BECKHOFF New Automation Technology

Handbuch | DE

TwinCAT 2

NC Camming

23.11.2022 | Version: 1.1

Inhaltsverzeichnis

1 Vorwort 5
1.1 Hinweise zur Dokumentation 5
1.2 Sicherheitshinweise 6
1.3 Hinweise zur Informationssicherheit 7
2 Übersicht 8
3 Kurvenscheiben-Achse - Motion Functions 9
3.1 Definition eines Punktes 9
3.2 Funktions Typen 11
3.3 Programmierung durch die SPS 12
3.4 Charakteristische Werte 14
3.5 Online Modifikation 16
4 Kurvenscheiben-Achse - Positionstabellen 18
4.1 Interfaces 18
4.2 Tabellentypen 19
4.3 Abarbeitungsmodi 21
4.4 Ablauf 24
4.5 Online Modifikation der Skalierung oder des Offsets 26
4.6 Multitabellen-Achse (Sonderform) 27
4.6.1 Interfaces und Konfiguration 28
4.6.2 Tabellentypen 28
4.6.3 Abarbeitungsmodi 30
4.6.4 Ablauf 31
5 TcNcCamming 36
5.1 PLCopen Bausteine und Motion Functions 36
5.1.1 MC_CamTableSelect 36
5.1.2 MC_CamIn 38
5.1.3 MC_CamInExt 39
5.1.4 Achskopplung mit Kurvenscheiben 41
5.1.5 MC_CamOut 44
5.1.6 MC_CamScaling 45
5.1.7 MC_ReadCamTableSlaveDynamics 46
5.1.8 MC_ReadMotionFunction 47
5.1.9 MC_WriteMotionFunction 48
5.1.10 MC_ReadMotionFunctionPoint 49
5.1.11 MC_WriteMotionFunctionPoint 50
5.1.12 MC_SetCamOnlineChangeMode 51
5.1.13 MC_ReadCamTableCharacteristics 52
5.1.14 MC_ReadMotionFunctionValues 52
5.1.15 MC_CamInfo 54
5.1.16 Datentypen 55
5.2 Bausteine zur Kompatibilität mit bestehenden Programmen 68
5.2.1 Tabellenbausteine 68
5.2.2 Bausteine für Kurvenscheiben-Achsen 74

5.2.3 Bausteine für Multitabellen-Achsen 78
5.3 Get_TcNcCamming_Version 81

1 Vorwort

1.1 Hinweise zur Dokumentation

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

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

Disclaimer

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

Marken

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

Patente

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

EtherCAT®

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

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Erklärung der Symbole

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

⚠ GEFAHR

Akute Verletzungsgefahr!

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

⚠ WARNUNG

Verletzungsgefahr!

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

⚠ VORSICHT

Schädigung von Personen!

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

HINWEIS

Schädigung von Umwelt oder Geräten

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

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

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

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

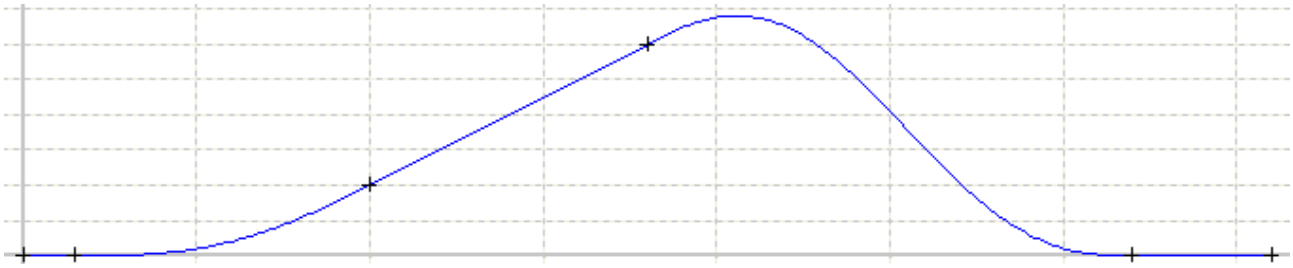
Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte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

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

2 Übersicht

Die nichtlineare Kopplung von elektrischen **Master**- und **Slave-Achsen** wird traditionell Kurvenscheibe genannt. TwinCAT bietet verschiedene Möglichkeiten für diese Art der Achskopplung. Zum einen gibt es die klassischen Positionstabellen, bei denen eine größere Anzahl von Punkten, d. h. eine Master-Position mit korrespondierender Slave-Position eine Kurve beschreibt. Zwischen diesen Punkten wird interpoliert. Diese Art der Kurvenscheibe hat sich in der Vergangenheit bewährt, hat aber den Nachteil, dass sie zur Laufzeit nur schwer zu verändern ist.

Als nächsten Schritt in diese Richtung bietet TwinCAT die so genannten **Motion Functions (MF)** an, die eine Kurvenscheibe auf andere Art beschreiben. Eine Motion Function hat eine üblicherweise geringe Anzahl von Stützstellen, zwischen denen eine mathematische Funktion, beispielsweise ein Polynom, den Verlauf der Kurve beschreibt. Die Slave-Positionen werden zur Laufzeit aus der mathematischen Beschreibung der Kurvenscheibe berechnet. Mit Motion Functions ist es sehr viel einfacher, den Funktionsverlauf auch zur Laufzeit zu ändern, da bereits die Manipulation einer Stützstelle den Verlauf verändert.



TwinCAT bietet einen **Kurvenscheibeneditor**, mit dem sich beide Arten von Kurvenscheiben, also Positionstabellen und Motion Functions erstellen lassen. Weiterhin gibt es eine **SPS-Bibliothek**, durch die Kurvenscheiben einfach programmierbar werden.

Kurvenscheiben mit Motion Functions [▶ 9]

Kurvenscheiben mit Positionstabellen [▶ 18]

SPS-Bibliothek Kurvenscheiben [▶ 36]

Kurvenscheibeneditor

3 Kurvenscheiben-Achse - Motion Functions

Eine **Motion Function (MF)** ist eine Beschreibung einer Kurvenscheibe mittels mathematischer Funktionen. Sie unterteilt den gesamten Kurvenverlauf in entsprechende **Segmente** (Abschnitte), auf denen unterschiedlichste **Bewegungsgesetze** – spezielle mathematische Funktionen - verwendet werden können (für Beispiele von Kurvenscheiben siehe: Beispiele des Cam Design Tools) Die Bewegungsgesetze für mechanische Kurvenscheiben sind insbesondere in der VDI-Richtlinie 2143 definiert. Die elektronischen Kurvenscheiben in TwinCAT verwenden unter anderem diese Funktionen. Die Motion Functions realisieren diese Bewegungsfunktionen direkt in dem Echtzeit Treiber der NC. Im Gegensatz zu den klassischen Tabellenkopplungen, die nur diskrete Schritte (Punktewolken) mittels größerer Datenmengen in die NC übertragen, liegen hier die vollständigen Informationen sehr kompakt in der NC vor. Probleme, die ihren Ursprung in der Diskretisierung der Daten (Positionsstützstellen) in der Tabelle haben, sind damit prinzipiell nicht mehr vorhanden.

Die Realisierung der Bewegungsgesetze in der NC hat aber unter anderem noch einen weiteren entscheidenden Vorteil: Ein **Bewegungsdiagramm** – die komplette Beschreibung der Bewegung einer Slave-Achse – lässt sich jetzt auch einfach und übersichtlich von der PLC aus definieren und modifizieren. Durch entsprechende **Funktionsbausteine** der **PLC** kann diese Funktionalität einfach verwendet werden. Hierbei hat man nicht nur auf die komplette Beschreibung der Bewegung sondern auch auf einzelne Segmente oder Teilbereiche Einfluss.

Um sicherzustellen, dass eine konstruierte Kurvenscheibe auch durch das Antriebssystem abgefahren werden kann, werden durch das System **charakteristische Werte** (Kennwerte wie Maximal- und Minimalwerte der Position, Geschwindigkeit und Beschleunigung etc.) berechnet, die vom Anwender auszuwerten sind. Die letztlich resultierenden Dynamikgrenzwerte sind dabei von der Bewegung des Masters abhängig und beziehen sich auf eine konstante Master-Geschwindigkeit. Somit sind die charakteristischen Werte unter der Annahme einer konstanten Mastergeschwindigkeit idealisiert errechnet worden.

Zusätzlich wird auch die mittlere Geschwindigkeit und die effektive Beschleunigung berechnet. Diese können zum Beispiel für die Berechnung des effektiven Moments oder des Arbeitspunktes P_A (n_m ; M_{Eff}) im Drehmoment- Drehzahl – Diagramm des Motors verwendet werden. Die PLC kann auf die aktuellen charakteristischen Werte der NC mittels Funktionsbausteine zugreifen.

Im Kurvenscheibeneditor **TwinCAT Cam Design Tool** ist die Entscheidung, ob klassische Tabellenkopplungen (Punktewolke) oder Motion Functions verwendet werden sollen, durch eine Auswahl konfigurierbar. Danach werden beim Aktivieren der Konfiguration entweder die Positionstabellen oder die Punkte der Motion Functions generiert. Im Falle der Motion Functions können diese Punkte einzeln nachträglich von der PLC modifiziert werden.

In Zukunft wird man die Einzelwerte oder ganze Bereiche der Motion Functions nach entsprechenden Regeln auch Online, d. h. während die Kurvenscheibe aktiv ist, ändern können. Somit können sehr flexible Kurvenscheiben realisiert werden.

3.1 Definition eines Punktes

	Function	X start	Y start	Y' start	Y'' start	Y''' start	X end	Y end	Y' end	Y'' end	Y''' end	Symmetry
1	Synchron	0.00...	0.00...	0.00...	0.000...	0.000...	30.00...	0.00...	0.000...	0.000000	0.000000	0.500000
2	Automatic	30.0...	0.00...	0.00...	0.000...	0.000...	150.0...	20.0...	0.222...	0.000000	0.000000	0.500000
3	Synchron	150....	20.0...	0.22...	0.000...	0.000...	240.0...	40.0...	0.222...	0.000000	0.000000	0.500000
4	Automatic	240....	40.0...	0.22...	0.000...	0.000...	340.0...	0.00...	0.000...	0.000000	0.000000	0.500000
5	Synchron	340....	0.00...	0.00...	0.000...	0.000...	360.0...	0.00...	0.000...	0.000000	0.000000	0.500000

Für die Definition der Bewegung in der NC reichen genau die Informationen aus, die in dem Cam Design Tool in der Tabelle enthalten sind. Wenn man sich diese MF-Tabelle aber genauer ansieht, so sind dort auch redundante Daten vorhanden. Durch die Beschreibung der Bewegung in Segmenten (Abschnitten) ist bei einfach zusammenhängenden Bewegungsdiagrammen, der Endpunkt eines Abschnittes mit dem

Anfangspunkt des nächsten Segmentes identisch. Die komplexeren Punkttypen des Cam Design Tools wie zum Beispiel der Slidepunkt (Gleitpunkt) sind im Moment noch nicht realisiert . Zusätzlich will man die Möglichkeit haben, in einem gegebenen Bewegungsdiagramm nachträglich einzelne Punkte inaktiv (**MOTIONPOINTTYPE_IGNORE**, im Folgenden als **IGNORE** bezeichnet) zu schalten. Diese Vorgaben führen zu einer Beschreibung, in der neben dem Startpunkt eines Segmentes inklusiv der Informationen des Punktes (Geschwindigkeit, Beschleunigung, Punkttyp) auch die des Segmentes (Funktionstyp, Symmetriewert) enthalten sind.

Punktstruktur

PointIndex	UINT32	Index des Punktes
FunctionType	UINT16	Funktionstyp [▶ 11]
PointType	UINT16	Punkttyp [▶ 10]
RelIndexNextPoint	INT32	relativer Index des Endpunktes (default: 0, entspricht dann der 1)
MasterPos	REAL64	Master Position
SlavePos	REAL64	Slave Position an dieser Stützstelle
SlaveVelo	REAL64	Slave Geschwindigkeit an dieser Stützstelle
SlaveAcc	REAL64	Slave Beschleunigung an dieser Stützstelle
SlaveJerk/Symmetry	REAL64	Slave Ruck an dieser Stützstelle bzw. Symmetriewert des Segmentes für Rast in Rast Bewegungsgesetze

Durch einen relativen Index wird in dieser Struktur auf den Punktindex des Endpunktes dieses Segmentes gezeigt. Damit für die einfach zusammenhängenden Bewegungsdiagramme die Definition einfach bleibt, werden die IGNORE Punkte wirklich komplett ignoriert. Also wird der relative Punktindex intern automatisch angepasst.

Der Defaultwert des relativen Punktindex darf also Null sein, obwohl er bei einer üblichen einfach verketten Liste der Wert Eins sein sollte. Diese Information muss der Anwender also nicht aktualisieren. Zu den möglichen Punkttypen des Cam Design Tools kommt also der IGNORE Punkt hinzu.

PunktTypen

MOTIONPOINTTYPE_IGNORE	0x0000	Ignore point	Ignorierter Punkt
MOTIONPOINTTYPE_REST	0x0001	Restpoint	Rastpunkt
MOTIONPOINTTYPE_VELOCITY	0x0002	Velocitypoint	Geschwindigkeitspunkt
MOTIONPOINTTYPE_TURN	0x0004	Turnpoint	Umkehrpunkt
MOTIONPOINTTYPE_MOTION	0x0008	Motionpoint	Bewegungspunkt

Da man während die MF aktiv ist keine Punkte hinzufügen kann, ist es durch den Punkttyp IGNORE möglich, entsprechende Punkte vorzusehen. Diese können später online durch Vorgabe der entsprechenden Werte (Punkttyp ungleich IGNORE) eingeschaltet werden.

Die Masterpositionen müssen entweder streng monoton steigend oder fallend sein. Andernfalls wird dies mit einer Fehlermeldung abgelehnt.

3.2 Funktions Typen

Von den Funktionstypen des TwinCAT Cam Design Tools ist bisher nur eine Teilmenge realisiert:

Funktionstyp	Nr.	Beschreibung	Randbedingungen
POLYNOM1	1	Polynom 1ter Potenz, Gerade oder synchrone Bewegung	VV
POLYNOM3	3	Polynom 3ter Potenz	RR
POLYNOM5	5	Polynom 5ter Potenz	RR
POLYNOM8	8	Polynom 8ter Potenz	RR
SINELINE	10	Sinus Gerade	RR
MODSINELINE	11	Modifizierte Sinuslinie	RR
BESTEHORN	12	Bestehorn	RR
ACCTRAPEZ	13	Beschleunigungstrapez	RR
POLYNOM5_MM	15	Polynom 5ter Potenz mit Randwertanpassung	MM
SINE_LINE_COMBI	16	Sinus Geraden Kombination	RR
HARMONIC_COMBI_RT	17	Harmonische Kombination	RT
HARMONIC_COMBI_TR	18	Harmonische Kombination	TR
HARMONIC_COMBI_VT	19	Harmonische Kombination	VT
HARMONIC_COMBI_TV	20	Harmonische Kombination	TV
ACCTRAPEZ_RT	21	Beschleunigungstrapez	RT
ACCTRAPEZ_TR	22	Beschleunigungstrapez	TR
MODSINELINE_VV	23	Modifizierte Sinuslinie	VV
STEPFUNCTION	99	Schrittfunktion	RR

Bei den Randbedingungen der Motion Functions steht das

R (Rest) für Rast (Geschwindigkeit = Null, Beschleunigung = Null),

V (Velocity) für Geschwindigkeit (Beschleunigung = Null),

T (Turn) für Umkehr (Geschwindigkeit = Null) und

M (Motion) für Bewegung (keine Bedingungen an die Geschwindigkeit oder die Beschleunigung) .

Die Randbedingung RT steht also zum Beispiel dafür, das der Anfangspunkt des Segments die Bedingung R eine Rast (Geschwindigkeit = Null, Beschleunigung = Null) und der Endpunkt die Bedingung T eine Umkehr (Geschwindigkeit = Null) erfüllt.

Das POLYNOM1 realisiert eine Gerade oder auch eine synchrone Bewegung. Da bei dem POLYNOM1 durch die Master- und Slavewerte an den Segmentgrenzen die Geschwindigkeit eindeutig bestimmt ist, muss diese nicht mit übergeben werden bzw. der vorgegebene Wert (z.B. eine Null) wird durch den intern berechneten Wert überschrieben. Das einzige Bewegungsgesetz, das beliebige Randbedingungen berücksichtigt, ist im Moment das POLYNOM5_MM. Allerdings sollte man dafür auch die entsprechenden Randparameter zur Verfügung stellen. Wenn die benachbarten Segmente eines POLYNOM5_MM Segmentes vom Funktionstyp POLYNOM1 oder die Randbedingung R für Rast erfüllen, werden die Randparameter von den internen Werten der Geschwindigkeit des POLYNOM1 bzw. den Werten der Rast

(Geschwindigkeit = Null, Beschleunigung = Null) überschrieben. Die Beispiele des Cam Design Tools geben einen Eindruck von den Möglichkeiten und zu verwendenden Bewegungsgesetzen für den Entwurf von Kurvenscheiben.

Bei den nicht Rast in Rast Typen werden die Randwerte benötigt. Eine Ausnahme ist die synchrone Bewegung (Polynom1) bei der aus den aktuellen Positionen der Punkte die aktuellen Geschwindigkeiten automatisch berechnet werden. Die direkt benachbarten Segmente aktualisieren entsprechend ihre internen Koeffizienten für die Berechnung.

Wenn im Cam Design Tool auf die Motion Function umgeschaltet wird, sind alle Funktionstypen, die nicht in der NC realisiert sind, in der Tabelle rot hinterlegt. Diese werden als Polynom 5ter Potenz mit Randwertanpassung vom Cam Design Tool der NC zur Verfügung gestellt.

3.3 Programmierung durch die SPS

In dem TwinCAT Cam Design Tool wird eine Basis Kurvenscheibe mit der maximal nötigen Anzahl von Punkten entworfen. Diese steht dann beim TwinCAT Systemstart automatisch zur Verfügung oder sie wird per Knopfdruck (Download) in die NC geladen. Mit Funktionsbausteinen der PLC können jetzt zum einen die einzelnen Punkte komplett ignoriert oder deren Werte manipuliert werden. Die aktuell in der NC vorhanden Daten lassen sich im Online Modus des Cam Design Tools visualisieren oder per Upload in das Tool laden. So lassen sich die Daten graphisch und numerisch überprüfen. Zusätzlich gibt es die Möglichkeit die Daten von der NC als Tabelle zu diskreten Masterpositionen inklusive der Ableitungen aufbereiten zu lassen. Diese Daten können dann von der PLC verwendet oder an eine Visualisierungssoftware weitergereicht werden. Die Visualisierung kann dann die Punktdaten inklusive der Ableitungen darstellen, ohne diese selbst berechnen zu müssen.

Da die Daten der Kurvenscheibe exakt so gefahren werden, wie sie in den Motion Functions definiert sind, muss man sich in der Konzeptphase genau überlegen, welche Funktionalität dem Endanwender (Maschinenbediener) zur Verfügung gestellt werden soll. Beziehungsweise wie man die Eingaben der Bedieneroberfläche so überprüft, dass nur sinnvolle Kurvenscheiben entstehen können. Die Berechnung der Charakteristischen Werte [▶ 14] ist dafür ein zusätzliches effizientes Hilfsmittel. Zur grafischen Kontrolle der Motion Functions in der Bedieneroberfläche kann die NC die Werte der Position, Geschwindigkeit und Beschleunigung in einer Tabelle (Punktewolke) zur Verfügung stellen.

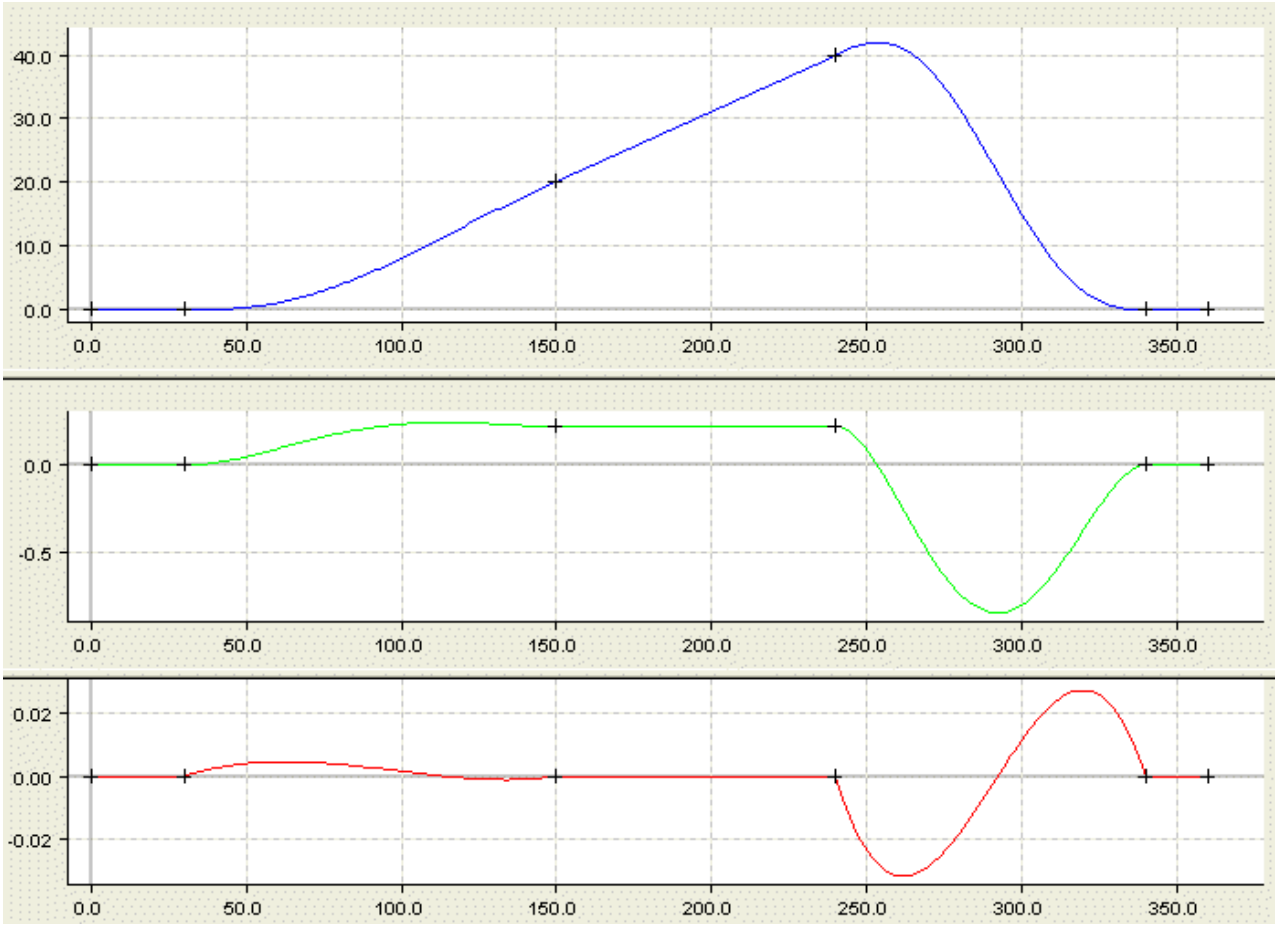
Beispiel einer Anwendung:

Bei einer Transfereinrichtung einer Pressenanlage wird die Bewegung des Transfers durch die Hubwinkel der Presse vorgegeben. Eine an diesen Hubwinkel gekoppelte Bewegung wird in der Bedieneroberfläche der Transfereinrichtung eingegeben. Diese Informationen werden von der PLC zusammen mit den intern gespeicherten Kenndaten der Mechanik verarbeitet. Die PLC überträgt die Kurvenscheibendaten an die NC für die Bewegung der Achsen. Für die Visualisierung holt die PLC die Kurvenscheibendaten (Position, Geschwindigkeit und Beschleunigung) in Tabellenform von der NC und stellt diese der Bedieneroberfläche zur Verfügung.

Für eine Optimierungsrechnung liest die PLC die Charakteristischen Werte [▶ 14] (insbesondere die mittlere Drehzahl und die effektive Beschleunigung) der aktuellen Kurvenscheibe für die Berechnung des Arbeitspunktes P_A (n_m ; M_{Eff}) im Drehmoment- Drehzahl – Diagramm des jeweiligen Motors. So kann iterativ die maximal zulässige Drehzahl der Transfereinrichtung ermittelt werden.

