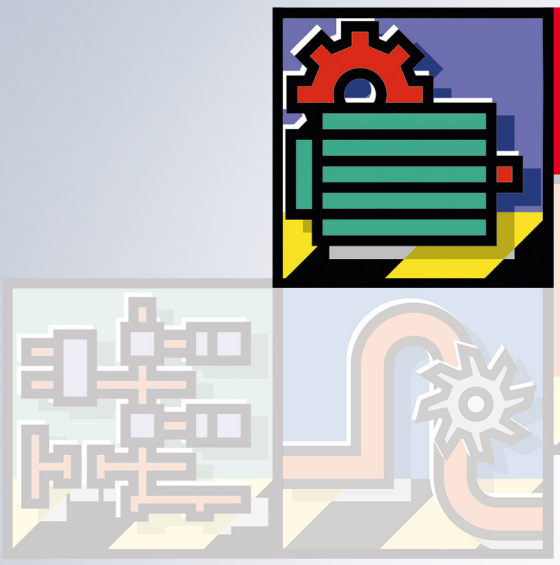
BECKHOFF New Automation Technology

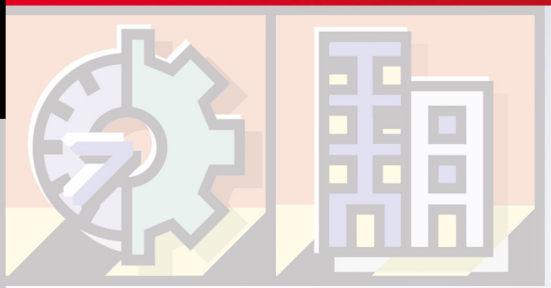
Handbuch | DE

TS5050

TwinCAT 2 | MC Camming



Supplement | Motion



03.07.2025 | Version: 1.4

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige 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 weiterentwickelt.
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®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Kennzeichnungen führen.

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

Copyright

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Signalwörter

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

Warnungen vor Personenschäden

⚠ GEFAHR

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

⚠ WARNUNG

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

⚠ VORSICHT

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

Warnung vor Umwelt- oder Sachschäden

HINWEIS

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

Information zum Umgang mit dem Produkt

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

1.3 Hinweise zur Informationssicherheit

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

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

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

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

2 Übersicht

In vielen Anwendungen ist es notwendig, zwei oder mehr Achsen miteinander zu synchronisieren. In der TwinCAT NC PTP können Achsen aneinander gekoppelt 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 TcMC2_Camming enthält Funktionsbausteine für den Umgang mit Kurvenscheiben. Es werden zwei Arten von Kurvenscheiben unterstützt.

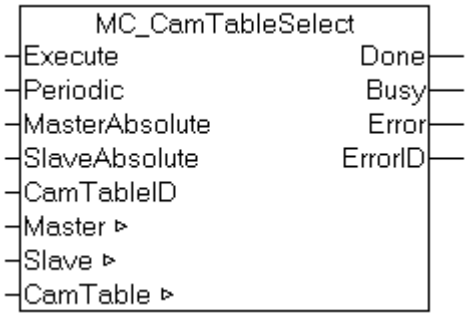
Zum einen kann eine Kurvenscheibe eine zweispaltige Tabelle von Master- und Slave-Positionen sein (Standardtabelle). Die Masterspalte definiert Stützstellen über den Verfahrweg des Masters von einem kleinsten Positionswert aufsteigend bis zu einem größten Wert. Mit den Stützstellen der Tabelle wird aus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

3 Kurvenscheiben

3.1 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 [▶ 13].

Eingänge

```
VAR_INPUT
   Execute           : BOOL;
   Periodic          : BOOL;
   MasterAbsolute    : BOOL;
   SlaveAbsolute     : BOOL;
   CamTableID        : MC_CAM_ID;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
Periodic	Wenn die Kurvenscheibe sich zyklisch wiederholt ist *Periodic* TRUE.
MasterAbsolute	Momentan nicht verwendet
SlaveAbsolute	Momentan nicht verwendet
CamTableID	ID der Kurvenscheibe mit der gekoppelt wird

Ausgänge

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

Done	Wird TRUE, wenn die Kurvenscheibe erfolgreich angelegt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

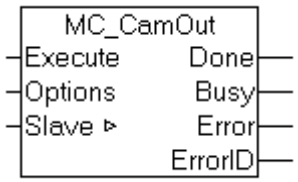
Ein/Ausgänge

```
VAR_IN_OUT
   Master             : AXIS_REF;
   Slave              : AXIS_REF;
   CamTable           : MC_CAM_REF;
END_VAR
```

Master	Achsdatenstruktur des Masters - wird zurzeit nicht verwendet.
Slave	Achsdatenstruktur des Slaves - wird zurzeit nicht verwendet.
CamTable	Datenstruktur vom Typ MC_CAM_REF [▶ 50] beschreibt den Datenspeicher für die Kurvenscheibe in der SPS

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

3.2 MC_CamOut



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

HINWEIS

Wenn eine Slave-Achse in der Bewegung abgekoppelt wird, so wird sie nicht automatisch gestoppt, sondern sie erreicht eine konstante Geschwindigkeit mit der sie endlos weiterfährt. Die Achse kann mit einem Stopp-Kommando angehalten werden.

HINWEIS

Aufruf während der Bewegung

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.

Eingänge

```
VAR_INPUT
   Execute          : BOOL;
   Options          : ST_GearOutOptions;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
Options	Zur Zeit nicht implementiert

Ausgänge

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

Done	Wird TRUE, wenn die Achse erfolgreich abgekoppelt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

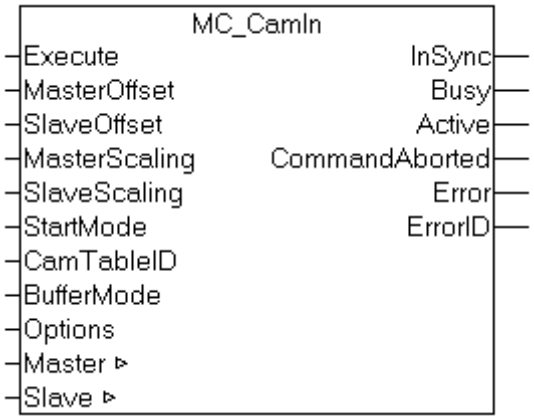
Ein/Ausgänge

```
VAR_IN_OUT
   Slave             : AXIS_REF;
END_VAR
```

Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

3.3 MC_CamIn



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

Mit dem Status-Flag Axis.Status.CamTableQueued (AXIS_REF) kann geprüft werden, ob eine Kurvenscheibe zur Umschaltung gepuffert ist.

Wichtig:

Nähere Erläuterungen zum Koppeln mit Kurvenscheiben [▶ 16]

ActivationMode [▶ 51] (Ankoppeln oder Umschalten von Kurvenscheiben)

StartMode [▶ 66]

ScalingMode [▶ 55]

Eingänge

```
VAR_INPUT
   Execute          : BOOL;
   MasterOffset     : LREAL;
   SlaveOffset      : LREAL;
   MasterScaling    : LREAL := 1.0;
   SlaveScaling     : LREAL := 1.0;
   StartMode        : MC_StartMode;
   CamTableID       : MC_CAM_ID;
   BufferMode       : MC_BufferMode;
   Options          : ST_CamInOptions;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
MasterOffset	Offset zur Masterposition der Kurvenscheibe
SlaveOffset	Offset zur Slaveposition aus der Kurvenscheibe
MasterScaling	Skalierung der Masterposition der Kurvenscheibe
SlaveScaling	Skalierung der Slaveposition der Kurvenscheibe
StartMode	StartMode [▶ 66] 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 [▶ 49] der Kurvenscheibe mit der gekoppelt wird
BufferMode	Zur Zeit nicht implementiert
Options	Datenstruktur [▶ 67] mit weiteren Koppel- und Umschaltoptionen:

	ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Kurvenscheibenkopplung bzw. -umschaltung stattfinden soll. Auch beim erstmaligen Koppeln eines Slaves kann ein *ActivationMode* angegeben werden.
	ActivationPosition	optionale Masterposition, an der abhängig vom *ActivationMode* eine Kurvenscheibe umgeschaltet wird. (Nicht notwendig bei erstmaliger Kopplung.) Falls der *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
	MasterScalingMode	optionaler Skalierungsmodus [▶ 55] für die Masterposition der Kurvenscheibe
	SlaveScalingMode	optionaler Skalierungsmodus [▶ 55] für die Slaveposition der Kurvenscheibe
	InterpolationType	Interpolationstyp [▶ 59] für Positionstabellen. Nicht notwendig für Motionfunctions.

Ausgänge

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

InSync	Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde und die Kurvenscheibe aktiv ist.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *InSync*, *CommandAborted* oder *Error* gesetzt.
Active	Active zeigt an, dass das Kommando ausgeführt wird. Bei einer Kurvenscheibenumschaltung wird *Active* TRUE, wenn das Koppelkommando erfolgreich ausgeführt wurde aber die Kurvenscheibe noch gepuffert ist. Wenn die Kurvenscheibe abhängig vom *ActivationMode* aktiviert wird, so wird *Active* FALSE und *InSync* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

