWORDs steht für Hilfsmarke1, Bit31 für Hilfsmarke 32. Anders als bei iLastHelpMark kann so ausgeschlossen werden, dass eine Hilfsmarke verloren geht.

dWayPos: LREAL

Beschreibung siehe Eingang dLastWayPos.

Der XInterpolator sucht, wenn er aktiv ist, zur aktuellen X-Position die passende Position auf der vorgegebenen Bahn und gibt den entsprechenden Bahn-Punkt in piSetPosition aus. Damit dies sprungfrei passieren kann, muss zu einer X-Position stets eine eindeutige Bahnposition existieren. Folgende Bahn wäre z.B. zulässig:



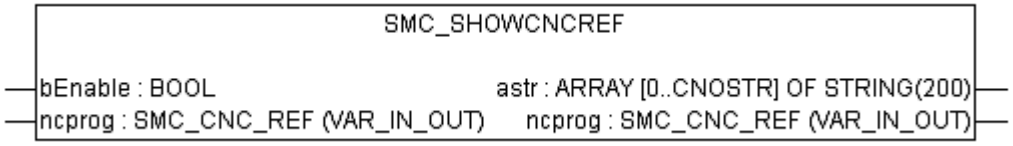
7 Die Bibliothek SM_CNCDiagnostic.lib

Diese Bibliothek enthält Hilfsmittel, die gerade während der Inbetriebnahme-Phase von großem Nutzen sein können, da sie dem Applikateur helfen, die Daten, die zwischen den Bausteinen ausgetauscht werden, darzustellen.

7.1 Funktionsbausteine zur Analyse von SMC_CNC_REF-Daten

7.1.1 Der Funktionsblock SMC_ShowCNCREF

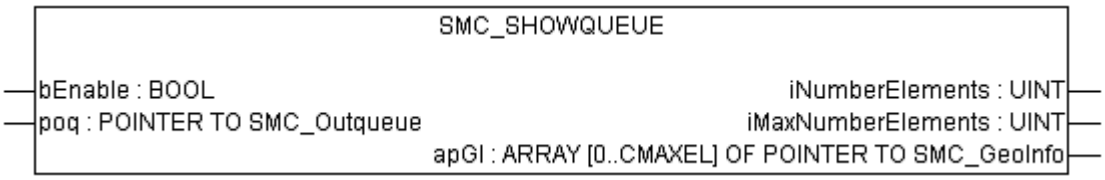
Dieser Funktionsblock kann die ersten zehn Zeilen eines NC-Programms, welches in Form einer Datenstruktur SMC_CNC_REF vorliegt, textuell (Din66025) darstellen. Als Ausgang ist ein Array of String angelegt (cnostr), welches die Textzeilen enthält. Das Visualisierungs-Template VISU_SMC_ShowCNCRef kann diese Ausgaben darstellen.



7.2 Funktionsbausteine zur Analyse von SMC_OutQueue-Daten

7.2.1 Der Funktionsblock SMC_ShowQueue

Dieser Funktionsblock stellt die ersten zehn SMC_GeoInfo-Objekte einer OutQueue in Form eines ARRAY OF POINTER TO SMC_GeoInfo zur Verfügung. Einige wichtige Elemente daraus können vom Visualisierungs-Template VISU_SMC_ShowQueue dargestellt werden. Dazu zählen: Objektnummer, Zeilennummer, Objekttyp, Startposition (X/Y/Z), Endposition (X/Y/Z), Sollgeschwindigkeit und Endgeschwindigkeit.



8 Die Bibliothek SM_Trafo.lib

8.1 Überblick

Diese **Bausteinbibliothek** ist eine Erweiterung der SM_CNC.lib und stellt Bausteine zur Verfügung, die dem IEC-Programmierer die Transformation von Welt- zu Maschinenkoordinaten und die Kontrolle der Achsen erleichtert.

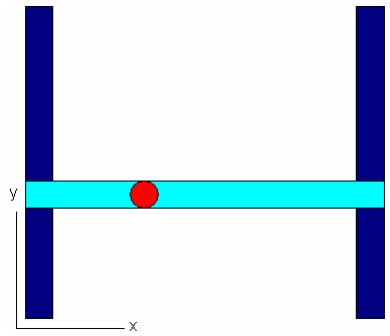
Dazu enthält sie zum einen Bausteine, die die Antriebe mit Sollwerten kontrollieren, gleichzeitig die Sollwerte überwachen und Sprünge detektieren können.

Außerdem enthält sie Bausteine für die mathematische Vorwärts- und Rückwärts-Transformation für einige gängige Kinematiken. Instanzen der Vorwärtstransformations-Bausteine können mit ebenfalls enthaltenen Visualisierungs-Templates verknüpft werden, die ein sofortiges und einfaches Darstellen ermöglichen.

8.2 Transformations-Funktionsblöcke

Die zu einer speziellen Kinematik gehörenden Bausteine gehören paarweise zusammen, wobei der Baustein, der mit SMC_TRAFO_<Kinematik> beginnt, eine Rückwärtsrechnung, der mit SMC_TRAFOF_<Kinematik> eine Vorwärtsrechnung durchführt. Jede Instanz eines SMC_TRAFOF_<Kinematik>-Bausteins kann mit einem Visualisierungs-Template mit dem Namen SMC_VISU_<Kinematik> verknüpft werden.

8.2.1 Portal-Systeme



Die Bausteine addieren, da für Portalsysteme keine Transformation durchgeführt werden muss, lediglich Offsets auf die x-, y- und z-Achsen.

SMC_TRAFO_Gantry2

pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

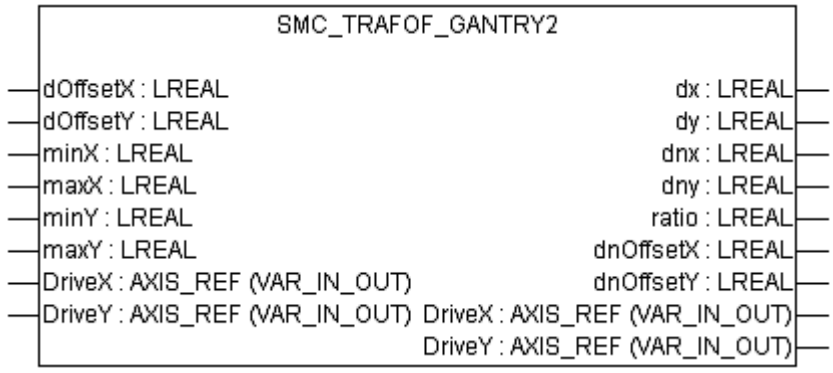
dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse.

dx, dy: LREAL

Sollwerte für x- und y-Achse.

SMC_TRAFOF_Gantry2



dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse. Gleiche Werte wie bei SMC_TRAFO_Gantry2.

minX, maxX, minY, maxY: LREAL

Bewegungsbereich (für Visualisierung).

DriveX, DriveY: AXIS_REF

x-, y-Achse.

dx, dy: LREAL

x-, y-Position in Weltkoordinaten.

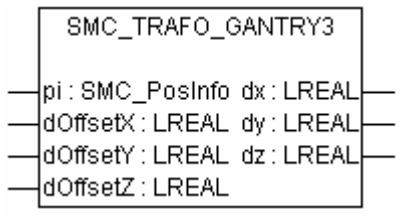
dnx, dny, dnOffsetX, dnOffsetY: LREAL

genormte x- und y-Position [0..1] und Offsets (für Visualisierung).

ratio: LREAL

Verhältnis zwischen x-Intervall und y-Intervall (für Visualisierung).

SMC_TRAFO_Gantry3



pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

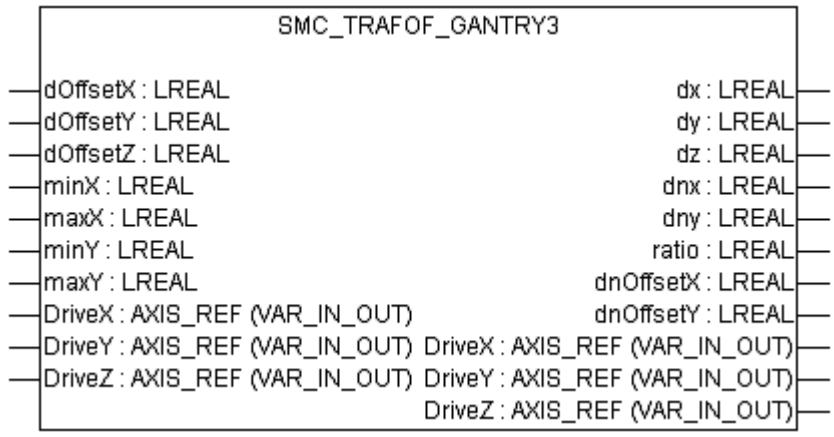
dOffsetX, dOffsetY, dOffsetZ: LREAL

Offset für x-, y- und z-Achse.

dx, dy, dz: LREAL

Sollwerte für x-, y- und z-Achse.

SMC_TRAFOF_Gantry3



dOffsetX, dOffsetY, dOffsetZ: LREAL

Offset für x-, y- und z-Achse. Gleiche Werte wie bei SMC_TRAFO_Gantry3.

minX, maxX, minY, maxY: LREAL

Bewegungsbereich (für Visualisierung).

DriveX, DriveY, DriveZ: AXIS_REF

x-, y-, z-Achse.

dx, dy, dz: LREAL

x-, y-, z-Position in Weltkoordinaten.

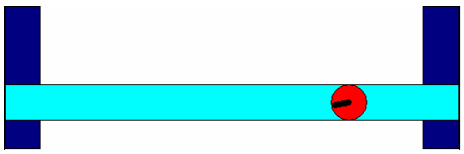
dnx, dny, dnOffsetX, dnOffsetY: LREAL

genormte x- und y-Position [0..1] und Offset (für Visualisierung).

ratio: LREAL

Verhältnis zwischen x-Intervall und y-Intervall (für Visualisierung).

GantryCutter



Die Bausteine SMC_TRAFO<n>_Gantry<n> existieren auch als SMC_TRAFO<n>_GantryCutter<n>. Diese Bausteine enthalten Transformationen für Portalsysteme mit einer Rotationsachse, die so gesteuert wird, dass sie entlang der aktuellen Bahntangente zeigt. Sie enthalten als zusätzliche Eingänge die Rotationsachse (DriveR), die als rotatorische Achse mit der Periode 360 angelegt sein muss, einen Offset (dOffsetX) und die Drehrichtung (iDirectionR). Der Rückwärtstransformationsbaustein benötigt zudem den Vektor der aktuellen Bahntangente (v), der als Ausgang am Interpolator anliegt.

SMC_TRAFOV_Gantry

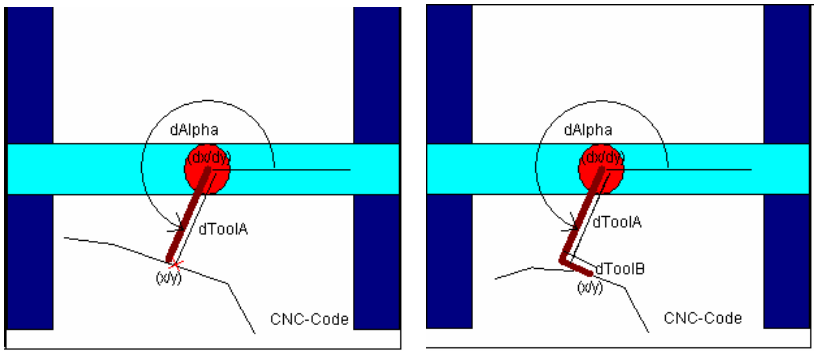
Einige der oben beschriebenen Rückwärtstransformationen sind in einer Version vorhanden, in welcher auch die Bahn-Geschwindigkeit und –Richtung als Steuergröße für die Achsen verwendet wird. Diese beginnen mit „SMC_TRAFOV_“ anstelle von „SMC_TRAFO_“. Sie benötigen als zusätzlichen Eingänge die Bahntangente (v) und -geschwindigkeit (dVel) aus dem Interpolator. Neben den Sollpositionen geben sie die Sollgeschwindigkeiten (dvx/dvy/dvz) aus. Der Vorteile dieser Methode ist, dass der Schleppfehler im Antrieb durch die Geschwindigkeitsvorsteuerung – vorausgesetzt der Antrieb unterstützt diesen Modus – minimiert werden kann. Jede Achse sollte deshalb über den Baustein SMC_ControlAxisByPosVel gesteuert werden.

8.2.2 Portal-Systeme mit Werkzeugversatz

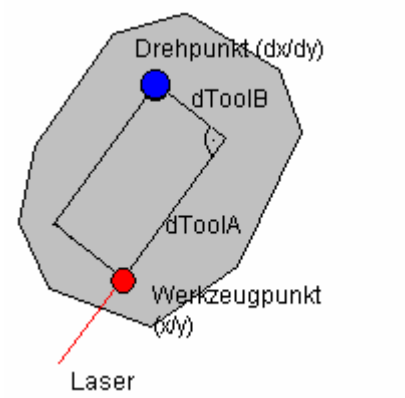
Es gibt Maschinenportale, deren xyz-Position nicht mit dem Werkzeug-Eingriffspunkt übereinstimmt, da dieser nicht axial in der z-Achse liegt, sondern versetzt liegt. Kann die z-Achse nicht gedreht werden, ist das ein konstanter xy-Offset, der als solcher in die Standard-Gantry-Transformation eingebracht werden kann.

Ist allerdings eine Drehachse um z im Spiel, handelt es sich nicht um einen konstanten Offset; die Verschiebung ist abhängig von der Stellung der C-Achse.

Zu unterscheiden ist, ob das Werkzeug als Gerade (dann, wenn der Vektor zwischen Werkzeugeingriffspunkt und Achse und die gewünschte Ausrichtung des Werkzeugs übereinstimmen) (->SMC_TRAFO_Gantry2Tool1) oder als Parallelogramm bzw. rechtwinkliges Dreieck angenähert werden muss (SMC_TRAFO_Gantry2Tool2):



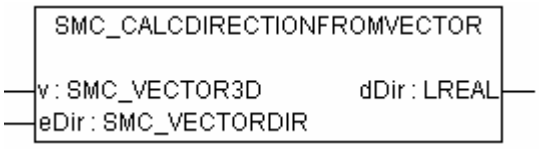
In diesem Beispiel kann das Werkzeug nicht als Gerade, sondern muss als rechtwinkliges Dreieck angenähert werden:



Prinzipiell kann anstelle der eindimensionalen Transformation mit Werkzeugversatz (Werkzeug kann als Gerade angenähert werden) auch die Bahn mittels SMC_Toolcorr entsprechend moduliert werden. Der Unterschied zwischen beiden Methoden liegt in der Geschwindigkeit des Werkzeugpunkts. Während bei der Lösung mit SMC_ToolCorr die Geschwindigkeit des Drehpunktes gemäß den Vorgaben aus dem CNC-Programm (F, E) gesteuert wird (wobei die des Werkzeugpunktes variieren kann), wird bei der Verwendung dieser Transformation die Geschwindigkeit des Werkzeugpunktes durch das CNC-Programm festgelegt.

Für die Berechnung der Ausrichtung des Werkzeugs (dAlpha) kann folgende Funktion verwendet werden:

SMC_CalcDirectionFromVector



v**: SMC_VECTOR3D**

Der Eingangsvektor v ist typischerweise der Interpolatorausgang vecActTangent.

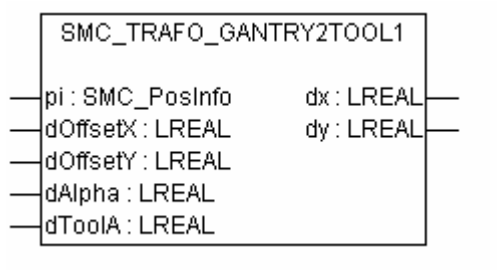
eDir: SMC_VECTORDIR

In eDir wird spezifiziert, ob die Richtung tangential zur Bahn (SMC_tangential), entgegengesetzt (SMC_opp_tangential) oder orthogonal dazu (SMC_orthogonal_r (rechts der Bahntangente) bzw. SMC_orthogonal_l (links der Bahntangente)) berechnet werden soll.

dDir: LREAL

Die Ausgabe dDir ist in Winkelgrad, und bleibt für Phasen, in denen der Interpolator steht (v ist Nullvektor) konstant. eDir wird meist als Sollwert (SMC_ControlAxisByPos) für die Richtungsachse und als Eingang dAlpha für die Transformation verwendet.

SMC_TRAFO_Gantry2Tool1



pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse.

dAlpha: LREAL

Ausrichtung des Werkzeugs in Winkelgrad.

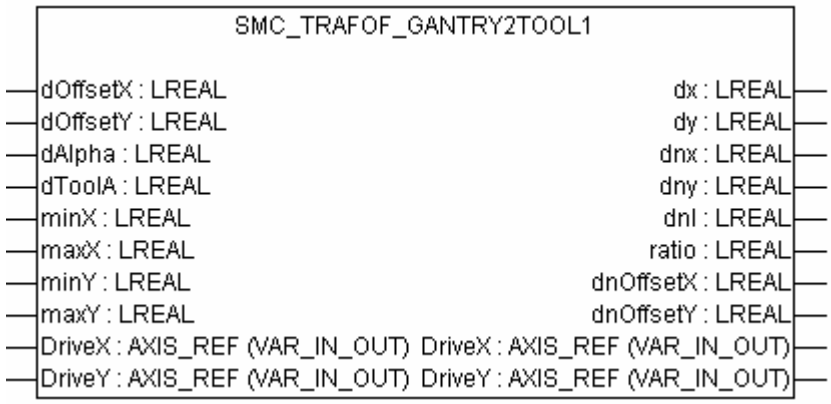
dToolA: LREAL

Länge des Werkzeugs; Abstand zwischen Drehpunkt und Werkzeugpunkt.

dx, dy: LREAL

Sollwerte für x- und y-Achse.

SMC_TRAFOF_Gantry2Tool1



dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse. Gleiche Werte wie bei SMC_TRAFO_Gantry2.

dAlpha: LREAL

Ausrichtung des Werkzeugs in Winkelgrad.

dToolA: LREAL

Länge des Werkzeugs; Abstand zwischen Drehpunkt und Werkzeugpunkt.

minX, maxX, minY, maxY: LREAL

Bewegungsbereich (für Visualisierung).

DriveX, DriveY: AXIS_REF

x-, y-Achse.

dx, dy: LREAL

x-, y-Position in Weltkoordinaten.