Beispiel einer Kurvenscheibe:

	Function	X start	Y start	Y' start	Y'' start	Y''' start	X end	Y end	Y' end	Y'' end	Y''' end	Symmetry
1	Synchron	0.00...	0.00...	0.00...	0.000...	0.000...	30.00...	0.00...	0.000...	0.000000	0.000000	0.500000
2	Automatic	30.0...	0.00...	0.00...	0.000...	0.000...	150.0...	20.0...	0.222...	0.000000	0.000000	0.500000
3	Synchron	150....	20.0...	0.22...	0.000...	0.000...	240.0...	40.0...	0.222...	0.000000	0.000000	0.500000
4	Automatic	240....	40.0...	0.22...	0.000...	0.000...	340.0...	0.00...	0.000...	0.000000	0.000000	0.500000
5	Synchron	340....	0.00...	0.00...	0.000...	0.000...	360.0...	0.00...	0.000...	0.000000	0.000000	0.500000



Die obige Kurvenscheibe soll als Motion Function realisiert werden:

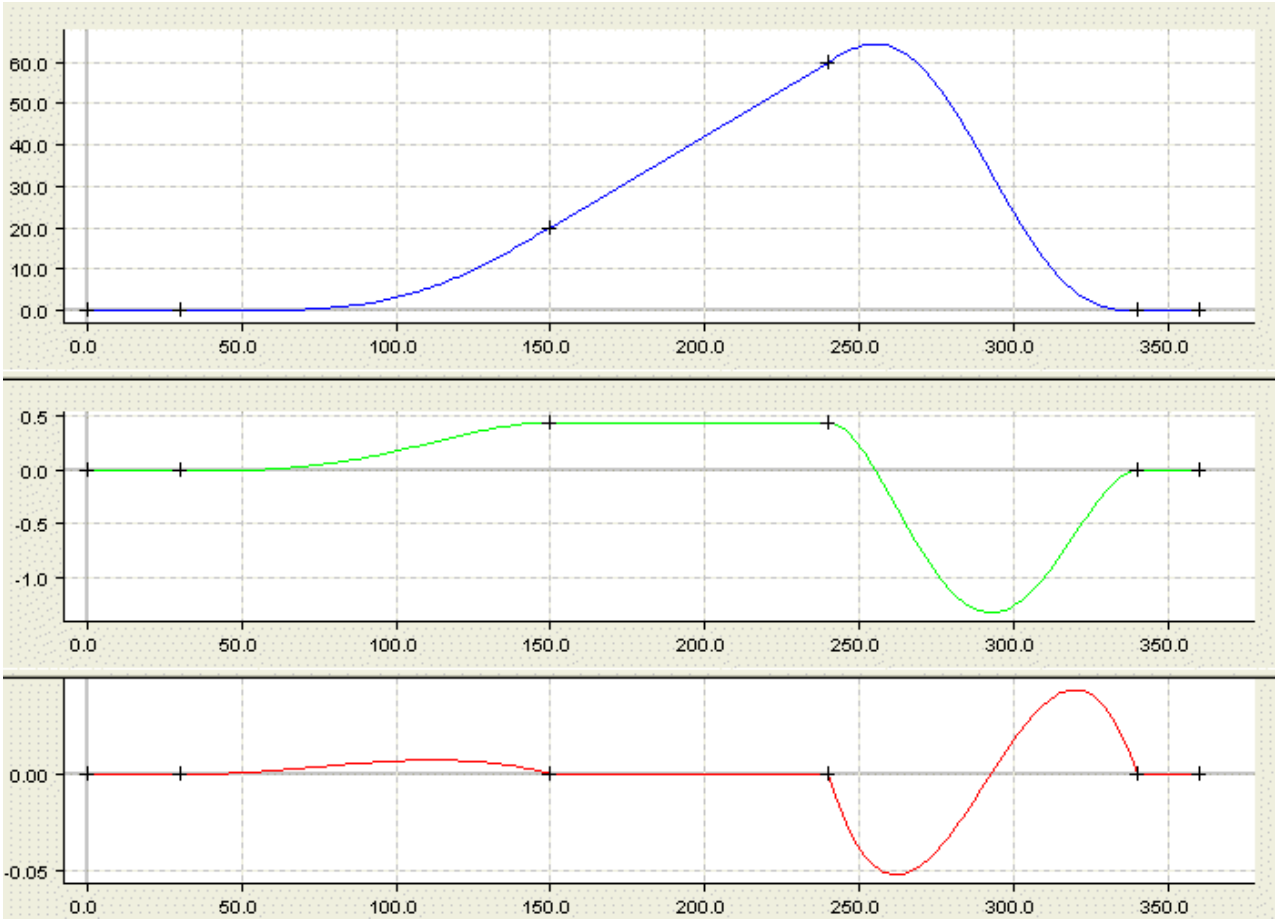
Point Index	Function Type	Point Type	RelIndex NextPoint	Master Pos	Slave Pos	Slave Velo	Slave Acc	Slave Jerk
1	POLYNOM1	REST	1 bzw. 0	0	0	0	0	0
2	POLYNOM5_MM	REST	1 bzw. 0	30	0	0	0	0
3	POLYNOM1	VELOCITY	1 bzw. 0	150	20	0	0	0
4	POLYNOM5_MM	VELOCITY	1 bzw. 0	240	40	0	0	0
5	POLYNOM1	REST	1 bzw. 0	340	0	0	0	0
6	POLYNOM1	REST	0	360	0	0	0	0

Die Geschwindigkeit muss in diesem Fall nicht vorgeben werden, da sie intern gerechnet wird (siehe Funktions Typen [▶ 11]). Auch der relative Index kann für alle Werte zu Null gesetzt werden, da das Diagramm nur aus einfach zusammenhängenden Punkten besteht. Nur bei komplexen Punktdefinitionen wie zum Beispiel dem Slidepunkt muss der Index zwingend ungleich Null gesetzt werden. Die Masterposition muss monoton steigend sein.

Änderung 1:

Durch das Schreiben eines einzigen Punktes kann jetzt die komplette Kurvenscheibe verändert werden:

4	POLYNOM5_MM	VELOCITY	1	240	60	0	0	0



Änderung 2:

oder durch das Ignorieren der zwei relevanten Punkt kann die komplette Kurvenscheibe auf Null gesetzt werden, dass heißt alle Slavewerte sind gleich Null.

3	POLYNOM1	IGNORE	1	150	20	0	0	0
4	POLYNOM5_MM	IGNORE	1	240	40	0	0	0

Die Aktivierung dieser Änderungen erfolgt im Moment noch erst nach der Kopplung, wird aber in Zukunft online nach entsprechenden Regeln möglich sein.

Weitere Beispiele von Kurvenscheiben sind in: Beispiele des Cam Design Tools (wobei das Beispiel 4 noch nicht mit Motion Functions realisiert werden kann.)

3.4 Charakteristische Werte

Durch die interne Berechnung der *Charakteristischen Werte (Characteristic Values)* kann das Risiko, durch falsche Eingabe von Werten bestimmte physikalische Grenzen der Achse zu verletzen, eliminiert werden.

Die in der Tabelle beschriebenen Größen, werden von der NC berechnet und können mit einem PLC FB oder per ADS gelesen werden. Die dynamischen Größen können nur für einen Master berechnet werden, der sich mit einer konstanten Geschwindigkeit bewegt. Die Geschwindigkeit des Masters ist ebenfalls in der Struktur enthalten. Standardmäßig werden die Werte für eine Geschwindigkeit von 1.0 berechnet, diese

Werte bezeichnet man auch als normierte Werte. Eine Umrechnung in eine andere konstante Mastergeschwindigkeit ist in der NC implementiert und lässt sich über einen FB der PLC aufrufen. Die umskalierten Werte sind in der gleichen Struktur gespeichert. Der Bezug, auf welche Mastergeschwindigkeit sich die charakteristischen Kennwerte beziehen, ist in der ersten Variable namens Mastergeschwindigkeit zu erkennen.

Für eine Bewegungsfunktion mit konstanter maximaler Beschleunigung, ist die Position des Masters die maximale Beschleunigung (fMPosAtSAccMax) mehrdeutig. Das gleiche gilt auch für die anderen Masterpositionen von Maximal- oder Minimalwerten, diese sind also mit der notwendigen Vorsicht zu behandeln. Entsprechendes gilt für die von den Masterpositionen abgeleitete Größen (fSVeloAtSAccMin, fSVeloAtSAccMax).

Die letztlich resultierenden Dynamikgrenzwerte sind von der Bewegung des Masters abhängig, daher sind diese berechneten Werte nur für eine Bewegung mit konstanter Mastergeschwindigkeit exakt.

Nr.	Namen	Beschreibung		Datentyp
Master				
1.	fMasterVeloNom	master nominal velocity (normed = 1.0)	1	REAL64
Start Point				
2.	fMasterPosStart	master start position	2	REAL64
3.	fSlavePosStart	slave start position	3	REAL64
4.	fSlaveVeloStart	slave start velocity	4	REAL64
5.	fSlaveAccStart	slave start acceleration	5	REAL64
6.	fSlaveJerkStart	slave start jerk	6	REAL64
End Point				
7.	fMasterPosEnd	master end position	7	REAL64
8.	fSlavePosEnd	slave end position	8	REAL64
9.	fSlaveVeloEnd	slave end velocity	9	REAL64
10.	fSlaveAccEnd	slave end acceleration	10	REAL64
11.	fSlaveJerkEnd	slave end jerk	11	REAL64
Minimal Slave Position				
12.	fMPosAtSPosMin	master position at slave minimum position	12	REAL64
13.	fSlavePosMin	slave minimum position	13	REAL64
Minimal Slave Velocity				
14.	fMPosAtSVeloMin	master position at slave minimum velocity	14	REAL64
15.	fSlaveVeloMin	slave minimum velocity	15	REAL64
Minimal Slave Acceleration				
16.	fMPosAtSAccMin	master pos. at slave minimum acceleration	16	REAL64
17.	fSlaveAccMin	slave minimum acceleration	17	REAL64
18.	fSVeloAtSAccMin	slave velocity at slave min. acceleration	18	REAL64

Minimal Slave Jerk/Dyn.Mom				
19.	fSlaveJerkMin	slave minimum jerk	19	REAL64
20.	fSlaveDynMomMin	slave minimum dynamic momentum	20	REAL64 **NOT SUPPORTED**
Maximal Slave Position				
21.	fMPosAtSPosMax	master position at slave maximum position	21	REAL64
22.	fSlavePosMax	slave maximum position	22	REAL64
Maximal Slave Velocity				
23.	fMPosAtSVeloMax	master position at slave maximum velocity	23	REAL64
24.	fSlaveVeloMax	slave maximum velocity	24	REAL64
Maximal Slave Acceleration				
25.	fMPosAtSAccMax	master pos. at slave maximum acceleration	25	REAL64
26.	fSlaveAccMax	slave maximum acceleration	26	REAL64
27.	fSVeloAtSAccMax	slave velocity at slave max. acceleration	27	REAL64
Maximal Slave Jerk/Dyn.Mom				
28.	fSlaveJerkMax	slave maximum jerk	28	REAL64
29.	fSlaveDynMomMax	slave maximum dynamic momentum	29	REAL64 **NOT SUPPORTED**
Mean Effective Values				
30.	fSlaveVeloMean	slave mean absolute velocity	30	REAL64
31.	fSlaveAccEff	slave effective acceleration	31	REAL64

3.5 Online Modifikation

Die Motion Function haben nicht nur den Vorteil, daß sie von der PLC per Funktionsbaustein definiert werden können, sondern auch zur Laufzeit modifiziert werden können. Das kann zum Beispiel die Veränderung des Slavewertes eines Umkehrpunktes sein, um den Hub zu verstellen,. Das können aber auch umfangreichere Änderungen für die Anfahrkurve einer komplexen zyklischen Bewegung (ohne Rast und konstante Geschwindigkeit) sein.

Die Online Änderung der **Motion Function (MF)** folgt dabei bestimmten Regeln. So kann die Aktivierung ab einer bestimmten Position der Achse (AtMasterAxisPos), sofort (Instantaneous) oder ab dem nächsten Zyklus erfolgen. Die Aktivierungsregel gilt dabei so lange, bis eine neue Regel übertragen wird. Mit dem Befehl zur Online-Änderungen von MF-Punkten können dann sowohl Einzelpunkte als auch zusammenhängende Bereiche von mehreren Punkten geändert werden. Für die Synchronisation ist in dem zyklischen Interface ein entsprechendes Bit vorgesehen. Also muss in einem Programm zum einen die Aktivierungsregel definiert werden (per Funktionsbaustein) und zum anderen die geänderten Werte der Motion Function Punkte übertragen werden. Diese Übertragung erfolgt mit den gleichen ADS-Befehlen oder Funktionsbausteinen der PLC mit denen bei Start des Systems die Motion Function Punkte definiert werden.

ActivationMode	
Instantaneous (Default)	0
AtMasterCamPos	1
AtMasterAxisPos	2
NextCycle	3
AsSoonAsPossible (ehemaliger Default)	5
Off	6
DeleteQueuedData	7

Der ActivationMode **Instantaneous** aktiviert im nächsten Tick die übertragenen Werte der Motion Function Punkte. Diese Funktionalität sollte man natürlich nur dann einsetzen, wenn die neuen Werte sich nicht direkt auf die aktuellen Werte der Slaveachse auswirken. So betrifft eine Änderung zum Beispiel ein anderes Segment ohne eine Rückwirkung auf das aktuelle Segment. Mit der Option **AsSoonAsPossible** wird überprüft welche Segment nicht geändert wurden und die Änderung wird erst dann realisiert, wenn der Slave sich in so einem Segment befindet. Die beiden Modi **AtMasterAxisPos** und **AtMasterCamPos**unterscheiden sich darin, dass **AtMasterAxisPos** die Achsposition als Referenz nimmt, während **AtMasterCamPos** sich auf die Position im Bewegungsdiagramm bezieht. Bei einer zyklischen Bewegung von 0 bis 360 wird die Achsposition 180 nur einmalig überfahren, die Position im Bewegungsdiagramm 180 aber in jedem Zyklus überschritten. Wenn die Achsposition schon überschritten wurde führt der Aktivierungsbefehl mit dem Modi **AtMasterAxisPos** zu einem Fehler. In dem Modus **AtMasterCamPos** wird der Aktivierungsbefehl in dem nächsten Zyklus ab der gegebenen Position ausgeführt. Mit **NextCycle** wird die Aktivierungsposition auf das Ende bzw. den Anfang des Zyklus gesetzt. Mit dem Mode **Off** werden die Änderungsbefehle nicht ausgeführt. Mit **DeleteQueuedData**werden die bereits übermittelten aber noch nicht aktivierten Punktedaten gelöscht. Die aktuell gefahrene Kurvenscheibe ist davon natürlich nicht betroffen.

Für eine komfortable Änderung gibt es die Möglichkeit den SlaveScalingMode auf **AutoOffset** zu setzen. Dabei werden der Offset des Slaves automatisch so korrigiert, daß in der Slaveposition kein Sprung entsteht. In der Geschwindigkeit oder der Beschleunigung sind hier Sprünge möglich. Wenn zum Beispiel die Aktivierungsposition in einem Segment konstanter Geschwindigkeit liegt und das geänderte Segment die gleiche Geschwindigkeit hat, kann hier ohne Probleme die Änderung vollzogen werden.

SlaveScalingMode	
UserDefined (Default)	0
AutoOffset	1
Off	2

Die Speicherverwaltung der Motion Function Punkte erlaubt nicht nachträglich noch Punkte hinzuzufügen, daher gibt es die Möglichkeit Punkte zu ignorieren (IGNORE). Für eine effiziente Suche des aktuellen Segmentes war es erforderlich, die Masterpositionen in einer monoton steigend Reihenfolge abzuspeichern. Also auch die Punkte die später eingeschaltet werden sollen, müssen sich schon von der Masterposition an der richtigen Stelle befinden. Beim Starten des System sollte man sich also darüber klar sein, welche maximale Menge an Punkten benötigt wird.

HINWEIS

Die Online Modifikation bietet viele Möglichkeiten der Änderung. Damit ergeben sich aber auch Situationen, die zu Fehlern in den Achsen oder größeren Probleme führen können. Dieses mächtige Werkzeug sollte man also nur mit Vorsicht und nach entsprechenden Simulationstests an einer Maschine zum Einsatz bringen.

4 Kurvenscheiben-Achse - Positionstabellen

Eine Slave-Achse, deren Positions-Sollwerte aus den Positions-Sollwerten einer anderen Achse mittels einer vorausberechneten **Tabelle** ermittelt werden heißt **Kurvenscheiben-Achse (Tabellen-Slave-Achse)**.

Die Tabelle enthält die zur jeweiligen Masterposition gehörige Slaveposition und wird linear interpoliert. Die Tabellen können in der Masterposition äquidistant monoton steigend oder nur monoton steigend sein. Die Tabellen können linear oder zyklisch abgearbeitet werden. Es ist möglich die Masterposition und/oder die Slaveposition mit einem Offset zu versehen. Diese Offsets sind on-line änderbar. Die Tabellen können absolut oder relativ in Bezug auf die Koppelpositionen (von Master und /oder Slave) abgearbeitet werden.

Interfaces

Interfaces [▶ 18]

Die Interfaces bestehen aus dem **System Manager**, **SPS-Bausteinen** und dem **zyklischen NC-SPS-Achs-Interface**.

Tabellentypen

Tabellentypen [▶ 19]

Es gibt vier Tabellentypen die nach Tabelleninhalt und Abarbeitungsmodus unterschieden werden.

- Äquidistante Tabellen bei denen die Masterposition in äquidistanten Schritten ansteigt:
  - **Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
  - **Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.
- Nicht-Äquidistante Tabellen bei denen die Masterposition in nicht notwendigerweise äqudistanten Schritten streng monoton ansteigt:
  - **Nicht-Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
  - **Nicht-Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.

Abarbeitungsmodi

Abarbeitungsmodi [▶ 21]

Es gilt Master-Offsets und/oder Slave-Offsets, unterschiedliche Bezugssysteme (relativ/absolut - in Bezug auf die Koppelposition), und die Möglichkeit der **linearen** oder **zyklischen** Abarbeitung.

Ablauf

Ablauf [▶ 24]

Der Ablauf besteht aus den Koppelvorbereitungen (Verfahren der Achsen auf Koppelposition), Koppeln, Starten der Master-Achse, eventuellen on-line Änderungen des/der Offsets, Bremsen der Slave-Achse (direkt über den Master oder indirekt über die Tabelle) und dem Abkoppeln.

4.1 Interfaces

Der Benutzer kann die Kurvenscheiben-Achse über den System Manager oder SPS-Bausteine bedienen. Außerdem steht ihm (im System Manager und über die SPS) ein zyklisches NC-SPS Interface zur Verfügung, das das **Achs-Prozessabbild** beinhaltet und Zugriff auf diverse Variablen ermöglicht.

System Manager

Der System Manager bietet den Zugriff auf alle Funktionalitäten der Kurvenscheiben-Achse:

- Logisches Koppeln mit Angabe der Masterachse und Setzen der Koppeltabelle.
- Abkoppeln.

SPS-Bausteine

Die SPS Library bietet den Zugriff auf alle Slave-Achs Funktionalitäten :

- Logisches Koppeln mit Angabe der Masterachse und Setzen der Koppeltabelle.
- Stopp.

NC-SPS-Interface

Das zyklische NC-SPS Achs-Interface beinhaltet das Achs-Prozessabbild und insbesondere Zugriff auf die Koppelinformation (Koppelflag).

4.2 Tabellentypen

Es gibt vier Tabellentypen die nach Tabelleninhalt und Abarbeitungsmodus unterschieden werden.

- Äquidistante Tabellen bei denen die Masterposition in äquidistanten Schritten ansteigt:
  - **Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
  - **Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.
- Nicht-Äquidistante Tabellen bei denen die Masterposition in nicht notwendigerweise äqudistanten Schritten streng monoton ansteigt:
  - **Nicht-Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
  - **Nicht-Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.

Die Multitabellen-Slaves arbeiten nur mit Nicht-Äquidistante Tabellen.

Art und Quelle der Gefahr

- Nicht-Äquidistante Tabellen basieren auf einem linearen Suchalgorithmus, der per Bisektion gestartet und reinitialisiert wird.
- Äquidistante Tabellen sind auch Nicht-Äquidistante Tabellen.
- Jede Tabelle kann linear oder zyklisch abgearbeitet werden.

Generelle Tabellen-Konventionen

- Die Tabellen enthalten nur Binärdaten.
- Die Tabelle besteht aus einem Kopf (erste Zeile) und den Tabellendaten (restliche Zeilen).
- Der Kopf enthält zwei Zahlen des Typs unsigned short. In der ersten Spalte steht die Anzahl der Zeilen (ohne Kopf), in der zweiten Spalte die Anzahl der Spalten (für Tabellen von Tabellen-Slaves: immer 2). Es gibt keine Trennzeichen zwischen den Daten.
- Bis auf den Kopf enthält die Tabelle nur Daten des Typs double.
- Die erste Spalte (abgesehen von der Kopfzeile) enthält die Masterposition, die zweite Spalte die zugehörige Slaveposition (jeweils in mm). Es gibt keine Trennzeichen zwischen den Daten.

Die Tabelle(n) muss der angestrebten Dynamik angepasst sein. An dynamisch kritischen Stellen, wo die Master-Achse langsam fährt, muss die Tabelle genug Daten enthalten, um präzise auflösen zu können.

Tabellen-Bezeichnungen

Die Tabelle *T* enthält *N* Zeilen und pro Zeile zwei Daten, die Masterposition **T***[k][0]* und die zugehörige Slaveposition. **T***[k][1], 0 ≤ k < N*. Bezeichnungen:
T*[k,0]* Masterwert, *0 ≤ k < N*,
T*[k,1]* Slavewert, *0 ≤ k < N*,
T*(y)* interpolierter Slave-Positionswert aus dem Intervall **T***[k,1]* und **T***[k+1,1]*, wobei **T***[k,0] < y ≤* **T***[k+1,0]* für nicht-äquidistante Tabellen und **T***[k,0] ≤ y <* **T***[k+1,0]* für äquidistante Tabellen gilt.

Zyklische Tabellen (Zyklische Abarbeitung)

Zyklische Tabellen sind durch folgende zusätzliche Parameter charakterisiert:

- Eine **Master-Periode**: es wird unterstellt, dass die Periode nicht-negativ ist und bei *0.0* anfängt. Die Tabellen erstrecken sich vereinbarungsgemäß von **T***[0][0] = 0.0* bis **T***[N-1][0]* = Periode.
- Eine **Slave-Differenz pro Master-Periode** (mit beliebigem Vorzeichen) die angibt, um wie viel mm sich die Slave-Position pro Periode ändert. Ist der Slave selber periodisch, dann ist diese Differenz *0.0*. Es gilt: **T***[N-1][1]* - **T***[0][1]* = Slave-Differenz pro Master-Periode.

Auswertung Linearer Tabellen

- Wenn ein Tabellenüberlauf/Tabellenunterlauf in der Masterposition vorliegt, wird als Slave-Positionswert der Tabellenwert am jeweiligen Rand der Tabelle ausgegeben.
- Insbesondere gilt: liegt die Masterposition bei Kopplung außerhalb der Tabelle, dann springt die Slave-Achse im ersten Schritt auf den jeweiligen Tabellenrandwert was im Allgemeinen einen Schleppabstand bewirkt.
- Die aktuelle Slaveposition wird durch lineare Interpolation ermittelt.
- Die aktuelle Slavegeschwindigkeit wird, falls das möglich ist, per Tabelle und Kettenregel aus der Master-Sollgeschwindigkeit berechnet (das setzt voraus, dass die Master-Sollgeschwindigkeit hinreichend glatt ist). Ansonsten wird die Slavegeschwindigkeit per numerischer Differentiation berechnet (und ist dementsprechend grob, wenn mehrfach innerhalb eines Zeilenpaars interpoliert wird).
- Die aktuelle Slavebeschleunigung wird immer per numerischer Differentiation berechnet und ist dementsprechend grob.

Auswertung Zyklischer Tabellen

- Die Tabelle verfügt über einen Zykluszähler, der bei Tabellenüberlauf inkrementiert und bei Tabellenunterlauf dekrementiert wird.
- Bei Tabellenüberlauf/Tabellenunterlauf wird die Slavedifferenz pro Masterperiode mal Zykluszähler auf die Slaveposition addiert.
- Die aktuelle Slaveposition wird durch lineare Interpolation ermittelt.
- Die aktuelle Slavegeschwindigkeit wird, falls das möglich ist, per Tabelle und Kettenregel aus der Master-Sollgeschwindigkeit berechnet (das setzt voraus, dass die Master-Sollgeschwindigkeit hinreichend glatt ist). Ansonsten wird die Slavegeschwindigkeit numerischer Differentiation berechnet (und ist dementsprechend grob, wenn mehrfach innerhalb eines Zeilenpaars interpoliert wird).
- Die aktuelle Slavebeschleunigung wird immer per numerischer Differentiation berechnet und ist dementsprechend grob.

4.3 Abarbeitungsmodi

Kurvenscheiben können mit gekoppelten Master/Slave-Achsen linear oder zyklisch abgearbeitet werden. Sie lassen sich sowohl Master- als auch Slave-seitig skalieren und mit einem Positions-Offset versehen. Das Bezugssystem einer Kurvenscheibe kann dabei absolut oder relativ zur Koppelposition der Achsen interpretiert werden.

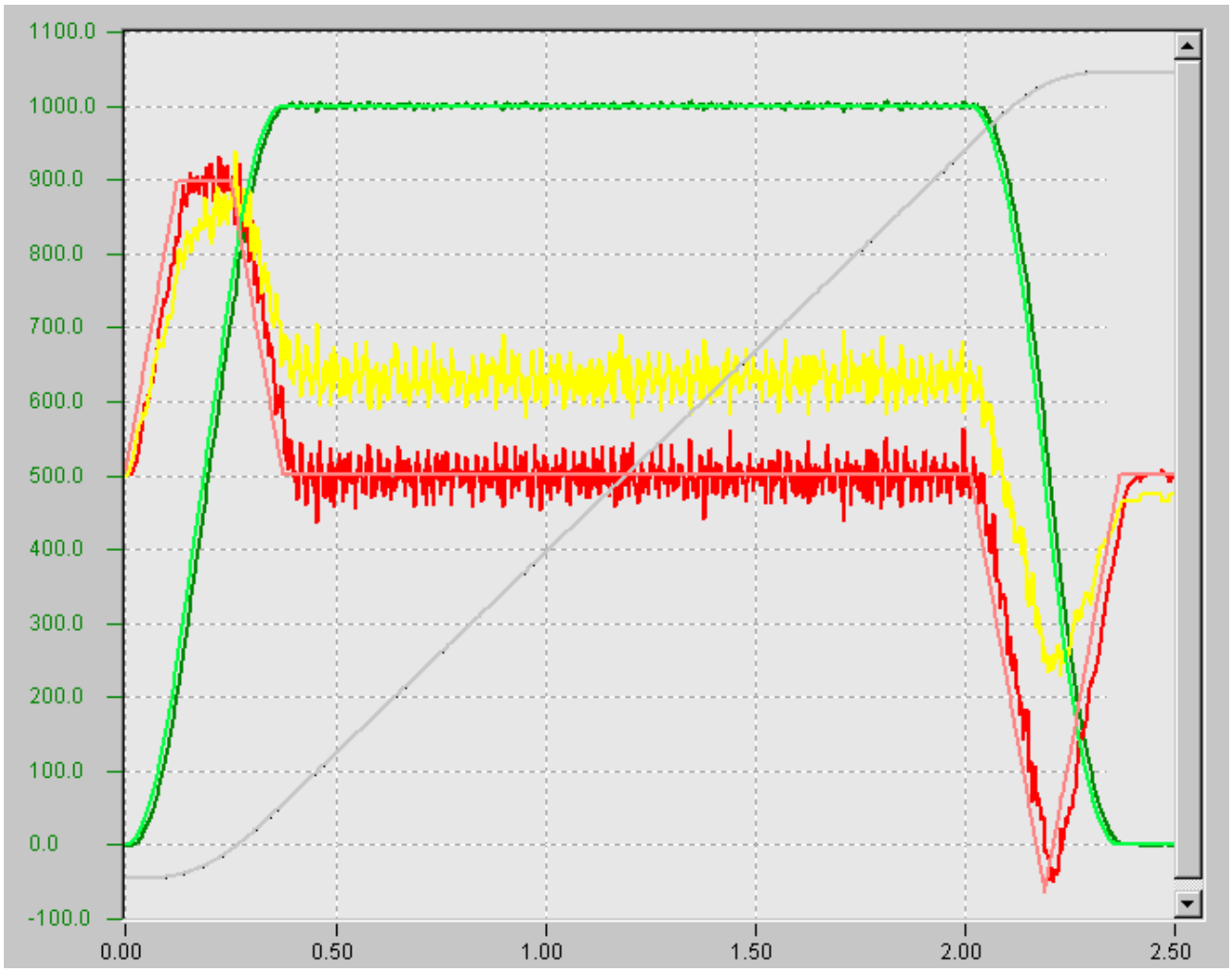


Abb. 1: Lineare Abarbeitung

Offsets

Es kann sowohl ein Slave-Offset, als auch ein Master-Offset angegeben werden:

- Der Master-Offset (*MO*) wird **vor** allen anderen Operationen auf die Masterposition addiert.
- Der Slave-Offset (*SO*) wird **nach** allen anderen Operationen auf die Slave-Position addiert.

Skalierung

Es gibt eine Slave-Skalierung, mit der alle Slave-Positionen multiplikativ skaliert werden. Ferner gibt es eine Master-Skalierung, mit der alle Master-Positionen dividiert werden. Die Skalierungen sind üblicherweise durch die Kopplung des Slaves mit 1.0 vorbesetzt, man kann aber von diesen Werten beliebig abweichen (Master-Skalierung darf nicht 0.0 sein). Eine Änderung der Skalierung ist auch online möglich, sollte dann aber nur in Maßen vorgenommen werden, da - auch im dynamischen Stillstand der Achsen - ein Schleppabstand erzeugt wird. Bei zyklischer Abarbeitung (s.u.) wird die Slave-Differenz (Slavehub) pro Masterperiode ebenfalls mit der Slave-Skalierung multipliziert.