```
VAR_IN_OUT
   Master           : AXIS_REF;
   Slave            : AXIS_REF;
END_VAR
```

Master	Achsdatenstruktur des Masters.
Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

3.4 MC_CamIn Anhang

Ankoppeln mit Kurvenscheiben

Mit dem Funktionsbaustein MC_CamIn [▶ 13] 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.

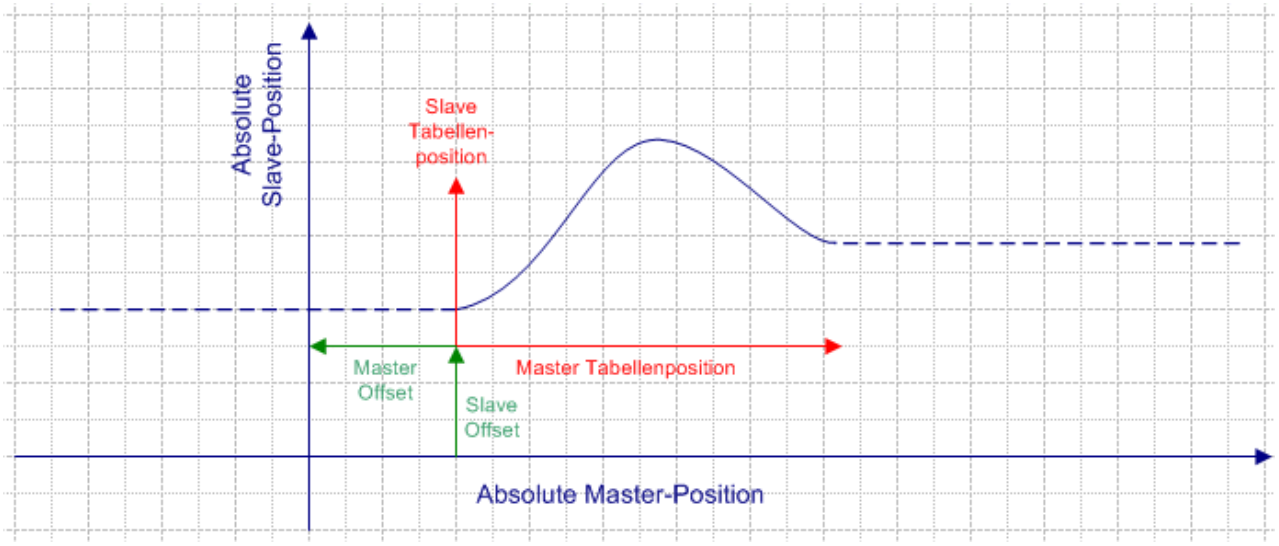
In der Praxis stellt sich die Frage, welche Position der Slave vor der Kopplung einnehmen muss und wie diese berechnet wird. Die folgenden Abbildungen veranschaulichen, wie hier vorgegangen werden kann.

HINWEIS

- 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 [▶ 41] 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.

MasterCamTablePosition := (MasterPosition + MasterOffset) / MasterScaling;

Die Master-Tabellenposition ist Eingangsparameter für den Funktionsbaustein MC_ReadCamTableSlaveDynamics [▶ 41]. Das Ergebnis wird gegebenenfalls mit Slave-Offset und -Skalierung auf eine absolute Slave-Position umgerechnet.

SlaveCamTablePosition := ReadSlaveDynamics.SlavePosition;

SlavePosition := (SlaveCamTablePosition * SlaveScaling) + SlaveOffset;

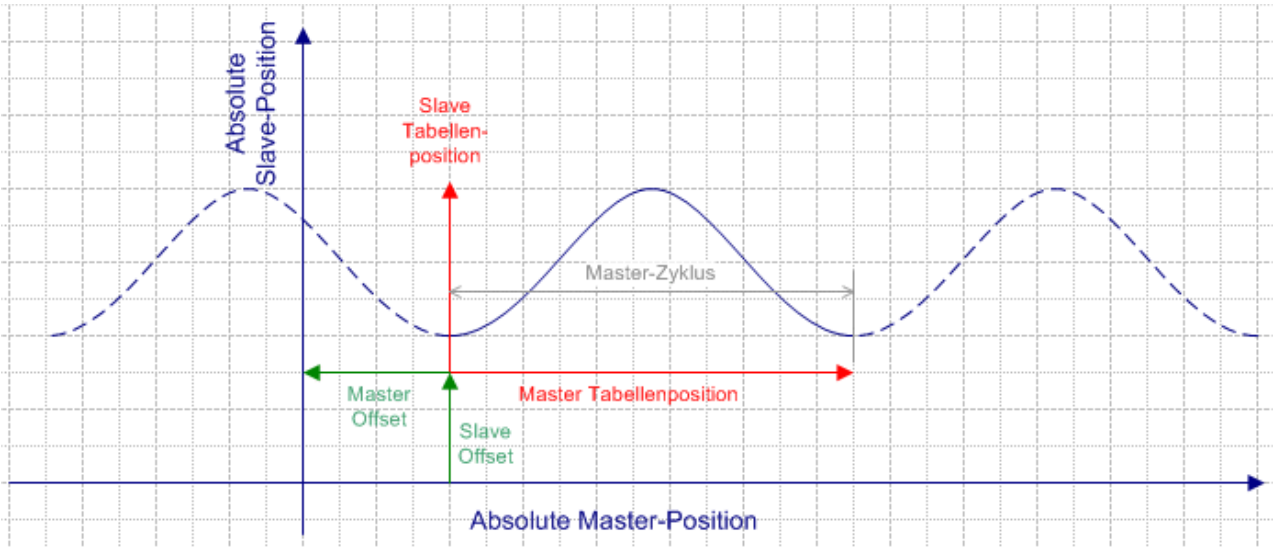
Der Slave wird vor der Kopplung an diese Position gefahren. Alternativ kann auch der Master zu einer Position gefahren werden, die mit der aktuellen Slave-Position korrespondiert. Es ist aber nicht allgemein möglich, diese Position aus der Kurvenscheibe zu ermitteln, da die Kurvenscheibe bei dieser Betrachtung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

HINWEIS

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

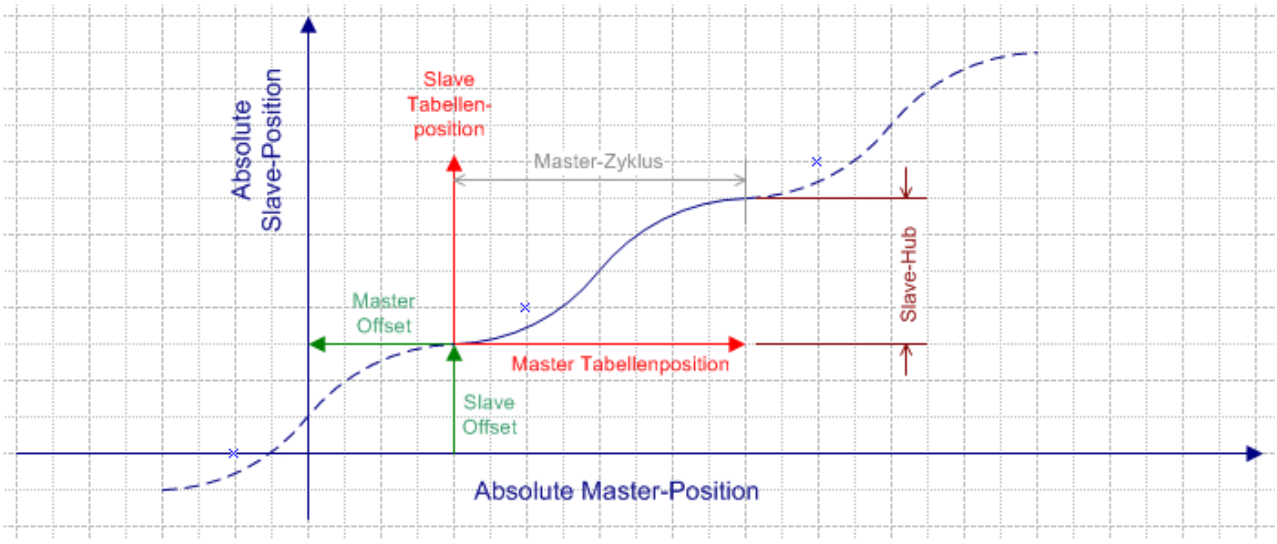
Eine zyklische Kurvenscheibe ohne Hub zeichnet sich dadurch aus, dass die Slave-Anfangs- und die Endposition in der Tabelle identisch sind. Der Slave bewegt sich dadurch zyklisch in einem definierten Bereich, ohne seine Position stetig in eine Richtung zu verändern.



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

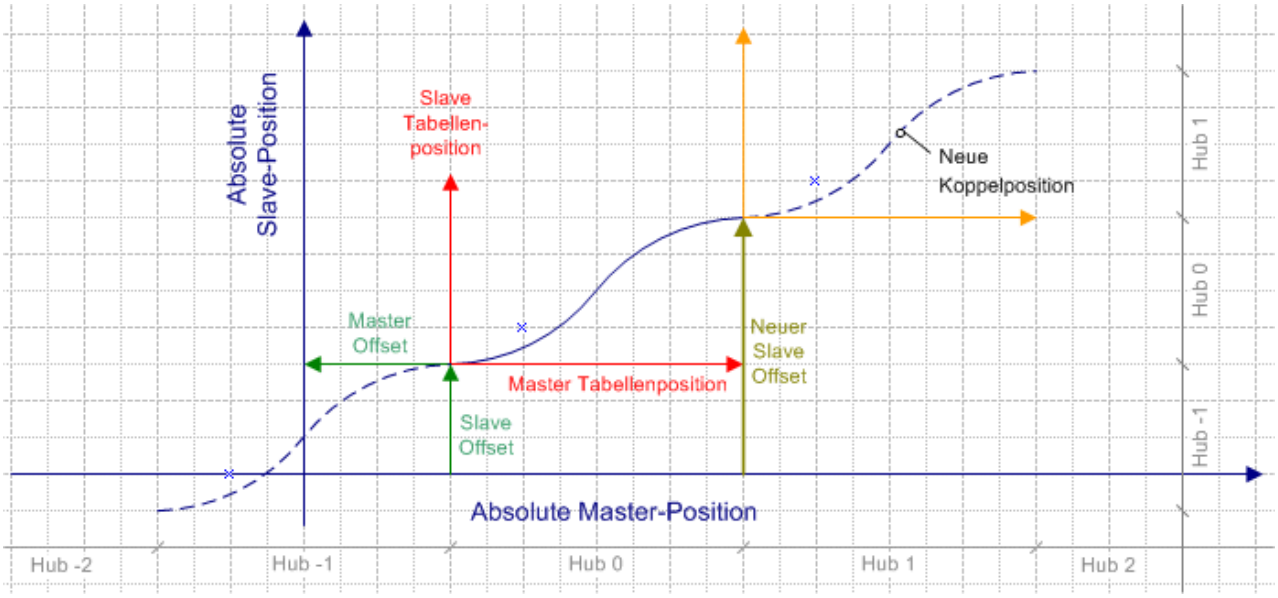
Der Hub einer zyklischen Kurvenscheibe ist die Differenz zwischen der letzten und der ersten Tabellenposition des Slaves.



Eine solchen 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), das heißt ohne aufaddierte Hübe, 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 Neuberechnung des Slave-Offsets zu berücksichtigen und auszugleichen.



MasterCamTablePos := (MasterPosition + MasterOffset) / MasterScaling;

Die Master-Tabellenposition ist Eingangsparameter für den Funktionsbaustein MC_ReadCamTableSlaveDynamics [▶ 41]. 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.

SlaveCamTablePosition := ReadSlaveDynamics.SlavePosition;

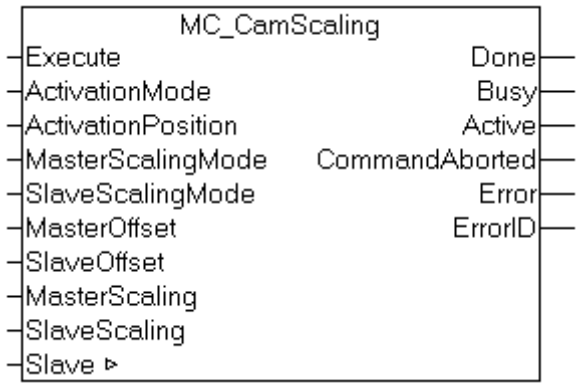
Hubanzahl := MODTURNS((SlavePosition - SlaveOffset), SlaveHub);

NewSlaveOffset := SlaveOffset + (SlaveHub * Hubanzahl);

SlavePosition := (SlaveCamTablePosition * SlaveScaling) + NewSlaveOffset;

Die Autooffset [▶ 55]-Funktion kann die Berechnung von Offsets insbesondere beim Umschalten von Kurvenscheiben vereinfachen.

3.5 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.

Mit dem Status-Flag Axis.Status.CamScalingPending (AXIS_REF) kann geprüft werden, ob eine Skalierung gepuffert ist.

Eingänge