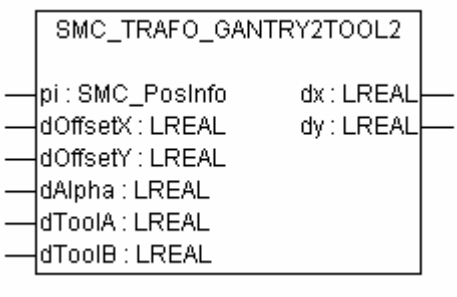
dnx, dny, dnl, dnOffsetX, dnOffsetY: LREAL

genormte x- und y-Position [0..1], Werkzeuglänge und Offsets (für Visualisierung).

ratio: LREAL

Verhältnis zwischen x-Intervall und y-Intervall (für Visualisierung).

SMC_TRAFO_Gantry2Tool2



pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse.

dAlpha: LREAL

Ausrichtung des Werkzeugs in Winkelgrad.

dToolA, dToolB: LREAL

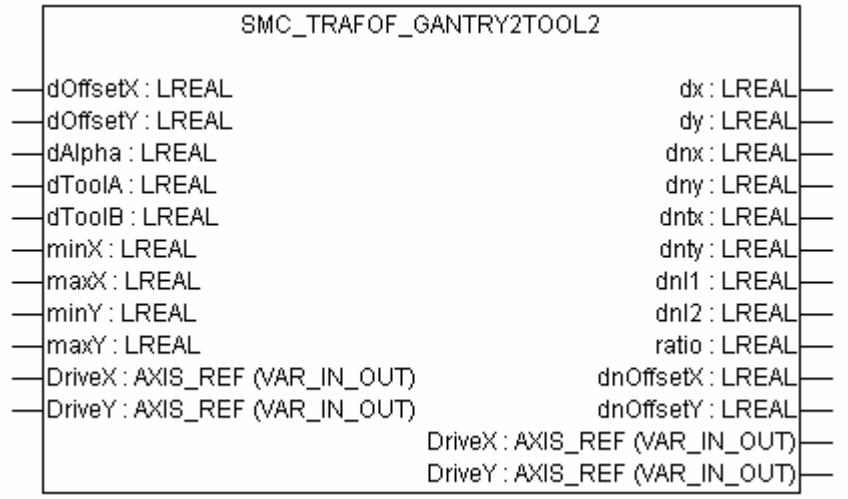
Kathetenlängen des rechtwinkligen Dreiecks, welches zwischen Drehpunkt und Werkzeugpunkt liegt. Dabei ist dToolA die Länge der zur Bahn tangentialen, dToolB die Länge der zur Bahn orthogonalen Kathete.

Ist dToolB positiv, so ist der Werkzeugpunkt (x/y) in Werkzeugrichtung gesehen links verschoben, ansonsten rechts.

dx, dy: LREAL

Sollwerte für x- und y-Achse.

SMC_TRAFOF_Gantry2Tool2



dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse. Gleiche Werte wie bei SMC_TRAFO_Gantry2.

dAlpha: LREAL

Ausrichtung des Werkzeugs in Winkelgrad.

dToolA, dToolB: LREAL

Länge des Werkzeugs; Abstand zwischen Drehpunkt und Werkzeugpunkt.

minX, maxX, minY, maxY: LREAL

Bewegungsbereich (für Visualisierung).

DriveX, DriveY: AXIS_REF

x-, y-Achse.

dx, dy: LREAL

x-, y-Position in Weltkoordinaten.

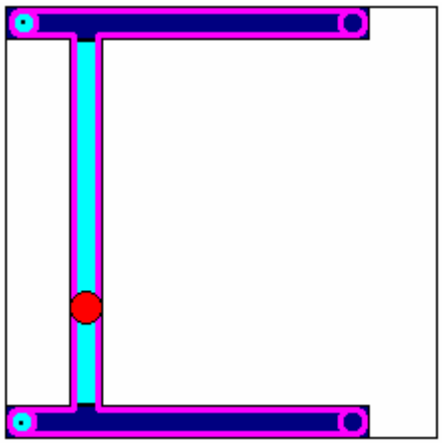
dnx, dny, dnl1, dnl2, dnOffsetX, dnOffsetY: LREAL

genormte x- und y-Position [0..1], Werkzeuglänge und Offsets (für Visualisierung).

ratio: LREAL

Verhältnis zwischen x-Intervall und y-Intervall (für Visualisierung).

8.2.3 H-Portal-System mit stationären Antrieben

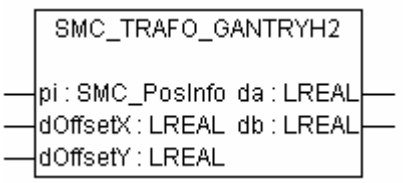


Dieses kinematische System ähnelt den zuvor beschriebenen Portal-Systemen, wobei hier die Antriebe stationär angebracht sind, und den Schlitten und die Y-Achse über einen mehrfach umgelenkten Riemen (im Bild pink) bewegen.

Die Transformation ist für folgende Antriebskonfigurationen ausgelegt; andere Konfigurationen können durch Vertauschen von x und y erreicht werden:

Man beachte, dass für diese Transformation eine spezielle Referenzfahrt nötig ist. Will man, dass einen Bewegung in Y-Richtung erfolgt, müssen die Antriebe A und B gleichläufig bewegt werden; für eine reine X-Bewegung sind diese gegenläufig zu drehen. Hat man die Referenzposition gefunden, verwendet man die vom Vorwärtstransformations-FB errechneten x- und y-Werte als Offset (dOffsetX und dOffsetY).

SMC_TRAFO_GantryH2



pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

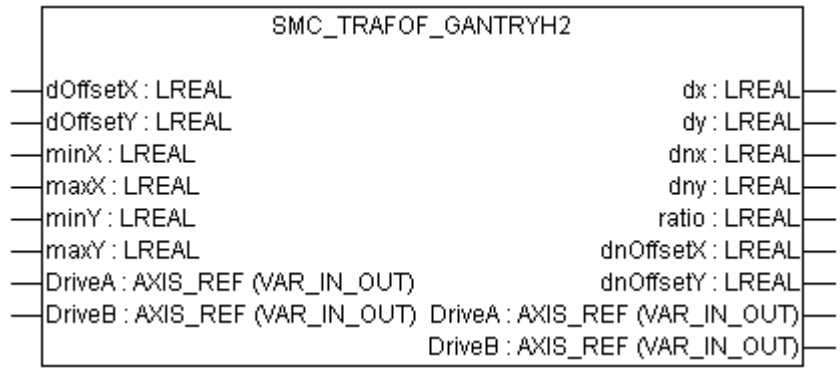
dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse.

da, db: LREAL

Sollwerte für A- und B-Achse.

SMC_TRAFOF_GantryH2



dOffsetX, dOffsetY: LREAL

Offset für x- und y-Achse. Gleiche Werte wie bei SMC_TRAFO_GantryH2.

minX, maxX, minY, maxY: LREAL

Bewegungsbereich (für Visualisierung).

DriveA, DriveB: AXIS_REF

A-, B-Achse.

dx, dy: LREAL

x-, y-Position in Weltkoordinaten.

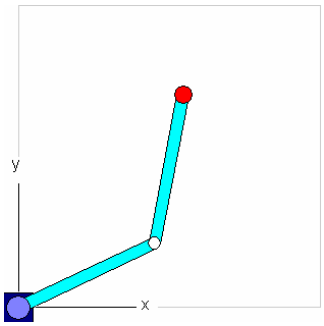
dnx, dny, dnOffsetX, dnOffsetY: LREAL

genormte x- und y-Position [0..1] und Offsets (für Visualisierung).

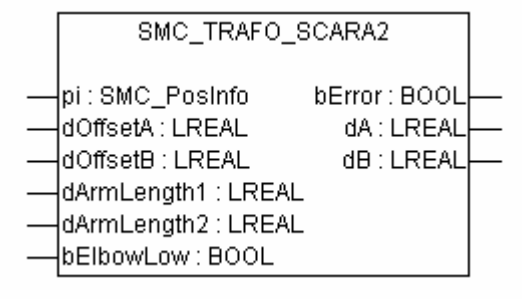
ratio: LREAL

Verhältnis zwischen x-Intervall und y-Intervall (für Visualisierung).

8.2.4 2-Gelenkige Scara-Systeme



SMC_TRAFO_Scara2



pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

dOffsetA, dOffsetB: LREAL

Offset für A- und B-Achse.

dArmLength1, dArmLength2: LREAL

Länge des ersten und zweiten Arms.

bElbowLow: BOOL

Ellbogen nach unten (TRUE) bzw. oben (FALSE)

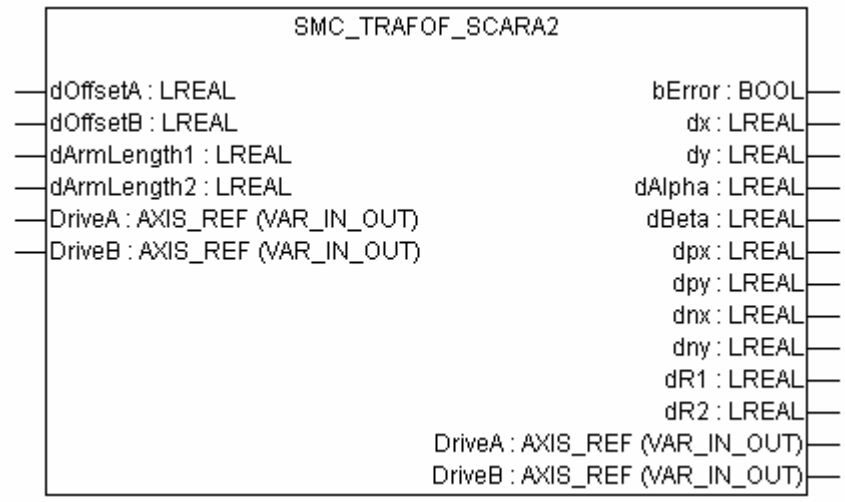
bError BOOL

TRUE: Unzulässige Werte.

dA, dB: LREAL

Achsstellung A- bzw. B-Achse.

SMC_TRAFOF_Scara2



dOffsetA, dOffsetB: LREAL

Offset für A- und B-Achse. Gleiche Werte wie bei SMC_TRAFO_Scara2.

dArmLength1, dArmLength2: LREAL

Länge des ersten und zweiten Arms.

DriveA, DriveB: AXIS_REF

A-, B-Achse.

bError BOOL

TRUE: Unzulässige Werte.

dx, dy: LREAL

x-, y-Position in Weltkoordinaten.

dAlpha, dBeta: LREAL

Gelenkwinkel (Achsstellungen ohne Offset). (für Visualisierung)

dpx, dpy: LREAL

Genormte Position des 1. Gelenks.]-1..1[(für Visualisierung)

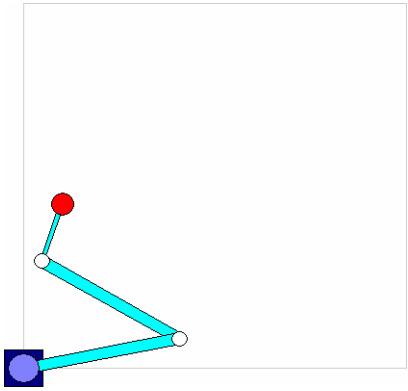
dnx, dny: LREAL

Genormte Position des Manipulators.]-1..1[(für Visualisierung)

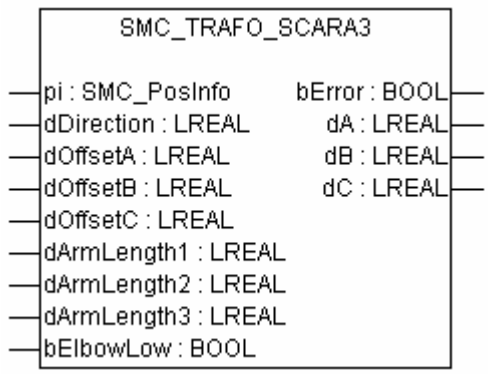
dR1, dR2: LREAL

Relative Armlängen. dR1+dR2 = 1.]0..1[(für Visualisierung)

8.2.5 3-Gelenkige Scara-Systeme



SMC_TRAFO_Scara3



pi: SMC_PosInfo

Soll-Positionsvektor. Ausgang des Interpolators.

dDirection: LREAL

Richtungswinkel des letzten Gelenks in Grad. (0° W, 90° N)

dOffsetA, dOffsetB, dOffsetC: LREAL

Offset für A-, B- und C-Achse.

dArmLength1, dArmLength2, dArmLength3: LREAL

Länge der Arme.

bElbowLow: BOOL

Ellbogen (1. und 2. Gelenk) nach unten (TRUE) bzw. oben (FALSE)

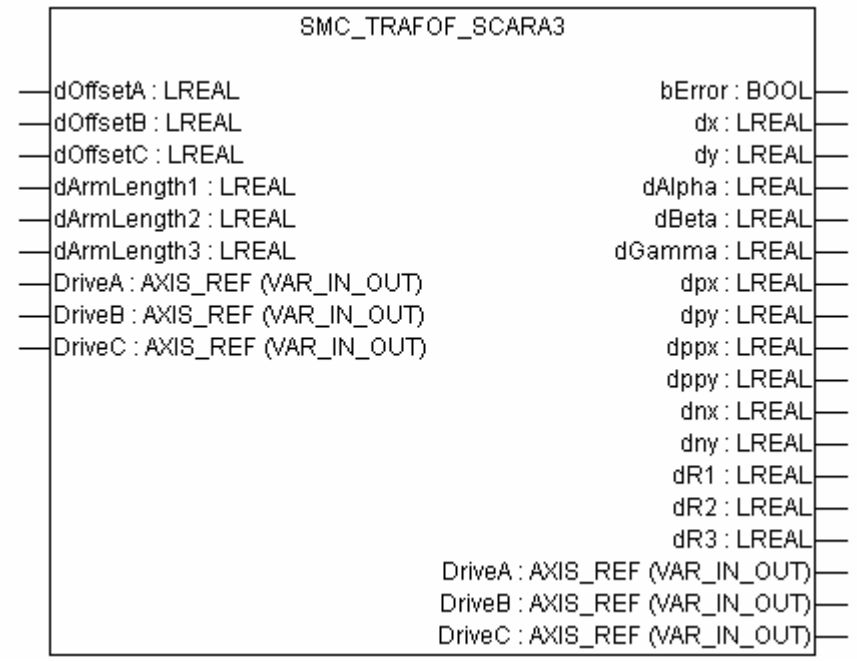
bError BOOL

TRUE: Unzulässige Werte.

dA, dB, dC: LREAL

Achsstellung A-, B-, C-Achse.

SMC_TRAFOF_Scara3



dOffsetA, dOffsetB, dOffsetC: LREAL

Offset für A-, B- und C-Achse. Gleiche Werte wie bei SMC_TRAFO_Scara3.

dArmLength1, dArmLength2, dArmLength3: LREAL

Länge der Arme.

DriveA, DriveB, DriveC: AXIS_REF

A-, B- und C-Achse.

bError BOOL

TRUE: Unzulässige Werte.

dx, dy: LREAL

x-, y-Position in Weltkoordinaten.

dAlpha, dBeta, dGamma: LREAL

Gelenkwinkel (Achsstellungen ohne Offset). (für Visualisierung)

dpx, dpy: LREAL

Genormte Position des 1. Gelenks.]-1..1[(für Visualisierung)

dppx, dppy: LREAL

Genormte Position des 2. Gelenks.]-1..1[(für Visualisierung)

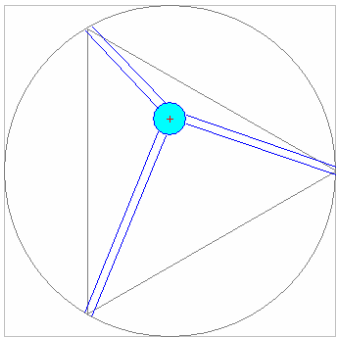
dnx, dny: LREAL

Genormte Position des Manipulators.]-1..1[(für Visualisierung)

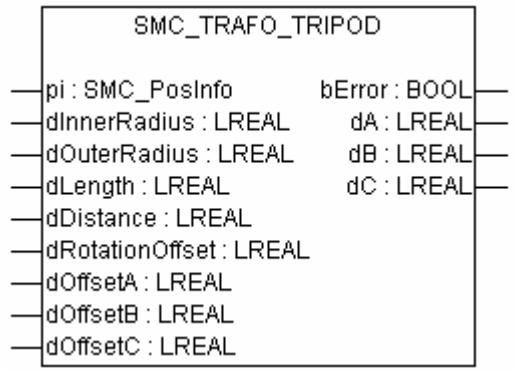
dR1, dR2, dR3: LREAL

Relative Armlängen. dR1+dR2+dR3 = 1.]0..1[(für Visualisierung)

8.2.6 Parallel-Kinematiken



SMC_TRAFO_Tripod



pi: SMC_PosInfo

Soll-Positionsvektor. Position des Mittelpunkts des inneren Rings. Ausgang des Interpolators.

dInnerRadius: LREAL

Radius des inneren Rings

dOuterRadius: LREAL

Radius des äusseren Rings

dLength: LREAL

Stablängen

dDistance: LREAL

Abstand zwischen zwei zusammengehörenden Stäben am äusseren und inneren Ring

dRotationOffset: LREAL

Richtung in Grad (mathem. Sinn), in welcher die Achse A vom Ursprung (0/0) aus gesehen liegt.

dOffsetA, dOffsetB, dOffsetC: LREAL

Offset der einzelnen Achsen

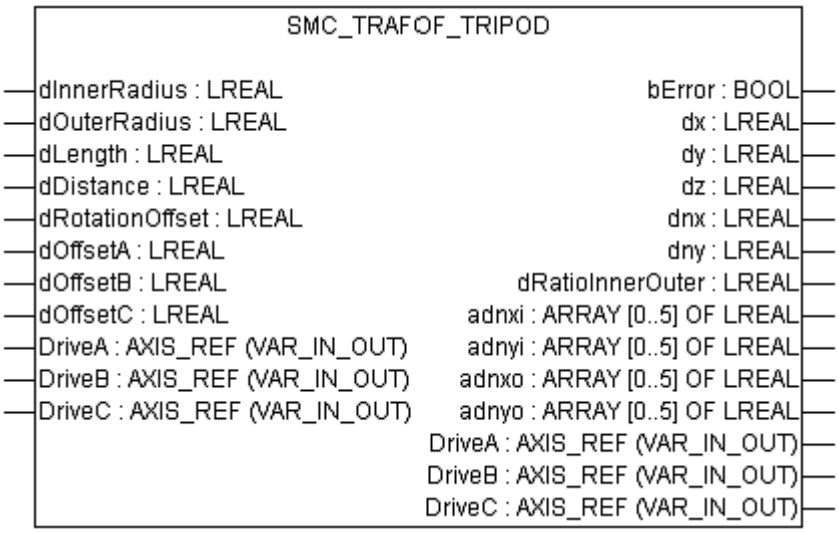
bError BOOL

TRUE: Unzulässige Werte. Arbeitsbereich verlassen

dA, dB, dC: LREAL

Achsstellung A-, B-, C-Achse.