- Die Master-Skalierung (*MS*) wird als **Divisor** verwendet.
- Die Slave-Skalierung (*SS*) wird als **Multiplikator** verwendet.

Zyklische Abarbeitung

Für zyklische Abarbeitung gilt: die Modulo-Operation erfolgt auf tiefster Ebene, d.h. unmittelbar vor Auswertung der Tabelle.

- Die Tabelle verfügt über einen **Zykluszähler** (Z_m), der bei Tabellenüberlauf inkrementiert und bei Tabellenunterlauf dekrementiert wird.
- Die Masterposition wird **absolut** übergeben, nicht Modulo.
- Bei Tabellenüberlauf/Tabellenunterlauf wird die **Slave-Differenz (Slavehub) pro Masterperiode** (*SH*) mal Zykluszähler auf die Slave-Position addiert.

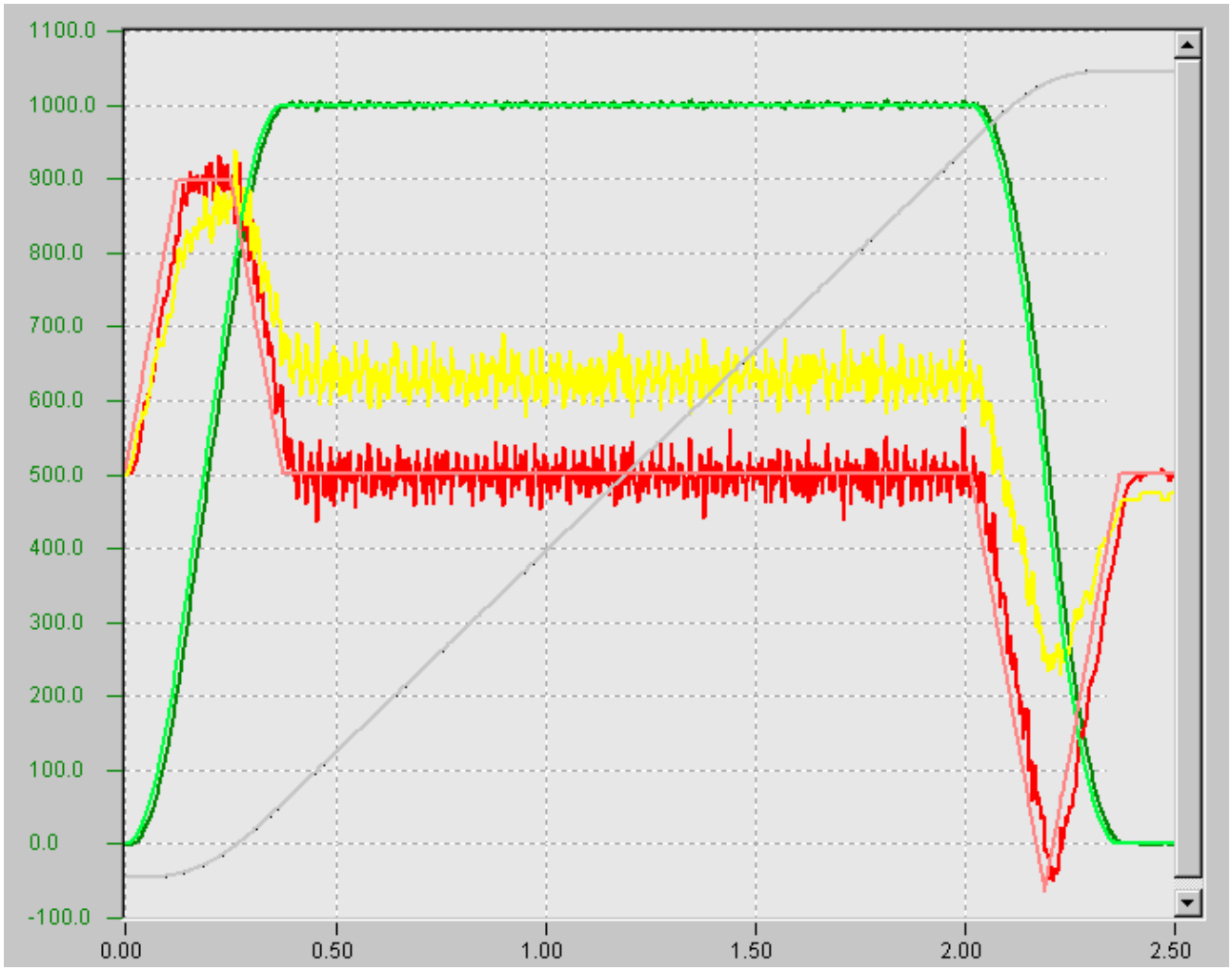


Abb. 2: Zyklische Abarbeitung

Bezeichnungen

1. P_m absolute Masterposition
2. P_{mc} Master-Koppelposition (Masterstartposition), nur für *relatives Bezugssystem* wirksam !
3. MO Master-Offset
4. MS Master-Skalierung
5. P_s absolute Slave-Position
6. P_{sc} Slave-Koppelposition (Slave-Startposition), nur für *relatives Bezugssystem* wirksam !
7. SO Slave-Offset
8. SS Slave-Skalierung
9. **T**(•) Tabelle
10. mod[•] Modulorechnung mit Masterperiode (Beispiele für eine Masterperiode von 360 Grad: mod[40] = 40, mod[400] = 40, mod[-20] = 340)
11. Z_m Zykluszähler des Masters

12. SH Slave-Differenz (Slavehub) pro Masterperiode

Bezugssysteme

Die Position einer Slave-Achse ergibt sich nach einer Berechnungsvorschrift aus den Daten der Kurvenscheibentabelle und der aktuellen Master-Position. Wenn ein relatives Bezugssystem für Master oder Slave gewählt wird, dann sollte die zugehörige Positionsspalte der Tabelle mit *0.0* beginnen. "Relativ" bezieht sich immer auf die Koppelposition der jeweiligen Achse. Im Folgenden werden die Berechnungsvorschriften für Kombinationen aus relativem und absolutem Bezug erläutert.
Oftmals ergibt sich für den konkreten Anwendungsfall die Notwendigkeit die nachfolgenden Formeln so umzustellen, dass sich der Slave-Offset (*SO*) bzw. der Master-Offset (*MO*) ergibt.

Bezugssystem für Lineare Tabellen

Allgemeiner Fall: Offsets ungleich 0.0 und Skalierungen ungleich 1.0

- Master absolut und Slave absolut: $P = SO\ (\ (MO + P\ ) / MS\ )_s + \mathbf{T}_m \cdot SS$
- Master relativ und Slave absolut: $P = SO\ (\ (MO + P - P\ ) / MS\ )_s + \mathbf{T}_{mmc} \cdot SS$
- Master absolut und Slave relativ: $P = P + SO + (\ (MO + P\ ) / MS\ )_{ssc}\mathbf{T}_m \cdot SS$
- Master relativ und Slave relativ: $P = P + SO + (\ (MO + P - P\ ) / MS\ )_{ssc}\mathbf{T}_{mmc} \cdot SS$

Vereinfachter Sonderfall 1: Offsets ungleich 0.0 und Skalierungen sind 1.0

- Master absolut und Slave absolut: $P = SO\ (\ (MO + P\ )\ )_s + \mathbf{T}_m$
- Master relativ und Slave absolut: $P = SO\ (\ (MO + P - P\ )\ )_s + \mathbf{T}_{mmc}$
- Master absolut und Slave relativ: $P = P + SO + (\ (MO + P\ )\ )_{ssc}\mathbf{T}_m$
- Master relativ und Slave relativ: $P = P + SO + (\ (MO + P - P\ )\ )_{ssc}\mathbf{T}_{mmc}$

Vereinfachter Sonderfall 2: Offsets sind 0.0 und Skalierungen sind 1.0

- Master absolut und Slave absolut: $P = (\ P\ )_s\mathbf{T}_m$
- Master relativ und Slave absolut: $P = (\ P - P\ )_s\mathbf{T}_{mmc}$
- Master absolut und Slave relativ: $P = P + (\ P\ )_{ssc}\mathbf{T}_m$
- Master relativ und Slave relativ: $P = P + (\ P - P\ )_{ssc}\mathbf{T}_{mmc}$

Bezugssystem für Zyklische Tabellen

Bei Tabellenüberlauf bzw. Tabellenunterlauf wird die Slavedifferenz (Slavehub) pro Masterperiode multipliziert mit dem Zykluszähler auf die Slaveposition addiert.

Allgemeiner Fall: Offsets und Slavehub sind ungleich 0.0 und Skalierungen ungleich 1.0

- Master absolut und Slave absolut: $P = SO\ (\ mod[\ (MO + P\ ) / MS\ ]\ )_s + \mathbf{T}_m \cdot SS + SH \cdot Z_M$
- Master relativ und Slave absolut: $P = SO\ (\ mod[\ (MO + P - P\ ) / MS\ ]\ )_s + \mathbf{T}_{mmc} \cdot SS + SH \cdot Z_M$
- Master absolut und Slave relativ: $P = P + SO + (\ mod[\ (MO + P\ ) / MS\ ]\ )_{ssc}\mathbf{T}_m \cdot SS + SH \cdot Z_M$
- Master relativ und Slave relativ: $P = P + SO + (\ mod[\ (MO + P - P\ ) / MS\ ]\ )_{ssc}\mathbf{T}_{mmc} \cdot SS + SH \cdot Z_M$

Vereinfachter Sonderfall 1: Offsets und Slavehub sind ungleich 0.0, Skalierungen sind 1.0

- Master absolut und Slave absolut: $P = SO\ (\ mod[\ (MO + P\ )\ ]\ ) + SH_s + \mathbf{T}_m \cdot Z_M$
- Master relativ und Slave absolut: $P = SO\ (\ mod[\ (MO + P - P\ )\ ]\ )_s + \mathbf{T}_{mmc} + SH \cdot Z_M$
- Master absolut und Slave relativ: $P = P + SO + (\ mod[\ (MO + P\ )\ ]\ ) + SH_{ssc}\mathbf{T}_m \cdot Z_M$
- Master relativ und Slave relativ: $P = P + SO + (\ mod[\ (MO + P - P\ )\ ]\ ) + SH_{ssc}\mathbf{T}_{mmc} \cdot Z_M$

Vereinfachter Sonderfall 2: Offsets und Slavehub sind 0.0, Skalierungen sind 1.0

- Master absolut und Slave absolut: $P = (\ mod[\ P\ ]\ )_s\mathbf{T}_m$
- Master relativ und Slave absolut: $P = (\ mod[\ P - P\ ]\ )_s\mathbf{T}_{mmc}$

- Master absolut und Slave relativ: $P = P + (mod[P])_{ssc}\mathbf{T}_{m}$
- Master relativ und Slave relativ: $P = P + (mod[P - P])_{ssc}\mathbf{T}_{mmc}$

Interfaces

4.4 Ablauf

Der Ablauf besteht aus den Koppelvorbereitungen (Verfahren der Achsen auf Koppelposition), Koppeln, Starten der Master-Achse, eventuellen on-line Änderungen des/der Offsets, Bremsen der Slave-Achse (direkt über den Master oder indirekt über die Tabelle) und dem Abkoppeln.

Vorbereitung

1. Die Master-Achse wird im PTP-Betriebszustand auf Koppelposition = Startposition gefahren.
2. Die zur Masterposition gehörige Slaveposition wird aus der Tabelle ermittelt. Das Lesen der Tabelle ergibt nur den Tabellenwert, dieser muß vom Benutzer noch mit den, bei Kopplung eventuell anvisierten Offsets und Abarbeitungsmodi (relativ/absolut) verrechnet werden.
3. Die Slave-Achse wird PTP-Modus auf Koppelposition = Startposition gefahren.

Koppeln/Start

Das Ankoppeln erfolgt off-line (Slave und Master stehen).

Der Slave startet dann, wenn der Master startet und der Tabelleninhalt es bedingt (die Slaveposition ist nicht konstant).

On-Line

Die Tabellen-Master-Slave-Kopplung ist eine **Streckensteuerung**.

Die **Offsets** können on-line geändert werden. Ändern des/der Offsets sollte bei Fahrt nur mit Maßen durchgeführt werden, da ein Schleppabstand erzeugt wird..

Bei Tabellenüberlauf/Tabellenunterlauf zyklischer Tabellen wird die Slavedifferenz pro Masterperiode mal Zykluszähler auf die Slaveposition addiert.

Es gibt keinen separaten Stop der Slave-Achse.

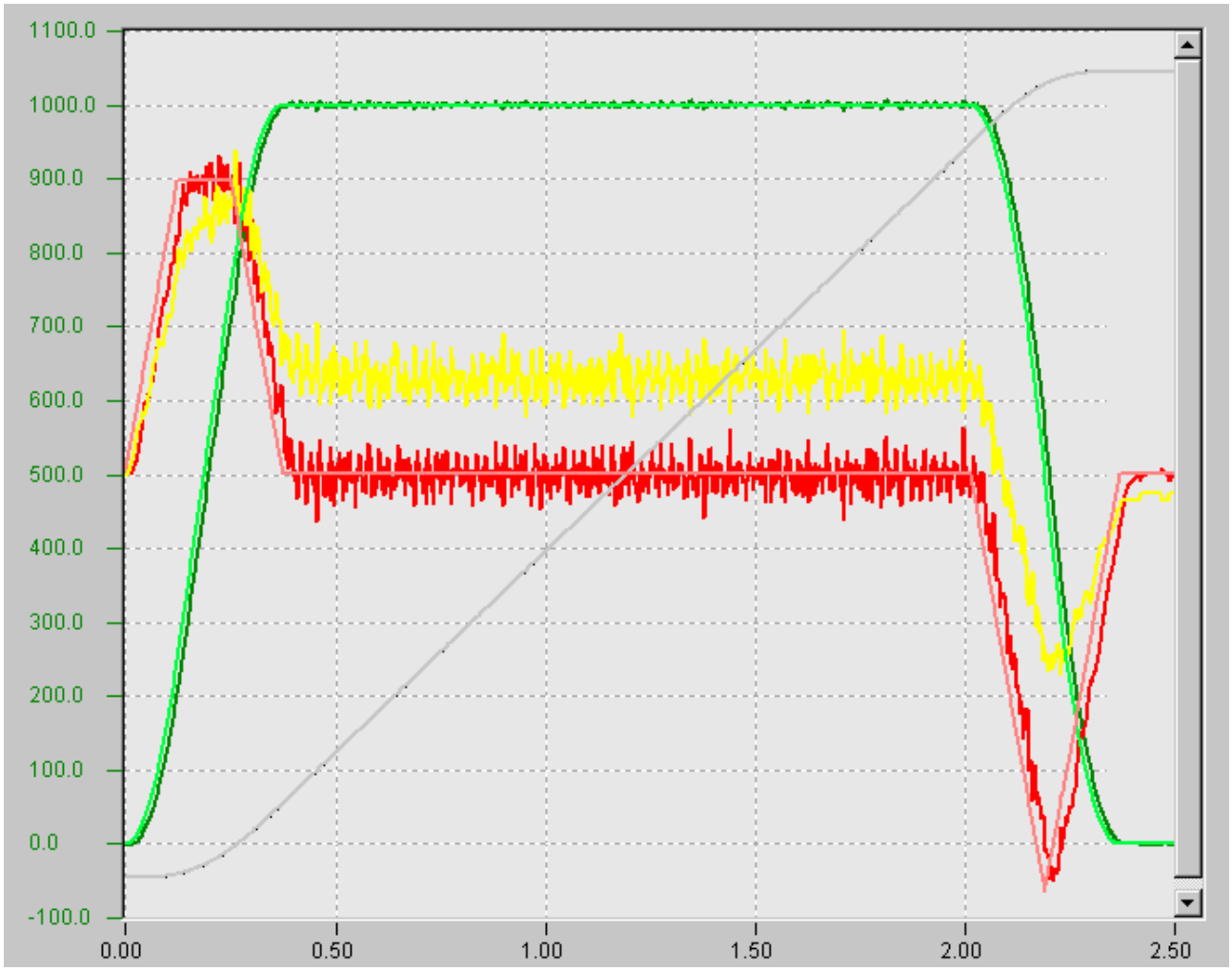


Abb. 3: On-line Offsetänderungen

Errorhandling

Es gibt **kein explizites Errorhandling**, wenn ein Tabellenüberlauf/Tabellenunterlauf linearer Tabellen passiert. Die Slave-Achse bleibt am jeweiligen Tabellenrand instantan stehen und erzeugt (je nach dynamischem Zustand) einen Schleppabstand.

Stop/Abkoppeln

Die Slave-Achse stoppt entweder, wenn der Master stoppt (Streckensteuerung) oder wenn der Tabelleninhalt es bedingt (die Slaveposition ist über einen weiten Bereich der Masterposition konstant).

Das Abkoppeln kann off-line (Master und Slave stehen) oder semi-off-line (nur der Slave steht) durchgeführt werden.

Interfaces

- **Tabelle Lesen**
  - 1. SPS-Bausteine
- **Ankoppeln**
  - 1. System Manager
  - 2. SPS-Bausteine
- **Abkoppeln**
  - 1. System Manager

◦ 2. SPS-Bausteine

4.5 Online Modifikation der Skalierung oder des Offsets

Die Online Änderung der Skalierung oder des Offset von Kurvenscheiben (Positionstabellen und Motion Function) folgt bestimmten Regeln. So kann die Aktivierung ab einer bestimmten Position der Achse (AtMasterAxisPos), sofort (Instantaneous) oder ab dem nächsten Zyklus erfolgen. Mit dem Befehl zur Online-Änderung oder dem Funktionsbaustein (**MC_CamScaling**) werden die Aktivierungsregel (ActvivationMode, ActivationPosition, MasterScalingMode und SlaveScalingMode) und die Parameter MasterOffset, SlaveOffset, MasterScaling und SlaveScaling übertragen.

ActivationMode	
Instantaneous (Default)	0
AtMasterCamPos	1
AtMasterAxisPos	2
NextCycle	3

Der ActivationMode **Instantaneous** aktiviert im nächsten Tick die übertragenen Werte. Die beiden Modi **AtMasterAxisPos**und **AtMasterCamPos** unterscheiden sich darin, dass **AtMasterAxisPos** die Achsposition als Referenz nimmt, während **AtMasterCamPos** sich auf die Position im Bewegungsdiagramm bezieht. Bei einer zyklischen Bewegung von 0 bis 360 wird die Achsposition 180 nur einmalig überfahren, die Position im Bewegungsdiagramm 180 aber in jedem Zyklus überschritten. Wenn die Achsposition schon überschritten wurde führt der Aktivierungsbefehl mit dem Modi **AtMasterAxisPos** zu einem Fehler. In dem Modus **AtMasterCamPos** wird der Aktivierungsbefehl in dem nächsten Zyklus ab der gegebenen Position ausgeführt. Mit **NextCycle** wird die Aktivierungsposition auf das Ende bzw. den Anfang des Zyklus gesetzt.

Für eine komfortable Änderung gibt es die Möglichkeit den SlaveScalingMode oder den Master ScalingMode auf **AutoOffset**zu setzen. Dabei werden der Offset des Slaves bzw. des Masters automatisch so korrigiert, daß in der Slaveposition kein Sprung entsteht. In der Geschwindigkeit oder der Beschleunigung sind hier Sprünge möglich.

MasterScalingMode / SlaveScalingMode	
UserDefined (Default)	0
AutoOffset	1
Off	2

Beispiele:

Die Kurvenscheibe hat am Zyklusanfang eine Rast (Geschwindigkeit und Beschleunigung sind gleich Null). Mit den Werten:

MasterOffset = 0, SlaveOffset =0, MasterScaling=1, SlaveScaling =0 , MasterScalingMode = Off, SlaveScalingMode = UserDefined , ActivationMode **=** NextCycle

kann die Bewegung des Slaves still gesetzt werden. Die Kopplung bleibt dabei erhalten. Mit dem entsprechenden Befehl mit SlaveScaling =1.0 kann die Bewegung später wieder eingeschaltet werden.

Die Kurvenscheibe hat eine Bereich konstanter Geschwindigkeit von 200 bis 250 Grad. Mit den Werten:

MasterOffset = 10, SlaveOffset =0, MasterScaling=1, SlaveScaling =1 , MasterScalingMode = UserDefined , SlaveScalingMode = AutoOffset, ActivationMode **=** AtMasterCamPos, ActivationPosition = 200

wird der MasterOffset absolut um 10 verändert. Durch SlaveScalingMode = AutoOffset wir der Slave automatisch so korrigiert, dass kein Sprung in der Position entsteht. Das Ergebnis entspricht einem instantanem Phasing.

HINWEIS

Die Online Modifikation bietet viele Möglichkeiten der Änderung. Damit ergeben sich aber auch Situationen, die zu Fehlern in den Achsen oder größeren Probleme führen können. Dieses mächtige Werkzeug sollte man also nur mit Vorsicht und nach entsprechenden Simulationstests an einer Maschine zum Einsatz bringen.

4.6 Multitabellen-Achse (Sonderform)

Eine Slave-Achse, deren Positions-Sollwerte aus den Positions-Sollwerten einer anderen Achse mittels mehrerer vorausberechneter Tabellen (seriell und/oder parallel) ermittelt werden heißt **Multitabellen-Slave-Achse**.

Die Tabellen enthalten die zur jeweiligen Masterposition gehörige Slaveposition, sowie eine Skalierung - bestehend aus Masteroffset, Slaveoffset, Mastersteigung und Slavesteigung - die eine beliebige **Skalierung** der Tabelle ermöglichen. Im Normalfall hat man es mit einer normierten Starttabelle, normierten Arbeitstabellen und einer normierten Stopptabelle zu tun (Profiltabellen), die seriell hintereinander abgearbeitet werden. Dazu kommen Korrekturtabellen, die parallel zu den Profiltabellen abgearbeitet werden und die es ermöglichen, on-line Positionskorrekturen der Slave-Achse vorzunehmen.

Interfaces und Konfiguration

Interfaces und Konfiguration [▶ 28]

Die Interfaces bestehen aus dem **System Manager**, **SPS-Bausteinen** und dem **zyklischen NC-SPS-Achs-Interface**.

Tabellentypen

Tabellentypen [▶ 19]

Es gibt zwei Tabellentypen die nach Abarbeitungsmodus unterschieden werden. Es handelt sich dabei um Nicht-Äquidistante Tabellen bei denen die Masterposition in nicht notwendigerweise äqudistanten Schritten streng monoton ansteigt:

- **Nicht-Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
- **Nicht-Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.

Abarbeitungsmodi

Abarbeitungsmodi [▶ 30]

Es gibt für jede Tabelle einer Multi-Tabellen-Achse seine **Skalierung** bestehend aus Master-Offset, Master-Steigung, Slave-Offset, Slave-Steigung und Tabellengewicht, mit der die Tabelle beliebig affin in beiden Koordinaten skaliert werden kann. Damit besteht die Möglichkeit, sich auf eine geringe Anzahl von **normierten Tabellen** zu beschränken. Es gibt die Möglichkeit der linearen oder zyklische Abarbeitung.

Ablauf

Ablauf [▶ 31]

Der Ablauf besteht aus den Koppelvorbereitungen (Verfahren der Achsen auf Koppelposition), Koppeln, Starten der Master-Achse, parallelen Tabellenwechseln, eventuellen seriellen **on-line Aktivierungen von Korrekturtabellen**, Bremsen der Slave-Achse (direkt über den Master oder indirekt über die Tabelle(n)) und dem Abkoppeln.

4.6.1 Interfaces und Konfiguration

Der Benutzer kann die Multi-Tabellen-Slave-Achse über den System Manager oder SPS-Bausteine bedienen. Außerdem steht ihm (im System Manager und über die SPS) ein zyklisches NC-SPS Interface zur Verfügung, das das **Achs-Prozessabbild** beinhaltet und Zugriff auf diverse Variablen ermöglicht.

System Manager

Der System Manager bietet den Zugriff auf alle Funktionalitäten der Multi-Tabellen-Slave-Achse:

- Logisches Koppeln mit Angabe der Masterachse und der Koppel-Tabellen.
- Stopp.

SPS-Bausteine

Die SPS Library bietet den Zugriff auf alle Slave-Achs Funktionalitäten :

- Logisches Koppeln mit Angabe der Masterachse und der Koppel-Tabellen.
- Stopp.

NC-SPS-Interface

Das zyklische NC-SPS Achs-Interface beinhaltet das Achs-Prozessabbild und insbesondere Zugriff auf die Koppelinformation (Koppelflag).

4.6.2 Tabellentypen

Es gibt vier Tabellentypen, die nach Tabelleninhalt und Abarbeitungsmodus unterschieden werden.

- Äquidistante Tabellen bei denen die Masterposition in äquidistanten Schritten ansteigt:
  - **Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
  - **Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.
- Nicht-Äquidistante Tabellen bei denen die Masterposition in nicht notwendigerweise äqudistanten Schritten streng monoton ansteigt:
  - **Nicht-Äquidistante lineare Tabellen**, d.h. Tabellen die am Tabellenrand gemäß dem Tabelleninhalt abgearbeitet werden.
  - **Nicht-Äquidistante zyklische Tabellen**, d.h. Tabellen die am Tabellenrand periodisch fortgesetzt abgearbeitet werden.

Die Multitabellen-Slaves arbeiten nur mit Nicht-Äquidistante Tabellen.

- Nicht-Äquidistante Tabellen basieren auf einem linearen Suchalgorithmus, der per Bisektion gestartet und reinitialisiert wird.
- Äquidistante Tabellen sind auch Nicht-Äquidistante Tabellen
- Jede Tabelle kann linear oder zyklisch abgearbeitet werden.

Generelle Tabellen-Konventionen

- Die Tabellen enthalten nur Binärdaten.
- Die Tabelle besteht aus einem Kopf (erste Zeile) und den Tabellendaten (restliche Zeilen).
- Der Kopf enthält zwei Zahlen des Typs unsigned short. In der ersten Spalte steht die Anzahl der Zeilen (ohne Kopf), in der zweiten Spalte die Anzahl der Spalten (für Tabellen von Tabellen-Slaves: immer 2). Es gibt keine Trennzeichen zwischen den Daten.
- Bis auf den Kopf enthält die Tabelle nur Daten des Typs double.

- Die erste Spalte (abgesehen von der Kopfzeile) enthält die Masterposition, die zweite Spalte die zugehörige Slaveposition (jeweils in mm). Es gibt keine Trennzeichen zwischen den Daten.

Die Tabelle(n) muss der angestrebten Dynamik angepasst sein. An dynamisch kritischen Stellen, wo die Master-Achse langsam fährt, muss die Tabelle genug Daten enthalten, um präzise auflösen zu können.