```
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 einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Skalierung stattfinden soll.
ActivationPosition	Masterposition, an der abhängig vom ActivationMode [▶ 51] eine Kurvenscheibe skaliert wird. Falls der *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
MasterScalingMode	optionaler Skalierungsmodus [▶ 55] für die Masterposition der Kurvenscheibe
SlaveScalingMode	optionaler Skalierungsmodus [▶ 55] für die Slaveposition der Kurvenscheibe
MasterOffset	Offset zur Masterposition der Kurvenscheibe

SlaveOffset	Offset zur Slaveposition der Kurvenscheibe
MasterScaling	Skalierung der Masterposition der Kurvenscheibe
SlaveScaling	Skalierung der Slaveposition der Kurvenscheibe

Ausgänge

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

Done	Wird TRUE, wenn die Skalierung erfolgreich durchgeführt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done*, *CommandAborted* oder *Error* gesetzt.
Active	Active zeigt an, dass das Kommando ausgeführt wird. Sobald die Skalierung abhängig vom *ActivationMode* ausgeführt wurde, so wird *Active* FALSE und *Done* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

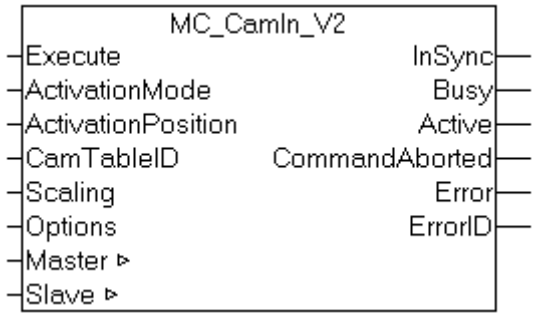
```
VAR_IN_OUT
   Slave                : AXIS_REF;
END_VAR
```

Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

4 Multi-Kurvenscheiben

4.1 MC_CamIn_V2



MC_CamIn_V2 ist eine Weiterentwicklung des Funktionsbausteins MC_CamIn [▶ 13] und ist in der Lage, mit mehren überlagerten Kurvenscheiben zu arbeiten (Multi-Cam). MC_CamIn_V2 stellt bei einem ersten Aufruf eine Master-Slave-Kopplung mit einer Kurvenscheibe her. Mit weiteren Aufrufen können zur Laufzeit bei gleicher Slave-Achse zusätzliche Kurvenscheiben additiv überlagert oder wieder aus dem Verbund entfernt werden. Dabei können die Regeln für die Umschaltung, insbesondere der genaue Zeitpunkt oder die Position bestimmt werden.