SMC_TRAFOF_Tripod



dInnerRadius, dOuterRadius, dLength, dDistance, dRotationOffset, dOffsetA, dOffsetB, dOffsetC: LREAL

siehe SMC_TRAFO_TRIPOD

DriveA, DriveB, DriveC: AXIS_REF

A-, B- und C-Achse.

bError BOOL

TRUE: Unzulässige Werte.

dx, dy, dz: LREAL

x-, y-, z-Position des Zentrums des inneren Rings in Weltkoordinaten.

dnx, dny: LREAL

Genormte Position des Manipulators. (für Visualisierung)

dRatioInnerOuter: LREAL

Verhältnis der Radien des inneren zum äusseren Ring. (für Visualisierung)

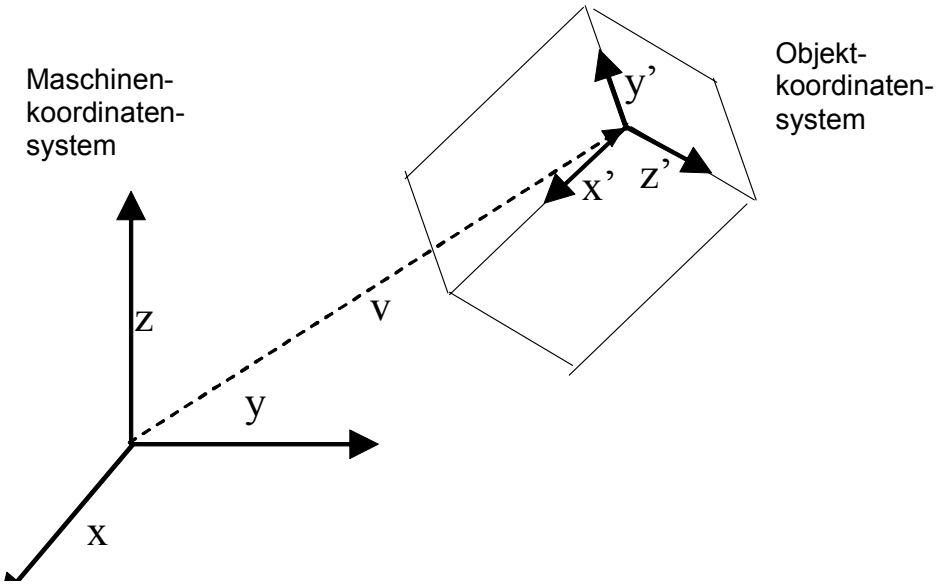
adnxi, adnyi, adnxo, adnyo: ARRAY[0..5] OF LREAL

Genormte Anfangs- und End-Position der Stäbe. (für Visualisierung)

8.3 Räumliche Transformationen

Manchmal unterscheiden sich die Koordinatensysteme der Maschine und des Bearbeitungsstückes. Obige Transformationsbausteine wandeln die kartesischen Koordinaten des Werkzeugpunkts in dessen Maschinenkoordinaten um. Vorausgehend kann es aber nötig sein (falls das Werkstück nicht exakt gemäß des CNC-Programms ausgerichtet ist), dass die vom Interpolator errechneten Bahnkoordinaten vor der Übergabe an die Maschinen-Transformation transformiert werden müssen.

Man stelle sich ein gewöhnliches Portal (X/Y/Z) vor. Den Werkzeugpunkt des Portals muss man nun auf der Fläche eines Werkstückes, welches schräg im Raum liegt, verfahren:



Es gibt mehrere Möglichkeiten, den Bezug zwischen zwei Koordinatensystemen zu beschreiben. Eine Koordinatentransformation besteht immer aus räumlicher Translation und räumlicher Rotation. Die Translation beschreibt man dabei mit einem dreidimensionalen Vektor, die Rotation entweder durch drei Winkel (z.B. YawPitchRoll) oder durch die drei Einheitsvektoren des neuen (Objekt-)Koordinatensystems x', y', z'.

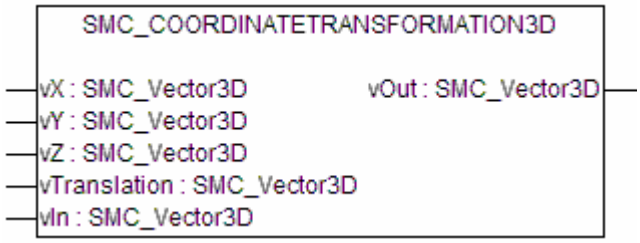
Wird die Methode der drei Rotationswinkel gewählt, können diese z.B. nach der Roll/Pitch/Yaw (RPY) Konvention definiert. Dabei geht das neue Koordinatensystem aus dem alten durch Drehung um verschiedene Achsen hervor. Man kann sich das Verfahren RPY (α, β, γ) auf zweierlei Arten, die das identische Ergenis liefern, vorstellen:

1. Ausgehend vom Koordinatensystem (x,y,z) dreht man das Koordinatensystem um die z-Achse um den Winkel γ im mathematisch positiven Sinn. Dadurch entsteht das neue Koordinatensystem (x1, y1, z1=z). Bei diesem hält man nun die Achse y1 fest und dreht das Koordinatensystem um β, woraus (x2, y2=y1, z2) entsteht. Schließlich dreht man dieses Koordinatensystem um x2 um den Winkel α. Dadurch erhält man (x'=x2, y', z').

2. Ausgehend vom Koordinatensystem (x,y,z) dreht man das Koordinatensystem um die x-Achse um den Winkel α. Das so entstandene Koordinatensystem (xa=x, ya, za) dreht man um die ursprüngliche y-Achse (nicht ya!!) um β (xb, yb, zb) udn anschließend um die ursprüngliche z-Achse um γ, wodurch (x', y', z') entsteht.

SMC_CoordinateTransformation 3D

Dieser Baustein berechnet die Koordinaten einer (im alten Koordinatensystem vorliegenden) Position bzgl. des neuen Koordinatensystems. Dafür wird die Koordinatentransformation des neuen bzgl. des alten Koordinatensystems mittels Translationsvektor und neuen Einheitsvektoren vorgegeben



vX, vY, vZ: SMC_Vector3D

Einheitsvektoren des neuen Koordinatensystems bzgl. des alten **.**

vTranslation: SMC_Vector3D

Verschiebungsvektor. Vektor vom alten Koordinatenursprung zum neuen bzgl des alten Koordinatensystems.

vIn: SMC_Vector3D

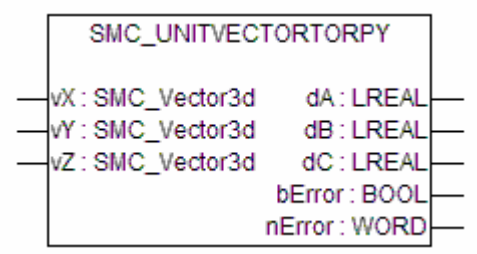
zu transformierende Position.

vOut: SMC_Vector3D

transformierte Position.

SMC_UnitVectorToRPY

Baustein zur Errechnung der RPY-Winkel bzgl. des alten Koordinatensystems aus den Einheitsvektoren des neuen Koordinatensystems.



vX, vY, vZ: SMC_Vector3D

Einheitsvektoren des neuen Koordinatensystems bzgl. des alten.

dA, dB, dC: LREAL

RPY-Winkel im Bogenmaß.

bError: BOOL

unmögliche Eingabe-Werte.

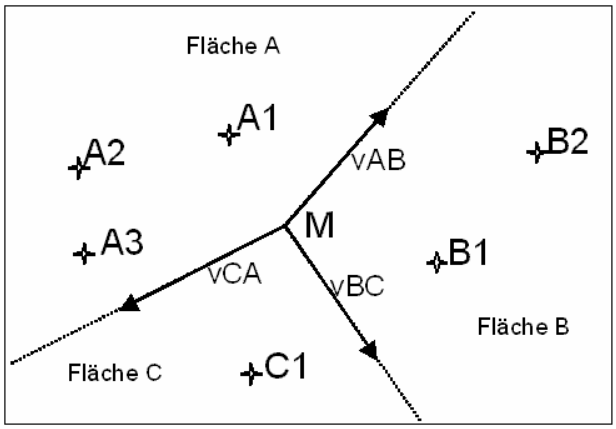
nError: WORD

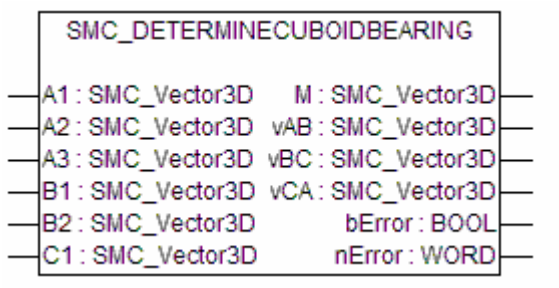
Fehlerbeschreibung:

0	kein Fehler (bError = FALSE)
1	Vektoren haben nicht die Länge 1
2	Vektoren stehen nicht senkrecht aufeinander
3	kein Rechtssystem

SMC_DetermineCuboidBearing

Baustein zur Bestimmung der Lage eines Quaders (Eckpunkt, Kantenrichtungen) im Raum durch Vorgabe von 6 (3/2/1) Punkten:





A1, A2, A3: SMC_Vector3D

Drei Punkte auf einer Seitenfläche des Quaders, die nicht auf einer Geraden liegen dürfen.

B1, B2: SMC_Vector3D

Zwei Punkte auf einer anderen Seitenfläche des Quaders, deren Projektion auf die Fläche A nicht identisch sein darf.

C1: SMC_Vector3D

Punkte auf einer weiteren Seitenfläche des Quaders.

M: SMC_Vector3D

Eckpunkt des Quaders

vAB, vBC, vCA: SMC_Vector3D

Einheitsvektoren auf den Kantenlinien des Quaders.

bError: BOOL

unmögliche Eingabe-Werte.

nError: WORD

Fehlerbeschreibung:

0	kein Fehler (bError = FALSE)
1	A1, A2, A3 liegen auf einer Geraden
2	Projektion von B1 und B2 auf Ebene A sind identisch

9 Die Bibliothek SM_Error.lib

Diese Bibliothek muss in jedem Projekt vorhanden sein, da sie alle Fehlerdefinitionen enthält. Sie enthält alle von SoftMotion-Funktionsbausteinen erzeugten Fehler und kann diese als String darstellen.

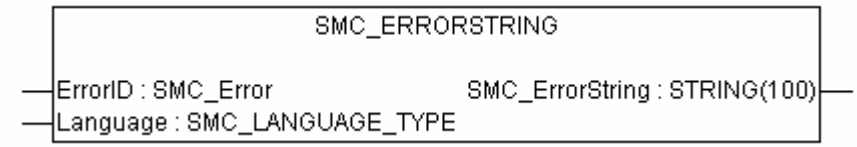
Grundsätzlich sollte der Applikateur beachten, dass zweierlei Fehlersorten in seinem Programm auftreten können. Zum einen **Antriebsfehler**, also Fehler die im Antrieb passieren (z.B. Schleppfehler, Stromversorgung fehlt etc.). Zum anderen **Bausteinfehler**, also Fehler, die von den Bausteinen über die Ausgänge Error und ErrorID zurückgemeldet werden, und häufig auf eine falsche Parametrierung zurückzuführen sind.

Antriebsfehler müssen über MC_ReadAxisError bzw. MC_ReadParameter gelesen und ggf. mit MC_Reset gelöscht werden. Antriebsfehler sind antriebsspezifisch und nicht genormt.

Bausteinfehler können mit den Funktionen der SM_Error.lib nach Bedarf in Strings umgewandelt werden. Da diese Fehler an allen SoftMotion-Bausteinen autreten können und von der Applikation gesammelt werden müssten, wurde eine zusätzliche Funktionalität in die AXIS_REF-Datenstruktur implementiert, die eine Liste der zuletzt aufgetretenen Fehler speichert. Mit dem Ausgang **FBErrorOccured** von MC_ReadStatus kann geprüft werden, ob bzw. welcher Bausteinfehler zuletzt auftrat. Der Funktionsblock **SMC_ReadFBError** gibt die Fehlernummer des ältesten aufgetretenen Fehlers zurück. Die Funktion SMC_**ClearFBError** löscht den ältesten Fehler.

9.1 Funktionsblöcke

9.1.1 SMC_ErrorString



Die Funktion SMC_ErrorString gibt in Abhängigkeit von den Eingaben ErrorID (SMC_Error) und Language (SMC_LANGUAGE_TYPE (english, deutsch)) eine String-Repräsentation des Fehlers zurück.

9.2 Die Enumeration SMC_Error

Die Enumeration SMC_Error enthält alle Fehlernummern, die von SoftMotion-FBs erzeugt werden:

Fehler nr.	Baustein	Enum-Wert	Beschreibung
0	alle	SMC_NO_ERROR	kein Fehler
1	DriveInterface	SMC_DI_GENERAL_COMMUNICATION_ERROR	Kommunikationsfehler (z.B. Sercos-Ringbruch)
10	DriveInterface	SMC_DI_SWLIMITS_EXCEEDED	Position außerhalb des erlaubten Fensters (SWLimit)
20	alle bewegungserzeugenden Bausteine	SMC_REGULATOR_OR_START_NOT_SET	Reglerfreigabe nicht erteilt oder Bremse gesetzt

Fehler nr.	Baustein	Enum-Wert	Beschreibung
30	DriveInterface	SMC_FB_WASNT_CALLED_DURING_MOTION	Bewegungserzeugender Baustein wurde vor dem Ende der Bewegung nicht mehr aufgerufen
31	alle Bausteine	SMC_AXIS_IS_NO_AXIS_REF	Eingegebene AXIS_REF-Variable ist nicht vom Typ AXIS_REF
32	Alle bewegungserzeugende Bausteine	SMC_AXIS_REF_CHANGED_DURING_OPERATION	Eingegebene AXIS_REF-Variable wurde getauscht, während der Baustein aktiv war
50	SMC_Homing	SMC_3SH_INVALID_VELACC_VALUES	ungültige Geschwindigkeits- oder Beschleunigungswerte
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLIMIT	Modus schreibt (aus Sicherheitsgründen) die Benutzung der Endschalter vor
70	SMC_SetControllerMode	SMC_SCM_NOT_SUPPORTED	Modus nicht unterstützt
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	Achse ist nicht im richtigen Regelungsmodus
80	SMC_ResetAxisGroup	SMC_RAG_ERROR_DURING_STARTUP	Fehler beim Achsgruppen-Hochlauf
90	SMC_ChangeGearingRatio	SMC_CGR_ZERO_VALUES	unzulässige Werte
91	SMC_ChangeGearingRatio	SMC_CGR_DRIVE_POWERED	Getriebeparameter dürfen nicht geändert werden solange der Antrieb in Regelung ist
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	Die Achse enthält keine Angabe über Zykluszeit (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	Achse war fehlerfrei
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER	Achse führt Fehler-Rücksetzen nicht aus.
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	Fehler ließ sich nicht zurücksetzen.
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	Kommunikation zur Achse funktionierte nicht.
130	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	Parameternummer unbekannt
131	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_REQUESTING_ERROR	Fehler beim Übertragen zu den Antrieben. Siehe Fehlernummer in Bausteininstanz ReadDriveParameter (SM_DriveBasic.lib)
140	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_PARAM_INVALID	Parameternummer unbekannt oder Schreiben verboten

Fehler nr.	Baustein	Enum-Wert	Beschreibung
141	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_SENDING_ERROR	Fehler beim Übertragen zu den Antrieben. Siehe Fehlernummer in Bausteininstanz WriteDriveParameter (SM_DriveBasic.lib)
170	MC_Home	SMC_H_AXIS_WASNT_STANDSTILL	Achse war nicht im Zustand 'standstill'
171	MC_Home	SMC_H_AXIS_DIDNT_START_HOMING	Starten der Homing-Aktion schlug fehl.
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	Kommunikationsfehler.
173	MC_Home	SMC_H_ERROR_WHEN_STOPPING	Fehler beim Stoppen nach Homing. Deceleration gesetzt?
180	MC_Stop	SMC_MS_UNKNOWN_STOPPING_ERROR	unbekannter Fehler beim Stoppen
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VALUES	unzulässige Geschwindigkeits- oder Beschleunigungswerte
182	MC_Stop	SMC_MS_DIRECTION_NOT_APPLICABLE	Direction=shortest/fastest nicht anwendbar
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	Antrieb ist im errorstop-Zustand. Stop kann nicht ausgeführt werden.
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_VALUES	unzulässige Geschwindigkeits- oder Beschleunigungswerte
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	Richtungsfehler
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VALUES	unzulässige Geschwindigkeits- oder Beschleunigungswerte
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	Richtungsfehler
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VALUES	unzulässige Geschwindigkeits- oder Beschleunigungswerte
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	Richtungsfehler
276	MC_MoveSuperImposed	SMC_MSI_INVALID_VELACC_VALUES	unzulässige Geschwindigkeits- oder Beschleunigungswerte
277	MC_MoveSuperImposed	SMC_MSI_INVALID_DIRECTION	Richtungsfehler
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VALUES	unzulässige Geschwindigkeits- oder Beschleunigungswerte
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APPLICABLE	Direction=shortest/fastest nicht anwendbar
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	fehlerhafte Arraygröße
326	MC_PositionProfile	SMC_PP_STEP0MS	Schrittzeit = t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	fehlerhafte Arraygröße
351	MC_VelocityProfile	SMC_VP_STEP0MS	Schrittzeit = t#0s

Fehler nr.	Baustein	Enum-Wert	Beschreibung
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	fehlerhafte Arraygröße
376	MC_AccelerationProfile	SMC_AP_STEP0MS	Schrittzeit = t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	Trigger bereits aktiv
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	DriveInterface unterstützt Fenster-Funktion nicht
402	MC_TouchProbe	SMC_TP_COMM_ERROR	Kommunikationsfehler
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	Trigger bereits frei
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	Kurvenscheibe enthält keine Tappets
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	Tappet-GroupID überschreitet MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	mehr als 32 Zugriffe auf eine CAM_REF
625	MC_CamIn	SMC_CI_NO_CAM_SELECTED	keine Kurvenscheibe gewählt
626	MC_CamIn	SMC_CI_MASTER_OUT_OF_SCALE	Masterachse außerhalb des zulässigen Bereichs
627	MC_CamIn	SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	für ramp_in-Funktion müssen Geschwindigkeits- und Beschleunigungswerte spezifiziert werden
628	MC_CamIn	SMC_CI_SCALING_INCORRECT	Skalierungsvariablen fEditor/TableMasterMin/Max falsch
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator = 0
676	MC_GearIn	SMC_GI_INVALID_ACC	Acceleration unzulässig
677	MC_GearIn	SMC_GI_INVALID_DEC	Deceleration unzulässig
725	MC_Phase	SMC_PH_INVALID_VELACCDEC	Geschwindigkeit, Brems- oder Beschleunigungswerte unzulässig
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD0	Rotationsachse mit fPositionPeriod = 0
750	Alle Bausteine, die MC_CAM_REF als Eingang benutzen	SMC_NO_CAM_REF_TYPE	Eingegebene Kurvenscheibe ist nicht vom Typ MC_CAM_REF
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	Bahn unfahrbar, da Soll-Geschwindigkeit = 0.
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	Allerletztes Bahnobjekt hat *Vel_End* > 0.