Tabellen-Bezeichnungen

Die Tabelle *T* enthält *N* Zeilen und pro Zeile zwei Daten, die Masterposition **T***[k][0]* und die zugehörige Slaveposition. **T***[k][1], 0 ≤ k < N*. Bezeichnungen:
T*[k,0]* Masterwert, *0 ≤ k < N*,
T*[k,1]* Slavewert, *0 ≤ k < N*,
T*(y)* interpolierter Slave-Positionswert aus dem Intervall **T***[k,1]* und **T***[k+1,1]*, wobei **T***[k,0] < y ≤* **T***[k+1,0]* für nicht-äquidistante Tabellen und **T***[k,0] ≤ y <* **T***[k+1,0]* für äquidistante Tabellen gilt.

Zyklische Tabellen (Zyklische Abarbeitung)

Zyklische Tabellen sind durch folgende zusätzliche Parameter charakterisiert:

- Eine **Master-Periode**: es wird unterstellt, dass die Periode nicht-negativ ist und bei *0.0* anfängt. Die Tabellen erstrecken sich vereinbarungsgemäß von **T***[0][0] = 0.0* bis **T***[N-1][0]* = Periode.
- Eine **Slave-Differenz pro Master-Periode** (mit beliebigem Vorzeichen) die angibt, um wie viel mm sich die Slave-Position pro Periode ändert. Ist der Slave selber periodisch, dann ist diese Differenz *0.0*. Es gilt: **T***[N-1][1]* - **T***[0][1]* = Slave-Differenz pro Master-Periode.

Auswertung Linearer Tabellen

- Wenn ein Tabellenüberlauf/Tabellenunterlauf in der Masterposition vorliegt, wird als Slave-Positionswert der Tabellenwert am jeweiligen Rand der Tabelle ausgegeben.
- Insbesondere gilt: liegt die Masterposition bei Kopplung außerhalb der Tabelle, dann springt die Slave-Achse im ersten Schritt auf den jeweiligen Tabellenrandwert was im Allgemeinen einen Schleppabstand bewirkt.
- Die aktuelle Slaveposition wird durch lineare Interpolation ermittelt.
- Die aktuelle Slavegeschwindigkeit wird, falls das möglich ist, per Tabelle und Kettenregel aus der Master-Sollgeschwindigkeit berechnet (das setzt voraus, dass die Master-Sollgeschwindigkeit hinreichend glatt ist). Ansonsten wird die Slavegeschwindigkeit per numerischer Differentiation berechnet (und ist dementsprechend grob, wenn mehrfach innerhalb eines Zeilenpaars interpoliert wird).
- Die aktuelle Slavebeschleunigung wird immer per numerischer Differentiation berechnet und ist dementsprechend grob.

Auswertung Zyklischer Tabellen

- Die Tabelle verfügt über einen Zykluszähler, der bei Tabellenüberlauf inkrementiert und bei Tabellenunterlauf dekrementiert wird.
- Bei Tabellenüberlauf/Tabellenunterlauf wird die Slavedifferenz pro Masterperiode mal Zykluszähler auf die Slaveposition addiert.
- Die aktuelle Slaveposition wird durch lineare Interpolation ermittelt.
- Die aktuelle Slavegeschwindigkeit wird, falls das möglich ist, per Tabelle und Kettenregel aus der Master-Sollgeschwindigkeit berechnet (das setzt voraus, dass die Master-Sollgeschwindigkeit hinreichend glatt ist). Ansonsten wird die Slavegeschwindigkeit numerischer Differentiation berechnet (und ist dementsprechend grob, wenn mehrfach innerhalb eines Zeilenpaars interpoliert wird).
- Die aktuelle Slavebeschleunigung wird immer per numerischer Differentiation berechnet und ist dementsprechend grob.

4.6.3 Abarbeitungsmodi

Es gibt für jede Tabelle einer Multi-Tabellen-Slave-Achse eine **Skalierung** bestehend aus Master-Offset, Master-Steigung, Slave-Offset, Slave-Steigung und Tabellengewicht, mit der die Tabelle beliebig affin in beiden Koordinaten skaliert werden kann. Damit besteht die Möglichkeit, sich auf eine geringe Anzahl von **normierten Tabellen** zu beschränken. Es gibt die Möglichkeit der linearen oder zyklische Abarbeitung

Bezugssystem

Die skalierten Achspositionen sind immer absolute Angaben.

Zyklische Abarbeitung

Für zyklische Abarbeitung gilt: die Modulo-Operation erfolgt auf **tiefster Ebene**, d.h. unmittelbar vor Auswertung der Tabelle.

- Die Tabelle verfügt über einen **Zykluszähler**, der bei Tabellenüberlauf inkrementiert und bei Tabellenunterlauf dekrementiert wird.
- Die Masterposition wird absolut übergeben, nicht Modulo.
- Bei Tabellenüberlauf/Tabellenunterlauf wird die Slavedifferenz pro Masterperiode mal Zykluszähler auf die Slaveposition addiert.

Tabellentypen

Bei einer Multitabellen-Slave-Achse ist die Gesamtstrecke üblicherweise in mehrere Segmente geteilt, denen eigene (normierte) Tabellen zugeordnet sind (**Profiltabellen**): Starttabelle, Arbeitstabelle(n), Stopptabelle. Diese Tabellen werden seriell hintereinander abgefahren. Hinzu kommen **Korrekturtabellen**, die nur in eingeschränkten Bereichen gültig sind, und die parallel zu den Profiltabellen abgearbeitet werden. Die Profiltabellen können linear oder zyklisch abgearbeitet werden; die Korrekturtabellen werden immer linear abgearbeitet.

Skalierung

Tab. 1: Skalierungsparameter einer Tabelle

Skalierungsparameter	Bedeutung und Randbedingungen
RangeLow RL	Untere Grenze des Gültigkeitsbereis der Tabelle (*RL* < *RH*)
RangeLow RH	Obere Grenze des Gültigkeitsbereis der Tabelle (*RH* > *RL*)
Tabellengewicht W	Gewicht mit dem die Tabelle in eine parallele konvexe Kombination von Tabellen eingeht (*W* > *0.0*)
Master-Offset MO	Offset auf die Masterposition *[0][0]***T**
Master-Steigung MS	Skalierungfaktor der Masterposition der Tabelle: *[k][0]* <- *MS* × *[k][0]***TT**
Slave-Offset SO	Offset auf die Slaveposition *[0][1]***T**
Slave-Steigung SS	Skalierungfaktor der Slaveposition der Tabelle: *[k][1]* <- *SS* × *[k][1]***TT**

Das on-line setzen der Skalierung dient nicht dazu die Parameter zu ändern, während die Tabelle aktiv ist, sondern die Parameter zu setzen, bevor die Tabelle aktiv wird. **Zur Korrektur einer aktiven Tabelle dienen alleine die Korrekturtabellen** (deren Parameter auch nicht geändert werden dürfen, während die Tabelle aktiv ist).

Bemerkung zur Skalierung von Korrekturtabellen:

- RangeLow = *-DBL_MAX* wird automatisch durch die NC-gesetzt..
- RangeHigh = *DBK_MAX* wird automatisch durch die NC-gesetzt..

- Tabellengewicht wird vom Benutzer gesetzt (üblicherweise *1.0*).
- Der Master-Offset wird automatisch durch die NC-gesetzt..
- Master-Steigung, Slave-Offset unf Slave-Steigung werden vom Benutzer gesetzt.

Interfaces

4.6.4 Ablauf

Der Ablauf besteht aus den Koppelvorbereitungen (Verfahren der Achsen auf Koppelposition), Koppeln, Starten der Master-Achse, parallelen Tabellenwechseln, eventuellen seriellen **on-line Aktivierungen von Korrekturtabellen**, Bremsen der Slave-Achse (direkt über den Master oder indirekt über die Tabelle(n)) und dem Abkoppeln.

Vorbereitung

1. Die Master-Achse wird im PTP-Betriebszustand auf Koppelposition = Startposition gefahren.
2. Die zur Masterposition gehörige Slaveposition wird aus der/den Tabelle(n) ermittelt. Das Lesen einer Tabelle ergibt nur den Tabellenwert, dieser muss vom Benutzer noch mit den, bei Kopplung eventuell anvisierten Offsets, Steigungen und Gewichten sowie der geplanten Tabellenzusammensatzung verrechnet werden.
3. Die Slave-Achse wird PTP-Modus auf Koppelposition = Startposition gefahren.

Koppeln/Start

Das Ankoppeln erfolgt off-line (Slave und Master stehen).

Der Slave startet dann, wenn der Master startet und der/die Tabelleninhalt(e) es bedingt/bedingen (die Slaveposition ist nicht konstant).

On-Line: Generell

Die Tabellen-Master-Slave-Kopplung ist eine **Streckensteuerung**.

Bei Tabellenüberlauf/Tabellenunterlauf zyklischer Tabellen wird die Slavedifferenz pro Masterperiode mal Zykluszähler auf die Slaveposition addiert.

Es gibt keinen separaten Stop der Slave-Achse.

On-Line: Tabellenwechsel und Tabellenzusammensatzung

Der Wechsel der Korrekturtabellen wird durch die SPS in dem Sinne vorbereitet, dass die Tabellenskalierung jeder Tabelle früh genug durch die SPS gesetzt wird. Zu den Skalierungsparametern gehört die Festlegung des (positionsmäßigen absoluten) Gültigkeitsintervalls jeder Tabelle. Der eigentliche Tabellenwechsel wird durch die NC vorgenommen.

Sind gleichzeitig mehrere Tabellen (mehrere Profiltabellen oder Profiltabellen und Korrekturtabellen) aktiv, dann wird die Slaveposition folgendermaßen berechnet (der Index *i* bezieht sich jeweils auf die *i*-te aktive Tabelle):

P_mi aktuelle absolute Masterposition,

SO_i Slave-Offset der *i*-ten Tabelle,

SS_i Slave-Steigung der *i*-ten Tabelle,

MO_i Master-Offset der *i*-ten Tabelle,

MS_i Master-Steigung der *i*-ten Tabelle,

W_i Gewicht der *i*-ten Tabelle.

Tab. 2: Ermittlung der Slaveposition über mehrere parallele Tabellen

Skalierungsparameter	Bedeutung und Randbedingungen
eine lineare Tabelle	P_si(P_m) = SO_i + SS_i × **T**_i(MO_i + MS_i × P_m)
eine zyklische Tabelle	P_si(P_m) = SO_i + SS_i × **T**_i([MO_i + MS_i × P_m]mod(•))
alle Profiltabellen	P_s = Σ_i W_i × P_si(P_m)

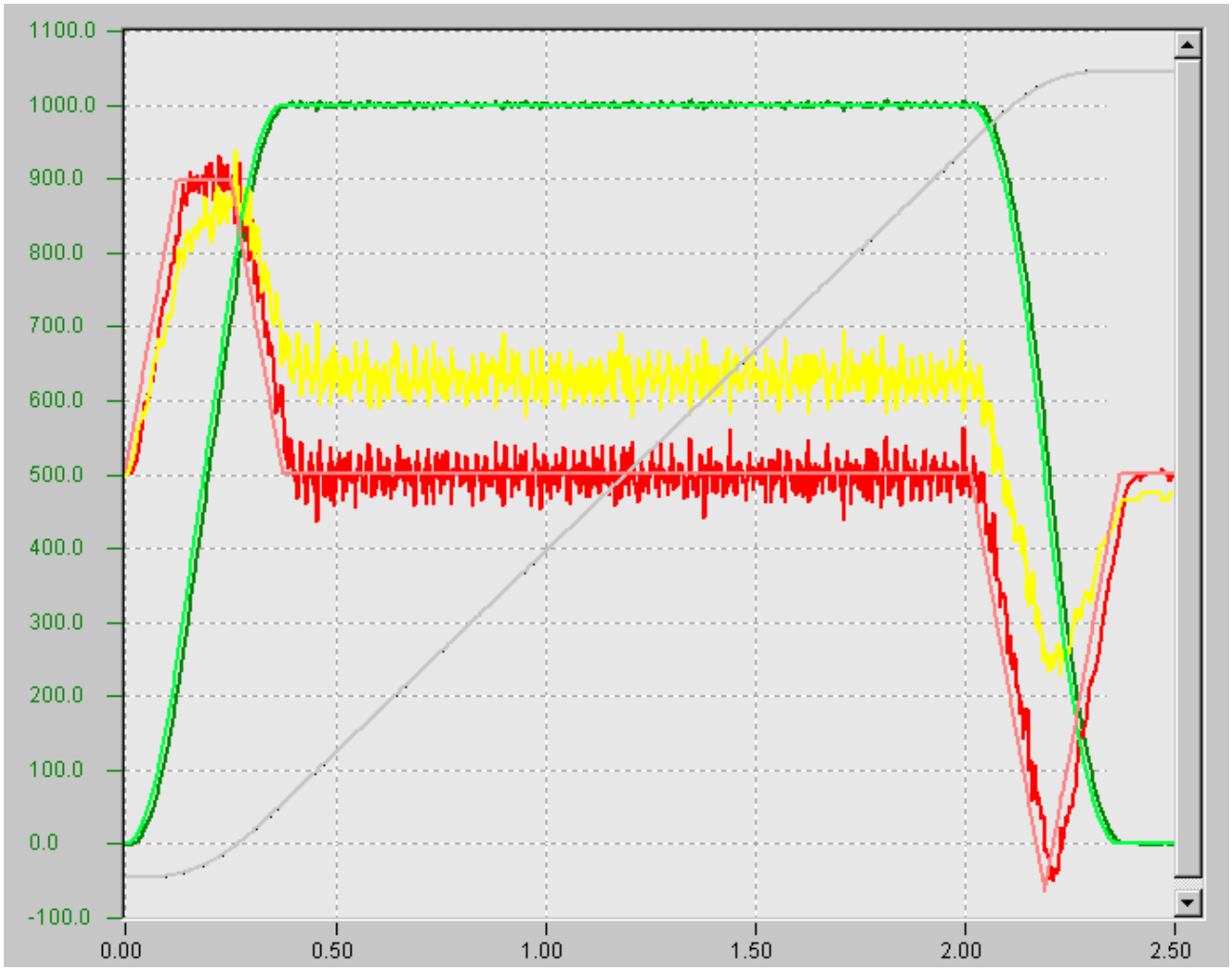


Abb. 4: Lineare serielle Abarbeitung

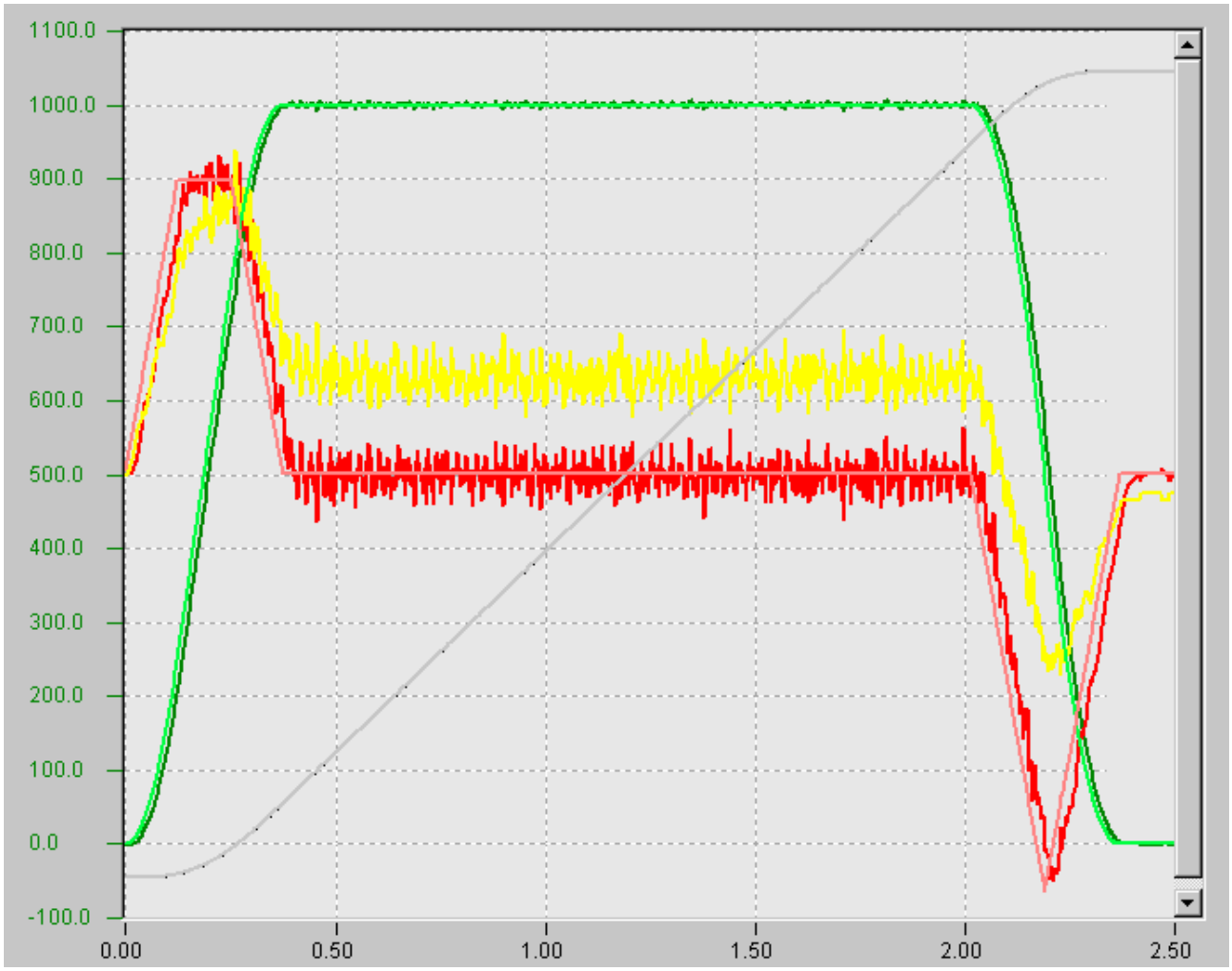


Abb. 5: Lineare/zyklische/lineare serielle Abarbeitung

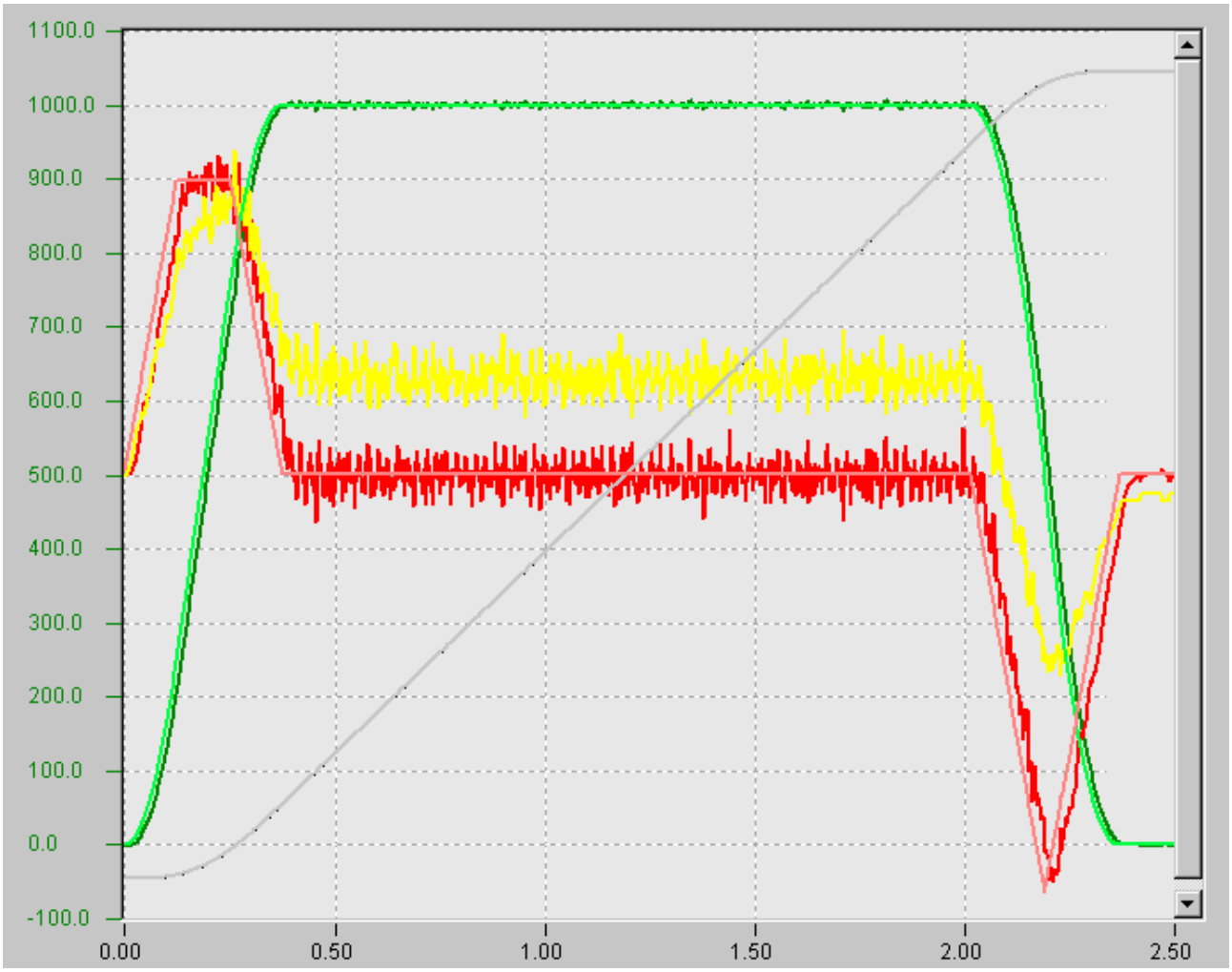


Abb. 6: Lineare/zyklische'lineare serielle Abarbeitung mit paralleler Korrektur

On-Line: Deaktivierung Zyklischer Tabellen

Das Ende einer zyklischen Abarbeitung einer Tabelle wird durch den Benutzer per SPS angefordert, und findet am Ende der jeweiligen Zyklischen Tabelle statt.

On-Line: Aktivierung von Korrekturtabellen

Es gibt zwei Möglichkeiten der Aktivierung von Korrekturtabellen:

- Instantane Aktivierung: Die Korrekturtabellen werden durch den Benutzer skaliert und aktiviert und nach Aufruf durch den Benutzer instantan in der NC aktiviert.
- Aktivierung bei Unter/Überschreiten einer Aktivierungsposition: Die Korrekturtabellen werden durch den Benutzer skaliert und vor Unter/Überschreiten der Aktivierungsposition mit Angabe der Aktivierungsposition logisch aktiviert. Bei Unter/Überschreiten der Aktivierungsposition wird die Korrektur automatisch in der NC aktiviert.

Die Korrekturtabellen werden parallel zu den Profiltabellen abgefahren. Die Korrekturtabellen deaktivieren sich automatisch nach Abfahren der Korrektur.

Bemerkung zur **Skalierung von Korrekturtabellen**:

- RangeLow = *-DBL_MAX* wird automatisch durch die NC-gesetzt…
- RangeHigh = *DBK_MAX* wird automatisch durch die NC-gesetzt…
- Tabellengewicht wird vom Benutzer gesetzt (üblicherweise *1.0*).
- Der Master-Offset wird automatisch durch die NC-gesetzt…

- Master-Steigung, Slave-Offset und Slave-Steigung werden vom Benutzer gesetzt.

Errorhandling

Es gibt **kein explizites Errorhandling**, wenn ein Tabellenüberlauf/Tabellenunterlauf linearer Tabellen passiert. Der Slave-Achsen-Positionswert bleibt am jeweiligen Tabellenrand instantan stehen und erzeugt (je nach dynamischem Zustand und Tabellenzusammensetzung) einen Schleppabstand.

Stop/Abkoppeln

Die Slave-Achse stoppt entweder, wenn der Master stoppt (Streckensteuerung) oder wenn der/die Tabelleninhalt(e) es bedingt/bedingen (die Slaveposition ist über einen weiten Bereich der Masterposition(en) konstant).

Das Abkoppeln kann off-line (Master und Slave stehen) oder semi-off-line (nur der Slave steht) durchgeführt werden.

Interfaces

- **Tabelle Lesen**
  - 1. SPS-Bausteine
- **Skalierung setzen**
  - 1. SPS-Bausteine
- **Ankoppeln**
  - 1. System Manager
  - 2. SPS-Bausteine
- **Abkoppeln**
  - 1. System Manager
  - 2. SPS-Bausteine

5 TcNcCamming

In vielen Anwendungen ist es notwendig, zwei oder mehr Achsen miteinander zu synchronisieren. In der TwinCAT NC PTP können Achsen aneinandergekoppelt werden. Eine Master-Achse wird dann aktiv verfahren und ein oder mehrere gekoppelte Slave-Achsen werden durch die NC synchron mitpositioniert.

Die einfachste Kopplungsart ist die Linearkopplung mit einem festen Übersetzungsverhältnis (elektronisches Getriebe).

Für manche Anwendungen ist eine komplexere, mathematisch nicht durch eine Formel beschreibbare Kopplung von Master und Slave notwendig. Diese Abhängigkeit kann durch eine Tabelle beschrieben werden, in der zu jeder Master-Position eine zugehörige Slave-Position festgehalten wird.

Die TwinCAT NC PTP bietet die Möglichkeit, eine Slave-Achse über eine Tabelle an eine Master-Achse zu koppeln (elektronische Kurvenscheibe). Dabei enthält die Tabelle eine Anzahl von vorgegebenen Stützstellen, zwischen denen die NC Position und Geschwindigkeit interpoliert.

Die Bibliothek TcNcCamming enthält Funtionsbausteine für den Umgang mit Kurvenscheiben. Es werden zwei Arten von Kurvenscheiben unterstützt.

Zum einen kann eine Kurvenscheibe eine zweispaltige Tabelle von Master- und Slave-Positionen sein (Standardtabelle). Die Masterspalte definiert Stützstellen über den Verfahrweg des Masters von einem kleinsten Positionswert aufsteigend bis zu einem größten Wert. Mit den Stützstellen der Tabelle wird aus der zweiten Spalte die zugehörige Slave-Position ermittelt. Dabei wird zwischen den Stützstellen interpoliert.

Zum anderen kann eine Kurvenscheibe als so genannte Motion Function definiert werden. Eine Motion Function ist eine einspaltige Tabelle von Stützstellen. Jede Stützstelle enthält aber nicht einfach nur eine Position, sondern eine vollständige Beschreibung des Kurvenverlaufs in einem Abschnitt (Segment) der Kurvenscheibe. Neben der Master- und Slave-Position am Anfang des Segmentes wird beispielsweise der Funktionsverlauf bis zur nächsten Stützstelle als mathematische Funktion festgelegt. Eine Motion Function benötigt dadurch nur sehr wenige Stützstellen. Trotzdem ist jeder Punkt zwischen den Stützstellen durch die mathematische Funktion exakt definiert und es gibt keine Interpolationsungenauigkeiten.