MC_CamIn_V2 kann nur alternativ zu *MC_CamIn* verwendet werden. Beide Funktionsbausteine können bei derselben Slave-Achse nicht gemeinsam genutzt werden. Für Addition, Austausch und Entfernen von Kurvenscheiben stehen alternativ die Funktionsbausteine MC_CamAdd [▶ 25], MC_CamExchange [▶ 27] und MC_CamRemove [▶ 29] zur Verfügung. Alle Operationen können aber auch mit *MC_CamIn_V2* durchgeführt werden.

Mit dem Status-Flag Axis.Status.CamTableQueued (AXIS_REF) kann geprüft werden, ob eine Kurvenscheibe zur Addition oder Umschaltung gepuffert ist.

MC_CamIn_V2 ist mit einem Laufzeitsystem ab Version TwinCAT 2.11 R2 einsetzbar.

Wichtig:

ActivationMode [▶ 51] (Zeitpunkt bzw. Position ab der eine Operation durchgeführt wird)

CamOperationMode [▶ 68] (Addieren, Umschalten oder Entfernen von überlagerten Kurvenscheiben)

ScalingMode [▶ 55]

Eingänge

```
VAR_INPUT
   Execute              : BOOL;
   ActivationMode       : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition   : LREAL;
   CamTableID           : MC_CAM_ID;
   Scaling              : ST_CamScalingData;
   Options              : ST_CamInOptions_V2;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Kurvenscheibenkopplung bzw. -umschaltung stattfinden soll. Auch beim erstmaligen Koppeln eines Slaves kann ein ActivationMode [▶ 51] angegeben werden.
ActivationPosition	optionale Masterposition, an der abhängig vom ActivationMode [▶ 51] eine Kurvenscheibe umgeschaltet wird. (Nicht notwendig bei erstmaliger Kopplung.)

	Falls der ActivationMode [▶ 51] MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.	
CamTableID	ID [▶ 49] der Kurvenscheibe mit der gekoppelt wird	
Scaling	Optionale Skalierungs-Parameter [▶ 69] für die Kurvenscheibe	
Options	Datenstruktur mit weiteren Koppel- und Umschaltoptionen:	
	InterpolationType	Interpolationstyp [▶ 59] für Positionstabellen. Nicht notwendig für Motionfunctions.
	CamOperationMode	Der CamOperationMode [▶ 68] legt fest wie die angegebene Kurvenscheibe (*CamTableID*) im den Koppelverbund wirken soll. Kurvenscheiben können addiert, getauscht oder entfernt werden.
	ReferenceCamTableID	Optionale ID [▶ 49] einer Kurvenscheibe, die bereits in der Kopplung aktiv ist. Diese ID wird insbesondere bei nicht eindeutigen Operationen benötigt, wie z. B. beim Austauschen bestimmter Kurvenscheiben bei Multi-Kopplungen. Bei eindeutigen Operationen kann der Wert 0 bleiben.

Ausgänge

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

InSync	Wird TRUE, wenn die Kurvenscheiben-Operation erfolgreich durchgeführt wurde. Bei Operationen mit Aktivierungsposition wird InSync erst nach der tatsächlichen Aktivierung TRUE.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *InSync*, *CommandAborted* oder *Error* gesetzt.
Active	Active zeigt an, dass das Kommando ausgeführt wird. *Active* wird TRUE, wenn das Kommando erfolgreich abgesetzt wurde aber die Operation noch gepuffert ist. Wenn die Kurvenscheibe abhängig vom *ActivationMode* aktiviert wird, so wird *Active* FALSE und *InSync* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

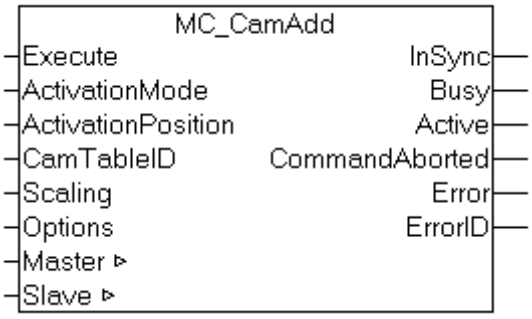
Ein/Ausgänge

```
VAR_IN_OUT
   Master               : AXIS_REF;
   Slave                : AXIS_REF;
END_VAR
```

Master	Achsdatenstruktur des Masters.
Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

4.2 MC_CamAdd



MC_CamAdd fügt eine Kurvenscheibe additiv zu einer Multi-Cam Kopplung hinzu. Die Kurvenscheibenkopplung wird anfangs mit MC_CamIn_V2 [▶ 22] hergestellt.

Die Addition einer Kurvenscheibe kann alternativ auch mit MC_CamIn_V2 durchgeführt werden.

Mit dem Status-Flag Axis.Status.CamTableQueued (AXIS_REF) kann geprüft werden, ob eine Kurvenscheibe zur Addition oder Umschaltung gepuffert ist.

MC_CamAdd ist mit einem Laufzeitsystem ab Version TwinCAT 2.11 R2 einsetzbar.

Wichtig:
ActivationMode [▶ 51] (Zeitpunkt bzw. Position ab der eine Operation durchgeführt wird)

CamOperationMode [▶ 68] (Addieren, Umschalten oder Entfernen von überlagerten Kurvenscheiben)

ScalingMode [▶ 55]

Eingänge

```
VAR_INPUT
   Execute              : BOOL;
   ActivationMode       : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition   : LREAL;
   CamTableID           : MC_CAM_ID;
   Scaling              : ST_CamScalingData;
   Options              : ST_CamInOptions_V2;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.	
ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Kurvenscheibenkopplung bzw. -umschaltung stattfinden soll. Auch beim erstmaligen Koppeln eines Slaves kann ein ActivationMode [▶ 51] angegeben werden.	
ActivationPosition	optionale Masterposition, an der abhängig vom ActivationMode [▶ 51] eine Kurvenscheibe umgeschaltet wird. (Nicht notwendig bei erstmaliger Kopplung.) Falls der ActivationMode [▶ 51] MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.	
CamTableID	ID [▶ 49] der Kurvenscheibe mit der gekoppelt wird	
Scaling	Optionale Skalierungs-Parameter [▶ 69] für die Kurvenscheibe	
Options	Datenstruktur mit weiteren Koppel- und Umschaltoptionen:	
	InterpolationType	Interpolationstyp [▶ 59] für Positionstabellen. Nicht notwendig für Motionfunctions.

	CamOperationMode	Der CamOperationMode [▶ 68] ist bei *MC_CamAdd* fest mit dem Modus *CAMOPERATIONMODE_ADDITIVE* vorbelegt.
	ReferenceCamTableID	Optionale ID [▶ 49] einer Kurvenscheibe, die bereits in der Kopplung aktiv ist. Diese ID wird nur bei sonst nicht eindeutigen Operationen benötigt, wie z. B. bei einer automatischen Offset-Justierung bezogen auf eine schon existierende Kurvenscheibe (Master-AutoOffset).

Ausgänge

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

InSync	Wird TRUE, wenn die Kurvenscheiben-Operation erfolgreich durchgeführt wurde. Bei Operationen mit Aktivierungsposition wird InSync erst nach der tatsächlichen Aktivierung TRUE.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *InSync*, *CommandAborted* oder *Error* gesetzt.
Active	Active zeigt an, dass das Kommando ausgeführt wird. *Active* wird TRUE, wenn das Kommando erfolgreich abgesetzt wurde aber die Operation noch gepuffert ist. Wenn die Kurvenscheibe abhängig vom *ActivationMode* aktiviert wird, so wird *Active* FALSE und *InSync* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

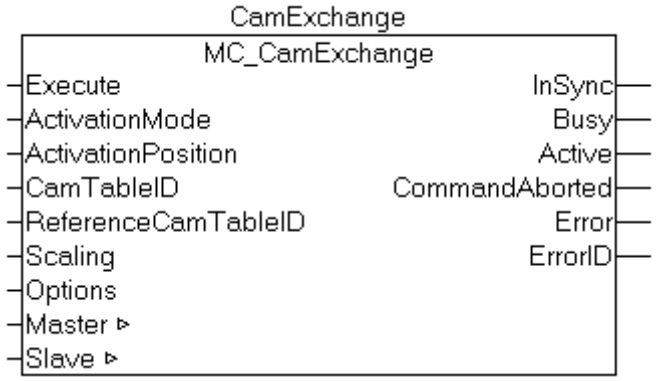
Ein/Ausgänge

```
VAR_IN_OUT
   Master               : AXIS_REF;
   Slave                : AXIS_REF;
END_VAR
```

Master	Achsdatenstruktur des Masters.
Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

4.3 MC_CamExchange



MC_CamExchange tauscht eine Kurvenscheibe in einer Multi-Cam Kopplung aus. Die Kurvenscheibenkopplung wird anfangs mit MC_CamIn_V2 [▶ 22] hergestellt.