Fehler nr.	Baustein	Enum-Wert	Beschreibung
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	Warnung: GEOINFO-Liste in DataIn abgearbeitet, aber Ende der Liste ist nicht gesetzt. Ursache: *EndOfList* der Queue in *DataIn* vergessen zu setzen oder SMC_Interpolator schneller als bahnerzeugende Bausteine.
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_STOP	Geschwindigkeit an Stopp-Punkt > 0.
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	Zu viele SMC_Interpolator-Rekursionen. SoftMotion-Fehler.
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIES	Die Eingangs-OutQueue DataIn hatte nicht den Baustein SMC_CHeckVelocities als letztes durchlaufen
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	Intener/numerischer Fehler
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_SMALL	Datenpuffer zu klein
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	Bahn passt nicht vollständig in die Queue
1080	SMC_Interpolator	SMC_WAR_INT_OUTQUEUE_TOO_SMALL	Warnung: OutQueue *DataIn* zu klein dimensioniert. Einhalten von Stopps kann nicht garantiert werden.
1081	SMC_Interpolator	SMC_WAR_END_VELOCITIES_INCORRECT	Warnung: Endgeschwindigkeiten inkonsistent.
1100	SMC_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	Geschwindigkeit, Brems- oder Beschleunigungswerte unzulässig
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	Beschleunigungswert unzulässig
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	Beschleunigungswert unzulässig
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	Data underrun. Queue wurde leergelesen.
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	Buffer zu klein
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	Bufferelemente haben falschen Typ
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	Aktuelle Zeile des Interpolators konnte nicht gefunden warden
1500	Alle Funktionsbausteine, die SMC_CNC_REF verwenden	SMC_NO_CNC_REF_TYPE	Eingegebenes CNC-Programm ist nicht vom Typ SMC_CNC_REF

Fehler nr.	Baustein	Enum-Wert	Beschreibung
1501	Alle Funktionsbausteine, die SMC_OUTQUEUE verwenden	SMC_NO_OUTQUEUE_TYPE	Eingegebene OutQueue ist nicht vom Typ SMC_OUTQUEUE
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	Datei existiert nicht
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	Kein Buffer angelegt
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	Buffer zu klein
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	Data underrun. Buffer wurde leer gelesen.
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE_REPLACED	Platzhaltervariable konnte nicht ersetzt werden
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	Eingang pvl zeigt nicht auf SMC_VARLIST-Objekt
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	Datei konnte nicht geöffnet werden
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	kein Buffer angegeben.
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	Buffer zu klein.
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	unerwartetes Dateiende
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	Datei konnte nicht geöffnet werden.
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OVERRUN	Buffer-Überlauf; WriteToFile öfter aufrufen
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	Datei konnte nicht geöffnet werden
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	gespeicherte Kurvenscheibe zu groß
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	falscher Übersetzungsmodus
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	Datei hat falsche Version
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	unerwartetes Dateiende
3001	SMC_WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	Parameter-Auslese-Kanal belegt
3002	SMC_WriteDriveParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	Datei konnte nicht erzeugt werden
3003	SMC_WriteDriveParamsToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	Fehler beim Lesen der Parameter
3004	SMC_WriteDriveParamsToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	Timeout beim Vorbereiten der Parameterliste
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	Nenner des Umrechnungsfaktors (dwRatioTechUnitsDenom) der Encoder-Referenz 0.
5001	SMC_Encoder	SMC_ENC_AXISUSEDBYOTHERFB	Anderer Baustein versucht, Bewegung auf Encoder-Achse auszuführen.

10 Die Bibliothek SM_FileFBs.lib

10.1 Überblick

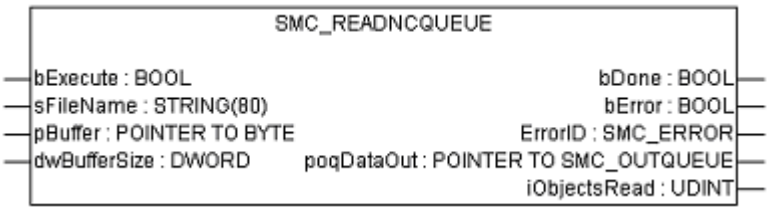
Diese **Bibliothek** stellt Bausteine zur Verfügung, die im Zusammenhang mit File-Funktionalität stehen.

Ihre Verwendung setzt die 3S-Systembibliotheken SysLibFile.lib und Standard.lib voraus.

10.2 CNC-Funktionsblöcke

SMC_ReadNCQueue

Dieser Baustein liest eine OutQueue-Datei, die vom CNC-Editor erzeugt wurde (siehe 3.3) vom Filesystem der Steuerung ein, und stellt eine OutQueue-Struktur zur Verfügung, die typischerweise vom Interpolator abgefahren wird.



Eingänge des Bausteins:

bExecute: BOOL

Der Baustein beginnt bei einer steigenden Flanke mit dem Einlesen der Queue.

sFileName: STRING(80)

Dateipfad und –name.

pBuffer: POINTER TO BYTE

Zeiger auf einen in der IEC-Applikation reservierten, genügend großen, freien Datenbereich.

dwBufferSize: DWORD

Größe des Datenbereichs in Byte.

Ausgänge des Bausteins:

bDone: BOOL

Wird gesetzt, wenn die Queue vollständig eingelesen wurde

bError: BOOL

TRUE: Fehler trat auf

ErrorID: SMC_ERROR

Fehlernummer.

poqDataOut: POINTER TO SMC_OUTQUEUE

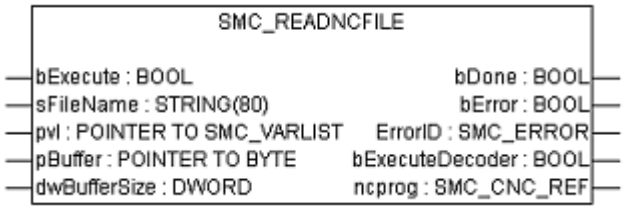
Zeiger auf die eingelesene Queue.

iObjectsRead: UDINT

Anzahl der bislang gelesenen und in die Queue geschriebenen SMC_GeoInfo-Objekte.

SMC_ReadNCFile

Dieser Baustein liest eine NC-ASCII-Datei vom Dateisystem der Steuerung ein, um sie dem Baustein SMC_NCDecoder zur Verfügung zu stellen. So kann zur Laufzeit ein NC-Programm eingelesen und umgesetzt werden.



Eingänge des Bausteins:

bExecute: BOOL

Der Baustein beginnt bei einer steigenden Flanke mit dem Einlesen des Programms.

sFileName: STRING(80)

Dateipfad und –name.

pvl: POINTER TO SMC_VARLIST

Pointer auf ein SMC_VARLIST-Objekt. (s.u.). Sollten keine Variablen im CNC-Programm verwendet werden, wird dieser Eingang nicht belegt.

pBuffer: POINTER TO BYTE

Zeiger auf einen in der IEC-Applikation reservierten, genügend großen, freien Datenbereich.

dwBufferSize: DWORD

Größe des Datenbereichs in Byte.

Ausgänge des Bausteins:

bDone: BOOL

Wird gesetzt, wenn das Programm vollständig eingelesen wurde

bError: BOOL

TRUE: Fehler trat auf

ErrorID: SMC_ERROR

Fehlernummer.

bExecuteDecoder: BOOL

Signal, mit welchem der Execute-Eingang des SMC_NCDecoder-Bausteins getriggert werden sollte.

ncprog: SMC_CNC_REF

CNC-Programm. Eingang des anschließenden SMC_NCDecoder-Bausteins.

SMC_VARLIST-Struktur

Die IEC1131-3 sieht keine Möglichkeit vor, aus dem symbolischen Variablennamen, der beispielsweise als String vorliegt, den Wert der zugehörigen Variablen zu gewinnen. Diese Möglichkeit müsste aber gegeben sein, um die Variablen-Funktionalität (siehe 3.2), die dem Anwender mit der Übersetzungsvariante ‚Programmvariable' (siehe 3.7) zur Verfügung steht, beim Einlesen des CNC-Programms über Datei ebenso nutzen zu können. Deshalb behilft man sich mit der Strukur SMC_VARLIST. Diese besteht aus der Variable *wNumberVars*, die die Anzahl aller verwendeten Variablen enthält, und einem Zeiger *psvVarList* auf das erste Element eines Arrays von ***SMC_SingleVar***, welches den Typ und einen Zeiger auf die Variable enthält. Ein *SMC_SingleVar*-Objekt enthält die Zeichenkette *strVarName*, die den Namen der Variable, wie er im NC-Programm vorkommt, in Großbuchstaben enthält. Außerdem besteht es aus einem Pointer auf die Variable und deren Typ. Dazu ein Beispiel:

Im NC-Programm, welches mit SMC_ReadNCFile aus Datei gelesen wird, kommen zwei Variablen g_rTestX (REAL) und g_byCommand (BYTE) vor:

```
N0 G$g_byCommand$ X$g_rTestX$
```

Man legt nun folgende Variablen an:

```
g_byCommand: BYTE;
g_rTest: REAL;
asv: ARRAY[0..1] OF SMC_SingleVar :=
            (strVarName:='G_BYCOMMAND', eVarType:=SMC_TYPE_BYTE),
            (strVarName:='G_rTESTX', eVarType:=SMC_TYPE_REAL);
vl: SMC_VarList:=(wNumberVars:=2);
```

Vor dem Aufruf des Bausteins SMC_ReadNCFile, dessen pvl-Eingang man dann mit ADR(vl) beschreibt, muss man den Zeiger auf die Variablen setzen:

```
asv[0].pAdr:=ADR(g_byCommand);
asv[1].pAdr:=ADR(g_rTest);
```

und muss die Verknüpfung zwischen SMC_VarList und AMC_SingleVar herstellen:

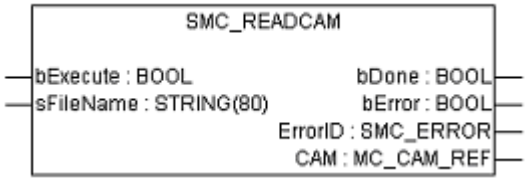
```
vl.psvVarList := ADR(asv[0]);
```

Kann eine Variable nicht ersetzt werden, wird ein Fehler ausgegeben und der Baustein bricht ab.

10.3 CAM-Funktionsblöcke

SMC_ReadCAM

Dieser Baustein wird dazu verwendet, eine Kurvenscheibe, die mit dem CAM-Editor erstellt und in eine *.CAM-Datei gespeichert wurde (siehe 4.4.3), zur Laufzeit zu Laden und den Bausteinen MC_CamTableSelect und MC-CamIn zur Verfügung zu stellen.



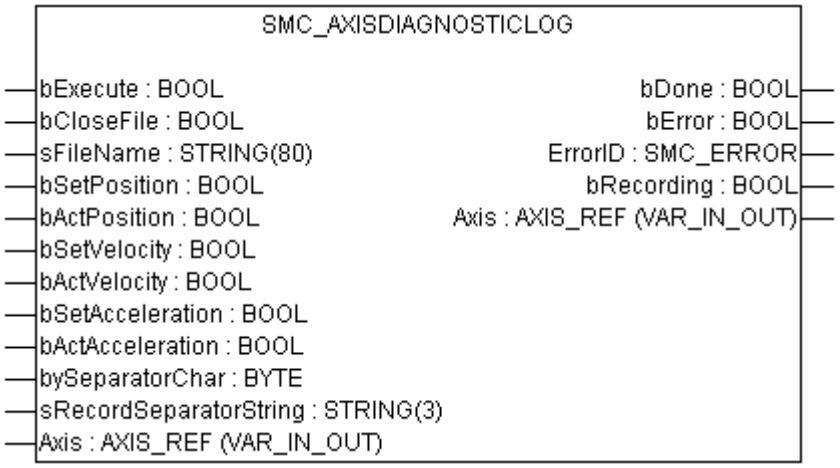
Die Größe einer ladbaren Kurvenscheibe wird begrenzt durch die globalen Konstanten *gc_SMC_FILE_MAXCAMEL* (Anzahl Elemente) und *gc_SMC_FILE_MAXCAMTAP* (Anzahl Nockenschaltaktionen).

10.4 Diagnose-Funktionsblöcke

SMC_AxisDiagnosticLog

Dieser Baustein kann dazu verwendet werden, ausgewählte Werte einer Achse zyklisch in eine Datei zu schreiben. Die so gewonnene Datei ist ideal für Diagnosezwecke verwendbar.

Da das Schreiben von Daten auf einen Datenträger in der Regel längere Zeit benötigt, speichert dieser Baustein die gesammelten Daten in einen Buffer der Größe 5kByte, und erst durch den Aufruf der Bausteinaktion **WriteToFile** werden die Daten geschrieben. Diesen Aktions-Aufruf sollte man deshalb in eine langsamere (etwa 50 ms), weniger priore Task verlagern, um die eigentliche Motion-Task nicht aufzuhalten und das Antriebsverhalten zu stören. Läuft der Buffer über, erzeugt der Baustein einen Fehler.



Eingänge des Bausteins:

bExecute: BOOL

Der Baustein beginnt bei einer steigenden Flanke. Er überschreibt eine evtl. vorhandene Datei des gewählten Namens.

bCloseFile: BOOL

Der Baustein schließt die Datei, sobald dieser Eingang TRUE wird.

sFileName: STRING(80)

Dateipfad und –name.

bSetPosition, bActPosition, bSetVelocity, bActVelocity, bSetAcceleration, bActAcceleration: BOOL

Diese Eingänge entscheiden, ob die zugehörigen werte der Achse in die Datei gespeichert werden sollen.

bySeparatorChar: BYTE (Default: TAB)

ASCII-Code des Buchstabens, der zwischen zwei Werte desselben Datums geschrieben werden soll.

sRecordSeparatorString: STRING(3) (Default: ‚RN')

String, der am Ende eines Datums geschrieben werden soll.

Axis: AXIS_REF;

Achse, die beobachtet werden soll.

Ausgänge des Bausteins:

bDone BOOL

TRUE: Logging beendet, Datei geschlossen.

bError: BOOL

TRUE: Fehler trat auf

ErrorID: SMC_ERROR

Fehlernummer.

bRecording: BOOL

Baustein zeichnet auf.

11 Programmierbeispiele

11.1 Überblick

Um mit einem CoDeSys Projekt mittels der SoftMotion Funktionalität eine Antriebs-Hardware zu steuern, sind folgende Punkte zu beachten:

- Die SoftMotion Funktion muss in den **Zielsystemeinstellungen**, Register 'Allgemein' aktiviert sein
- Die Bibliotheken **SM_DriveBasic.lib** und die herstellerspezifische **<BusInterfaceBezeichnung>Drive.lib** müssen im CoDeSys Projekt eingebunden werden, um das Drive Interface für die Kommunikation mit den Antrieben verwenden zu können.
- Im **Drive Interface** (Steuerungskonfiguration) muss die Struktur der Antriebs-Hardware abgebildet und parametriert werden; nach einem Übersetzungslauf des Projekts werden dafür automatisch globale Variablen angelegt.
- Eine **Taskkonfiguration** muss aufgesetzt werden.
- Ein **IEC Programm** mit einem CoDeSys Editor muss erstellt werden, das die gewünschten Bewegungen durch Aufrufe passender Bausteine ausführt. Damit für diese Verarbeitung die entsprechenden SoftMotion Funktionen verfügbar sind, müssen die Bibliotheken **SM_CNC.lib** bzw. **SM_PLCopen.lib** eingebunden werden. Im CNC- bzw. CAM-Editor können Mehr-Achsbewegungen bzw. Kurvenscheiben für die Steuerung der Antriebe grafisch und textuell programmiert werden; CoDeSys erzeugt daraus automatisch entsprechende globale Datenstrukturen (CNC Data, CAM Data), mit denen das IEC Programm arbeiten kann.

Sehen Sie folgende Programmierbeispiele:

- Steuerungskonfiguration für Antriebe erstellen (Kapitel 11.2)
- Bewegungssteuerung einzelner Achsen (single-axis) (Kapitel 11.3)
- Bewegungssteuerung einzelner Achsen in CFC mit Visualisierungs-Template (single-axis) (Kapitel 11.4)
- Kurvenscheiben-Antriebssteuerung mit Hilfe einer virtuellen Zeitachse (Kapitel 11.5)
- Beispiel: wechselnde Kurvenscheiben (siehe Kapitel 11.6)
- Antriebssteuerung mit Hilfe des CNC-Editors (Kapitel 11.7)
  - Direktes Erzeugen der Queue (Kapitel 11.7.1)
  - online Decodierung, Verwendung von Variablen (Kapitel 11.7.2)
  - Bahnvorverarbeitung online (Kapitel 11.7.3)
- Dynamische SoftMotion-Programmierung (Kapitel 11.8)

11.2 Beispiel: Drive Interface: Steuerungskonfiguration für Antriebe erstellen

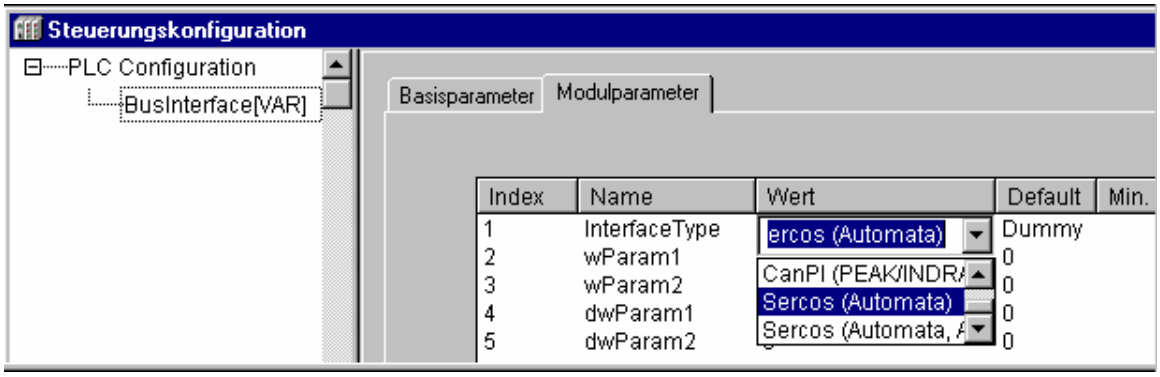
(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt DriveInterface.pro, basierend auf der Konfigurationsdatei softmotion.cfg)

Dieses Beispiel beschreibt, wie eine vorhandene physikalische Antriebsstruktur ins IEC-Programm abgebildet wird. Durch diese Konfiguration stehen dem IEC-Programm Datenstrukturen zur Verfügung, auf denen die SoftMotion-Funktionsbausteine arbeiten und dadurch Bewegungen erzeugen können.