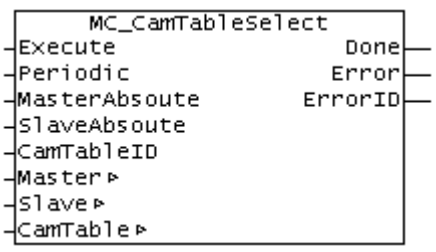
Im Gegensatz zu einer Standardtabelle können die Punkte einer Motion Function auch zur Laufzeit manipuliert werden. Dabei achtet das System darauf, das eine Manipulation erst wirkt, wenn die Änderung keinen direkten Einfluss auf den Slave hat. Positionssprünge werden so vermieden.

Mit der Erweiterung der Kurvenscheibenfunktionalität in Richtung Motion Functions wurden PLCopen konforme Funktionsbausteine eingeführt. Es wird empfohlen, diese Bausteine mit dem Namen *MC...* an Stelle der älteren Bausteine mit dem Namen *CamTable...* zu verwenden. Die Funktion ist aber auch bei den älteren Bausteinen nach wie vor gegeben.

Zur einfachen Handhabung der Achskopplung über Tabellen dient die TwinCAT PLC Library **TcNcCamming.lib**, die als zusätzliches Produkt erhältlich ist. Beispielprogramme zum Thema Kurvenscheiben und Motion Functions verwenden diese Bibliothek.

5.1 PLCopen Bausteine und Motion Functions

5.1.1 MC_CamTableSelect



Mit dem Funktionsbaustein MC_CamTableSelect kann eine Tabelle spezifiziert und in die NC geladen werden. Der Baustein legt eine neue Tabelle an und füllt sie gleichzeitig mit Daten, die von der SPS bereitgestellt werden.

MC_CamTableSelect muss nicht benutzt werden, wenn eine mit dem TwinCAT Kurvenscheibeneditor erstellte Tabelle benutzt werden soll. In diesem Fall reicht das einfache Ankoppeln mit MC_CamIn.

VAR_INPUT

```
VAR_INPUT
    Execute :       BOOL;
    Periodic :      BOOL;
    MasterAbsoute : BOOL;
    SlaveAbsoute :  BOOL;
    CamTableID :    MC_CAM_ID;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Periodic : TRUE = periodische Ausführung, FALSE = keine periodische Ausführung

MasterAbsolute : TRUE = Master absolut, FALSE = Master relativ

SlaveAbsolute : TRUE = Slave absolut, FALSE = Slave relativ.

CamTableID : ID der geladenen Tabelle. (Siehe: MC_CAM_ID [▶ 55])

VAR_OUTPUT

```
VAR_OUTPUT
    Done :      BOOL;
    Error :         BOOL;
    ErrorID :       UDINT;
END_VAR
```

Done : Wird TRUE, wenn Funktion erfolgreich ausgeführt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    Master :    NCTOPLC_AXLESTRUCT;
    Slave :     NCTOPLC_AXLESTRUCT;
    CamTable :  MC_CAM_REF;
ND_VAR
```

Master : Achsstruktur des Masters.

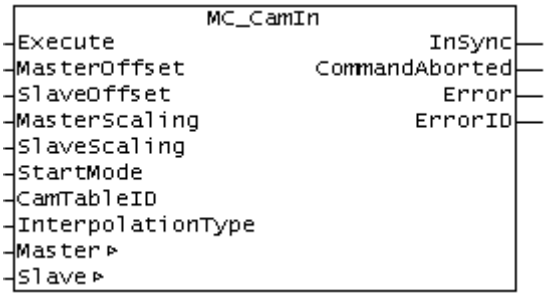
Slave : Achsstruktur des Slaves.

CamTable : Verweis auf die Tabelle (Struktur: MC_CAM_REF [▶ 55]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.2 MC_CamIn



Mit dem Funktionsbaustein MC_CamIn wird eine Master-Slave-Kopplung mit einer bestimmten Kurvenscheibe aktiviert.

Alternativ steht der erweiterte Funktionsbaustein MC_CamInExt [▶ 39] zur Verfügung, mit dem auch eine Kurvenscheibe gewechselt werden kann.

Wichtig: Nähere Erläuterungen zum Koppeln mit Kurvenscheiben [▶ 41]

VAR_INPUT

```
VAR_INPUT
    Execute :          BOOL;
    MasterOffset :     LREAL;
    SlaveOffset :      LREAL;
    MasterScaling :    LREAL;
    SlaveScaling :     LREAL;
    StartMode :        MC_StartMode;
    CamTableID :       MC_CAM_ID;
    InterpolationType: MC_InterpolationType;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

MasterOffset : Offset für die Masterpositionen.

SlaveOffset : Offset für die Slavepositionen.

MasterScaling : Faktor für Masterpositionen (default 1.0).

SlaveScaling : Faktor für Slavepositionen (default 1.0).

StartMode : Legt fest [▶ 56], ob die verwendete Kurvenscheibe *absolut* oder *relativ* zur aktuellen Masterposition verwendet werden soll. Gültige Werte sind START_ABSOLUTE und START_RELATIVE.

CamTableID : Tabellen ID [▶ 55].

InterpolationType : Interpolationsart der Tabellendaten. Nicht notwendig für Motionfunctions.

VAR_OUTPUT

```
VAR_OUTPUT
    InSync :          BOOL;
    CommandAborted :  BOOL;
    Error :           BOOL;
    ErrorID :         UDINT;
END_VAR
```

InSync : Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.

CommandAborted : Wird TRUE, wenn die Kopplung nicht durchgeführt werden konnte.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    Master :    NCTOPLC_AXLESTRUCT;
    Slave :     NCTOPLC_AXLESTRUCT;
END_VAR
```

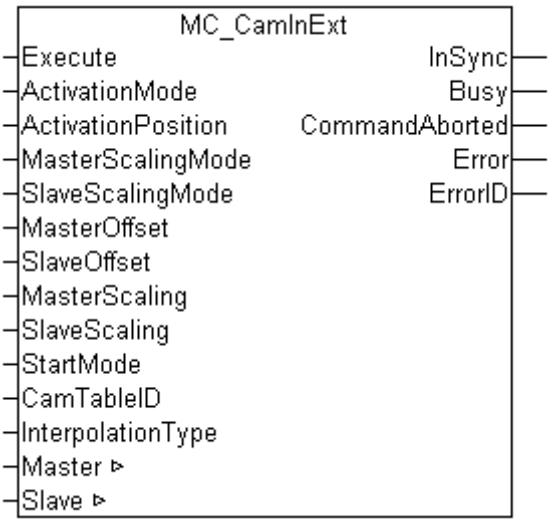
Master : Achsstruktur des Masters.

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.3 MC_CamInExt



Mit dem Funktionsbaustein *MC_CamInExt* wird eine Master-Slave-Kopplung mit einer bestimmten Kurvenscheibe aktiviert. Im Gegensatz zum Standardbaustein MC_CamIn [▶ 38] ist es mit *MC_CamInExt* möglich, im gekoppelten Zustand auf eine neue Kurvenscheibe umzuschalten. Dabei können die Regeln für die Umschaltung, insbesondere der genaue Zeitpunkt oder die Position bestimmt werden.

Um die erweiterten Möglichkeiten des *MC_CamInExt* zu nutzen muss er schon beim ersten Ankoppeln verwendet werden. Wenn eine Achse mit *MC_CamIn* gekoppelt wurde, ist es später nicht möglich, mit *MC_CamInExt* auf eine andere Kurvenscheibe umzuschalten.

Wichtig:

Nähere Erläuterungen zum Koppeln mit Kurvenscheiben [▶ 41]

ActivationMode [▶ 61] (Ankoppeln oder Umschalten von Kurvenscheiben)

StartMode [▶ 56]

ScalingMode [▶ 64]

VAR_INPUT

```
VAR_INPUT
    Execute             : BOOL;
    ActivationMode      : MC_CamActivationMode;
    ActivationPosition  : LREAL;
    MasterScalingMode   : MC_CamScalingMode;
    SlaveScalingMode    : MC_CamScalingMode;
    MasterOffset        : LREAL;
```

```
    SlaveOffset         : LREAL;
    MasterScaling       : LREAL := 1.0;
    SlaveScaling        : LREAL := 1.0;
    StartMode           : MC_StartMode := START_ABSOLUTE;
    CamTableID          : MC_CAM_ID;
    InterpolationType   : MC_InterpolationType := MC_InterpolationType_Linear;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

ActivationMode : Mit dem ActivationMode [▶ 61] wird der Zeitpunkt bzw. die Position festgelegt, an der die Umschaltung stattfinden soll. (nur bei bereits aktiver Kopplung)

ActivationPosition : Aktivierungsposition, die für positionsbasierte Aktivierungs-Modi notwendig ist.

MasterScalingMode : Legt die Art der Skalierung der Kurvenscheibe für die Master-Skalierung fest (MC_CamScalingMode [▶ 64]).

SlaveScalingMode : Legt die Art der Skalierung der Kurvenscheibe für die Slave-Skalierung fest (MC_CamScalingMode [▶ 64]).

MasterOffset : Offset für die Master-Positionen.

SlaveOffset : Offset für die Slave-Positionen.

MasterScaling : Faktor [▶ 64] für Master-Positionen (default 1.0).

SlaveScaling : Faktor für Slave-Positionen (default 1.0).

StartMode : StartMode [▶ 56] bestimmt, ob die Kurvenscheibenpositionen absolut oder relativ zur Koppelposition interpretiert werden.
StartMode kann für Master (X-Koordinate) und Slave (Y-Koordinate) jeweils relativ oder absolut sein.

CamTableID : ID der Kurvenscheibe mit der gekoppelt wird .

InterpolationType : Interpolationsart der Kurvenscheibendaten [▶ 57].

VAR_OUTPUT

```
VAR_OUTPUT
    InSync      : BOOL;
    Busy        : BOOL;
    CommandAborted  : BOOL;
    Error       : BOOL;
    ErrorID         : UDINT;
END_VAR
```

InSync : Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.

Busy : wird TRUE, sobald der Funktionsbaustein aktiv wird und wird FALSE sobald die Funktion beendet wurde und neu getriggert werden kann.

CommandAborted : Wird TRUE, wenn die Kopplung nicht durchgeführt werden konnte.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    Master  : NCTOPLC_AXLESTRUCT;
    Slave   : NCTOPLC_AXLESTRUCT;
END_VAR
```

Master : Achsstruktur des Masters.

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1251	PC (i386)	TcNcCamming.Lib

5.1.4 Achskopplung mit Kurvenscheiben

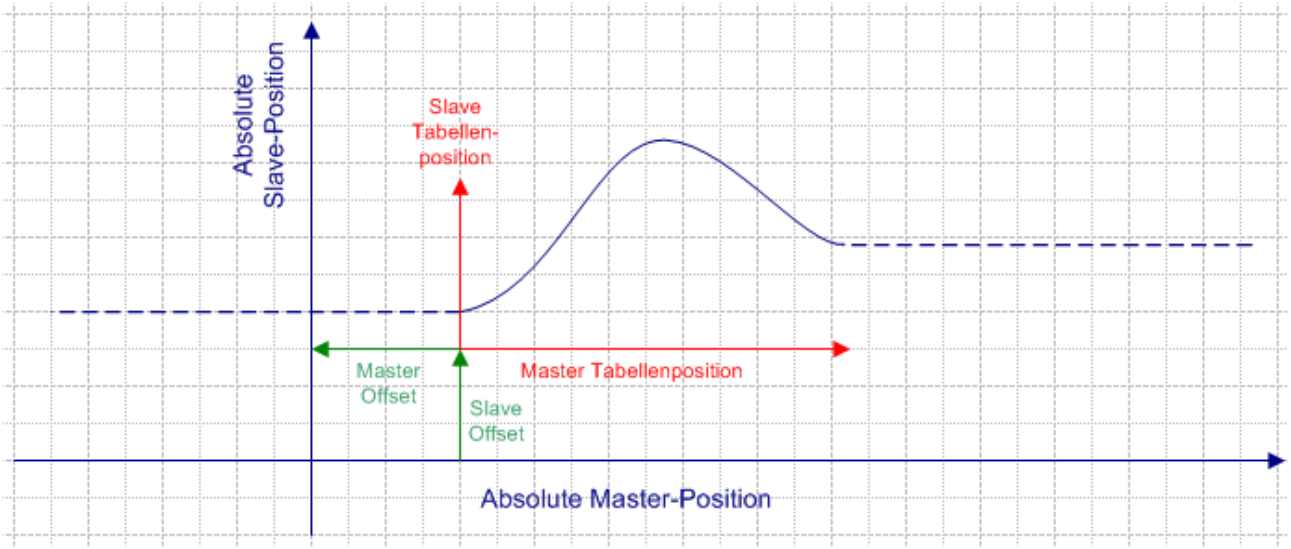
Mit dem Funktionsbaustein MC_CamIn [▶ 38] kann eine Kurvenscheibenkopplung (auch Tabellenkopplung) zwischen einer Master- und einer Slave-Achse hergestellt werden. Dabei ist zu beachten, dass die Slave-Achse bereits vor der Kopplung auf einer durch die Kurvenscheibe definierten Position stehen muss. Nach dem Koppeln und Starten des Masters wird die Slave-Position direkt aus der Kurvenscheibe berechnet. Die Slave-Achse wird also nicht langsam mit der Kurvenscheibe synchronisiert sondern springt, wenn sie nicht schon an der Tabellenposition steht.

In der Praxis stellt sich die Frage, welche Position der Slave vor der Kopplung einnehmen muss und wie diese berechnet wird. Die folgenden Abbildungen veranschaulichen, wie hier vorgegangen werden kann.

- Alle folgenden Berechnungen werden ausschließlich mit Sollpositionen der Achsen durchgeführt. Die Ist-Positionen gehen in die Berechnungen nicht ein und führen gerade bei zyklischen Kurvenscheiben zu Berechnungsfehlern, wenn sie dennoch verwendet werden.
- Ferner werden hier nur absolute Tabellenkopplungen betrachtet. Bei relativen Kopplungen geht die Koppelposition der Master- bzw. Slave-Achse als zusätzlicher Offset mit in die Berechnungen ein.

Lineare Kurvenscheiben

Eine lineare Kurvenscheibe ist nur über einen begrenzten Master-Positionsbereich definiert. Außerhalb dieses Bereiches ist die Slave-Position durch die erste bzw. letzte Position in der Tabelle definiert. Der Slave bleibt also an den Tabellenrändern stehen sobald der Master aus dem definierten Bereich herausfährt.



Die Abbildung zeigt, dass das absolute Achskoordinatensystem (blau) nicht mit dem Kurvenscheibenkoordinatensystem (rot) übereinstimmen muss. Das Koordinatensystem der Kurvenscheibe kann durch einen Master-Offset und einen Slave-Offset verschoben sein und eine Skalierung ist ebenfalls möglich.

Die zu einer bestimmten Master-Position gehörende Slave-Position kann durch den Funktionsbaustein MC_ReadCamTableSlaveDynamics [▶ 46] bestimmt werden. Der Baustein bezieht sich auf die Rohdaten der Tabelle, so dass Offsets und Skalierungen durch das SPS-Programm selbst berücksichtigt werden müssen. Zunächst wird der Master-Offset zur aktuellen Masterposition addiert und falls die Kurvenscheibe skaliert werden soll, wird durch diese Skalierung dividiert.

```
MasterTablePos := (MasterPosition + MasterOffset) / MasterScale;
```

Die Master-Tabellenposition ist Eingangsparameter für den Funktionsbaustein MC_ReadCamTableSlaveDynamics [▶ 46]. Das Ergebnis wird gegebenenfalls mit Slave-Offset und -Skalierung auf eine absolute Slave-Position umgerechnet.

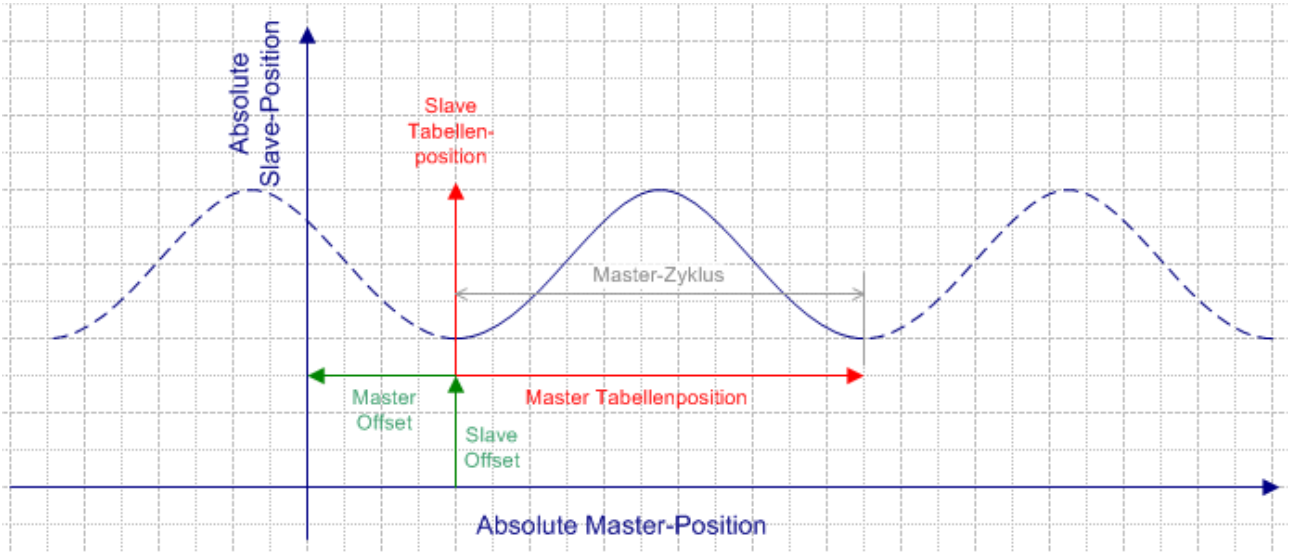
```
SlaveTablePosition := ReadSlaveDynamics.SlavePosition;

SlavePosition := (SlaveTablePosition * SlaveScale) + SlaveOffset;
```

Der Slave wird vor der Kopplung an diese Position gefahren. Alternativ kann auch der Master zu einer Position gefahren werden, die mit der aktuellen Slave-Position korrespondiert. Es ist aber nicht allgemein möglich, diese Position aus der Kurvenscheibe zu ermitteln, da die Kurvenscheibe bei dieser Betrachtung mehrdeutig sein kann.

Da der Master-Offset additiv in die erste Formel eingeht, führt ein positiver Offset zu einer Verschiebung des Kurvenscheibenkoordinatensystems nach links in negative Richtung. Der Master-Offset in der Abbildung ist demnach negativ. Ein positiver Slave-Offset führt zu einer Verschiebung des Kurvenscheibenkoordinatensystems in positive Richtung nach oben.

Zyklische Kurvenscheiben ohne Hub

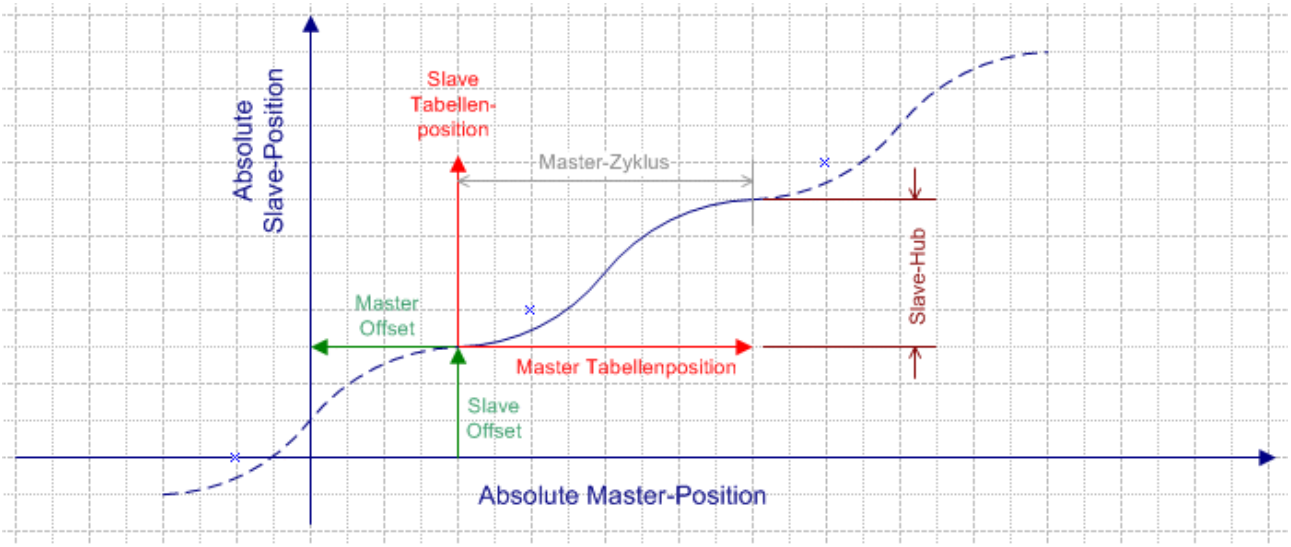
Eine zyklische Kurvenscheibe ohne Hub zeichnet sich dadurch aus, dass die Slave-Anfangs- und die Endposition in der Tabelle identisch sind. Der Slave bewegt sich dadurch zyklisch in einem definierten Bereich, ohne seine Position stetig in eine Richtung zu verändern.



Die Master-Slave-Kopplung erfordert bei diesem Kurvenscheibentypen dieselbe Vorbereitung wie bei einer linearen Kurvenscheibe. Die Ausgangsposition des Slaves kann also wie oben berechnet werden. Es ist nicht notwendig, die Modulo-Position des Masters zur Berechnung heranzuziehen, da die absolute Position bereits durch das Koppelkommando korrekt berücksichtigt wird.

Zyklische Kurvenscheiben mit Hub

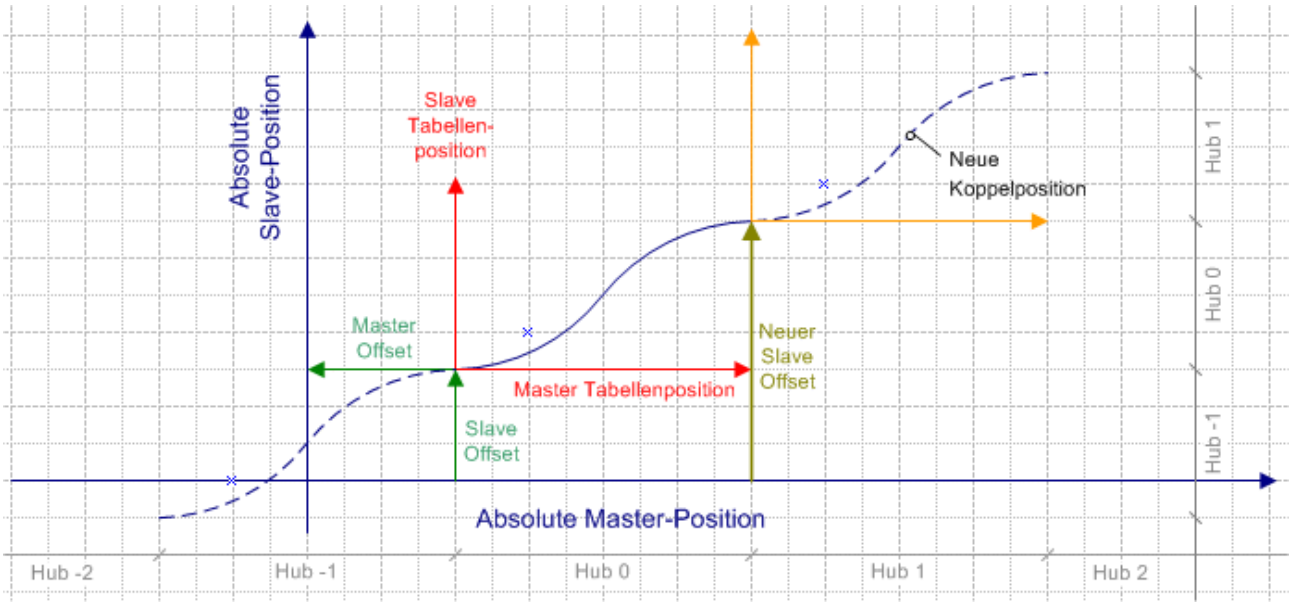
Der Hub einer zyklischen Kurvenscheibe ist die Differenz zwischen der letzten und der ersten Tabellenposition des Slaves.



Eine solche Kurvenscheibe wird am Tabellenende zyklisch fortgesetzt. Dabei springt die Slave-Position nicht zurück auf den Tabellen-Anfangswert, sondern die Bewegung wird kontinuierlich fortgesetzt. Mit jedem neuen Zyklus wird also der Hub als zusätzlicher interner Slave-Offset aufaddiert bzw. bei Bewegungsumkehr subtrahiert.

Abkoppeln und Wiederankoppeln bei zyklischen Kurvenscheiben mit Hub

Wenn ein Slave mit einer Kurvenscheibe mit Hub gekoppelt wird, so wird immer im Grundzyklus (rotes Koordinatensystem) angekoppelt. Wird der Slave nach einigen Zyklen abgekoppelt, und dann erneut angekoppelt, fällt die Position des Slaves in den Grundzyklus zurück. Dieses Verhalten ist gegebenenfalls durch eine Nebenrechnung des Slave-Offsets zu berücksichtigen und auszugleichen.




```
MasterTablePos := (MasterPosition + MasterOffset) / MasterScale;
```

Die Master-Tabellenposition ist Eingangsparameter für den Funktionsbaustein MC_ReadCamTableSlaveDynamics [▶ 46]. Das Ergebnis wird gegebenenfalls mit Slave-Offset und -Skalierung auf eine absolute Slave-Position umgerechnet. Zusätzlich muss die Anzahl der bereits aufgelaufenen Hübe berechnet und zur Slave-Position addiert werden.

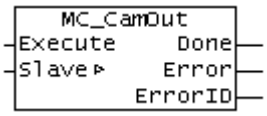
```
SlaveTablePosition := ReadSlaveDynamics.SlavePosition;
```

```
Hubanzahl := MODTURNS( (SlavePosition - SlaveOffset), SlaveHub );
```
(Siehe MODTURNS-Dokumentation)

```
NewSlaveOffset := SlaveOffset + (SlaveHub * Hubanzahl);
```

```
SlavePosition := (SlaveTablePosition * SlaveScale) + NewSlaveOffset;
```

5.1.5 MC_CamOut



Mit dem Funktionsbaustein MC_CamOut wird eine Master-Slave-Kopplung deaktiviert.

HINWEIS

Wenn der Sollwertgeneratortype der Achse auf "7 Phasen (optimiert)" eingestellt ist, wird die Slaveachse nach dem Abkoppeln beschleunigungsfrei gefahren und mit der sich einstellenden konstanten Geschwindigkeit weitergefahren. Es erfolgt keine Positionierung um den mit dem Koppelfaktor umgerechneten Masterverfahrweg, sondern es stellt sich ein Verhalten wie nach einem MC_MoveVelocity ein.

In TwinCAT 2.10 ist der Sollwertgeneratortyp wählbar.

Ab TwinCAT 2.11 ist der Sollwertgeneratortype fest auf "7 Phasen (optimiert)" eingestellt.

Bei der Umstellung eines Projektes von TwinCAT 2.10 auf TwinCAT 2.11 ergibt sich damit das hier beschriebene Verhalten.