Der Tausch einer Kurvenscheibe kann alternativ auch mit MC_CamIn_V2 durchgeführt werden.

Mit dem Status-Flag Axis.Status.CamTableQueued (AXIS_REF) kann geprüft werden, ob eine Kurvenscheibe zur Addition oder Umschaltung gepuffert ist.

MC_CamExchange ist mit einem Laufzeitsystem ab Version TwinCAT 2.11 R2 einsetzbar.

Wichtig:

ActivationMode [▶ 51] (Zeitpunkt bzw. Position ab der eine Operation durchgeführt wird)

CamOperationMode [▶ 68] (Addieren, Umschalten oder Entfernen von überlagerten Kurvenscheiben)

ScalingMode [▶ 55]

Eingänge

```
VAR_INPUT
   Execute              : BOOL;
   ActivationMode       : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition   : LREAL;
   CamTableID           : MC_CAM_ID;
   ReferenceCamTableID  : MC_CAM_ID;
   Scaling              : ST_CamScalingData;
   Options              : ST_CamInOptions_V2;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Kurvenscheibenkopplung bzw. -umschaltung stattfinden soll. Auch beim erstmaligen Koppeln eines Slaves kann ein ActivationMode [▶ 51] angegeben werden.
ActivationPosition	optionale Masterposition, an der abhängig vom ActivationMode [▶ 51] eine Kurvenscheibe umgeschaltet wird. (Nicht notwendig bei erstmaliger Kopplung.) Falls der ActivationMode [▶ 51] MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
CamTableID	ID [▶ 49] der Kurvenscheibe mit der gekoppelt wird

ReferenceCamTableID	Optionale ID einer Kurvenscheibe, die bereits in der Kopplung aktiv ist. Diese ID wird insbesondere bei nicht eindeutigen Operationen benötigt, wie z. B. beim Austauschen bestimmter Kurvenscheiben bei Multi-Kopplungen. Bei eindeutigen Operationen kann der Wert 0 bleiben.	
Scaling	Optionale Skalierungs-Parameter [▶ 69] für die Kurvenscheibe	
Options	Datenstruktur mit weiteren Koppel- und Umschaltoptionen:	
	InterpolationType	Interpolationstyp [▶ 59] für Positionstabellen. Nicht notwendig für Motionfunctions.
	CamOperationMode	Der CamOperationMode [▶ 68] ist bei *MC_CamExchange* fest mit dem Modus *CAMOPERATIONMODE_EXCHANGE* vorbelegt.
	ReferenceCamTableID	ist fest mit dem Wert des Eingangs ReferenceCamTableID vorbelegt

Ausgänge

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

InSync	Wird TRUE, wenn die Kurvenscheiben-Operation erfolgreich durchgeführt wurde. Bei Operationen mit Aktivierungsposition wird InSync erst nach der tatsächlichen Aktivierung TRUE.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *InSync*, *CommandAborted* oder *Error* gesetzt.
Active	zeigt an, dass das Kommando ausgeführt wird. *Active* wird TRUE, wenn das Kommando erfolgreich abgesetzt wurde aber die Operation noch gepuffert ist. Wenn die Kurvenscheibe abhängig vom ActivationMode [▶ 51] aktiviert wird, so wird *Active* FALSE und *nSync* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

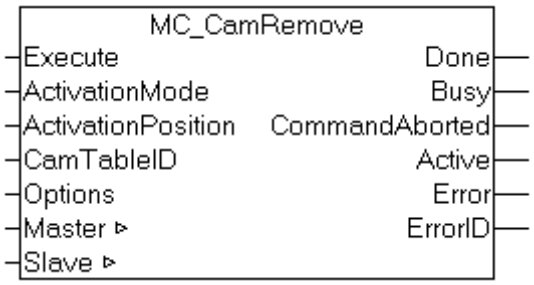
Ein/Ausgänge

```
VAR_IN_OUT
   Master               : AXIS_REF;
   Slave                : AXIS_REF;
END_VAR
```

Master	Achsdatenstruktur des Masters.
Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

4.4 MC_CamRemove



MC_CamRemove entfernt eine Kurvenscheibe aus einer Multi-Cam-Umgebung. Siehe auch MC_CamIn_V2 [▶ 22].

MC_CamRemove ist mit einem Laufzeitsystem ab Version TwinCAT 2.11 R2 einsetzbar.

Wichtig:

ActivationMode [▶ 51] (Zeitpunkt bzw. Position ab der eine Operation durchgeführt wird)

Eingänge

```
VAR_INPUT
   Execute              : BOOL;
   ActivationMode       : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition   : LREAL;
   CamTableID           : MC_CAM_ID;
   Options              : ST_CamInOptions_V2;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Kurvenscheibenkopplung bzw. -umschaltung stattfinden soll. Auch beim erstmaligen Koppeln eines Slaves kann ein ActivationMode [▶ 51] angegeben werden.
ActivationPosition	optionale Masterposition, an der abhängig vom ActivationMode [▶ 51] eine Kurvenscheibe umgeschaltet wird. (Nicht notwendig bei erstmaliger Kopplung.) Falls der ActivationMode [▶ 51] *MC_CAMACTIVATION_ATMASTERCAMPOS* verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
CamTableID	ID [▶ 49] der Kurvenscheibe, die aus dem Koppelverbund entfernt wird.
Options	nicht verwendet

Ausgänge

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

Done	Wird TRUE, wenn die Kurvenscheiben-Operation erfolgreich durchgeführt wurde. Bei Operationen mit Aktivierungsposition wird Done erst nach der tatsächlichen Deaktivierung TRUE.

Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *InSync*, *CommandAborted* oder *Error* gesetzt.
Active	Active zeigt an, dass das Kommando ausgeführt wird. *Active* wird TRUE, wenn das Kommando erfolgreich abgesetzt wurde aber die Operation noch gepuffert ist. Wenn die Kurvenscheibe abhängig vom ActivationMode [▶ 51] aktiviert wird, so wird *Active* FALSE und *InSync* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

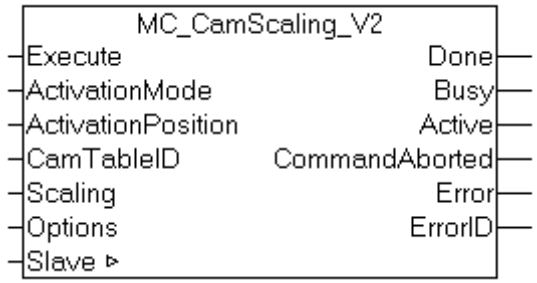
Ein/Ausgänge

```
VAR_IN_OUT
   Master               : AXIS_REF;
   Slave                : AXIS_REF;
END_VAR
```

Master	Achsdatenstruktur des Masters.
Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

4.5 MC_CamScaling_V2



Mit dem Funktionsbaustein *MC_CamScaling_V2* 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.

Mit dem Status-Flag Axis.Status.CamScalingPending (AXIS_REF) kann geprüft werden, ob eine Skalierung gepuffert ist.

MC_CamScaling_V2 ist mit einem Laufzeitsystem ab Version TwinCAT 2.11 R2 einsetzbar.