- Zunächst öffne man die **Steuerungskonfiguration** und wähle im Menü "Extras" "Standardkonfiguration".

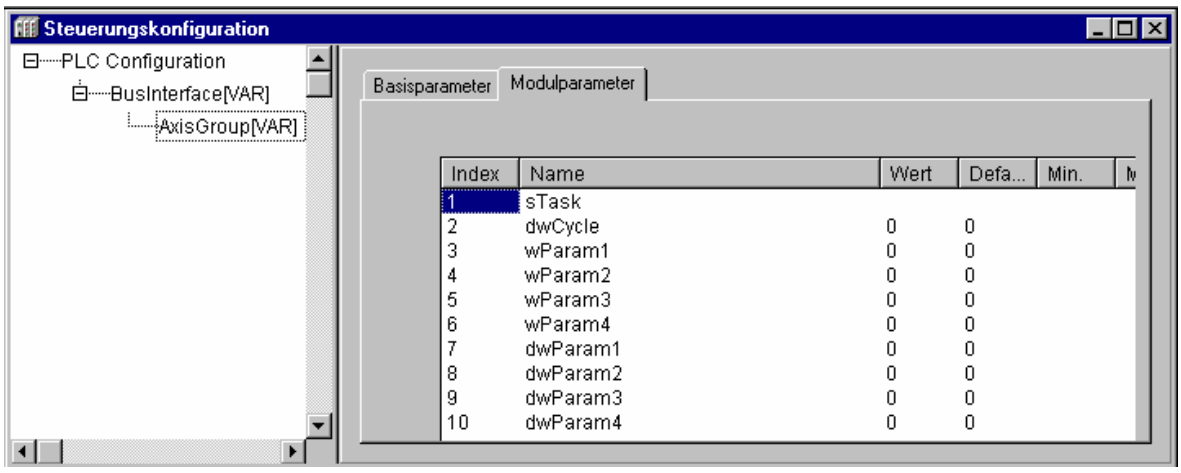
- Als Feldbus werde Sercos verwendet. Dazu wird beispielsweise eine ISA-Bus-Steckkarte von der Firma Automata verwendet. Deshalb wählt man aus dem Kontextmenü, das man durch einen Rechtsklick auf "PLC Configuration" erhält: "**BusInterface anhängen**".

- Für dieses BusInterface werden nun die entsprechenden **Modulparameter** eingestellt. Zunächst der InterfaceType. In unserem Beispiel steht keine Hardware-Anbindung zur Verfügung und wir können mit der Auswahl "Dummy" eine Art Simulationsmodus wählen.



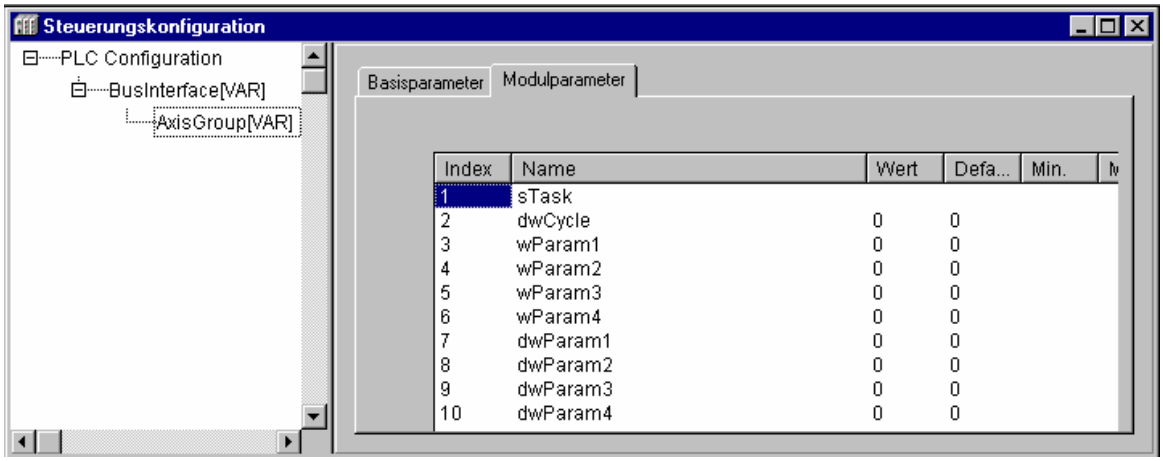
- Von dieser Auswahl hängen die weiteren Parameter ab, die einzustellen sind. Bei "Sercos (Automata)" sind dies beispielsweise für wParam1 die Interrupt-Nummer, für wParam2 der HW-Typ und für dwParam1 die Basisadresse der Karte.

  Diese Karte habe einen Ring, an dem vier Antriebe hängen, zu bedienen. Deshalb fügt man durch Rechtsklick auf das BusInterface eine "**AxisGroup**" ein:

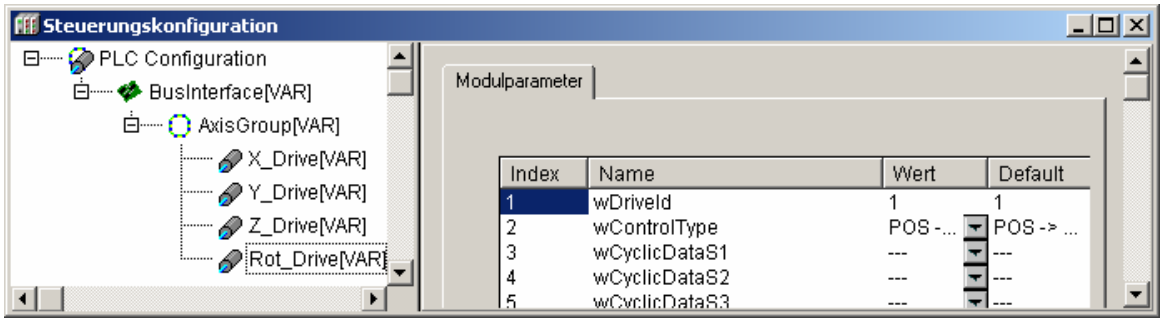


- Auch hier müssen die **Modulparameter** eingegeben werden. Zunächst wird der Name der **Task** in "sTask" eingegeben, von der aus die Antriebe bedient werden sollen (z.B. BusTask), in die Zeile darunter die Zykluszeit der Task in µsec (z.B. 3000).
  (Das Einrichten der Task wird im nächsten Punkt beschrieben.)
  Die weiteren Parameter der AxisGroup werden in Abhängigkeit von der Wahl des BusInterfaces gemacht. In unserem Beispiel wären dies für wParam1 die Baudrate in MBit (z.B. 2) und in wParam2 die Helligkeit der LED.

- Wir öffnen die **Taskkonfiguration** und erstellen eine Task mit diesen Eigenschaften:



- Nun geht es daran, die **Antriebe einzuhängen**. Es seien vier Antriebe vorhanden, drei Linearantriebe, die ein X-, Y-, Z-Portalsystem bedienen, sowie ein Antrieb, der das Werkzeug in der Z-Achse dreht.
  Wenn der Eintrag "AxisGroup" im Konfigurationsbaum selektiert ist (gepunkteter Rahmen), können jeweils durch Klick mit der rechten Maustaste und Anwählen von "Drive anhängen" nacheinander vier Antriebe eingehängt werden. Deren Namen können dann mit beliebigen IEC-Bezeichnern belegt werden, wenn durch Klicken auf den Eintrag ein Editierfeld um den Namen geöffnet wird. Für unser Beispiel geben Sie folgende Bezeichnungen ein:



- Die Antriebe müssen nun nach und nach parametriert werden:

- Zunächst sollten für alle Antriebe gemäß deren Konfiguration die ID "wDriveId" eingestellt werden. Im folgenden wird beschrieben, wie die Einstellungen beim Reiter „Modulparameter“ gemacht werden müssen; viel einfacher und selbst erklärend geht dies auch über die Dialoge.
  Die **Portalantriebe** können jeweils zwischen -50cm und +50cm bewegt werden. Ihre Wichtung sei translatorisch konfiguriert. Ein Inkrement für alle Positions- und Geschwindigkeitsdaten betrage 10-7m bzw. 10-7m/s. Wollen wir stattdessen, dass alle IEC-Positions- und Geschwindigkeitsdaten sich auf mm bzw. mm/sec beziehen, tragen wir in *dwRatioTechUnitsDenom* "10000", in *iRatioTechUnitsNum* "1" ein. Da es sich um einen Linearantrieb handelt, hat *fPositionPeriod* keine Bedeutung. Es sollten jetzt lediglich noch die Daten spezifiziert werden, die zyklisch gesendet und empfangen werden, bspw. wenn wir uns auf ein Vorzugstelegramm beziehen wollen, genügt es, aus "*wControlType*" "POS, VEL -> POS, VEL" auszusuchen. Damit werden zyklisch die Positions- und Geschwindigkeits-Sollwerte geschickt und die aktuellen Positions- und Geschwindigkeits-Istwerte erhalten. Um das Verlassen des zulässigen Bereichs (-50cm = -5000mm, 50 cm = 5000mm) zusätzlich zu überwachen (die Applikation sollte so programmiert sein, dass ein Verlassen dieses Bereiches nicht möglich ist), wird mit dem Setzen von SWLimitEnable = TRUE, SWLimitNegative = -5000 und SWLimitPositive = 5000 eine Überwachungsfunktion aktiviert.

- Bei dem verwendeten **rotatorische Antrieb** bestehe eine Umdrehung aus 65536 Inkrementen. Wir tragen deshalb, um eine interne Einheit von Winkelgrad zu erhalten in *dwRatioTechUnitsDenom* "65536", in *iRatioTechUnitsNum* "360" ein. Dieser Antrieb sei beispielsweise dafür gemacht, um auf eine Flasche einen Schraubverschluss drehen zu können.

Deshalb wollen wir zyklisch Position und Drehmoment schicken, um später durch Änderung der Betriebsart vom Positionier- auf ein Drehmoment-geregeltes Verhalten umschalten zu können. Dazu schalte man *wControlType* auf "CONFIGURABLE" und trage unter *wCyclicDataS1/2* "fSetPosition" und "fSetTorque" ein. Um den aktuellen Positionswert zurückzuerhalten trage man unter *wCyclicDataR1* "fActPosition" ein.

- Damit sich das Programm fehlerfrei übersetzen lässt, muss noch ein **Programmaufruf an die Task** "BusTask" angefügt werden. Beispielsweise **erstellt man ein Programm** "Ipo", in welchem später die Bewegungssteuerung stattfinden soll, und ruft es von "BusTask" aus auf.

Jetzt sollte ein fehlerfreier **Übersetzungslauf** möglich sein und das Programm kann **auf die Steuerung geladen und gestartet** werden. Automatisch erzeugt werden die folgenden Variablen und Strukturen:

→ Im **Globale-Variablen**-Ordner "**Drive Configuration Data**" befinden sich nun drei Instanzen der Bausteine "SercosDriveExecute_Start", "SercosDriveExecute_End" und "SercosDriveInit" aus der Sercos-Bibliothek mit den Namen "AxisGroupStartCycle", "AxisGroupEndCycle" und "AxisGroupInit", welche für die Kommunikation mit den Antrieben verantwortlich sind.

→ Im Globale-Variablen-Ordner "Drive_Globale_Variablen" der Bibliothek "**SM_DriveBasic.lib**" existiert eine Strukturvariable **g_DRIVESTRUCT**, welche alle Einträge der Steuerungskonfiguration, d.h. alle BusInterfaces, AxisGroups und Drives enthält.

→ Ausserdem wurden **für jeden Antrieb global eine Strukturvariable** erstellt, z.B. "X_Drive", die man sich z.B. mit dem Watch- und Rezepturverwalter ansehen kann. Auf dieser Strukturvariablen arbeiten die SoftMotion-Bausteine und das DriveInterface sorgt für die Aktualität der Struktur.

Auf diesen Antriebsstrukturen arbeiten die Motion-Bausteine des IEC-Programms, welches wir jetzt in Ipo erstellen könnten.

11.3 Beispiel: Bewegungssteuerung einzelner Achsen in ST (single-axis)

(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt PLCopenSingle.pro, basierend auf der Konfigurationsdatei softmotion.cfg)

Dieses Beispiel demonstriert, wie ein Antrieb mittels der nach PLCopen standardisierten Funktionsblöcke gesteuert werden kann.

Abgesehen von den **Bibliotheken** des Drive Interfaces muss die Bibliothek SM_PLCopen.lib im Projekt eingebunden sein.

In der **Steuerungskonfiguration** sei ein linearer Antrieb mit dem Namen "Drive" definiert:

In der **Taskkonfiguration** rufe man das Programm "Ipo" auf, das eine Bewegung auf dieser Achse erzeugen wird.

Dieses **Programm** soll im folgenden erstellt werden, dazu wird im Object Organizer ein Programm in der Sprache ST angelegt und gefüllt:

Bevor wir beginnen können, den Antrieb zu bewegen, muss gesichert sein, dass der Treiber den Antrieb gefunden und initialisiert hat. Wenn dies passiert ist, dann sollten wir dem Antrieb die Reglerfreigabe erteilen und ggfs. die Bremse lösen. Diese Funktion übernimmt der Baustein MC_Power:

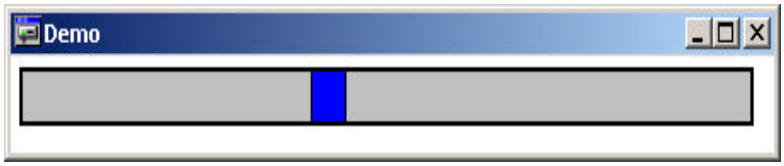
```
PROGRAM Ipo
VAR
       Init: BOOL := FALSE;
       Power: MC_Power;
END_VAR
IF NOT Init THEN
       Power(Enable:=TRUE, bRegulatorOn:=TRUE, bDriveStart:=TRUE,Axis:=Drive);
       Init:= Power.Status;
ELSE
END_IF
```

Im ELSE-Zweig der ersten IF-Bedingung können wir jetzt also den Antrieb steuern. In diesem Beispiel wollen wir das mit dem **Positionier-Baustein** MC_MoveAbsolute machen. Dazu deklarieren wir eine Instanz dieses Bausteins und eine Zielposition p, die wir mit 100 initialisieren wollen. Jetzt rufen wir jeden Zyklus diese Bausteininstanz mit den benötigten Werten auf. Wenn die programmierte Position erreicht wurde, ist der Done-Ausgang des Bausteins gesetzt und wir müssen den Execute-Eingang auf FALSE setzen, wenn wir eine neue Bewegung starten wollen, da der Baustein nur auf eine steigende Flanke eine neue Bewegung beginnt:

```
...    (* Fortsetzung des oben gezeigten Programms *)
ELSE
       MoveAbsolute(Execute:=TRUE, Position:=p, Velocity:=100,
Acceleration:=100, Deceleration:=100, Axis:=Drive);
       IF MoveAbsolute.Done THEN
              MoveAbsolute(Execute:=FALSE, Axis:=Drive);
       END_IF
END_IF
```

Jetzt können wir das Programm fehlerfrei übersetzen, online gehen und starten. Wenn wir mit einer Watchliste oder der Traceaufzeichnung die Istposition Drive.fActPosition beobachten, sehen wir, wie der Antrieb auf diese zusteuert. Forcen wir den Wert von p, so bewegt sich die Achse nach Erreichen des vorigen Zieles auf das neue Ziel zu.

Um die Bewegung grafisch betrachten zu können, stehen in der Bibliothek SM_DriveBasic.lib Visualisierungs-Templates für Antriebe (siehe Kapitel 2.2.10) zur Verfügung. Um diese zu verwenden, geht man zunächst wieder offline, erstellt eine neue Visualisierung, fügt darin ein Visualisierungs-Objekt ein und wählt aus der erscheinenden Liste „LinDrive“. Dann führt man einen Doppelklick auf das neu erstellte Objekt aus und trägt unter „Visualisierung“, „Platzhalter“, „AXISREF“ als Ersetzung den Namen der Antriebsstruktur (hier: Drive) ein. Die so gewonnene Visualisierung stellt die Position des Antriebs dar:



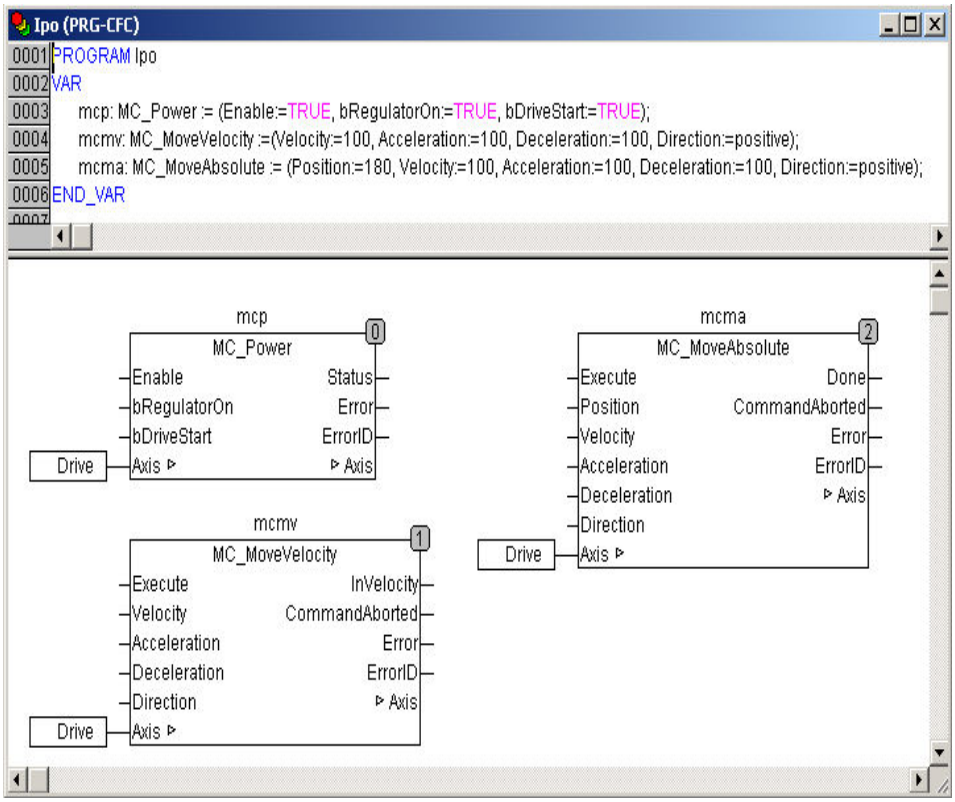
11.4 Beispiel: Bewegungssteuerung einzelner Achsen in CFC mit Visualisierungs-Template (single-axis)

(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt PLCopenSingle2.pro, basierend auf der Konfigurationsdatei softmotion.cfg)

Anstelle von ST kann auch jede andere IEC-Sprache als Programmiersprache dienen, wie folgendes Beispiel zeigt.