Ein Update bestehender Applikationen auf Version 2.11 kann daher, je nach Anwendung, eine Anpassung des SPS-Programms erforderlich machen.

VAR_INPUT

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done :      BOOL;
    Error :     BOOL;
    ErrorID :   UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Entkopplung erfolgreich durchgeführt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

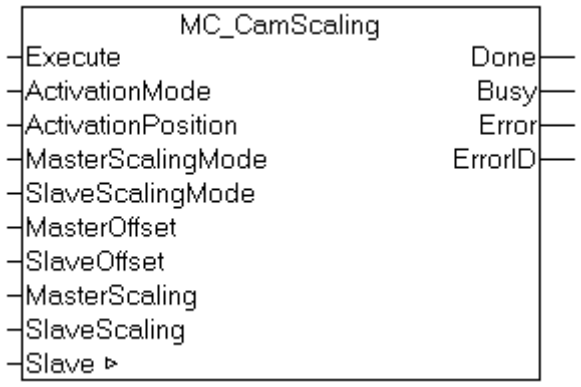
```
VAR_IN_OUT
    Slave : NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.6 MC_CamScaling



Mit dem Funktionsbaustein MC_CamScaling kann eine Kurvenscheibenkopplung skaliert werden. Dabei werden nicht die Tabellenrohdaten der Kurvenscheibe beeinflusst, sondern die Skalierung bezieht sich auf eine bestehende Master-Slave-Kopplung. Einstellbar sind die Skalierungsfaktoren für Master und Slave und die Offsets zur Verschiebung der Kurvenscheibe im Koordinatensystem.

Optional wirkt die Änderung erst ab einer bestimmten Master-Position wodurch die Skalierung punktgenau während der Fahrt geändert werden kann. Bei der Skalierung während der Fahrt ist Vorsicht geboten. Die Slave-Position zum Zeitpunkt der Skalierung darf durch die Änderung nur in geringem Maße beeinflusst werden.

VAR_INPUT

```
VAR_INPUT
    Execute            : BOOL;
    ActivationMode     : MC_CamActivationMode;
    ActivationPosition : LREAL;
    MasterScalingMode: : MC_CamScalingMode;
    SlaveScalingMode   : MC_CamScalingMode;
    MasterOffset       : LREAL;
    SlaveOffset        : LREAL;
    MasterScaling      : LREAL := 1.0;
    SlaveScaling       : LREAL := 1.0;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

ActivationMode : Definiert, wann und wie die Skalierung durchgeführt wird. (MC_CamActivationMode [▶ 61])

ActivationPosition : Optionale Master-Position, an der die Skalierung durchgeführt wird (je nach ActivationMode).

MasterScalingMode : Art der der Master-Skalierung. (MC_CamScalingMode [▶ 64])

SlaveScalingMode : Art der der Slave-Skalierung. (MC_CamScalingMode [▶ 64])

MasterOffset : Offset für die Masterpositionen.

SlaveOffset : Offset für die Slavepositionen.

MasterScaling : Faktor für Masterpositionen (default 1.0).

SlaveScaling : Faktor für Slavepositionen (default 1.0).

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Skalierung erfolgreich durchgeführt wurde.

Busy : wird TRUE, sobald der Funktionsbaustein aktiv wird und wird FALSE sobald die Funktion beendet wurde und neu getriggert werden kann.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

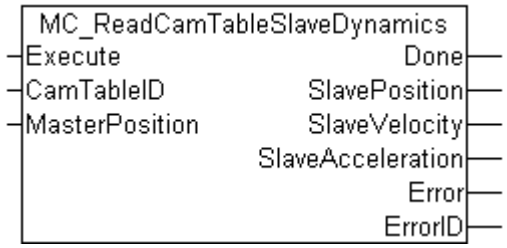
```
VAR_IN_OUT
    Slave :     NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 Build 1001	PC (i386)	TcNcCamming.Lib

5.1.7 MC_ReadCamTableSlaveDynamics



Mit dem Funktionsbaustein MC_ReadCamTableSlaveDynamics kann die Slave-Dynamik an einem bestimmten Punkt einer Kurvenscheibentabelle bestimmt werden. Die Funktion wertet die Tabellenrohdaten aus, eine eventuelle Skalierung der Kurvenscheibe bleibt unberücksichtigt.

Bei älteren Kurvenscheibentabellentypen [▶ 56] können nicht alle Dynamikparameter ermittelt werden. Die folgende Übersicht zeigt, welches Ergebnis zu erwarten ist:

MC_TableType_TABULARTYPE_MOTIONFUNC : Slave-Position, Geschwindigkeit und Beschleunigung werden ermittelt.

MC_TableType_TABULARTYPE_EQUIDIST : Slave-Position und Geschwindigkeit werden ermittelt. Die Beschleunigung ist immer 0.

MC_TableType_TABULARTYPE_NONEQUIDIST : Slave-Position wird ermittelt. Geschwindigkeit und Beschleunigung sind immer 0.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    CamTableID      : MC_CAM_ID;
    MasterPosition  : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : Tabellen ID [▶ 55].

MasterPosition : Master-Position innerhalb der Tabelle zu der die Slave-Dynamik ermittelt werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    SlavePosition    : LREAL;
    SlaveVelocity    : LREAL;
    SlaveAcceleration       : LREAL;
    Error       : BOOL;
    ErrorID          : UDINT;
END_VAR
```

Done : Wird TRUE, wenn das Kommando erfolgreich ausgeführt wurde.

SlavePosition : Position des Slaves in der Kurvenscheibentabelle an der angegebenen MasterPosition.

SlaveVelocity : Geschwindigkeit des Slaves in der Kurvenscheibentabelle an der angegebenen MasterPosition.

SlaveAcceleration : Beschleunigung des Slaves in der Kurvenscheibentabelle an der angegebenen MasterPosition.

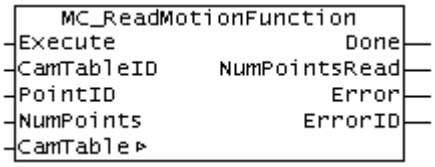
Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.8 MC_ReadMotionFunction



Mit dem Funktionsbaustein MC_ReadMotionFunction können die Daten einer Motion Function gelesen werden. Dabei kann die gesamte Funktion mit allen Stützstellen oder auch nur ein Teil gelesen werden. Die Daten werden in der durch CamTable beschriebenen Struktur in der SPS abgelegt.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    CamTableID      : MC_CAM_ID;
    PointID         : MC_MotionFunctionPoint_ID;
    NumPoints       : UDINT; (* 0 = fill MFsize *)
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : ID der geladenen Tabelle [▶ 55].

PointID : Punkt-ID [▶ 58] des ersten zu lesenden Punktes der Motion Function.

NumPoints : Anzahl der zu lesenden Punkte der Motion Function. Um alle Punkte zu lesen kann hier der Wert 0 angegeben werden, die tatsächlich gelesene Anzahl wird in der Ausgangsvariablen *NumPointsRead* zurückgeliefert.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    NumPointsRead    : UDINT; (* return value <= NumPoints *)
```

```
    Error       : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Daten erfolgreich gelesen wurden.

NumPointsRead : Die Anzahl der tatsächlich gelesenen Punkte. Die Anzahl kann kleiner oder gleich *NumPoints* sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    CamTable :      MC_CAM_REF;
ND_VAR
```

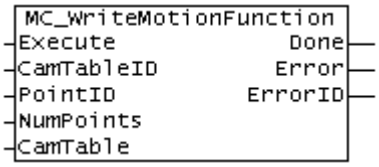
MC_CAM_REF [▶ 55]

CamTable : Verweis auf die Tabelle (Struktur).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.9 MC_WriteMotionFunction



Mit dem Funktionsbaustein *MC_WriteMotionFunction* können die Daten einer Motion Function in die NC geschrieben werden. Dabei kann die gesamte Funktion mit allen Stützstellen oder auch nur ein Teil geschrieben werden. Die Daten werden zuvor in der durch *CamTable* beschriebenen Struktur in der SPS abgelegt.

Mit dem Funktionsbaustein MC_SetCamOnlineChangeMode [▶ 51] kann festgelegt werden, wann die Daten in die Kurvenscheibe übernommen werden. Sollen die Daten nicht sofort, sondern beispielsweise erst an einer bestimmten Position des Masters aktiv werden, so puffert das System zunächst die geschriebenen Daten, um sie dann an der Masterposition zu aktivieren. Mit der Funktion AxisCamDataQueued kann geprüft werden, ob Daten gepuffert sind, das heißt geschrieben aber noch nicht aktiviert wurden.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    CamTableID      : MC_CAM_ID;
    PointID         : MC_MotionFunctionPoint_ID;
    NumPoints       : UDINT;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : ID der geladenen Tabelle [▶ 55].

PointID : Punkt-ID [▶ 58] des ersten zu schreibenden Punktes der Motion Function.

NumPoints : Anzahl der zu schreibenden Punkte der Motion Function.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Daten erfolgreich gelesen wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

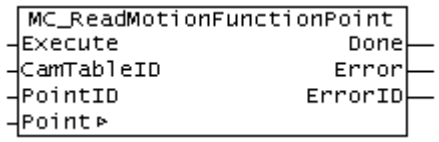
```
VAR_IN_OUT
    CamTable :      MC_CAM_REF;
ND_VAR
```

CamTable : Verweis auf die Tabelle [▶ 55] (Struktur). Die Startadresse der Tabellen-Datenstruktur (CamTable.pArray) zeigt auf den ersten Punkt, der geschrieben werden soll.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.10 MC_ReadMotionFunctionPoint



Mit dem Funktionsbaustein MC_ReadMotionFunctionPoint können die Daten einer Stützstelle einer Motion Function gelesen werden.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    CamTableID      : MC_CAM_ID;
    PointID         : MC_MotionFunctionPoint_ID;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : ID der geladenen Tabelle [▶ 55].

PointID : Punkt-ID [▶ 58] des ersten zu lesenden Punktes der Motion Function.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Error       : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Daten erfolgreich gelesen wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

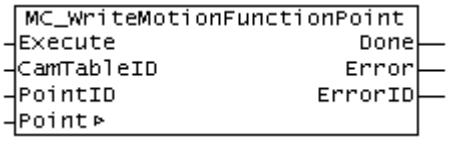
```
VAR_IN_OUT
    Point : MC_MotionFunctionPoint;
ND_VAR
```

Point : Datenstruktur [▶ 57] mit den Daten einer Stützstelle einer Motion Function

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.11 MC_WriteMotionFunctionPoint



Mit dem Funktionsbaustein MC_WriteMotionFunctionPoint können die Daten einer Stützstelle einer Motion Function beschrieben werden.

Mit dem Funktionsbaustein MC_SetCamOnlineChangeMode [▶ 51] kann festgelegt werden, wann die Daten in die Kurvenscheibe übernommen werden. Sollen die Daten nicht sofort, sondern beispielsweise erst an einer bestimmten Position des Masters aktiv werden, so puffert das System zunächst die geschriebenen Daten, um sie dann an der Masterposition zu aktivieren. Mit der Funktion AxisCamDataQueued kann geprüft werden, ob Daten gepuffert sind, das heißt geschrieben aber noch nicht aktiviert wurden.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    CamTableID      : MC_CAM_ID;
    PointID         : MC_MotionFunctionPoint_ID;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : ID der geladenen Tabelle [▶ 55].

PointID : Punkt-ID [▶ 58] des ersten zu schreibenden Punktes der Motion Function.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Daten erfolgreich geschrieben wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

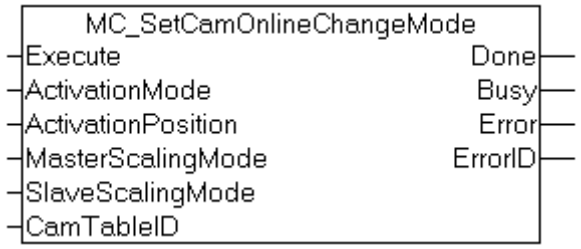
```
VAR_IN_OUT
    Point : MC_MotionFunctionPoint;
ND_VAR
```

Point : Datenstruktur [▶ 57] mit den Daten einer Stützstelle einer Motion Function

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.12 MC_SetCamOnlineChangeMode



Kurvenscheibendaten können während der Laufzeit durch die SPS geändert werden (siehe MC_WriteMotionFunction [▶ 48], MC_WriteMotionFunctionPoint [▶ 50]). Mit dem Funktionsbaustein MC_SetCamOnlineChangeMode wird festgelegt, wann und wie diese Änderungen übernommen werden.

Diese Funktion legt den Aktivierungsmodus für Änderungen fest, führt aber keine Änderung oder Umschaltung von Kurvenscheiben aus.

VAR_INPUT

```
VAR_INPUT
    Execute             : BOOL;
    ActivationMode      : MC_CamActivationMode;
    ActivationPosition  : LREAL;
    MasterScalingMode   : MC_CamScalingMode;
    SlaveScalingMode    : MC_CamScalingMode;
    CamTableID          : MC_CAM_ID;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

ActivationMode : Definiert, wann und wie die Skalierung durchgeführt wird. (MC_CamActivationMode [▶ 61])

ActivationPosition : Optionale Master-Position, an der die Skalierung durchgeführt wird (je nach ActivationMode).

MasterScalingMode : Art der der Master-Skalierung. (MC_CamScalingMode [▶ 64])

SlaveScalingMode : Art der der Slave-Skalierung. (MC_CamScalingMode [▶ 64])

CamTableID : Tabellen ID [▶ 55].

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Funktion erfolgreich durchgeführt wurde.

Busy : wird TRUE, sobald der Funktionsbaustein aktiv wird und wird FALSE sobald die Funktion beendet wurde und neu getriggert werden kann.

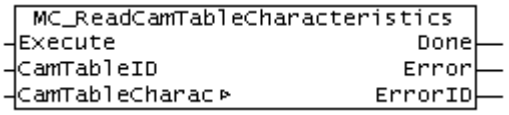
Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 Build 1001	PC (i386)	TcNcCamming.Lib

5.1.13 MC_ReadCamTableCharacteristics



Mit dem Funktionsbaustein MC_ReadCamTableCharacteristics werden die charakteristischen Kenngrößen einer Motion Function berechnet und ausgelesen.

VAR_INPUT

```
VAR_INPUT
    Execute :       BOOL;
    CamTableID :    MC_CAM_ID;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : Tabellen ID [▶ 55].

VAR_OUTPUT

```
VAR_OUTPUT
    Done :      BOOL;
    Error :     BOOL;
    ErrorID :   UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Berechnung erfolgreich durchgeführt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

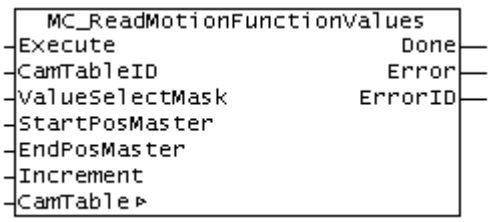
```
VAR_IN_OUT
    CamTableCharac : MC_TableCharacValues;
END_VAR
```

CamTableCharac: Datenstruktur [▶ 59] mit charakteristischen Kenngrößen der Motion Function

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 & TwinCAT v2.8	PC (i386)	TcNcCamming.Lib

5.1.14 MC_ReadMotionFunctionValues



Mit dem Funktionsbaustein MC_ReadMotionFunctionValues können die interpolierten Daten einer Motion Function in Form einer Tabelle gelesen werden.

VAR_INPUT

```
VAR_INPUT
    Execute          : BOOL;
    CamTableID       : MC_CAM_ID;
    ValueSelectMask  : UINT;  (* MC_ValueSelectType; position, velocity, acceleration, jerk… *)
    StartPosMaster   : LREAL; (* master position of first point *)
    EndPosMaster     : LREAL; (* master position of last point *)
    Increment        : LREAL; (* increment of master position *)
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

CamTableID : ID der geladenen Tabelle [▶ 55] (Typ Motion Function).

ValueSelectMask : Mit der Auswahlmaske [▶ 60] kann festgelegt werden, welche Daten interpoliert und zurückgeliefert werden soll. Die Spaltenzahl der durch CamTable beschriebenen Datenstruktur muss der Anzahl der durch ValueSelectMask bestimmten Spaltenzahl entsprechen. Wenn zum Beispiel nur Positionsdaten gelesen werden, ist die Spaltenzahl 2 (Master- und Slave-Position). Mit jeder weiteren Ableitung (Geschwindigkeit, Beschleunigung, Ruck) kommt eine Spalte hinzu.

StartPosMaster : Positionswert der Master-Achse des ersten interpolierten Punktes

EndPosMaster : Positionswert der Master-Achse des letzten interpolierten Punktes

Increment : Schrittweite der Interpolation

VAR_OUTPUT

```
VAR_OUTPUT
    Done     : BOOL;
    Error    : BOOL;
    ErrorID  : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Daten erfolgreich gelesen wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

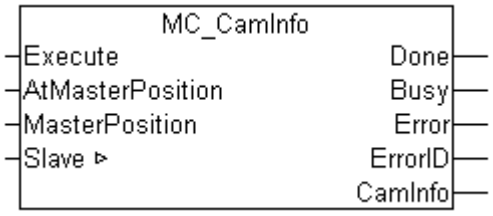
```
VAR_IN_OUT
    CamTable :      MC_CAM_REF;
ND_VAR
```

CamTable : Verweis auf die Tabelle [▶ 55] (Struktur).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.15 MC_CamInfo



Der Funktionsbaustein MC_CamInfo ermittelt Daten zum aktuellen Zustand und zur aktuellen Parametrierung einer Kurvenscheibenkopplung. Das Kommando setzt voraus, dass die Slave-Achse über eine Kurvenscheibe gekoppelt ist. Wenn der Eingang AtMasterPosition TRUE ist, wird nicht der aktuelle Zustand, sondern der Zustand bezogen auf die angegebene Master-Position ermittelt. Die ermittelten Daten werden in der Datenstruktur CamInfo abgelegt.

Wenn die gekoppelte Achsgruppe in eine Fehlersituation gerät (z. B. Notaus), so gibt der Funktionsbaustein den letzten gültigen Zustand der Kopplung zurück. Der Funktionsbaustein muss vor dem Abkoppeln des Slaves aufgerufen werden. Mit den ermittelten Daten kann dann die Kopplung an der ursprünglichen Achsposition wiederhergestellt werden.

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    AtMasterPosition  : BOOL;
    MasterPosition    : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

AtMasterPosition : Wenn *AtMasterPosition* TRUE ist, werden die Daten bezogen auf die angegebene *MasterPosition* ermitteln. Anderenfalls beziehen sich die Daten auf die aktuelle Masterposition.

MasterPosition : Master-Position auf die sich die ermittelten Daten beziehen. Diese Eingangsparameter ist nicht notwendig, wenn *AtMasterPosition* FALSE ist.

VAR_OUTPUT

```
VAR_OUTPUT
    Done     : BOOL;
    Busy     : BOOL;
    Error    : BOOL;
    ErrorID  : UDINT;
    CamInfo  : MC_CamInfoData;
END_VAR
```

Done : Wird TRUE, wenn die Funktion erfolgreich durchgeführt wurde.

Busy : wird TRUE, sobald der Funktionsbaustein aktiv wird und wird FALSE sobald die Funktion beendet wurde und neu getriggert werden kann.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

CamInfo : Die Datenstruktur CamInfo [▶ 66] enthält alle ermittelten Daten der Kurvenscheibenkopplung

VAR_IN_OUT

```
VAR_IN_OUT
    Slave   : NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1026	PC (i386)	TcNcCamming.Lib

5.1.16 Datentypen

5.1.16.1 MC_CAM_ID

```
TYPE
    MC_CAM_ID : UDINT;
END_TYPE
```

Typdeklaration für die Tabellen ID.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.2 MC_CAM_REF

```
TYPE MC_CAM_REF :
STRUCT
    pArray      : UDINT;     (* pointer to data *)
    nArraySize  : UDINT;     (* size of data in bytes
*)
    nTableType  : MC_TableType;  (* cam
table type *)
    nNoOfRows   : UDINT;
    nNoOfCols   : UDINT;
END_STRUCT
END_TYPE
```

Strukturdefinition der Tabelle. Der erste Parameter *pArray* ist ein Pointer auf eine Datenstruktur, die die Tabellendaten hält. Diese Datenstruktur kann abhängig vom Tabellentyp [▶ 56] *nTableType* unterschiedlich aufgebaut sein. In der Komponenten nNoOfRows wird die Anzahl der Zeilen eingetragen, in nNoOfCols die Anzahl der Spalten (normalerweise 1 oder 2).

Beispiel 1: Strukturbeschreibung einer Punktetabelle

pArray: Adresse eines zweidimensionalen Arrays. Die erste Spalte enthält eine aufsteigende Liste von Master-Positionen. Die zweite Spalte enthält die dazugehörigen Slave-Positionen.

nArraySize: Größe des Datenarrays. Die Größe sollte mit SIZEOF(Array) berechnet werden.

nTableType: Der Typ der Tabelle ist TABULARTYPE_EQUIDIST, wenn die Master-Positionen einen gleichen Abstand haben oder TABULARTYPE_NONEQUIDIST bei wechselndem Abstand.

nNoOfRows: Die Zeilenanzahl entspricht der Anzahl der Tabellenpunkte

nNoOfCols: Die Spaltenanzahl ist 2

Beispiel 2: Strukturbeschreibung einer Motion Function

pArray: Adresse eines eindimensionalen Arrays vom Typ *MC_MotionFunctionPoint*. Jedes Array-Element enthält eine Beschreibung einer Stützstelle der Kurvenscheibe.

nArraySize: Größe des Datenarrays. Die Größe sollte mit SIZEOF(Array) berechnet werden.

nTableType: Der Typ der Tabelle ist TABULARTYPE_MOTIONFUNC.

nNoOfRows: Die Zeilenanzahl entspricht der Anzahl der Tabellenpunkte

nNoOfCols: Die Spaltenanzahl ist 1

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.3 MC_StartMode

```
TYPE
 (
    MC_STARTMODE_ABSOLUTE := 1,      (* cam table is absolute for
master and slave *)
    MC_STARTMODE_RELATIVE,       (* cam table is relative for
master and slave *)
    MC_STARTMODE_MASTERABS_SLAVEREL, (* cam table is absolute for
master and relative for slave *)
    MC_STARTMODE_MASTERREL_SLAVEABS  (* cam table is relative for
master and absolute for slave *)
);
END_TYPE
```

Der *StartMode* wird beim Koppeln mit Kurvenscheiben durch MC_CamIn [▶ 38] oder MC_CamInExt [▶ 39] verwendet und legt fest ob eine Kurvenscheibe absolut zum Ursprung des Achskoordinatensystems oder relativ zur Koppelposition interpretiert wird. Der Mode kann für beide Koordinatenachsen getrennt absolut oder relativ festgelegt werden.

Beim *StartModeabsolut* liegt das Kurvenscheibenkoordinatensystem deckungsgleich über dem Achskoordinatensystem und kann gegebenenfalls durch einen Offset (Master- oder Slave-Offset) verschoben werden.

Beim *StartModerelativ* liegt der Ursprung des Kurvenscheibenkoordinatensystems an der Achsposition der jeweiligen Achse (Master oder Slave) zum Zeitpunkt der Kopplung oder Kurvenscheibenumschaltung. Zusätzlich kann die Kurvenscheibe durch einen Offset verschoben werden.

Die Modi MC_STARTMODE_RELATIVE und MC_STARTMODE_MASTERREL_SLAVEABS können nicht in Verbindung mit einer automatischen Master-Offsetberechnung (MC_CamScalingMode [▶ 64]) verwendet werden, da es hier einen Widerspruch gibt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1329	PC (i386)	TcNcCamming.Lib

5.1.16.4 MC_TableType

```
TYPE MC_TableType :
(
    (* n*m tabular with equidistant ascending master values
*)
    MC_TableType_TABULARTYPE_EQUIDIST := 10,
    (* n*m tabular with strictly monotone ascending master
values
    (not imperative equidistant) *)
    MC_TableType_TABULARTYPE_NONEQUIDIST := 11,
    (* motion function calculated in runtime *)
    MC_TableType_TABULARTYPE_MOTIONFUNC := 22 );
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.5 MC_InterpolationType

```
TYPE MC_InterpolationType :
(
    MC_InterpolationType_Linear := 0, (* linear 2 point
interpolation *)
    MC_InterpolationType_4Point := 1, (* 4 point
interpolation
                    (for equidistant tables
only) *)
    MC_InterpolationType_Spline := 2  (* spline interpolation
(tangential
                    or cyclic, depending on
table)*)
);
END_TYPE
```

Interpolationsart für Tabellendaten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.6 MC_MotionFunctionPoint

```
TYPE MC_MotionFunctionPoint :
STRUCT
    PointIndex    : MC_MotionFunctionPoint_ID;

    FunctionType      : MC_MotionFunctionType;

    PointType     : MC_MotionPointType;

    RelIndexNextPoint : MC_MotionFunctionPoint_ID;

    MasterPos     : LREAL; (* X *)
    SlavePos      : LREAL; (* Y *)
    SlaveVelo     : LREAL; (* Y' *)
    SlaveAcc      : LREAL; (* Y'' *)
    SlaveJerk     : LREAL; (* Y''' *)
END_STRUCT
END_TYPE
```

Die Datenstruktur *MC_MotionFunctionPoint* beschreibt eine Stützstelle einer Motion Function. Eine Motion Function ist eine eindimensionale Liste (Array) vom Typ *MC_MotionFunctionPoint*.

PointIndex: Absoluter Index [▶ 58] dieser Stützstelle innerhalb der Motion Function. Der Punktindex aller Stützstellen muss streng monoton steigend, lückenlos und größer 0 sein.

FunctionType: Typ MC_MotionFunctionType [▶ 59] der mathematischen Funktion zwischen dieser und der nachfolgenden Stützstelle

PointType: Typ MC_MotionPointType [▶ 58] dieser Stützstelle.

RelIndexNextPoint: Relativer Verweis auf die nachfolgende Stützstelle (normalerweise 1).