Eingänge

```
VAR_INPUT
   InSync               : BOOL;
   Execute              : BOOL;
   ActivationMode       : MC_CamActivationMode;
   ActivationPosition   : LREAL;
   CamTableID           : MC_CAM_ID;
   Scaling              : ST_CamScalingData;
   Options              : ST_CamScalingOptions_V2;
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
ActivationMode	Mit dem ActivationMode [▶ 51] wird der Zeitpunkt bzw. die Position festgelegt, an der die Skalierung stattfinden soll.
ActivationPosition	Masterposition, an der abhängig vom ActivationMode [▶ 51] eine Kurvenscheibe skaliert wird. Falls der ActivationMode [▶ 51] MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
CamTableID	ID [▶ 49] der Kurvenscheibe, die skaliert wird.
Scaling	Skalierungsdaten wie Modus, Offset und Skalierungsfaktor
Options	nicht verwendet

Ausgänge

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

Done	Wird TRUE, wenn die Skalierung erfolgreich durchgeführt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done*, *CommandAborted* oder *Error* gesetzt.
Active	Active zeigt an, dass das Kommando ausgeführt wird. Sobald die Skalierung abhängig vom ActivationMode [▶ 51] ausgeführt wurde, so wird *Active* FALSE und *Done* wird gesetzt.
CommandAborted	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

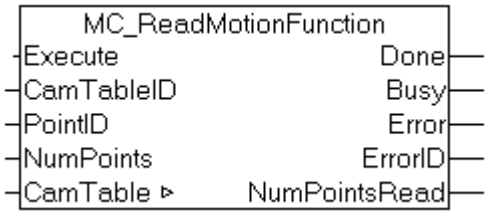
```
VAR_IN_OUT
   Slave                : AXIS_REF;
END_VAR
```

Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

5 Motion Functions

5.1 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 [▶ 50] beschriebenen Struktur in der SPS abgelegt.

Eingänge

```
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 [▶ 49] der geladenen Tabelle.
PointID	Punkt-ID [▶ 61] 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.

Ausgänge

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

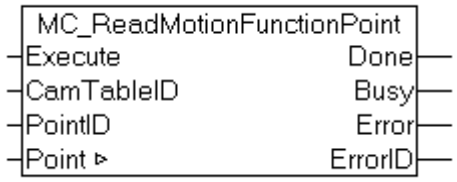
Done	Wird TRUE, wenn die Daten erfolgreich gelesen wurden.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
NumPointsRead	Die Anzahl der tatsächlich gelesenen Punkte. Die Anzahl kann kleiner oder gleich *NumPoints* sein.

Ein/Ausgänge

```
VAR_IN_OUT
   CamTable             : MC_CAM_REF;
END_VAR
```

CamTable	Referenz [▶ 50] auf die Tabelle (Struktur).

5.2 MC_ReadMotionFunctionPoint



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

Eingänge

```
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 [▶ 49] der geladenen Tabelle.
PointID	Punkt-ID [▶ 61] des ersten zu lesenden Punktes der Motion Function.

Ausgänge

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

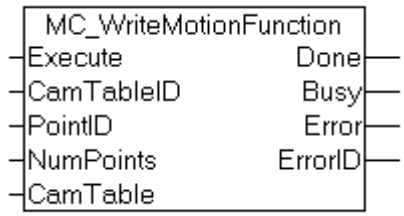
Done	Wird TRUE, wenn die Daten erfolgreich gelesen wurden.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

```
VAR_IN_OUT
   Point                : MC_MotionFunctionPoint;
END_VAR
```

Point	Datenstruktur [▶ 60] mit den Daten einer Stützstelle einer Motion Function

5.3 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 [▶ 50] beschriebenen Struktur in der SPS abgelegt.

Mit dem Funktionsbaustein MC_SetCamOnlineChangeMode [▶ 37] 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 dem Status-Flag Axis.Status.CamDataQueued (AXIS_REF)kann geprüft werden, ob Daten gepuffert sind, das heißt geschrieben aber noch nicht aktiviert wurden.

Eingänge

```
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 [▶ 49] der geladenen Tabelle.
PointID	Punkt-ID [▶ 61] des ersten zu schreibenden Punktes der Motion Function.
NumPoints	Anzahl der zu schreibenden Punkte der Motion Function.

Ausgänge

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

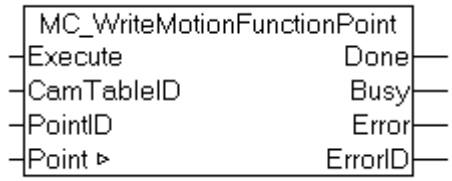
Done	Wird TRUE, wenn die Daten erfolgreich gelesen wurden.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

```
VAR_IN_OUT
   CamTable             : MC_CAM_REF;
END_VAR
```

CamTable	Referenz [▶ 50] auf die Tabelle (Struktur). Die Startadresse der Tabellen-Datenstruktur (CamTable.pArray) zeigt auf den ersten Punkt, der geschrieben werden soll.

5.4 MC_WriteMotionFunctionPoint



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

Mit dem Funktionsbaustein MC_SetCamOnlineChangeMode [▶ 37] 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 dem Status-Flag Axis.Status.CamDataQueued (AXIS_REF) kann geprüft werden, ob Daten gepuffert sind, das heißt geschrieben aber noch nicht aktiviert wurden.

Eingänge

```
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 [▶ 49] der geladenen Tabelle.
PointID	Punkt-ID [▶ 61] des ersten zu schreibenden Punktes der Motion Function.

Ausgänge

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

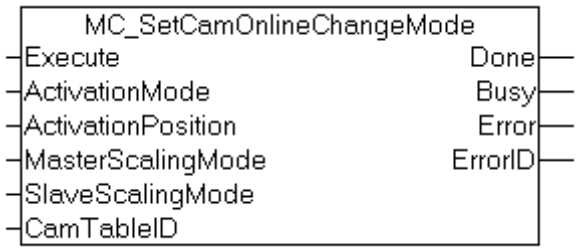
Done	Wird TRUE, wenn die Daten erfolgreich geschrieben wurden.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

```
VAR_IN_OUT
   Point                : MC_MotionFunctionPoint;
END_VAR
```

Point	Datenstruktur [▶ 60] mit den Daten einer Stützstelle einer Motion Function

5.5 MC_SetCamOnlineChangeMode



Der Funktionsbaustein *MC_SetCamOnlineChangeMode* legt den Modus für Schreibzugriffe auf Kurvenscheibendaten fest.

Kurvenscheiben können während der Laufzeit durch die SPS geändert werden (siehe MC_WriteMotionFunction [▶ 35], MC_WriteMotionFunctionPoint [▶ 36]). Mit *MC_SetCamOnlineChangeMode* wird festgelegt, wann und wie diese Änderungen übernommen werden. Der eingestellte Modus wirkt auf alle nachfolgenden Schreibvorgänge. Es ist also nicht notwendig den Baustein vor jedem Schreibzugriff erneut aufzurufen.

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

Eingänge

```
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 [▶ 51])
ActivationPosition	Optionale Master-Position, an der die Skalierung durchgeführt wird (je nach ActivationMode [▶ 51]). Falls der ActivationMode [▶ 51] MC_CAMACTIVATION_ATMASTERCAMPOS verwendet wird, bezieht sich die Position auf die unskalierte Kurvenscheibe. Wenn sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
MasterScalingMode	Art der der Master-Skalierung. (MC_CamScalingMode [▶ 55])
SlaveScalingMode	Art der der Slave-Skalierung. (MC_CamScalingMode [▶ 55])
CamTableID	Tabellen ID [▶ 49].

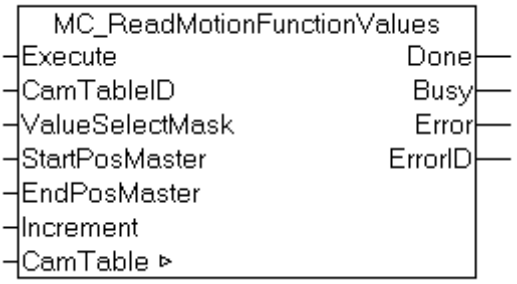
Ausgänge

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

Done	Wird TRUE, wenn die Funktion erfolgreich durchgeführt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.

Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

5.6 MC_ReadMotionFunctionValues



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

Diese Funktion kann beispielsweise zur Visualisierung einer Motion Function verwendet werden. Die gesamte Kurve wird mit parametrierbarer Schrittweite digitalisiert. Die ermittelten Daten lassen sich einfacher zur Anzeige bringen als eine Motion Function.