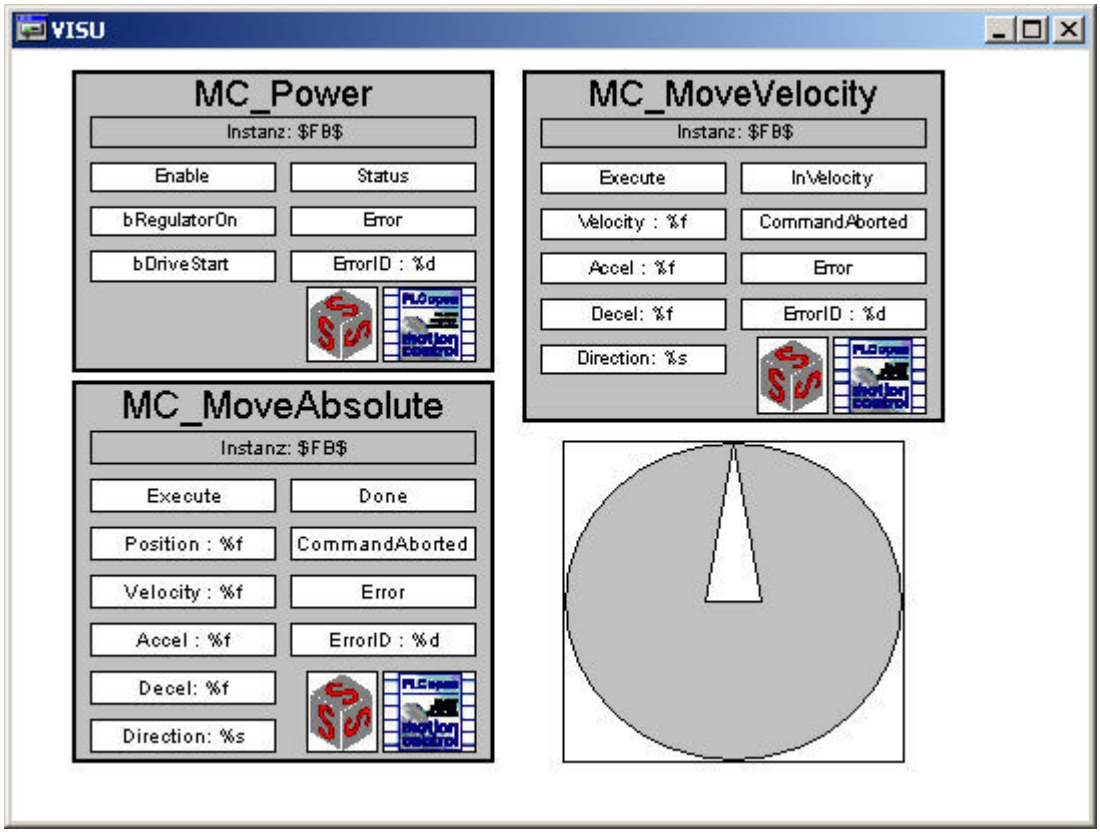
Dieses Beispiel dient dazu, den Start- und Unterbrechungsmechanismus der Funktionsbausteine am Beispiel nachzuvollziehen. Außerdem können die verschiedenen Startmodi für den Baustein MC_MoveAbsolute für rotatorische Achsen ausprobiert werden.

Man erstelle eine **Steuerungskonfiguration** und **Taskkonfiguration** wie im letzten Beispiel, nur dieses Mal für einen rotatorischen Antrieb mit der Peiode 360. Das Programm "Ipo" wird in CFC geschrieben und beinhaltet lediglich drei Aufrufe von Instanzen der Bausteine MC_Power (notwendig, um die Achse zu aktivieren), MC_MoveAbsolute und MC_MoveVelocity:



Die Initialisierung der Bausteineingänge empfiehlt sich, damit wir später beim Starten dieser Testapplikation die Werte nicht stets von Neuem eingeben müssen.

Außerdem erstellen wir eine Bedien-Visualisierung, wozu wir die in den Bibliotheken vorhandenen Baustein-Templates verwenden und über das Platzhalter-Konzept mit den Bausteininstanzen verknüpfen:



Nun können wir das Projekt fehlerfrei übersetzen, uns auf die Steuerung einloggen und starten. Durch Drücken des Execute-Eingangs von MoveVelocity beginnt sich der Antrieb zu drehen. Drücken vom Execute des MoveAbsolute positioniert den Antrieb auf die eingestellte Position, wobei entsprechend der Angabe Direction: positive nur in positive Richtung gedreht wird. Der Baustein MoveVelocity wird dadurch unterbrochen. es empfiehlt sich, mit diesen Bausteinen zu spielen, verschiedene Geschwindigkeiten und Beschleunigungen auszuprobieren und die Richtungsmodi (positive/negative/current/shortest/fastest) von MoveAbsolute zu testen.

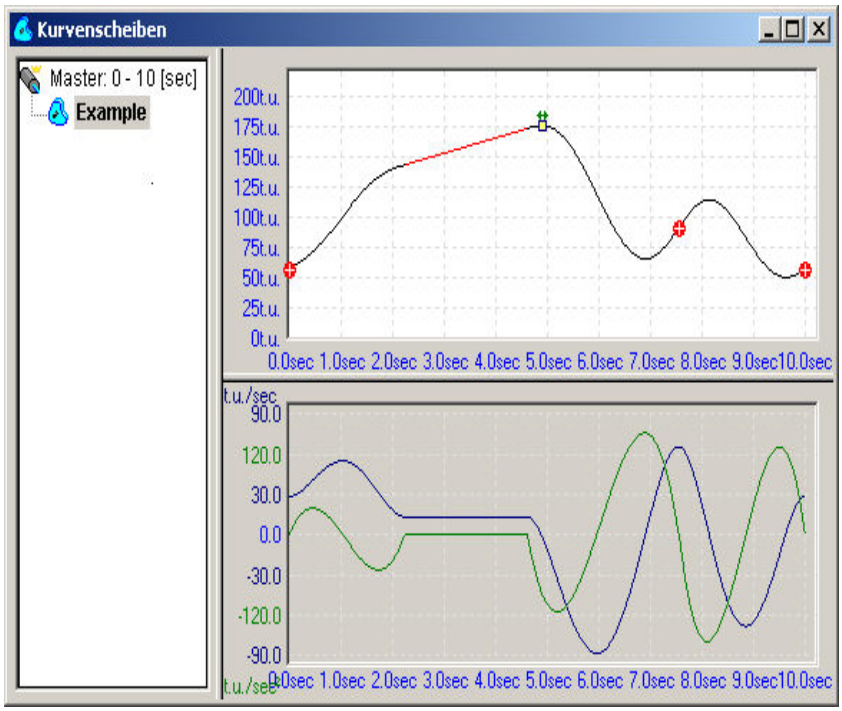
11.5 Beispiel: Kurvenscheiben-Antriebssteuerung mit Hilfe einer virtuellen Zeitachse

(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt PLCopenMulti.pro, basierend auf der Konfigurationsdatei softmotion.cfg)

(Voraussetzung: Die Bibliotheken DriveBasic.lib und SM_PLCopen.lib sind eingebunden.)

Das folgende Beispiel demonstriert, wie eine periodische Kurvenscheibe auf einem linearen Antrieb umgesetzt werden kann. Zusätzlich wird die Nockenfunktion demonstriert.

1. Dazu erstellt man im CoDeSys Kurvenscheiben-Editor zunächst eine beliebige periodische Kurvenscheibe, die sich auf eine Masterachse zwischen 0 und 10 sec bezieht, und mindestens eine invertierende Nocke mit ID1 enthält. Z.B.:



2. Im Drive Interface (**Steuerungskonfiguration)** sei ein linearer Antrieb mit dem Namen "Drive" definiert:

3. Das Programm Ipo sei ein FUP erstellt und beeinhaltet die Aufrufe dieser Bausteine:

```
PROGRAM Ipo
VAR
      Power: MC_Power;
      TimeAxis : SMC_TimeAxisFB;
      TableSelect: MC_CamTableSelect := (SlaveAbsolute:=TRUE);
      CamIn: MC_CamIn:=(StartMode:=ramp_in, VelocityDiff:=100,
Acceleration:=100, Deceleration:=100);
      Tappet: SMC_GetTappetValue;
END_VAR
```

Nach dem Power-Baustein (**Power**) für die Slave-Achse, wird zunächst der Zeitachsen-Baustein (**TimeAxis**) aufgerufen. Als Periode erhält diese 10 Sekunden, da die Kurvenscheibe auf diese Zeit ausgelegt ist. Die Task-Zykluszeit muss von Hand eingegeben werden. **TableSelect** wählt die gewünschte Kurvenscheibe aus, und **CamIn** setzt diese um. Der Baustein **Tappet** überprüft die Stellung des Nockenschalters. Da dieser invertiert programmiert wurde, wird er alle 10 Sekunden umschalten.

Nun kann das Programm **übersetzt** und auf der Steuerung **gestartet** werden.

Um die Soll- bzw. Istposition zu kontrollieren, erstelle man wieder eine Visualisierung, mit deren Hilfe man die einzelnen Bausteine und die Position der Achsen überprüfen kann.

Man beachte, dass der Master der Kurvenscheibe nicht nur eine virtuelle Zeitachse, sondern natürlich jede beliebige AXIS_REF-Datenstruktur sein kann. Dabei werden bei Antrieben die in Regelung sind, die Sollwerte herangezogen, bei freien Antriebe die Istwerte.

11.6 Beispiel: wechselnde Kurvenscheiben

(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt PLCopenMultiCAM.pro, basierend auf der Konfigurationsdatei softmotion.cfg)

(Voraussetzung: Die Bibliotheken DriveBasic.lib und SM_PLCopen.lib sind eingebunden.)

Dieses Beispiel zeigt, wie eine Kurvenscheiben-Bewegung mit zwei sich abwechselnden Kurvenscheiben realisiert werden kann. Es wurde in ST programmiert und führt die selben Aktionen wie das vorige Beispiel aus. Am Ende der ersten Kurvenscheibe setz der MC_CamIn-Baustein den Ausgang bEndOfProfile, aufgrund dessen der jeweils andere MC_CamTableSelect zur Anwendung kommt und MC_CamIn neu gestartet wird.

11.7 Beispiel: Antriebssteuerung mit Hilfe des CNC-Editors

Im folgenden wird die Grundstruktur eines möglichen IEC-Programms unter CoDeSys beschrieben, welches im CNC-Editor geplante Bahnen umzusetzen vermag.

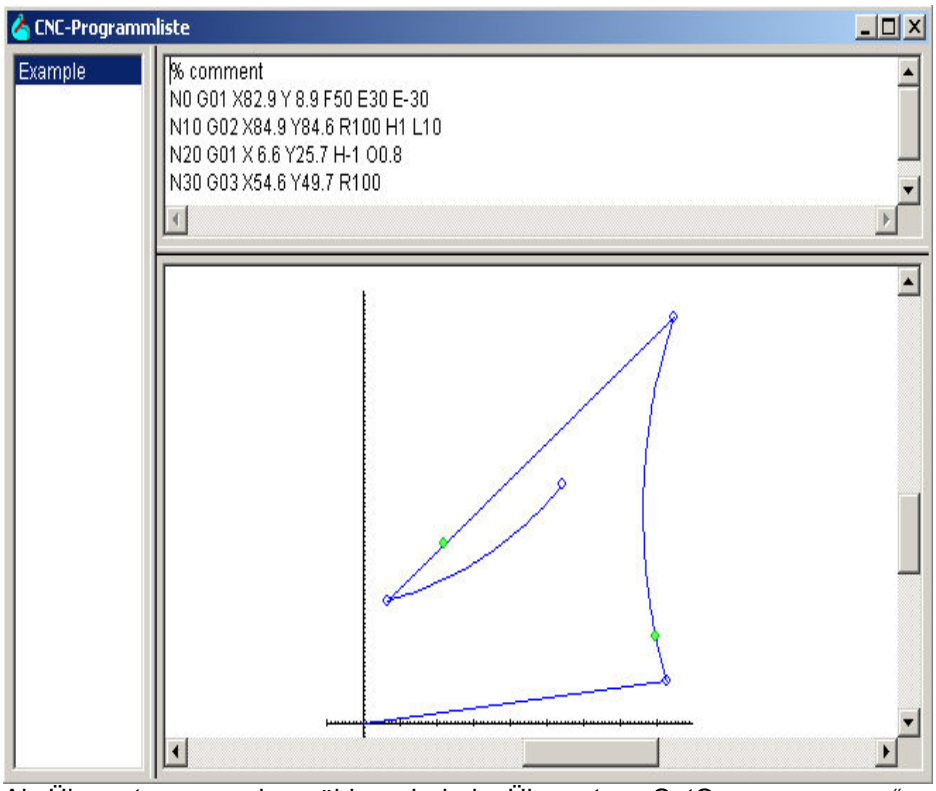
Wie unter 3.1 beschrieben, bestehen zwei Möglichkeiten ein CNC-Programm zu übersetzen und zu verwenden. Das erste Beispiel illustriert die direkte Erzeugung einer OutQueue, das zweite decodiert das Programm online unter Verwendung von Variablen. Das dritte Beispiel wendet online zusätzlich einen Bahnvorverarbeitungsbaustein an.

11.7.1 Direktes Erzeugen der OutQueue

(Sehen Sie das mit SoftMotion mitgelieferte Beispielprojekt CNCdirect.pro)

1. Erstellen des NC-Programms im CNC-Editor:

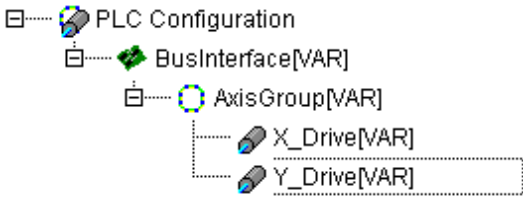
Wir erstellen wie in 0 beschrieben ein Beispielprogramm, welches sich zwischen x aus [0,100] und y aus [0,100] abspielt. Außerdem definieren wir Geschwindigkeiten und Beschleunigungen für die Bahn und setzen zwei Wegschaltpunkte auf der Bahn. Z.B.:



Als Übersetzungsmodus wählen wir „beim Übersetzen OutQueue erzeugen“.

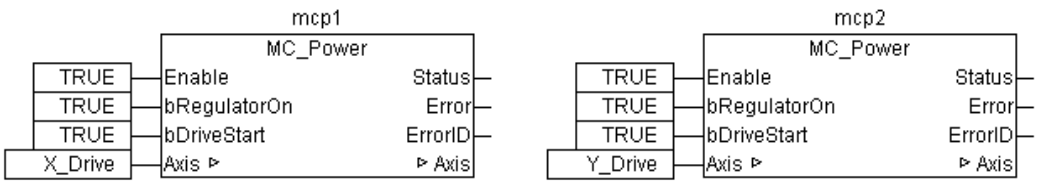
2. Drive Interface, Steuerungskonfiguration:

Eine Antriebsstruktur mit 2 linearen Antrieben wird definiert; dabei muss auch die maximale Geschwindigkeit etc. gesetzt werden.



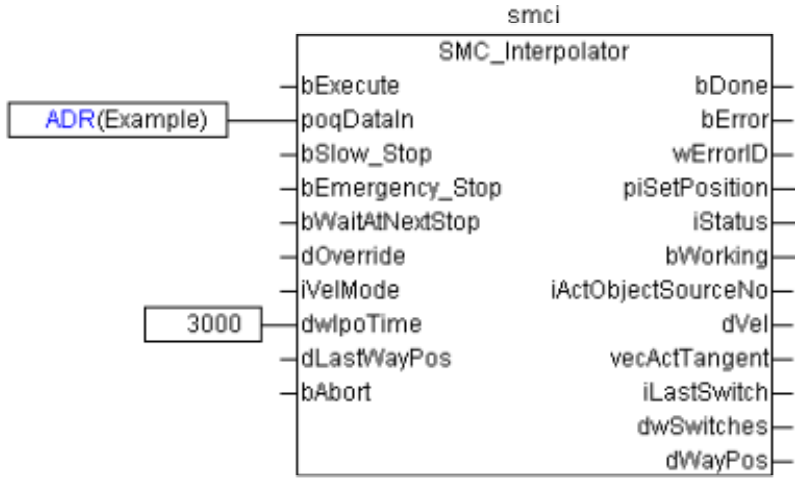
3. Erstellen des IEC-Programms:

Wir müssen zunächst die Antriebe mit dem MC_Power-Baustein aktivieren:

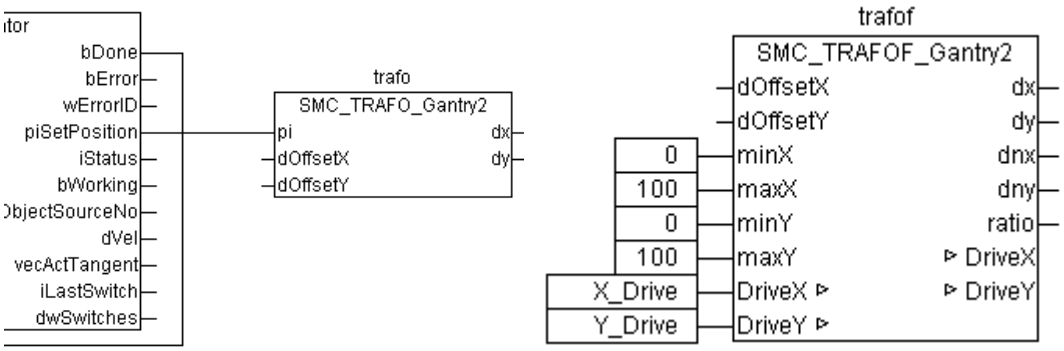


Als weiteres wichtiges Element benötigen wir einen SMC_Interpolator-Baustein. Dieser erhält als poqDataIn-Eingang die Adresse des erstellten CNC-Programms. Zudem muss die IEC-Task-

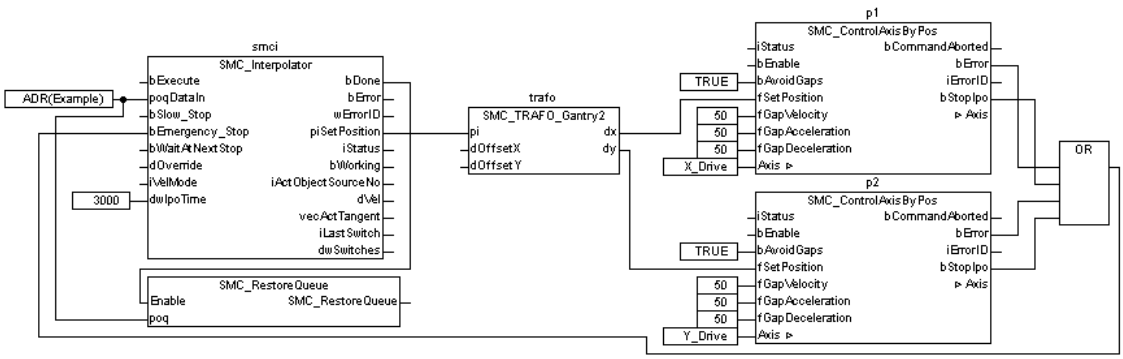
Zykluszeit in dwIpoTime geschrieben werden. Diese kann man entweder als konstanten Wert dem Eingang dwIpoTime eingeben, oder man benützt die Variable dwCycle der Achsgruppen-Struktur aus der Steuerungskonfiguration, was den Vorteil hat, dass, wenn man die Zykluszeit der Task ändert, automatisch die richtige Zeit als Interpolator-Eingang verwendet wird.