MasterPos: Position der Master-Achse an dieser Stützstelle

SlavePos: Position der Slave-Achse an dieser Stützstelle

SlaveVelo: Geschwindigkeit der Slave-Achse an dieser Stützstelle

SlaveAcc: Beschleunigung der Slave-Achse an dieser Stützstelle

SlaveJerk: Ruck der Slave-Achse an dieser Stützstelle

Einige Bewegungsgesetze (MC_MotionFunctionType) lassen eine Symmetrieeinstellung zu. Da bei all diesen Funktionen der Ruck an den Randpunkten 0 ist, kann in diesen Fällen der Ruckparameter SlaveJerk zur Einstellung der Symmetrie verwendet werden. Ein Wert von 0 bedeutet eine Symmetrie von 50%. Bei folgenden Funktionen kann der Wert im Bereich +/- 0.5 eingestellt werden:

POLYNOM3
POLYNOM5
POLYNOM8
POLYNOM11
SINUSLINIE
MODSINUSLINIE
BESTEHORN
BESCHLTRAPEZ
SINUS_GERADE_KOMBI

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.7 MC_MotionFunctionPoint_ID

```
TYPE
    MC_MotionFunctionPoint_ID : UDINT;
END_TYPE
```

Typdeklaration für die Punkt-ID der Punkte einer Motion Function.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.8 MC_MotionPointType

```
TYPE MC_MotionPointType :
(
    MOTIONPOINTTYPE_IGNORE,         (* Ignore point *)
    MOTIONPOINTTYPE_REST       := 16#0001, (* Restpoint - Rastpunkt
*)
    MOTIONPOINTTYPE_VELOCITY   := 16#0002, (* Velocity Point -
Geschwindigkeitspunkt *)
    MOTIONPOINTTYPE_TURN       := 16#0004, (* Turn Point -
Umkehrpunkt *)
    MOTIONPOINTTYPE_MOTION     := 16#0008, (* Motion Point -
Bewegungspunkt *)
    MOTIONPOINTTYPE_ADD    := 16#0F00, (* Addieren von
Segmenten *)
    MOTIONPOINTTYPE_ACTIVATION := 16#2000  (* 1: activation point
*)
);
END_TYPE
```

Typdeklaration für die Tabellen ID.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.9 MC_MotionFunctionType

```
TYPE MC_MotionFunctionType :
(
    MOTIONFUNCTYPE_NOTDEF,
    MOTIONFUNCTYPE_POLYNOM1 := 1,     (* 1: polynom with order
1 *)
    MOTIONFUNCTYPE_POLYNOM3 := 3,     (* 3: polynom with order
3 (rest <-> rest) *)
    MOTIONFUNCTYPE_POLYNOM5 := 5,     (* 5: polynom with order
5 (rest <-> rest) *)
    MOTIONFUNCTYPE_POLYNOM8 := 8,     (* 8: polynom with order
8 (rest <-> rest) *)
    MOTIONFUNCTYPE_SINUSLINIE := 10,
    MOTIONFUNCTYPE_MODSINUSLINIE := 11,
    MOTIONFUNCTYPE_BESTEHORN := 12,
    MOTIONFUNCTYPE_BESCHLTRAPEZ := 13,    (* 13:
Beschleunigungstrapez *)
    MOTIONFUNCTYPE_POLYNOM5_MM := 15,     (* 15: polynom with order
5 (motion <-> motion) *)
    MOTIONFUNCTYPE_SINUS_GERADE_KOMBI := 16,
    MOTIONFUNCTYPE_HARMONIC_KOMBI_RT := 17,
    MOTIONFUNCTYPE_HARMONIC_KOMBI_TR := 18,
    MOTIONFUNCTYPE_HARMONIC_KOMBI_VT := 19,
    MOTIONFUNCTYPE_HARMONIC_KOMBI_TV := 20,
    MOTIONFUNCTYPE_BESCHLTRAPEZ_RT := 21, (* 21:
Beschleunigungstrapez (rest <-> turn) *)
    MOTIONFUNCTYPE_BESCHLTRAPEZ_TR := 22, (* 22:
Beschleunigungstrapez (turn <-> rest) *)
    MOTIONFUNCTYPE_MODSINUSLINIE_VV := 23,
    MOTIONFUNCTYPE_POLYNOM7_MM := 24,     (* 24: polynom with order
7 (motion <-> motion) *)
    MOTIONFUNCTYPE_POLYNOM6STP := 27,     (* 27: polynom with order
6 *)
    MOTIONFUNCTYPE_POLYNOM6WDP := 28,     (* 28: polynom with order
6 *)
    MOTIONFUNCTYPE_STEPFUNCTION := 99
);
END_TYPE
```

Typdeklaration für Motion Functions

Der im TwinCAT Cam Design Editor verwendete Motion-Function-Typ Automatic entspricht MOTIONFUNCTYPE_POLYNOM5_MM.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.10 MC_TableCharacValues

```
TYPE MC_TableCharacValues :
STRUCT
    (* Master Velocity*)
    fMasterVeloNom : LREAL; (* 1. master nominal velocity (normed:
=> 1.0) *)
    (* characteristic slave data *)
    (*==========================*)
    (* Start of cam table *)
    fMasterPosStart : LREAL;    (* 2. master start position
*)
    fSlavePosStart : LREAL;     (* 3. slave start position *)
    fSlaveVeloStart : LREAL;    (* 4. slave start velocity *)
    fSlaveAccStart : LREAL;     (* 5. slave start acceleration
*)
    fSlaveJerkStart : LREAL;    (* 6. slave start jerk *)
    (* End of cam table*)
    fMasterPosEnd : LREAL;      (* 7. master end position *)
    fSlavePosEnd : LREAL;       (* 8. slave end position *)
    fSlaveVeloEnd : LREAL;      (* 9. slave end velocity *)
    fSlaveAccEnd : LREAL;       (* 10. slave end acceleration
*)
```

```
    fSlaveJerkEnd : LREAL;      (* 11. slave end jerk *)
    (* minimum slave position *)
    fMPosAtSPosMin : LREAL;     (* 12. master position AT slave
minimum position *)
    fSlavePosMin : LREAL;       (* 13. slave minimum position
*)
    (* minimum Slave velocity *)
    fMPosAtSVeloMin : LREAL;    (* 14. master position AT slave
minimum velocity *)
    fSlaveVeloMin : LREAL;      (* 15. slave minimum velocity
*)
    (* minimum slave acceleration *)
    fMPosAtSAccMin : LREAL;     (* 16. master position AT slave
minimum acceleration *)
    fSlaveAccMin : LREAL;       (* 17. slave minimum acceleration
*)
    fSVeloAtSAccMin : LREAL;    (* 18. slave velocity AT slave
minimum acceleration *)
    (* minimum slave jerk and dynamic momentum *)
    fSlaveJerkMin : LREAL;      (* 19. slave minimum jerk *)
    fSlaveDynMomMin : LREAL;    (* 20. slave minimum dynamic
momentum (NOT SUPPORTED YET !) *)
    (* maximum slave position *)
    fMPosAtSPosMax : LREAL;     (* 21. master position AT slave
maximum position *)
    fSlavePosMax : LREAL;       (* 22. slave maximum position
*)
    (* maximum Slave velocity *)
    fMPosAtSVeloMax : LREAL;    (* 23. master position AT slave
maximum velocity *)
    fSlaveVeloMax : LREAL;      (* 24. slave maximum velocity
*)
    (* maximum slave acceleration *)
    fMPosAtSAccMax : LREAL;     (* 25. master position AT slave
maximum acceleration *)
    fSlaveAccMax : LREAL;       (* 26. slave maximum acceleration
*)
    fSVeloAtSAccMax : LREAL;    (* 27. slave velocity AT slave
maximum acceleration *)
    (* maximum Slave slave jerk and dynamic momentum *)
    fSlaveJerkMax : LREAL;      (* 28. slave maximum jerk *)
    fSlaveDynMomMax : LREAL;    (* 29. slave maximum dynamic
momentum (NOT SUPPORTED YET !) *)
    (* mean and effective values *)
    fSlaveVeloMean : LREAL;     (* 30. slave mean absolute velocity
(NOT SUPPORTED YET !) *)
    fSlaveAccEff : LREAL;       (* 31. slave effective acceleration
(NOT SUPPORTED YET !) *)
    (* reserved space for future extension *)
    reserved : ARRAY[32..47] OF LREAL;
    (* organization structure of the cam table *)
    CamTableID : UDINT;
    NumberOfRows : UDINT;       (* number of cam table entries, e.
g. number of points *)
    NumberOfColumns : UDINT;    (* number of table columns,
typically 1 or 2 *)
    TableType : UINT;       (* MC_TableType *)
    Periodic : BOOL;
    reserved2 : ARRAY[1..121] OF BYTE;
END_STRUCT
END_TYPE
```

Typdeklaration für die charakteristischen Kenngrößen einer Motion Function.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.11 MC_ValueSelectType

```
TYPE MC_ValueSelectType :
(
    (* a bitmask can be created by adding the following values
*)
    MC_VALUETYPE_POSITION     := 1,
```

```
    MC_VALUETYPE_VELOCITY     := 2,
    MC_VALUETYPE_ACCELERATION := 4,
    MC_VALUETYPE_JERK     := 8
);
END_TYPE
```

Typdeklaration für den Zugriff auf Wertetabellen mit dem Funktionsbaustein MC_ReadMotionFunctionValues.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.12 MC_TableErrorCodes

```
TYPE
    (* Cam Table Error Codes *)
    MC_ERROR_POINTER_INVALID     := 16#4B30, (* invalid pointer
(address) value *)
    MC_ERROR_ARRAYSIZE_INVALID   := 16#4B31, (* invalid size of
data structure *)
    MC_ERROR_CAMTABLEID_INVALID  := 16#4B32, (* invalid cam table
ID (not [1..255]) *)
    MC_ERROR_POINTID_INVALID     := 16#4B33, (* invalid point ID
*)
    MC_ERROR_NUMPOINTS_INVALID   := 16#4B34,
    MC_ERROR_MCTABLETYPE_INVALID := 16#4B35,
    MC_ERROR_NUMROWS_INVALID     := 16#4B36,
    MC_ERROR_NUMCOLUMNS_INVALID  := 16#4B37,
    MC_ERROR_INCREMENT_INVALID   := 16#4B38
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.13 MC_CamActivationMode

```
TYPE
MC_CamActivationMode :

(

    MC_CAMACTIVATION_INSTANTANEOUS,     (* instantaneous change
*)

    MC_CAMACTIVATION_ATMASTERCAMPOS,    (* modify the data at a
defined master

                        position referring to
the cam tables

                    1      master position *)

    MC_CAMACTIVATION_ATMASTERAXISPOS,   (* modify the data at a
defined master

                        position referring to
the absolute

                        master axis position
*)

    MC_CAMACTIVATION_NEXTCYCLE,     (* modify the data at the
beginning of

                        the next cam table cycle
*)
   MC_CAMACTIVATION_NEXTCYCLEONCE,     (* not
```

```
yet implemented

                        modify the data at the
beginning of

                        the next cam table
cycle, activation

                        is valid for one cycle
only *)
    MC_CAMACTIVATION_ASSOONASPOSSIBLE,  (*
modify the data as soon as the

                        cam table is in a safe
state to

                        change its data *)

    MC_CAMACTIVATION_OFF,           (* don't accept any
modification *)

    MC_CAMACTIVATION_DELETEQUEUEDDATA   (* delete all data which
was written

                        to modify the cam table
but is

                        still not activated
*)

);

END_TYPE
```

MC_CamActivationMode legt Zeitpunkt und Art der Änderung einer Kurvenscheibe fest. Änderungen können einerseits durch Skalierung oder aber durch Ändern der Kurvenscheibendaten durchgeführt werden. Es sind jeweils nicht alle Modi möglich.

In den einzelnen Fällen sind die folgenden Modi möglich:

Skalieren von Kurvenscheiben

Kurvenscheiben können mit dem Funktionsbaustein MC_CamScaling [▶ 45] skaliert werden. Dabei sind folgende Aktivierungsmodi gültig.

MC_CAMACTIVATION_INSTANTANEOUS	Die Skalierung wird sofort ausgeführt.
MC_CAMACTIVATION_ATMASTERCAMPOS	Die Skalierung wird an einer bestimmten Kurvenscheibenposition (Master-Position innerhalb der Kurvenscheibe) durchgeführt. Das Skalierungs-Kommando muss vor dieser Position ausgeführt werden.
MC_CAMACTIVATION_ATMASTERAXISPOS	Die Skalierung wird an einer bestimmten absoluten Position der Master-Achse durchgeführt. Das Skalierungs-Kommando muss vor dieser Position ausgeführt werden.
MC_CAMACTIVATION_NEXTCYCLE	Die Skalierung wird bei einer zyklischen Kurvenscheibe am Übergang zur nächsten Periode durchgeführt.
MC_CAMACTIVATION_OFF	Es wird keine Skalierung durchgeführt. Dadurch kann beispielsweise die Skalierung nur für eine Achse (Master oder Slave) ausgeführt werden.

Setzen des Modus für Online-Änderung einer Kurvenscheibe (Schreiben von Punktedaten)

Mit MC_SetCamOnlineChangeMode [▶ 51] wird festgelegt, wann geänderte Kurvenscheibendaten aktiv werden (siehe auch MC_WriteMotionFunction [▶ 48] und MC_WriteMotionFunctionPoint [▶ 50]).

In beiden Fällen sind folgende Modi möglich:

MC_CAMACTIVATION_INSTANTANEOUS	Die Änderung wird sofort durchgeführt.
MC_CAMACTIVATION_ATMASTERCAMPOS	Die Änderung wird an einer bestimmten Kurvenscheibenposition (Master-Position innerhalb der Kurvenscheibe) durchgeführt. Das Kommando muss vor dieser Position ausgeführt werden.
MC_CAMACTIVATION_ATMASTERAXISPOS	Die Änderung wird an einer bestimmten absoluten Position der Master-Achse durchgeführt. Das Kommando muss vor dieser Position ausgeführt werden.
MC_CAMACTIVATION_NEXTCYCLE	Die Änderung wird bei einer zyklischen Kurvenscheibe am Übergang zur nächsten Periode durchgeführt.
MC_CAMACTIVATION_ASSOONASPOSSIBLE	Geänderte Kurvenscheibendaten werden übernommen, sobald das aus dynamischen Gründen möglich ist.
MC_CAMACTIVATION_OFF	Änderungen der Kurvenscheibendaten werden ignoriert.
MC_CAMACTIVATION_DELETEQUEUEDDATA	Gepufferte Kurvenscheibendaten werden gelöscht. Daten werden beispielsweise gepuffert, wenn die Änderung an einer bestimmten Masterposition oder am Zyklusende angefordert wurde.

Ankoppeln mit Kurvenscheiben

Mit dem Funktionsbausteinen MC_CamInExt [▶ 39] können Achsen mit Kurvenscheiben gekoppelt werden. Über den *ActivationMode* kann optional festgelegt werden, ab welcher Position die Slave-Achse aktiv wird.

MC_CAMACTIVATION_INSTANTANEOUS	Die Kurvenscheibenkopplung wird sofort aktiv und der Slave bewegt sich gemäß der Kurvenscheibendaten.
MC_CAMACTIVATION_ATMASTERCAMPOS	Die Kurvenscheibenkopplung ist schwebend aktiv. Der Slave bewegt sich erst ab einer definierten Kurvenscheibenposition (Master-Position innerhalb der Kurvenscheibe) gemäß der Kurvenscheibendaten. Dieser *ActivationMode* kann beim Ankoppeln nicht in Verbindung mit MC_StartMode [▶ 56] = MC_STARTMODE_RELATIVE oder MC_STARTMODE_MASTERREL_SLAVEABS verwendet werden.
MC_CAMACTIVATION_ATMASTERAXISPOS	Die Kurvenscheibenkopplung ist schwebend aktiv. Der Slave bewegt sich erst ab einer definierten absoluten Position der Master-Achse gemäß der Kurvenscheibendaten.
MC_CAMACTIVATION_NEXTCYCLE	Die Kurvenscheibenkopplung ist schwebend aktiv. Der Slave bewegt sich ab dem nächsten Zyklusübergang (bei zyklischen Kurvenscheiben). Dieser *ActivationMode* kann beim Ankoppeln nicht in Verbindung mit MC_StartMode [▶ 56] = MC_STARTMODE_RELATIVE oder MC_STARTMODE_MASTERREL_SLAVEABS verwendet werden.

Umschalten von Kurvenscheiben

Mit dem Funktionsbausteinen MC_CamInExt [▶ 39] kann im gekoppelten Zustand von einer Kurvenscheibe auf eine andere umgeschaltet werden. Über den *ActivationMode* kann festgelegt werden, an welcher Position die Umschaltung statt findet.

MC_CAMACTIVATION_INSTANTANEOUS	Die Kurvenscheibe wird sofort umgeschaltet und der Slave bewegt sich gemäß der neuen Kurvenscheibendaten.
MC_CAMACTIVATION_ATMASTERCAMPOS	Die Kurvenscheibenumschaltung wird an einer definierten Kurvenscheibenposition (Master-Position innerhalb der Kurvenscheibe) durchgeführt .
MC_CAMACTIVATION_ATMASTERAXISPOS	Die Kurvenscheibenumschaltung wird an einer definierten absoluten Position der Master-Achse durchgeführt.
MC_CAMACTIVATION_NEXTCYCLE	Die Kurvenscheibenumschaltung wird bei zyklischen Kurvenscheiben am nächsten Zyklusübergang durchgeführt. Bei linearen Kurvenscheiben findet die Umschaltung an den Rändern des definierten Bereiches statt.
MC_CAMACTIVATION_DELETEQUEUEDDATA	Eine schwebende und noch nicht aktiv gewordene Kurvenscheibenumschaltung wird verworfen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1329	PC (i386)	TcNcCamming.Lib

5.1.16.14 MC_CamScalingMode

```
TYPE MC_CamScalingMode :
(
    MC_CAMSCALING_USERDEFINED, (* user defines scaling parameters -
slope and offset *)
    MC_CAMSCALING_AUTOOFFSET,  (* offset is calculated
automatically for best result *)
    MC_CAMSCALING_OFF      (* no modification accepted *)
);
END_TYPE
```

Typ und Umfang der Skalierung einer Kurvenscheibenkopplung mit dem Funktionsbaustein MC_CamScaling [▶ 45].

MC_CAMSCALING_USERDEFINED : Die Skalierung und der Offset werden unverändert übernommen. Skalierung und Offset müssen vom Anwender so berechnet werden, dass kein Sprung in der Position entsteht.

MC_CAMSCALING_AUTOOFFSET : Die Skalierung wird übernommen und der Offset wird vom System so angepasst, dass kein Sprung in der Position entsteht. Die Skalierung sollte dennoch in einer Phase mit Slave-Geschwindigkeit 0 durchgeführt werden, weil sonst eine Sprung in der Geschwindigkeit nicht vermieden werden kann.

MC_CAMSCALING_OFF : Die Skalierung und der Offset werden ignoriert. Dieser Modus wird beispielsweise eingesetzt, wenn nur eine Slave-Skalierung aber keine Master-Skalierung durch geführt werden soll.

Autooffset

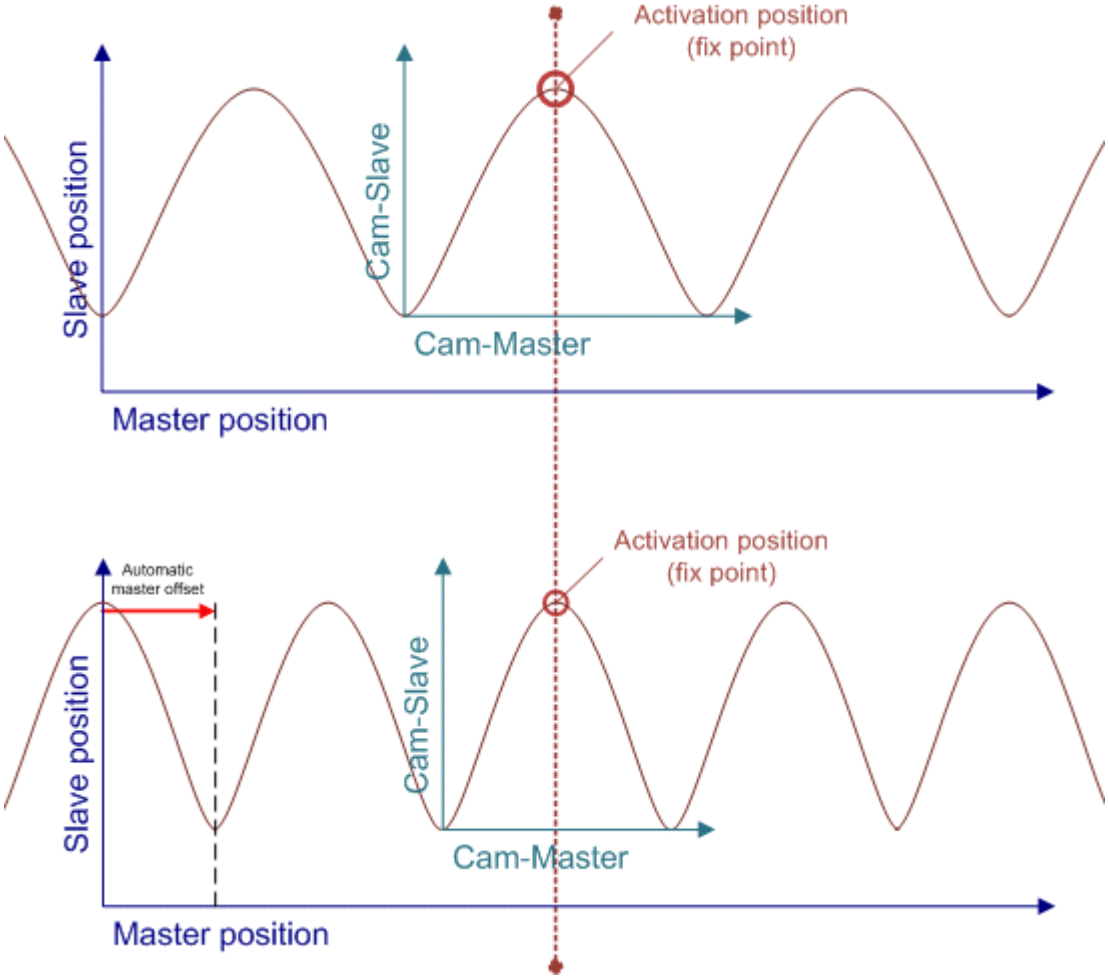
Der *Autooffset* Modus sorgt für eine automatische Anpassung eines Kurvenscheibenoffsets. *Autooffset* kann unabhängig für die Master- oder die Slave-Achse einer Kurvenscheibe angewendet werden und wirkt sowohl bei der Umschaltung als auch bei der Skalierung von Kurvenscheiben. Die Funktion arbeitet nach den im Folgenden beschriebenen Regeln.

Master-Autooffset

Master-Autooffset sorgt beim Umschalten von Kurvenscheiben mit eventuell verschiedenem Master-Zyklus oder beim Skalieren von Kurvenscheiben (Master-Skalierung), dass die Master-Position der Kurvenscheibe im Achs-Koordinatensystem nicht springt. Diese Funktion ist notwendig, da die relative Position einer Kurvenscheibe im Achs-Koordinatensystem vom Masterzyklus abhängt. Wird der Master-Zyklus z. B. durch Skalierung verändert, so würde die Position sich verändern.

Master-Autooffset bestimmt den Master-Offset der Kurvenscheibe so, dass die Masterposition innerhalb der Kurvenscheibe beibehalten wird. Bei einer Skalierung oder Umschaltung auf eine Kurvenscheibe mit anderem Master-Zyklus bedeutet das, dass die relative (prozentuale) Position vor und nach der Umschaltung identisch ist.

Beispiel: Eine Kurvenscheibe hat einen Master-Zyklus von 360° und wird um den Faktor 2 auf 720° skaliert. Die Skalierung wird an der Position 90° innerhalb der Kurvenscheibe durchgeführt, also bei 25% vom Zyklusanfang. Nach der Skalierung ist die relative Master-Position in der Kurvenscheibe bei 180° also ebenfalls 25% vom Zyklusanfang.



Bei einer Umschaltung an den Rändern einer Kurvenscheibe (siehe MC_CamActivationMode [▶ 61]*MC_CAMACTIVATION_NEXTCYCLE*), sorgt Master-Autooffset für ein nahtloses Aneinanderreihen der Kurvenscheiben sowohl bei zyklischen als auch bei linearen Kurvenscheiben.

Master-Autooffset kann nicht verwendet werden, wenn eine Kurvenscheibe relativ angekoppelt oder umgeschaltet wird, da sich diese Funktionen wiedersprechen. Beim ersten Ankoppeln gibt es weitere Einschränkungen, die aus der folgenden Tabelle ersichtlich sind.

Activation Mode	Coupling with Cam Tables			
Master-ScalingMode:	**without** Master-AutoOffset		**with** Master-AutoOffset	
StartMode:	Absolute	Relative	Absolute	Relative
Instantaneous (default)	√	√	—	—
AtMasterAxisPos	√	√	—	—
AtMasterCamPos	√	—	—	—
NextCycle	√	—	—	—
DeleteQueuedData	—	—	—	—

Activation Mode	Switching of Cam Tables			
Master-ScalingMode:	**without** Master-AutoOffset		**with** Master-AutoOffset	
StartMode:	Absolute	Relative	Absolute	Relative
Instantaneous (default)	√	√	√	—
AtMasterAxisPos	√	√	√	—
AtMasterCamPos	√	√	√	—
NextCycle	√	√	√	—
DeleteQueuedData	√	√	√	—

Slave-Autooffset

Slave-Autooffset berechnet einen Slave-Offset so, dass es durch eine Kurvenscheiben-Umschaltung oder durch eine Skalierung keinen Sprung in der Slave-Position gibt. Der Slave-Offset wird also so angepasst, dass die Slave-Position vor und nach der Aktion identisch ist.

Werden sowohl *Master-Autooffset als* auch *Slave-Autooffset* bei einer Kurvenscheibenumschaltung oder Skalierung verwendet, so wird zuerst der Master-Offset berechnet und anschließend der Slave-Offset.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1329	PC (i386)	TcNcCamming.Lib

5.1.16.15 MC_CamInfoData

```
TYPE MC_CamInfoData :
STRUCT
    CamTableID            : MC_CAM_ID;
    TableType             : MC_TableType;
    Periodic              : BOOL;
    InterpolationType         : MC_InterpolationType;

    NumberOfRows          : UDINT; (* number of cam table
entries, e. g. number of points *)
    NumberOfColumns           : UDINT; (* number of table
columns, typically 1 or 2 *)
    MasterCamStartPos         : LREAL; (* Master position of
the first cam table point *)
    SlaveCamStartPos          : LREAL; (* Slave position of the
first cam table point *)
    RawMasterPeriod           : LREAL;
    RawSlaveStroke        : LREAL;
    MasterAxisCouplingPos     : LREAL; (* Master axis position
when slave has been coupled *)
    SlaveAxisCouplingPos      : LREAL; (* Slave axis position
when slave has been coupled *)
    MasterAbsolute        : BOOL; (* raw unscaled distance
from first to last master cam table position *)
    SlaveAbsolute         : BOOL; (* raw unscaled distance
from first to last slave cam table position *)
```

```
    MasterOffset          : LREAL; (* total master offset
*)
    SlaveOffset           : LREAL; (* total slave offset
*)
    MasterScaling         : LREAL; (* total master scaling
factor *)
    SlaveScaling          : LREAL; (* total slave scaling
factor *)
    SumOfSlaveStrokes         : LREAL; (* sum OF the slave
strokes up TO ActualMasterAxisPos *)
    SumOfSuperpositionDistance    : LREAL; (* sum of additional
moves through MC_MoveSuperimposed *)
    ActualMasterAxisPos       : LREAL; (* absolute set position
of the master axis *)
    ActualSlaveAxisPos        : LREAL; (* absolute set position
of the slave axis *)
    ActualMasterCamPos        : LREAL; (* absolute cam table
position of the master *)
    ActualSlaveCamPos         : LREAL; (* absolute cam table
position of the slave *)
    (* mode for the scaling of cam tables *)
    ScalingPending        : BOOL; (* a change is currently
pending *)
    ScalingActivationMode     : MC_CamActivationMode;

    ScalingActivationPos      : LREAL;
    ScalingMasterScalingMode      : MC_CamScalingMode;