Eingänge

```
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 [▶ 49] der geladenen Tabelle (Typ Motion Function).
ValueSelectMask	Mit der Auswahlmaske kann festgelegt werden, welche Daten interpoliert und zurückgeliefert werden soll. Der Wert entsteht durch Addition von Einzelwerten des Datentyps MC_ValueSelectType [▶ 65]. Die Spaltenzahl der durch CamTable [▶ 50] beschriebenen Datenstruktur muss der Anzahl der durch ValueSelectMask [▶ 65] 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

Ausgänge

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

Done	Wird TRUE, wenn die Daten erfolgreich gelesen wurden.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

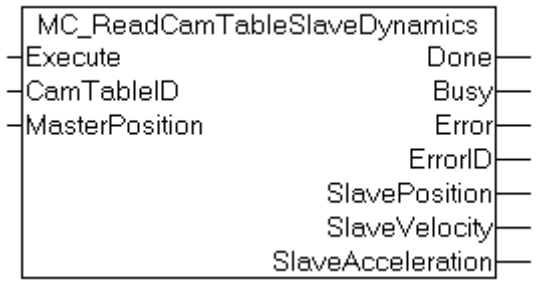
Ein/Ausgänge

```
VAR_IN_OUT
   CamTable             : MC_CAM_REF;
END_VAR
```

CamTable	Referenz [▶ 50] auf die Tabelle (Struktur).

6 Status

6.1 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 [▶ 65] können nicht alle Dynamikparameter ermittelt werden. Die folgende Übersicht zeigt, welches Ergebnis zu erwarten ist:

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

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

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

Eingänge

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

Execute	Mit der steigenden Flanke wird das Kommando ausgeführt.
CamTableID	Tabellen ID [▶ 49].
MasterPosition	Master-Position innerhalb der Tabelle zu der die Slave-Dynamik ermittelt werden soll.

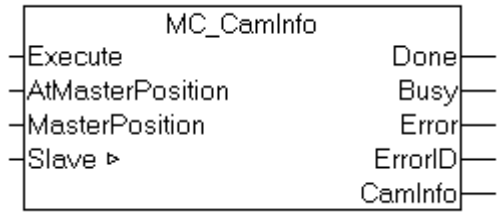
Ausgänge

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

Done	Wird TRUE, wenn das Kommando erfolgreich ausgeführt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
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.

6.2 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.

HINWEIS

Wenn die gekoppelte Achsgruppe in eine Fehlersituation gerät (z. B. Not-Halt), 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.

Eingänge

```
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.

Ausgänge

```
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	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
CamInfo	Die Datenstruktur CamInfo [▶ 58] enthält alle ermittelten Daten der Kurvenscheibenkopplung

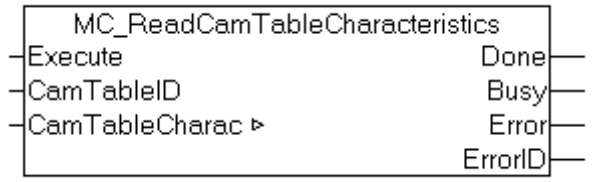
Ein/Ausgänge

```
VAR_IN_OUT
   Slave                : AXIS_REF;
END_VAR
```

Slave	Achsdatenstruktur des Slaves.

Die Achsdatenstruktur vom Typ AXIS_REF adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

6.3 MC_ReadCamTableCharacteristics



Mit dem Funktionsbaustein *MC_ReadCamTableCharacteristics* werden die charakteristischen Kenngrößen einer Motion Function berechnet und ausgelesen. Dazu gehören beispielsweise die Minimal- und Maximalwerte von Position, Geschwindigkeit, Beschleunigung und Ruck.

Eingänge

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

Execute	Mit der steigenden Flanke wird das Kommando ausgeführt.
CamTableID	Tabellen ID [▶ 49]

Ausgänge

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

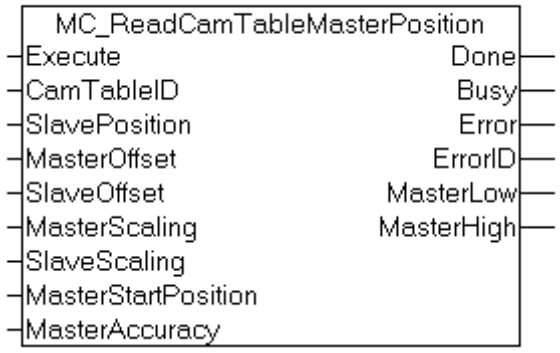
Done	Wird TRUE, wenn die Berechnung erfolgreich durchgeführt wurde.
Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Ein/Ausgänge

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

CamTableCharac	Datenstruktur [▶ 64] mit charakteristischen Kenngrößen der Motion Function

6.4 MC_ReadCamTableMasterPosition



Mit dem Funktionsbaustein*MC_ReadCamTableMasterPosition* kann die Masterposition für eine gegebene Slaveposition berechnet werden. Während die Slaveposition für eine gegebene Masterposition eindeutig sein muss, ist dies für den inversen Fall nicht so. Damit man nun als Ausgabe für den Funktionsbaustein eine beschränkte Anzahl von Werten für Master ausgeben kann, werden nur für eine gegebene Masterposition (*MasterStartPosition*) die jeweils kleinere (*MasterLow*) und größere Masterposition (*MasterHigh)* zu dem Slavewert ausgegeben.

So wird für die Kurvenscheibe vom Bild 1 für den Slavewert von 80 und einem Masterstartwert von 180 die Werte für *MasterHigh* von 225 und für *MasterLow* von 135 ausgegeben. Wenn die Kurvenscheibe zyklisch ist, werden für einen Masterstartwert von 90 neben dem *MasterHigh* von 135 auch der *MasterLow* von -135 berechnet. Im linearen Fall (nichtzyklisch) der Kurvenscheibe wird im Bild 2 nur der Wert *MasterHigh* als gültig dargestellt.

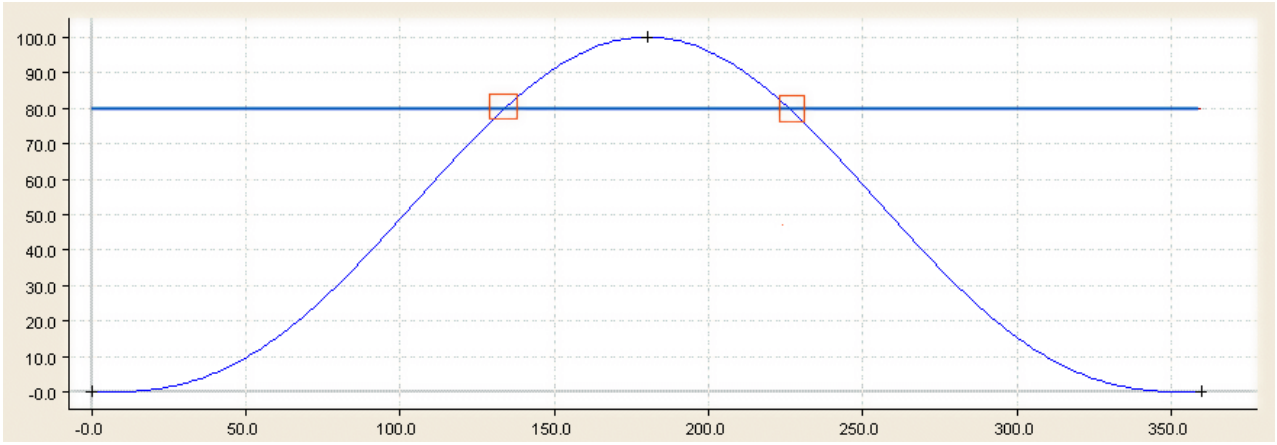


Bild1

Bei zyklischen Kurvenscheiben mit Hub kann die Masterposition nicht nur in einem der benachbarten Zyklen der StartMasterpos liegen, sondern bei entsprechender Slaveposition auch mehrere Zyklen weiter.

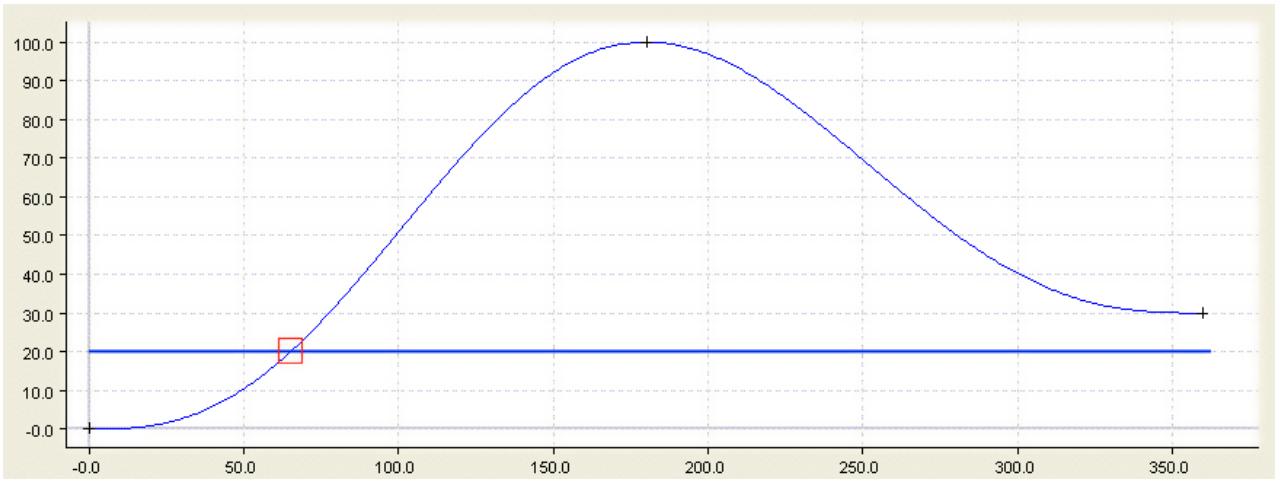


Bild2

Die Berechnung der Masterposition erfolgt mit numerischen Algorithmen, deren Genauigkeit über die Variable *MasterAccuracy* eingestellt werden kann.

Eingänge