In unserem Beispiel wollen wir ein Portal-System steuern. Deshalb fügen wir eine Instanz der Rückwärts- und Vorwärtstransformations-Bausteine aus der Bibliothek SM_Trafo.lib ein. Der Vorwärtstransformationsbaustein erhält als Eingänge die Antriebe (der Z-Antrieb wird mit einer sonst nicht verwendeten Variable dummy vom Typ AXIS_REF belegt); der Rückwärtstrafo-Baustein muss die Soll-Position des Interpolators erhalten:



Die Ausgänge des Bausteins, also die Achs-Koordinaten, müssen nun auf die Antriebe geschrieben werden. Dafür existieren die Funktionsblöcke SMC_ControlAxisByPos. Da unsere Applikation nicht gewährleistet, dass die Ausgaben des Interpolators stetig sind, (zum Beispiel endet die Bahn an einem anderen Punkt als sie anfängt), sollten wir die Sprungvermeidung aktivieren (bAvoidGaps, fGapVelocity, fGapAcceleration, fGapDeceleration), den StopIpo-Ausgang mit dem bEmergency_Stop des Interpolators verbinden und den Interpolator-Ausgang iStatus mit den entsprechenden Eingängen der Achskontroll-Bausteine verbinden.



Man beachte bei der Programmierung mit CFC vor allem auch auf die richtige Reihenfolge der Bausteine!

4. Erstellen der Bedien- und Testoberfläche:

Wir binden in eine neue Visualisierung zwei Visualisierungsobjekte ein. Zum einen das Template des Interpolators, zum anderen das Template der Transformation. Diese werden über das Platzhalterkonzept mit den entsprechenden Baustein-Instanzen (hier: Ipo.smci bzw. Ipo.trafof) verknüpft.

5. Inbetriebnahme:

Das so erstellte Programm lässt sich fehlerfrei übersetzen und starten und führt das CNC-Programm aus, sobald der Execute-Eingang des Interpolators gesetzt wurde. Wurde es vollständig abgefahren, kann durch eine neue steigende Flanke ein erneutes Abfahren erreicht werden.

Man beachte auch die Funktion der Wegschalter, die in der Visualisierung des Interpolations-Bausteins ebenfalls dargestellt sind.

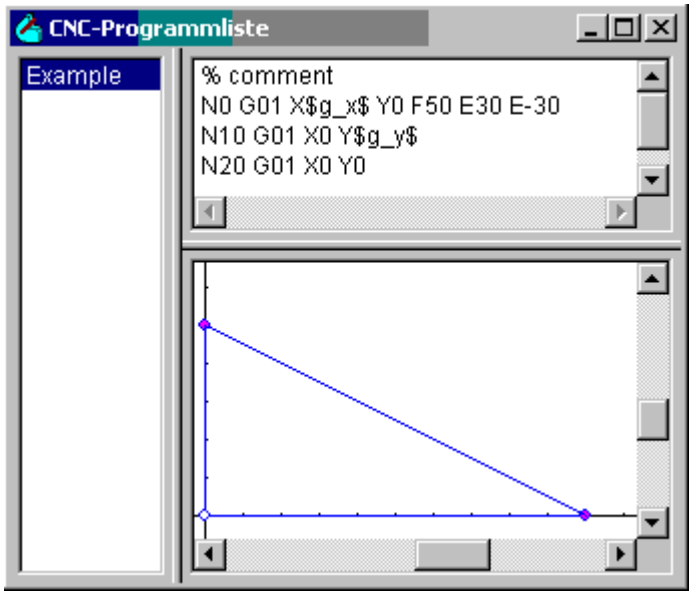
11.7.2 Decodierung online mit Verwendung von Variablen

(Sehen Sie das mit SoftMotion mitgelieferte Beispielprojekt CNConline.pro)

1. Erstellen des NC-Programms im CNC-Editor:

Wir erstellen wie im vorigen Beispiel ein CNC-Programm und verwenden darin zwei globale Variablen g_x und g_y. Z.B.:

```
VAR_GLOBAL
    g_x: REAL:=100;
    g_y:REAL:=50;
END_VAR
```



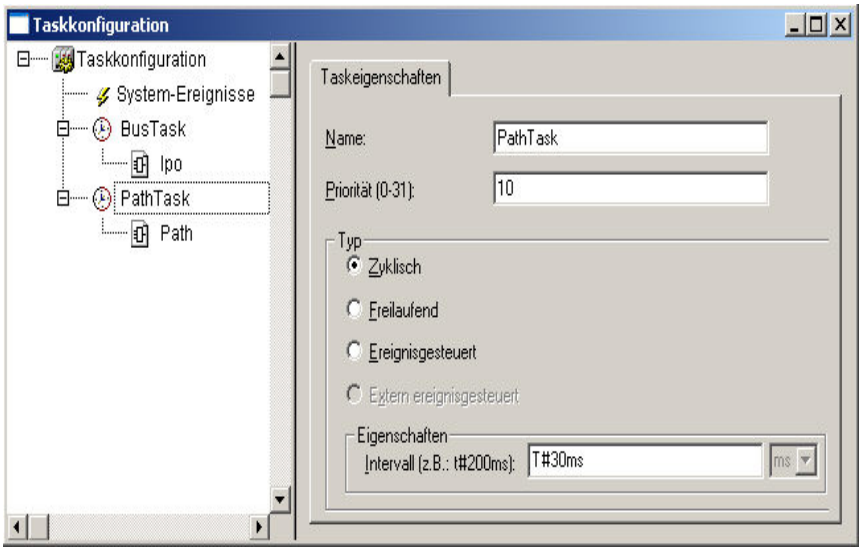
Als Übersetzungsmodus wählen wir diesmal aber „beim Übersetzen Programm-Variable erzeugen“, da wir Variablen in unserem Programm verwenden.

2. Drive Interface, Steuerungskonfiguration:

Die Antriebsstruktur entspricht der des vorhergehenden Beispiels.

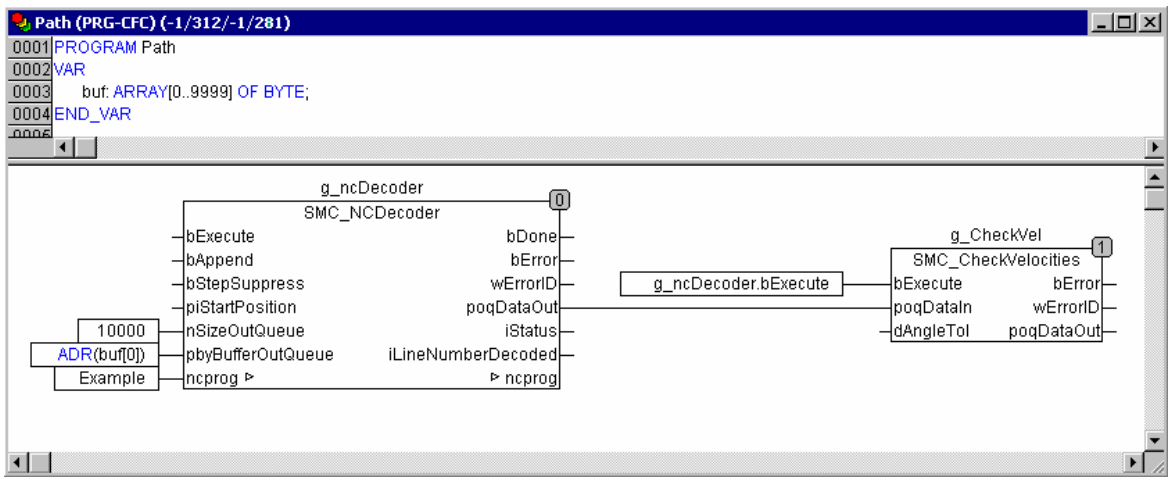
3. Erstellen des IEC-Programms:

Da wir diesmal den anderen Übersetzungsmodus gewählt haben, müssen wir das Decodieren und Bahnvorverarbeiten im IEC-Programm durchführen. Da dieser eher laufzeitintensive Vorgang nicht im Interpolatortakt durchgeführt werden muss (es wird ja pro Decoder-Aufruf ein Wegobjekt erzeugt, welches typischerweise für viele Interpolator-Aufrufe reicht), wird er häufig in eine weniger priore und seltener aufgerufene Task ausgelagert:



Der dahinter stehende Mechanismus ist folgender: In der langsamen Task wird anfänglich pro Zyklus etwa ein GEOINFO-Objekt erzeugt, welches in der OUTQUEUE-Struktur des Decoder-Bausteins gespeichert ist. Ist diese OUTQUEUE voll, pausieren die Bausteine der langsamen Task solange, bis die OUTQUEUE nicht mehr voll ist, sprich bis die schnelle Task das erste GEOINFO-Objekt abgearbeitet hat und dieses aus der OUTQUEUE löscht. Dann werden die Bausteine der langsamen Task wieder aktiv und befüllen die OUTQUEUE-Struktur. In der schnellen Task wird pro Zyklus ein Bahnpunkt aus der OUTQUEUE-Struktur, auf die der Eingang DataIn des Interpolators zeigt, berechnet und verarbeitet. Da ein GEOINFO-Objekt ja i.d.R. aus mehreren Bahnpunkten besteht, dauert es einige Zyklen bis das erste GEOINFO-Objekt abgearbeitet ist und vom Interpolator automatisch gelöscht wird. Da das Abarbeiten eines GEOINFO-Objekts offenbar mehrerer Zyklen bedarf als das Erstellen, kann die langsame Task tatsächlich seltener aufgerufen werden als die schnelle. Die Task-Zeiten müssen jedoch so gewählt werden, dass in der letzten OUTQUEUE der langsamen Task immer genügend GEOINFO-Objekte gelagert sind, sodass kein Data-Underrun auftritt. Dies tritt auf, wenn dem Interpolator aus DataIn keine GEOINFO-Objekte mehr zur Verfügung stehen, und das Bahnende noch nicht erreicht ist. In diesem Fall bremst der Interpolator und bleibt so lange stehen, bis wieder neue Datenelemente verfügbar sind.

Im Programm Path findet die Decodierung des NC-Programms zur OutQueue und die Geschwindigkeitsüberprüfung statt:



Der interpolierende Teil des IEC-Programm bleibt fast gleich, nur dass der Daten-Eingang des Interpolators nicht wie zuvor dem CNC-Programmnamen (ADR(Example)) entspricht, sondern aus dem OutQueue-Ausgang der bahnvorverarbeitenden Bausteine (g_CheckVel.poqDataOut) besteht.

4. Erstellen der Bedien- und Testoberfläche:

Zu der Visualisierung des vorigen Beispiels ist es sinnvoll, Templates der neuen Bausteine (SMC_NCDecoder und SMC_CheckVelocities) hinzuzufügen. Außerdem sollte man die globalen Variablen g_x und g_y ändern können, um später bei der Inbetriebnahme deren Funktion überprüfen zu können.

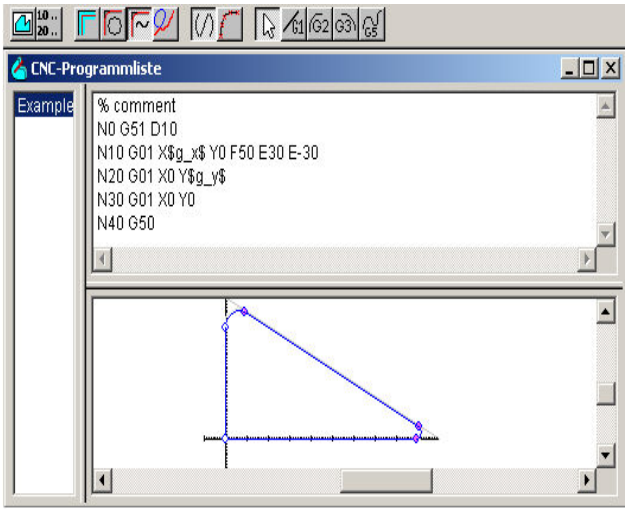
5. Inbetriebnahme:

Das so erstellte Programm lässt sich fehlerfrei übersetzen und starten und führt das CNC-Programm aus, nachdem die Execute-Eingänge des Decoders und Interpolators gesetzt wurden. Ändert man die Werte der globalen Variablen, werden diese bei einem Neustart des Decoders übernommen und die Bahn passt sich entsprechend an. Man beachte auch die Funktion des Append-Eingangs des Decoders.

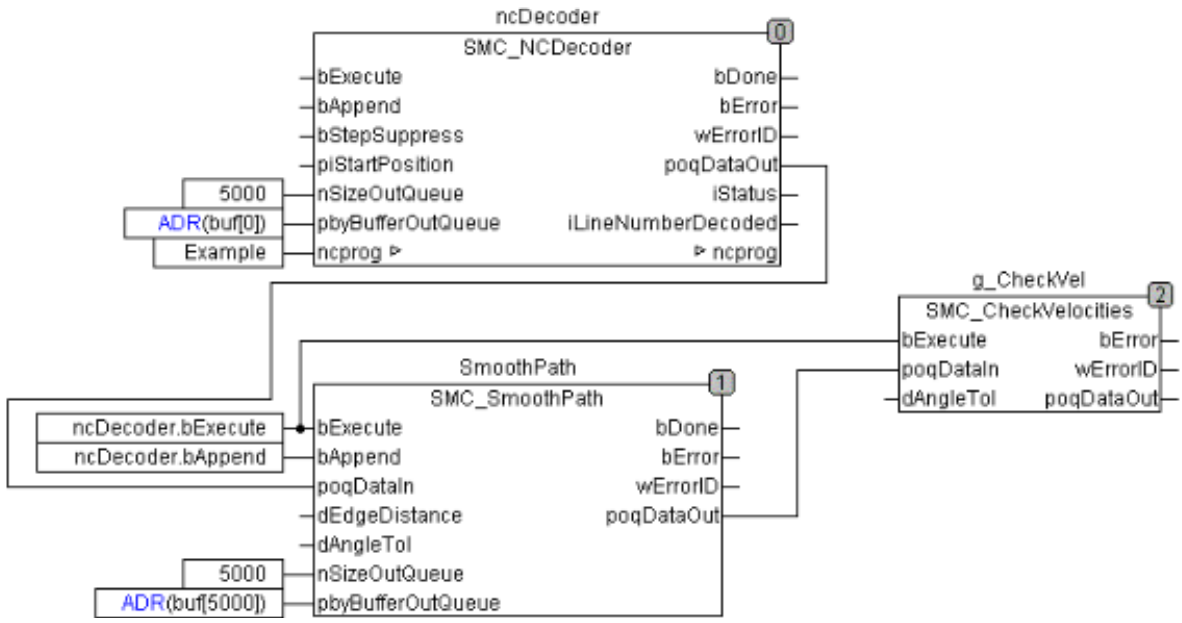
11.7.3 Bahnvorverarbeitung online

(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt CNCprepro.pro)

Wir wollen das vorige Beispiel um eine Bahnvorverarbeitung erweitern: Die Ecken des letzten Programms sollen durch Splines verrundet werden. Dies führt der Baustein SMC_SmoothPath durch. Das CNC-Programm muss von den Worten G51/G50 umklammert werden, dann ergibt sich:



Würden wir keine Variablen verwenden, könnten wir das Programm in dieser Form als Queue übersetzen und direkt in den Interpolator eingeben; da wir aber Variablen verwenden, müssen wir das Dekodieren und die Eckverschleifung selbst durchführen. Deshalb deklarieren wir einen neuen Baustein vom Typ SMC_SmoothPath und rufen ihn nach dem Decoder auf:



Der Daten-Eingang des Interpolator-Bausteins muss wie immer auf den Ausgang poqDataOut des CheckVelocities-Bausteins gesetzt werden.

Dieses Programm kann fehlerfrei übersetzt werden und stoppt – im Unterschied zum vorigen – nicht mehr in den Ecken des NC-Programms, da die Ecken der Bahn durch die Bahnvorverarbeitung knickfrei gemacht wurde.

11.8 Beispiel: Dynamische SoftMotion-Programmierung

(Sehen Sie hierzu das mit SoftMotion mitgelieferte Beispielprojekt CNCDynamicPath.pro, basierend auf der Konfigurationsdatei softmotion.cfg)

Einer der Vorteile von SoftMotion ist es, dem Programmierer und auch dem Benutzer nicht nur Einfluss auf die Abarbeitung einer Bahn zu gewähren, sondern diese auch bei laufendem Programm generieren oder ändern zu können. Um dies zu erreichen, muss der Programmierer nur den **Decoder-Baustein durch einen eigenen Bahngenerator ersetzen** und kann sich trotzdem der Bahnvorverarbeitung und vor allem des Interpolators wie gewohnt bedienen.

Um den Decoder-Baustein zu ersetzen, muss also auf anderem Weg **ein OUTQUEUE-Strukturobjekt erstellt** werden, das mit GEOINFO-Objekten, die die gewünschte Bahn verkörpern, befüllt und an den entsprechenden Folge-Baustein (z.B. Interpolator) weitergegeben wird.