    ScalingSlaveScalingMode       : MC_CamScalingMode;
    (* mode for online changes of cam table data *)
    CamDataQueued         : BOOL; (* a change is currently
pending *)
    OnlineChangeActivationMode    : MC_CamActivationMode;

    OnlineChangeActivationPos     : LREAL;
    OnlineChangeMasterScalingMode : MC_CamScalingMode;

    OnlineChangeSlaveScalingMode  : MC_CamScalingMode;
    (* mode for exchanging cam tables with MC_CamIn *)
    CamTableQueued        : BOOL; (* a change is currently
pending *)
    CamExchangeCamTableID     : MC_CAM_ID;
    CamExchangeActivationMode     : MC_CamActivationMode;

    CamExchangeActivationPos      : LREAL;
    CamExchangeMasterScalingMode  : MC_CamScalingMode;

    CamExchangeSlaveScalingMode   : MC_CamScalingMode;
END_STRUCT
END_TYPE
```

MC_CamScalingMode [▶ 64], MC_CamActivationMode [▶ 61], MC_CAM_ID [▶ 55], MC_InterpolationType [▶ 57] MC_TableType [▶ 56]

Die Datenstruktur *MC_CamInfoData* enthält Daten zum aktuellen Zustand einer Kurvenscheibenkopplung. Die Daten werden mit dem Funktionsbaustein MC_CamInfo [▶ 54] ermittelt.

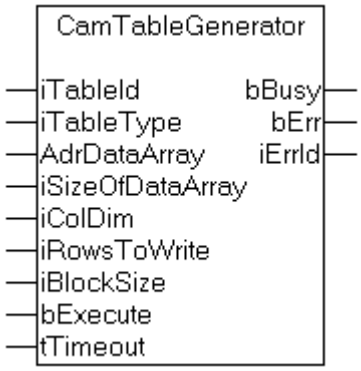
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1026	PC (i386)	TcNcCamming.Lib

5.2 Bausteine zur Kompatibilität mit bestehenden Programmen

5.2.1 Tabellenbausteine

5.2.1.1 CamTableGenerator



CamTableGenerate erstellt eine neue Tabelle im TwinCAT NC-Kontext und füllt sie mit Daten. Eine eventuell existierende Tabelle mit gleicher Tabellen-ID wird dabei gelöscht.

Interface

```
VAR_INPUT
    iTableId : INT;
    iTableType : UDINT;
    AdrDataArray : POINTER TO LREAL; (* PLC: ARRAY[ ROWS, COLUMNS ] OF LREAL *)
    iSizeOfDataArray : UDINT;
    iColDim : UDINT; (* second array dimension (COLUMNS) *)
    iRowsToWrite : UDINT; (* number of rows to write <= RowDim *)
    iBlockSize : UDINT; (* number of bytes to write in one internal step *)
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID : Tabellen-ID unter der die Tabelle künftig angesprochen wird

iTableType : Tabellentyp der Tabelle

(* (n*m) tabular with equidistant ascending master values *)
NC_TABULARTYPE_EQUIDIST_LINEAR : UDINT := 1;

(* same but cyclic *)
NC_TABULARTYPE_EQUIDIST_CYCLE : UDINT := 2;

(* (n*m) tabular with strictly monotone ascending master values (not imperative equidistant) *)
NC_TABULARTYPE_NONEQUIDIST_LINEAR : UDINT := 3;

(* same but cyclic *)
NC_TABULARTYPE_NONEQUIDIST_CYCLE : UDINT := 4;

AdrDataArray : Adresse eines Datenfeldes, das die Positionsdaten für Master- und Slave-Achsen enthält. Die erste Dimension des Feldes beschreibt die Tabellenzeilen, die zweite Dimension die Tabellenspalten.

iSizeOfDataArray : Gesamtgröße des Datenfeldes in Byte

iColDim : Anzahl der Spalten des Datenfeldes. Dieser Wert muss der tatsächlichen Größe der zweiten Felddimension entsprechen.

iRowsToWrite : Anzahl der Tabellenzeilen. Dieser Wert darf kleiner oder gleich der Größe der ersten Felddimension sein.

iBlockSize : Anzahl der Datenbytes, die der Funktionsbaustein in einem zusammenhängenden Block in die Tabelle im NC-Kontext überträgt. Bei iBlockSize=0 wird ein Default-Wert verwendet, der in den meisten Fällen zu empfehlen ist. Nur bei genauer Kenntnis der internen Vorgänge sollte iBlockSize>0 angegeben werden.

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

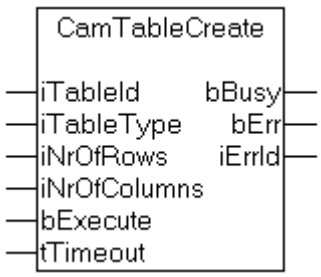
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.1.2 CamTableCreate



CamTableCreate erstellt eine neue, zunächst leere Tabelle im TwinCAT NC-Kontext.

Interface

```
VAR_INPUT
    iTableId : INT;
    iTableType : UDINT; (* table type *)
    iNrOfRows : UDINT; (* number of table lines (n) *)
    iNrOfColumns : UDINT; (* number of table columns (m) *)
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID : Tabellen-ID unter der die Tabelle künftig angesprochen wird

iTableType : Tabellentyp der Tabelle

(* (n*m) tabular with equidistant ascending master values *)
NC_TABULARTYPE_EQUIDIST_LINEAR : UDINT := 1;

(* same but cyclic *)
NC_TABULARTYPE_EQUIDIST_CYCLE : UDINT := 2;

(* (n*m) tabular with strictly monotone ascending master values (not imperative equidistant) *)
NC_TABULARTYPE_NONEQUIDIST_LINEAR : UDINT := 3;

(* same but cyclic *)
NC_TABULARTYPE_NONEQUIDIST_CYCLE : UDINT := 4;

iNrOfRows : Anzahl der Tabellenzeilen

iNrOfColumns : Anzahl der Tabellenspalten

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

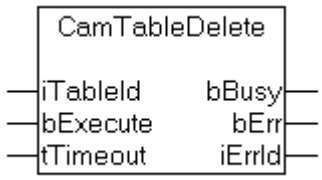
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.1.3 CamTableDelete



CamTableDelete löscht eine Tabelle aus dem NC-Kontext

Interface

```
VAR_INPUT
    iTableId : INT;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID : Tabellen-ID der zu löschenden Tabelle

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

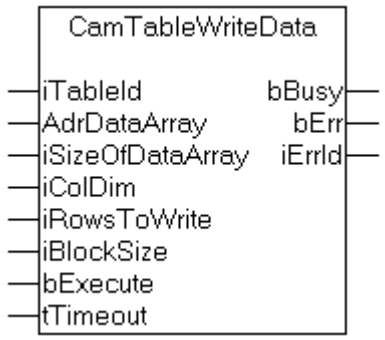
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.1.4 CamTableWriteData



CamTableWriteData schreibt Tabellendaten aus einer SPS-Variablen in eine Tabelle im NC-Kontext.

Interface

```
VAR_INPUT
    iTableId : INT;
    AdrDataArray : POINTER TO LREAL; (* PLC: ARRAY[ ROWS, COLUMNS ] OF LREAL *)
    iSizeOfDataArray : UDINT;
    iColDim : UDINT; (* second array dimension (COLUMNS) *)
    iRowsToWrite : UDINT; (* number of rows to write <= RowDim *)
    iBlockSize : UDINT; (* number of bytes to write in one internal step *)
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID : Tabellen-ID der Tabelle aus der die Daten gelesen werden sollen

AdrDataArray : Adresse eines Datenfeldes, dass die Positionsdaten für Master- und Slave-Achsen enthält. Die erste Dimension des Feldes beschreibt die Tabellenzeilen, die zweite Dimension die Tabellenspalten.

iSizeOfDataArray : Gesamtgröße des Datenfeldes in Byte

iColDim : Anzahl der Spalten des Datenfeldes. Dieser Wert muss der tatsächlichen Größe der zweiten Felddimension entsprechen.

iRowsToWrite : Anzahl der Tabellenzeilen. Dieser Wert darf kleiner oder gleich der Größe der ersten Felddimension sein.

iBlockSize : Anzahl der Datenbytes, die der Funktionsbaustein in einem zusammenhängenden Block in die Tabelle im NC-Kontext überträgt. Bei iBlockSize=0 wird ein Default-Wert verwendet, der in den meisten Fällen zu empfehlen ist. Nur bei genauer Kenntnis der internen Vorgänge sollte iBlockSize>0 angegeben werden.

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

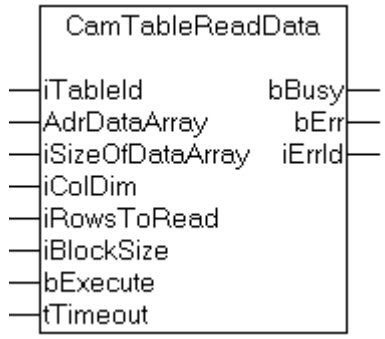
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.1.5 CamTableReadData



CamTableReadData liest Tabellendaten aus einer Tabelle im NC-Kontext und legt die Daten in einer SPS-Variablen ab

Interface

```
VAR_INPUT
    iTableId : INT;
    AdrDataArray : POINTER TO LREAL; (* PLC: ARRAY[ ROWS, COLUMNS ] OF LREAL *)
    iSizeOfDataArray : UDINT;
    iColDim : UDINT; (* second array dimension (COLUMNS) *)
    iRowsToRead : UDINT; (* number of rows to read <= RowDim *)
    iBlockSize : UDINT; (* number of bytes to read in one internal step *)
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID : Tabellen-ID der Tabelle aus der die Daten gelesen werden sollen

AdrDataArray : Adresse eines Datenfeldes, dass die Positionsdaten für Master- und Slave-Achsen aufnehmen soll. Die erste Dimension des Feldes beschreibt die Tabellenzeilen, die zweite Dimension die Tabellenspalten.

iSizeOfDataArray : Gesamtgröße des Datenfeldes in Byte

iColDim : Anzahl der Spalten des Datenfeldes. Dieser Wert muss der tatsächlichen Größe der zweiten Felddimension entsprechen.

iRowsToRead : Anzahl der Tabellenzeilen. Dieser Wert darf kleiner oder gleich der Größe der ersten Felddimension sein.

iBlockSize : Anzahl der Datenbytes, die der Funktionsbaustein in einem zusammenhängenden Block in die Tabelle im NC-Kontext überträgt. Bei iBlockSize=0 wird ein Default-Wert verwendet, der in den meisten Fällen zu empfehlen ist. Nur bei genauer Kenntnis der internen Vorgänge sollte iBlockSize>0 angegeben werden.

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

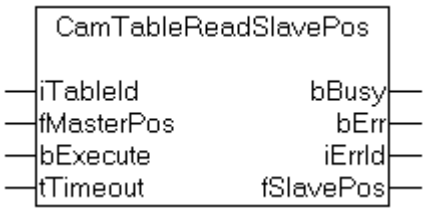
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flange an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.1.6 CamTableReadSlavePos



CamTableReadSlavePos liest zu einer Tabellenposition einer Master-Achse die über die Tabelle definierte Slave-Position. Dabei werden gegebenenfalls Positionsdaten für die Slave-Achse interpoliert, wenn die Master-Position nicht exakt einem Tabelleneintrag entspricht.

Interface

```
VAR_INPUT
    iTableId : INT;
    fMasterPos : LREAL;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
    fSlavePos : LREAL;
END_VAR
```

iTableID : Tabellen-ID unter der die Tabelle künftig angesprochen wird

fMasterPos : Position der Master-Achse

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

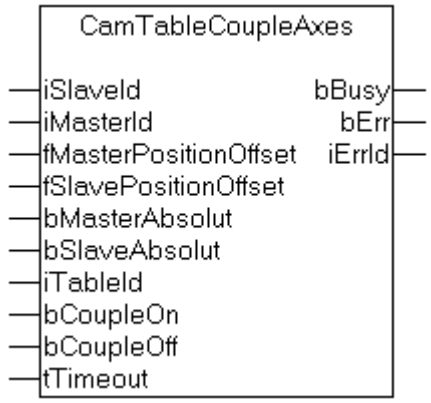
bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

fSlavePos : Ermittelte Position der Slave-Achse

5.2.2 Bausteine für Kurvenscheiben-Achsen

5.2.2.1 CamTableCoupleAxes



CamTableCoupleAxes koppelt eine Slave-Achse über eine existierende Tabelle an eine Master-Achse

Interface

```
VAR_INPUT
    iSlaveId : INT;
    iMasterId : INT;
    fMasterPositionOffset : LREAL;
    fSlavePositionOffset : LREAL;
    bMasterAbsolut : BOOL;
    bSlaveAbsolut : BOOL;
    iTableId : INT;
    bCoupleOn : BOOL;
    bCoupleOff : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iSlaveId : Achs-ID der Slave-Achse die gekoppelt werden soll

iMasterId : Achs-ID der Master-Achse an die gekoppelt werden soll

fMasterPositionOffset : Master-Positionsoffset auf die Tabellendaten

fSlavePositionOffset : Slave-Positionsoffset auf die Tabellendaten;

bMasterAbsolut : Tabellendaten für den Master als absolute Positionen werten

bSlaveAbsolut : Tabellendaten für den Slave als absolute Positionen werten

iTableId : Tabellen-ID der Tabelle über die gekoppelt wird

bCoupleOn : Flankengetriggertes Signal zum Ausführen der Achskopplung

bCoupleOff : Flankengetriggertes Signal zum Entkoppeln der Achsen

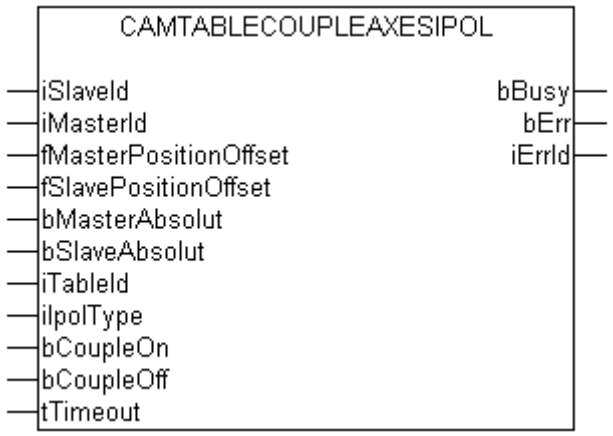
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bCoupleOn oder bCoupleOff TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.2.2 CamTableCoupleAxesIpol



CamTableCoupleAxesIpol koppelt eine Slave-Achse über eine existierende Tabelle an eine Master-Achse. Weiterhin kann der Typ der Interpolation zwischen den Tabellen-Stützstellen festgelegt werden.

Interface

```
VAR_INPUT
    iSlaveId : INT;
    iMasterId : INT;
    fMasterPositionOffset : LREAL;
    fSlavePositionOffset : LREAL;
    bMasterAbsolut : BOOL;
    bSlaveAbsolut : BOOL;
    iTableId : INT;
    iIpolType : UINT;
    bCoupleOn : BOOL;
    bCoupleOff : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iSlaveId : Achs-ID der Slave-Achse die gekoppelt werden soll

iMasterId : Achs-ID der Master-Achse an die gekoppelt werden soll

fMasterPositionOffset : Master-Positionsoffset auf die Tabellendaten

fSlavePositionOffset : Slave-Positionsoffset auf die Tabellendaten;

bMasterAbsolut : Tabellendaten für den Master als absolute Positionen werten

bSlaveAbsolut : Tabellendaten für den Slave als absolute Positionen werten

iTableId : Tabellen-ID der Tabelle über die gekoppelt wird

iIpolType : Typ der Interpolation zwischen den Tabellen-Stützstellen. Gültige Werte sind:

```
NC_INTERPOLATIONTYPE_LINEAR : UINT := 0; (* standard, uses 2 points *)
   NC_INTERPOLATIONTYPE_4POINT : UINT := 1; (* uses 4 points *)
   NC_INTERPOLATIONTYPE_SPLINE : UINT := 2; (* spline interpolation *)
```

bCoupleOn : Flankengetriggertes Signal zum Ausführen der Achskopplung

bCoupleOff : Flankengetriggertes Signal zum Entkoppeln der Achsen

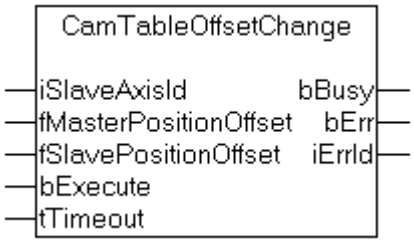
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bCoupleOn oder bCoupleOff TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.2.3 CamTableOffsetChange



CamTableOffsetChange ändert den Master- und Slave-Offset der Tabelle online, das heißt während Master- und Slave-Achse über die Tabelle gekoppelt sind.

Interface

```
VAR_INPUT
    iSlaveAxisId : UINT;
    fMasterPositionOffset : LREAL;
    fSlavePositionOffset : LREAL;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iSlaveAxisId : Achs-Id der Slave-Achse. Der zugehörige Master ist durch die bestehende Achskopplung bekannt.

fMasterPositionOffset : Positions-Offset auf die Master-Positionsdaten

fSlavePositionOffset : Positions-Offset auf die Slave-Positionsdaten

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

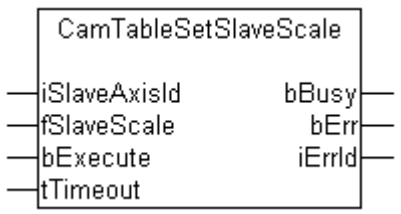
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.2.4 CamTableSetSlaveScale



CamTableSetSlaveScale skaliert die Slave-Spalte einer Tabelle für eine bestehende Master-Slave-Tabellenkopplung.

Interface

```
VAR_INPUT
    iSlaveAxisId : UINT;
    fSlaveScale : LREAL;
   bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iSlaveId : Achs-ID der Slave-Achse, die über eine Tabelle an eine Master-Achse gekoppelt ist

fSlaveScale : Skalierungsfaktor der Slave-Spalte der Tabelle

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

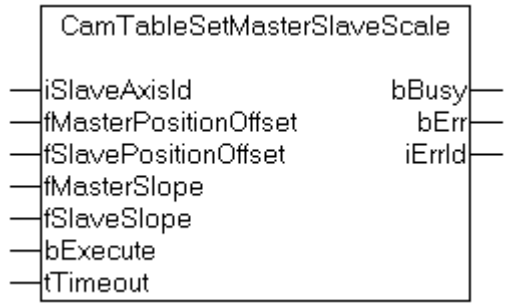
tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.2.5 CamTableSetMasterSlaveScale



CamTableSetMasterSlaveScale skaliert Master- und Slave-Spalte einer Tabelle für eine bestehende Master-Slave-Tabellenkopplung.

Interface

```
VAR_INPUT
    iSlaveAxisId : UINT;
    fMasterPositionOffset : LREAL;
    fSlavePositionOffset : LREAL;
    fMasterSlope : LREAL;
    fSlaveSlope : LREAL;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iSlaveId : Achs-ID der Slave-Achse, die über eine Tabelle an eine Master-Achse gekoppelt ist

fMasterOffset : Master-Positionsoffset der Tabelle

fSlaveOffset : Slave-Positionsoffset der Tabelle

fMasterSlope : Master-Skalierung der Tabelle

fSlaveSlope : Slave-Skalierung der Tabelle

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

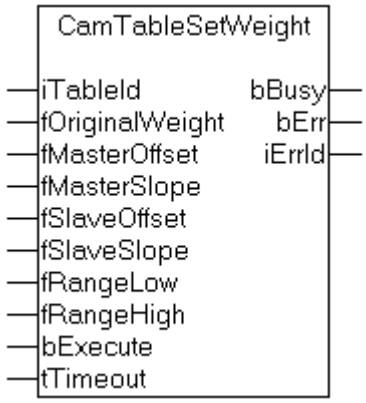
bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.3 Bausteine für Multitabellen-Achsen

5.2.3.1 CamTableSetWeight

(nur für Multitabellen)



CamTableSetWeight setzt die Skalierungs-Parameter einer skalierbaren *Multi*-Tabelle.

Interface

```
VAR_INPUT
    iTableId    : INT;
    fOriginalWeight : LREAL;
    fMasterOffset   : LREAL;
    fMasterSlope    : LREAL;
    fSlaveOffset    : LREAL;
    fSlaveSlope     : LREAL;
    fRangeLow       : LREAL;
    fRangeHigh      : LREAL;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr  : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID : Tabellen-ID unter der die Tabelle künftig angesprochen wird

fOriginalWeight : Originalgewicht der Tabelle
fMasterOffset : Master-Positionsoffset der Tabelle
fMasterSlope : Master-Skalierung der Tabelle
fSlaveOffset : Slave-Positionsoffset der Tabelle
fSlaveSlope : Slave-Skalierung der Tabelle
fRangeLow : untere Bereichsgrenze, in der die Tabelle gültig ist
fRangeHigh : obere Bereichsgrenze, in der die Tabelle gültig ist

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

tTimeout : ADS-Timeout (ca. 1 Sekunde)

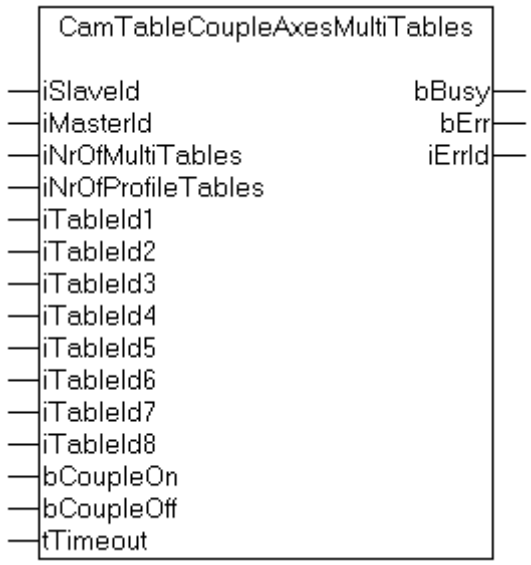
bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.3.2 CamTableCoupleAxesMultiTables

(nur für Multitabellen)



CamTableCoupleAxesMultiTables koppelt eine Slave-Achse über mehrere existierende Tabellen an eine Master-Achse (*Multi*-Tabellenkopplung)

Interface

```
VAR_INPUT
    iSlaveId        : INT;
    iMasterId       : INT;
    iNrOfMultiTables    : INT;
    iNrOfProfileTables  : INT;
    iTableId1       : INT;
    iTableId2       : INT;
    iTableId3       : INT;
    iTableId4       : INT;
    iTableId5       : INT;
    iTableId6       : INT;
    iTableId7       : INT;
    iTableId8       : INT;
    bCoupleOn       : BOOL;
    bCoupleOff      : BOOL;
    tTimeout        : TIME;
END_VAR
VAR_OUTPUT
    bBusy           : BOOL;
    bErr        : BOOL;
    iErrId          : UDINT;
END_VAR
```

iSlaveId : Achs-ID der Slave-Achse die gekoppelt werden soll

iMasterId : Achs-ID der Master-Achse an die gekoppelt werden soll

iNrOfMultiTables : Gesamtanzahl der tatsächlich genutzten Tabellen (1 bis 8)

iNrOfProfileTables : Anzahl der Profiltabellen (1 bis iNrOfMultiTables). Übrige Tabellen sind zuschaltbare Korrekturtabellen.

iTableId1 : Tabellenindex der ersten Tabelle

iTableId2 : Tabellenindex der 2. Tabelle

iTableId3 : Tabellenindex der 3. Tabelle

iTableId4 : Tabellenindex der 4. Tabelle

iTableId5 : Tabellenindex der 5. Tabelle

iTableId6 : Tabellenindex der 6. Tabelle

iTableId7 : Tabellenindex der 7. Tabelle

iTableId8 : Tabellenindex der 8. Tabelle

bCoupleOn : Flankengetriggertes Signal zum Ausführen der Achskopplung

bCoupleOff : Flankengetriggertes Signal zum Entkoppeln der Achsen

tTimeout : ADS-Timeout (ca. 1 Sekunde)

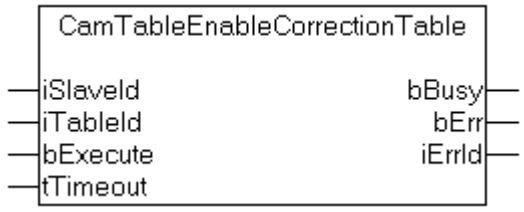
bBusy : wird mit steigender Flanke an bCoupleOn oder bCoupleOff TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.3.3 CamTableEnableCorrectionTable

(nur für Multitabellen)



CamTableEnableCorrectionTable aktiviert eine Korrekturtabelle bei *Multi*-Tabellen-Anwendungen.

Interface

```
VAR_INPUT
    iSlaveId    : INT;
    iTableId    : INT;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
VAR_OUTPUT
    bBusy       : BOOL;
    bErr    : BOOL;
    iErrId      : UDINT;
END_VAR
```

iSlaveAxisId : Achs-Id der Slave-Achse. Der zugehörige Master ist durch die bestehende Achskopplung bekannt.

iTableID : Tabellen-ID der Tabelle aus der die Daten gelesen werden sollen

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

tTimeout : ADS-Timeout (ca. 1 Sekunde)

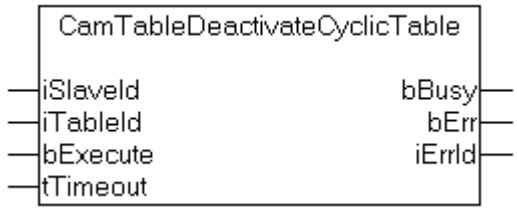
bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.2.3.4 CamTableDeactivateCyclicTable

(nur für Multitabellen)



CamTableDeactivateCyclicTable schaltet eine aktive zyklische *Multi*-Tabelle am Zyklusende ab.

Interface

```
VAR_INPUT
    iSlaveId    : INT;
    iTableId    : INT;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
VAR_OUTPUT
    bBusy       : BOOL;
    bErr    : BOOL;
    iErrId      : UDINT;
END_VAR
```

iSlaveAxisId : Achs-Id der Slave-Achse. Der zugehörige Master ist durch die bestehende Achskopplung bekannt.

iTableID : Tabellen-ID der Tabelle aus der die Daten gelesen werden sollen

bExecute : Flankengetriggertes Signal zum Ausführen des Kommandos

tTimeout : ADS-Timeout (ca. 1 Sekunde)

bBusy : wird mit steigender Flanke an bExecute TRUE und bleibt TRUE, solange der Baustein das Kommando ausführt

bErr : wird TRUE, wenn beim Ausführen des Kommandos ein Fehler aufgetreten ist.

bErrId : Fehlernummer (ADS- oder NC-Fehlernummer)

5.3 Get_TcNcCamming_Version

Get_TcNcCamming_Version ermittelt die Versionsnummer der SPS-Bibliothek TcNcCamming.lib. Der Funktionsaufruf liefert die Versionsnummer in einem String zurück.

Interface

```
FUNCTION Get_TcNcCamming_Version : STRING(20)
VAR_INPUT
    bGet : BOOL;
END_VAR
```

bGet : Signal zum Ausführen des Kommandos

Mehr Informationen:
www.beckhoff.de

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