Vorbereitende Schritte:

- Im Deklarationsteil muss ein OUTQUEUE-, ein GEOINFO-Strukturobjekt und ein Buffer gewünschter Größe deklariert sein:

  ```
  QUEUE: SMC_OUTQUEUE;
  BUF: ARRAY[0..49] OF SMC_GEOINFO;
  GEO: SMC_GEOINFO:=(dT1:=0, dT2:=1, dToolRadius:=0, dVel:=100, dVel_End:=100,
  dAccel:=200, dDecel:=500, iObj_Nr:=0);
  ```

- In einem Init-Schritt im Programmrumpf muss die OUTQUEUE-Struktur initialisiert werden:

  ```
  QUEUE.nSize := SIZEOF(BUF);
  QUEUE.pbyBuffer := ADR(BUF);
  ```

Dynamische Bahn-Programmierung

Im Programmrumpf müssen an der Stelle, wo die Bahn erstellt werden soll, für jedes GEOINFO-Objekt der Bahn folgende Schritte durchgeführt werden:

- Start-Position setzen (erstes Objekt)
  ```
  GEO.piStartPos.dX := 0;
  ...
  ```
  ...bzw. vom vorhergehenden Objekt kopieren.
  ```
  GEO.piStartPos := GEO.piDestPos;
  ```

- Bewegungstyp festlegen. Z.B.:
  ```
  GEO.iMoveType := CCWL;
  ```
  oder
  ```
  GEO.iMoveType := LIN;
  ```

- dem Bewegungstyp entsprechende Parameter setzen. Wenn Sie einen Kreisbogen (z.B. CCWL) verwenden, beachten Sie, dass die folgenden Positionen gesetzt werden müssen (siehe Struktur SMC_GEOINFO, Kapitel 6.5):

- GEO.dP1 := 200;
  GEO.dP2 := 100;
  GEO.dP3 := 50;
  GEO.dT1 := 0;
  GEO.dT2 := 90;

- ggf. Anfangs- oder Endbit in InternMark für die Bahnvorverarbeitung setzen Gerade (siehe Struktur SMC_GEOINFO, Kapitel 6.5).

- End-Position berechnen:
  SMC_CalcEndPnt(ADR(GEO));

- Länge des Objekts berechnen (lassen):
  SMC_CalcLengthGeo(ADR(GEO));

- Objekt in OUTQUEUE speichern:
  SMC_AppendObj(POQ:=ADR(QUEUE), PGI:=ADR(GEO));

- Ist die Bahn vollständig erstellt, muss die OUTQUEUE-Liste geschlossen werden: QUEUE.bEndOfList := TRUE;

Es gilt dabei zu beachten, dass, wenn die OUTQUEUE voll ist, d.h. wenn QUEUE.bFULL = TRUE, nicht versucht wird, weitere Objekte anzuhängen. Die Erstellung der Bahn muss dann solange unterbrochen werden, bis das erste Objekt der OUTQUEUE verarbeitet worden ist. Dann kann ein weiteres Objekt angefügt werden. Will man dies vermeiden, muss die OUTQUEUE so groß gewählt werden, dass alle GEOINFO-Objekte der gewünschten Bahn darin Platz finden.

Die Objektliste Queue wird erst dem CheckVelocities-Baustein und schließlich dem Interpolator übergeben, der diese weiter verarbeitet.

In diesem Beispiel wird außerdem gezeigt, wie eine kinematische Transformation, die in der von 3S zur Verfügung gestellten Bibliothek SM_Trafo.lib nicht enthalten ist, selbst programmiert werden kann. Die in dieses Projekt eingehängten Bausteine SMC_TRAFO und SMC_TRAFOF zeigen dies am Beispiel für ein kartesisches X/Y-System..

12 Index

A

Achse 5-2
Achsgruppe 2-2
Achsgruppenkonfiguration 2-2
Achsstatus 5-2
Antrieb 2-2
Antriebsfehler 9-1
Antriebskonfiguration 2-4
Antriebsparametrierung 2-27
Antriebstreiber 2-21
äquidistant 4-7
Ausgabegrösse anpassen 3-12
Automatische Codegenerierung 2-8
AXIS_REF-Struktur 2-23
Axisgroup 2-2
AxisGroup Konfiguration 2-2

B

Bahnabschnitt 6-41
Bahnecken verrunden 6-29
Bahnrotation 6-40
Bahntranslation 6-40
Bahnvorverarbeitung 6-29
Bausteinfehler 9-1
Bedingte Sprünge 3-6
Beschleunigungs-Sollwerte 5-19
Bibliothek
  Herstellerspezifische Drive-Lib 2-8
  SM_CNCDiagnostic.lib 7-1
  SM_DriveBasic.lib 4-12
  SM_Error.lib 9-1
  SM_FileFBs.lib 10-1
  SM_PLCopen.lib 5-1
BusInterface Konfiguration 2-2
BusInterface..Drive.lib 2-21

C

CAM Kurvenscheibe aus ASCII-Tabelle importieren 4-8
CAM-Editor 1-2, 4-1
  Start 4-1
CAM-Funktionsblöcke 10-3
CAMTableSelect 4-9
CAN-Achsgruppe 2-3
CAN-Antriebskonfiguration 2-7
ClearFBError 9-1
CNC - Programmbeispiel 11-8
CNC Data 3-1
CNC-Bibliothek 1-2
CNC-Code in Visualisierung 6-25
CNC-Editor 1-2
  Bearbeitungsmodus 3-11
  Graphischer Editor 3-11
  Texteditor 3-10
CNC-Programm
  Fahrbefehl 3-2, 3-3
  Satz 3-2
  Umwandeln in SMC_GEOINFO Strukturobjekte 6-23
  Wortkennung 3-2
CNC-Programm anlegen 3-7
CNC-Programm aus Datei lesen 3-10
CNC-Programm in Datei schreiben 3-10
CNC-Programm umbenennen 3-8
CNC-Programmname 3-7

D

Datei für CNC-Programm 3-1
Datenstruktur für CNC-Programm 3-1
Datenstrukturen für Kurvenscheibe 4-12
Dekodierung 6-23
Diagnose 2-19
Diagnose 10-3
DIN66025 3-2
Drive 2-2
Drive ID 2-4
Drive Interface 1-1
Drive Interface Beispielkonfiguration 11-1
Drive Konfiguration 2-4
DummyDrive.lib 2-21
DXF-Datei importieren 3-10
Dynamische Programmierung 11-14

E

Eckverrundung 3-13, 6-31
Eckverschleifung 3-13, 6-29
Einstellungen im CAM-Editor 4-6
Einstellungen Kurvenscheibe 4-2
Element selektieren im CAM-Editor 2-9, 4-8
elementoptimiert 4-7
Ellipsen 3-13
Ellipseninterpolation 3-6
Encoder 2-20
Encoder Konfiguration 2-7
Epsilonwerte für Null ändern 3-14
Extrema anzeigen 4-6

F

Fahrbefehl 3-2
FBErrorOcurred 9-1
Fehlerbehandlung 9-1
Fehlernummern 9-1

G

G20 3-6
G36 3-6
G37 3-6
GantryCutter 8-3
G-Code 6-25
Geber 2-2
Gerade einfügen 4-8
Geschwindigkeits-Rampentyp 2-5
Geschwindigkeits-Sollwerte 2-13, 2-15, 2-16
Geschwindigkeits-Sollwerte 5-19
Globale Variablen in SM_CNC
  lib 6-40

H

H-Funktion 3-4

I

Info 3-8
Initialisierungsdaten für Antrieb 2-7
Interpolation zeigen 3-14
IPO_UNKNOWN 6-37, 6-48

J

Jerk 2-5

K

Kreisinterpolation 3-5
Kreis-Links-Einfügemodus 3-12
Kreis-Rechts-Einfügemodus 3-12
Kurve komplettieren 4-6
Kurvenscheibe 4-1
  editieren 4-3
  Master-Achse 4-1
  Periode 4-9
  periodisch 4-2
  Periodisch 4-9
  Slave-Achse 4-1
  Umschalten 4-11
Kurvenscheibe - Programmbeispiel 11-7
Kurvenscheibe als ASCII-Tabelle exportieren 4-8
Kurvenscheibe aus Datei lesen 4-8
Kurvenscheibe in Datei schreiben 4-7
Kurvenscheiben-Baum 4-3

L

LIN 6-43
LinDrive 2-20
Linien-Einfügemodus 3-12
LINPOS 6-43
Löschen 3-8

M

Master-Achse 4-1
Mathematische Hilfsbausteine 2-8
MC_AbortTrigger 5-14
MC_AccelerationProfile 5-13
MC_CAM_REF 4-13
MC_CamIn 4-10, 5-16
MC_CamOut 5-17
MC_CamTableSelect 5-14
MC_CAMXYVA 4-14
MC_GearIn 5-17
MC_GearOut 5-18
MC_Home 5-5
MC_MoveAbsolute 5-6
MC_MoveAdditive 5-7
MC_MoveRelative 5-8
MC_MoveSuperImposed 5-9
MC_MoveVelocity 5-10
MC_Phasing 5-18
MC_PositionProfile 5-11
MC_Power 5-4
MC_ReadActualPosition 5-4
MC_ReadActualTorque 5-4
MC_ReadActualVelocity 5-4
MC_ReadAxisError 5-2
MC_ReadBoolParameter 5-3
MC_ReadParameter 5-3
MC_ReadStatus 5-2
MC_Reset 5-2
MC_Stop 5-5
MC_TouchProbe 5-13
MC_VelocityProfile 5-12
MC_WriteBoolParameter 5-3
MC_WriteParameter 5-3
Menü CNC-Programm
  CNC-Programm aus Datei lesen 3-10
  CNC-Programm in Datei schreiben 3-10
  CNC-Programm umbenennen 3-8
  DXF-Datei importieren 3-10
  Info 3-8
  Löschen 3-8
  Neues CNC-Programm 3-7
  Objekt teilen 3-9
  OutQueue in Datei schreiben 3-10
  Programm drehen 3-9
  Programm strecken 3-9
  Programm verschieben 3-9
  Queue-Grösse festlegen 3-8
  Richtung umdrehen 3-9
  Startposition festlegen 3-8
  Winkeltoleranz für Stop festlegen 3-9
Menü Einfügen
  Element selektieren im CAM-Editor 2-9, 4-8
  Gerade einfügen im CAM-Editor 4-8
  Nocke einfügen im CAM-Editor 4-9
  Punkt einfügen im CAM-Editor 4-8
Menü Extras
  Ausgabegrösse anpassen 3-12
  CAM Kurvenscheibe aus ASCII-Tabelle importieren 4-8
  Eckverrundung 3-13
  Eckverschleifung 3-13
  Eigenschaften 4-4
  Einstellungen 4-2, 4-6
  Epsilonwerte für Null ändern 3-14
  Extrema anzeigen 4-6
  Interpolation zeigen 3-14
  Kreis-Links-Einfügemodus 3-12
  Kreis-Rechts-Einfügemodus 3-12
  Kurve komplettieren 4-6
  Kurvenscheibe als ASCII-Tabelle exportieren 4-8
  Kurvenscheibe aus Datei lesen 4-8
  Kurvenscheibe in Datei schreiben 4-7
  Linien-Einfügemodus 3-12
  Programm neu nummerieren 3-10
  Satzunterdrückung 3-14
  Schleifenvermeidung 3-13
  Selektiermodus 3-12
  Spline-Einfügemodus 3-12
  Splines/Ellipsen durch Geraden ersetzen 3-13
  Übersetzungsoptionen 4-6
  Werkzeugradiuskorrektur 3-13
M-Funktion 3-5, 6-38
Modulo-Wert 2-8
Modulparameter 2-2
multi-axis Bewegungssteuerung 11-8

N

NC-Programm 3-7
Neue Kurvenscheibe erstellen 4-2
Neues CNC-Programm 3-7
nicht übersetzen 4-7

Nocke 4-5
Nocke einfügen 4-9
Normsprache 3-2

O

Objekt teilen 3-9
OUTQUEUE 6-41
OutQueue in Datei schreiben 3-10

P

PackProfile 2-7
Parameter der M-Funktion 6-38
Periode 4-9
Periodisch 4-9
periodische Kurvenscheibe 4-2
PLCopen-Bibliothek 1-2
polynomial 4-6
Portal-Systeme 8-1, 8-8
Portal-Systeme\mit Werkzeugversatz 8-4
Positionsdaten 6-41
Positions-Sollwerte 2-12, 2-13, 2-15
Positions-Sollwerte 5-19
Positionsspeicherung 6-41
Programm drehen 3-9
Programm neu nummerieren 3-10
Programm strecken 3-9
Programm verschieben 3-9
Punkt einfügen 4-8

Q

Queue-Grösse festlegen 3-8

R

Referenzfahrt 2-17, 5-5
Richtung umdrehen 3-9
RotDrive 2-20

S

Satz 3-3
Satzunterdrückung 3-3, 3-14
Scara-Systeme 8-9, 8-11
Schaltpunkt 3-4
Schleifenfreie Bahn erzeugen 6-28
Schleifenvermeidung 3-13
Selektiermodus 3-12
Sercos-Antriebskonfiguration 2-6
SercosDrive.lib 2-21
Sercos-Inteface 2-2
single-axis Bewegungssteuerung 11-4, 11-5
Slave-Achse 4-1
SM_CAN.lib 2-21
SM_CNC.lib 1-2, 6-23
SM_CNCDiagnostic.lib 7-1
SM_DriveBasic.lib 2-8, 4-12
SM_Error.lib 9-1
SM_Error.lib 1-2
SM_FileFBs.lib 1-2
SM_FileFBs.lib 10-1
SM_PLCopen.lib 5-1
SM_Trafo.lib 8-1
SM_Trafo.lib 1-2
SMC_ CalcDirectionFromVector 8-5
SMC_ CheckVelocities 6-32
SMC_AvoidLoop 6-28
SMC_AxisDiagnosticLog 10-3
SMC_CAMEditor 5-20
SMC_CAMRegister 5-21
SMC_CAMTable_<Variablen-Typ>_<Anzahl Elemente>_1 4-14
SMC_CAMTable_<Variablen-Typ>_<Anzahl Elemente>_2 4-14
SMC_CAMVisu 5-20
SMC_ChangeGearingRatio 2-10
SMC_CheckLimits 2-14
SMC_CNC_REF 6-46
SMC_CNC_REF-Daten 7-1
SMC_ControlAxisByPos 2-15
SMC_ControlAxisByPosVel 2-15
SMC_ControlAxisByVel 2-16
SMC_DetermineCuboidBearing 8-16
SMC_Error 9-1
SMC_ErrorString 9-1
SMC_FollowPosition 2-12
SMC_FollowPositionVelocity 2-13
SMC_FollowVelocity 2-13
SMC_GCode_Word 6-46
SMC_GCodeViewer 6-25
SMC_GEOINFO 6-41
SMC_GEOINFO Objekte verwalten 6-44
SMC_GetAxisGroupState 2-9
SMC_GetCamSlavePosition 5-18, 5-19
SMC_GetMaxSetAccDec 2-19
SMC_GetMaxSetVelocity 2-19
SMC_GetMParameters 3-5, 6-38
SMC_GetTappetValue 5-22
SMC_GetTrackingError 2-19
SMC_Homing 2-17
SMC_Interpolator 6-34, 6-38
SMC_Interpolator2Dir 6-39
SMC_Interpolator2Dir_SlowTask 6-39
SMC_IsAxisGroupReady 2-9
SMC_NCDECODER 6-23, 6-24
SMC_OutQueue-Daten 7-1
SMC_POSINFO 6-41
SMC_ReadCAM 10-3
SMC_ReadCANParameter 2-21
SMC_ReadFBError 9-1
SMC_ReadNCFile 10-2
SMC_ReadNCQueue 10-1
SMC_ReadSetPosition 5-22
SMC_ResetAxisGroup 2-9, 2-10
SMC_ROTATEQUEUE2D 6-40
SMC_RoundPath 6-31
SMC_SCALEQUEUE3D 6-40
SMC_SetControllerMode 2-11
SMC_SetPosition 5-13
SMC_SetTorque 5-22
SMC_ShowCNCREF 7-1
SMC_ShowQueue 7-1
SMC_SmoothPath 6-29
SMC_ToolCorr 6-26
SMC_TRAFO_Gantry2 8-1
SMC_TRAFO_Gantry2Tool1 8-5
SMC_TRAFO_Gantry2Tool2 8-6
SMC_TRAFO_Gantry3 8-2
SMC_TRAFO_GantryH2 8-8
SMC_TRAFO_Scara2 8-10
SMC_TRAFO_Scara3 8-11

SMC_TRAFO_Tripod 8-13
SMC_TRAFOF_Gantry2 8-2
SMC_TRAFOF_Gantry2Tool1 8-5
SMC_TRAFOF_Gantry2Tool2 8-7
SMC_TRAFOF_Gantry3 8-3
SMC_TRAFOF_GantryH2 8-9
SMC_TRAFOF_Scara2 8-10
SMC_TRAFOF_Scara3 8-12
SMC_TRAFOF_Tripod 8-14, 8-16
SMC_TRAFOV_Gantry 8-3
SMC_TRANSLATEQUEUE3D 6-40
SMC_VARLIST 10-2
SMC_VECTOR3D 6-41, 6-44
SMC_VECTOR6D 6-44
SMC_WriteCANParameter 2-21
SMC_WriteDriveParamsToFile 2-28
SoftMotion Drive Interface 2-1
SoftMotion Komponenten 1-1
SoftMotion_CNC_Globals 6-40
Sollwerte 5-19
Sollwert-Überschreitung 2-14
Sollwertvorgabe 2-12
Spline einfügen 3-12
Spline-Einfügemodus 3-12
Splineinterpolation 3-6
Splines 3-13
Sprung 3-6
StartMode 4-10
Startposition festlegen 3-8
Steigung 4-6
Steuerungskonfiguration für Antriebe - Beispiel 11-1
Steuerungskonfiguration für SoftMotion 2-2
Strukturbefüllung 4-12

T

Texteditor 3-10
Transformation 8-1
Transformations-Funktionsblöcke 8-1

U

Übersetzungsoptionen 4-6
Umrechnungsfaktor für Antriebe 2-4
Umschalten zwischen Kurvenscheiben 4-11

V

Variabelnwerte über CNC-Programm verändern 3-6
Variablen in CNC-Programm 3-5
Vektorspeicherung 6-44
Versetzte Bahn 6-26, 6-27
Virtuelle Zeitachse 11-7
Virtuelle Zeitachse 2-16
Visualisierungs-Template 11-5
Visualisierungs-Templates 2-20
Vorzeichenwert 2-8

W

Wechselnde Kurvenscheiben - Programmbeispiel 11-8
Wegobjekt 6-41
Werkzeugradiuskorrektur 3-13
Winkeltoleranz für Stop festlegen 3-9
Winkelwert 2-8
Wort
  Satznummer 3-2
Wortkennung 3-2
WriteToFile 10-3

Z

Zusatz-Funktion 3-5
zyklische Kommunikations-Daten 2-4
Zykluszeitmessung 2-8