```
VAR_INPUT
   Execute              : BOOL;
   CamTableID           : MC_CAM_ID;
   SlavePosition        : LREAL; (* absolute slave axis position *)
   MasterOffset         : LREAL; (* E *)
   SlaveOffset          : LREAL; (* E *)
   MasterScaling        : LREAL := 1.0; (* E *)
   SlaveScaling         : LREAL := 1.0; (* E *)
   MasterStartPosition  : LREAL; (* Master position to start the iteration from *)
   MasterAccuracy       : LREAL; (* Master iteration accuracy *)
END_VAR
```

Execute	Mit einer steigenden Flanke am Eingang *Execute* wird das Kommando ausgeführt.
CamTableID	ID [▶ 49] der Kurvenscheibe für die die Berechnung durchgeführt wird
SlavePosition	Die Slaveposition deren Masterposition gesucht ist
MasterOffset	Offset zur Masterposition der Kurvenscheibe
SlaveOffset	Offset zur Slaveposition aus der Kurvenscheibe
MasterScaling	Skalierung der Masterposition der Kurvenscheibe
SlaveScaling	Skalierung der Slaveposition der Kurvenscheibe
MasterStartPosition	Startposition des Masters
MasterAccuracy	Genauigkeit für die Berechnung

Ausgänge

```
VAR_OUTPUT
   Execute              : BOOL;
   Done                 : BOOL;
   Busy                 : BOOL;
   Active               : BOOL;
   Error                : BOOL;
   ErrorID              : UDINT;
   MasterLow            : ST_CamMasterData; (* position information of the lower bound master
position *)
   MasterHigh           : ST_CamMasterData; (* position information of the upper bound master
position *)
END_VAR
```

Done	Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde und die Kurvenscheibe aktiv ist.

Busy	Der *Busy*-Ausgang wird TRUE, sobald das Kommando mit *Execute* gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn *Busy* wieder FALSE wird, so ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge *Done* oder *Error* gesetzt.
Error	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
MasterLow	Masterposition kleiner als die MasterStartPosition in der Datenstruktur ST_CamMasterData [▶ 68]
MasterHigh	Masterposition kleiner als die MasterStartPosition in der Datenstruktur ST_CamMasterData [▶ 68]

7 Datentypen

7.1 Datentyp MC_CAM_ID

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

Typdefinition für die Tabellen ID.

7.2 Datentyp MC_CAM_REF

```
TYPE MC_CAM_REF :
STRUCT
   pArray               : UDINT;
   ArraySize            : UDINT;
   TableType            : MC_TableType;
   NoOfRows             : UDINT;
   NoOfColumns          : UDINT;
END_STRUCT
END_TYPE
```

Die Datenstruktur *MC_CAM_REF* beschreibt den Datenspeicher einer Kurvenscheibe in einer weiteren SPS-Variablen (ARRAY).

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

Tab. 1: Beispiel 1: Strukturbeschreibung einer Positionstabelle

pArray	Adresse eines zweidimensionalen Arrays. Die erste Spalte enthält eine aufsteigende Liste von Master-Positionen. Die zweite Spalte enthält die dazugehörigen Slave-Positionen. Die Adresse kann mit der ADR Funktion zugewiesen werden. Beispiel: Table1 : ARRAY[0..360, 0..1] OF LREAL; pArray := ADR(Table1);
ArraySize	Speichergröße des zweidimensionalen Arrays, die mit der SIZEOF Funktion ermittelt werden kann. Beispiel: ArraySize := SIZEOF(Table1);
TableType	Der Typ [▶ 65] der Tabelle ist *MC_TABLETYPE_EQUIDISTANT*, wenn die Master-Positionen einen gleichen Abstand haben Oder *MC_TABLETYPE_NONEQUIDISTANT* bei wechselndem Abstand.
NoOfRows	Die Zeilenanzahl entspricht der Anzahl der Tabellenpunkte.
NoOfColumns	Die Spaltenanzahl ist 2.

Tab. 2: Beispiel 2: Strukturbeschreibung einer Motion Function

pArray	Adresse eines eindimensionalen Arrays vom Typ MC_MotionFunctionPoint. [▶ 60] Jedes Array-Element enthält eine Beschreibung einer Stützstelle der Kurvenscheibe. Beispiel: MotionFunction : ARRAY[1..10] OF MC_MotionFunctionPoint; pArray := ADR(MotionFunction);
ArraySize	Speichergröße des eindimensionalen Arrays, die mit der SIZEOF Funktion ermittelt werden kann. Beispiel: ArraySize := SIZEOF(MotionFunction);
TableType	Der Typ der Tabelle ist *MC_TABLETYPE_MOTIONFUNCTION*.
NoOfRows	Die Zeilenanzahl entspricht der Anzahl der Tabellenpunkte.
NoOfColumns	Die Spaltenanzahl ist 1.

7.3 Datentyp MC_CamActivationMode

```
TYPE MC_CamActivationMode :
(
   (* instantaneous change *)
   MC_CAMACTIVATION_INSTANTANEOUS,

   (* modify the data at a defined master position referring to the cam tables master position *)
   MC_CAMACTIVATION_ATMASTERCAMPOS,

   (* modify the data at a defined master position referring to the absolute master axis position *)
   MC_CAMACTIVATION_ATMASTERAXISPOS

   (* modify the data at the beginning of the next cam table cycle *)
   MC_CAMACTIVATION_NEXTCYCLE,

   (* not yet implemented!
   modify the data at the beginning of the next cam table cycle, activation is valid for one cycle
only *)
   MC_CAMACTIVATION_NEXTCYCLEONCE,

   (* modify the data as soon as the cam table is in a safe state to change its data *)
   MC_CAMACTIVATION_ASSOONASPOSSIBLE,

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

   (* delete all data which was written to modify the cam table but is still not activated *)
   MC_CAMACTIVATION_DELETEQUEUEDDATA,

   (* special mode at a defined master axis position in a defined positive direction *)
   MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION,

   (* special mode at a defined master axis position in a defined negative direction *)
   MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION
);
END_TYPE
```

MC_CamActivationMode legt Zeitpunkt und Art der Änderung einer Kurvenscheibe fest. Änderungen können einerseits durch Skalierung, durch Ändern der Kurvenscheibendaten oder durch Umschalten von Kurvenscheiben durchgeführt werden.

In den einzelnen Fällen sind die folgenden Modi möglich:

Skalieren von Kurvenscheiben

Kurvenscheiben können mit dem Funktionsbaustein MC_CamScaling [▶ 20] 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. Die Position bezieht sich auf die unskalierte Kurvenscheibe. Falls sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert 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 [▶ 37] wird festgelegt, wann geänderte Kurvenscheibendaten aktiv werden (siehe auch MC_WriteMotionFunction [▶ 35] und MC_WriteMotionFunctionPoint [▶ 36]).

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. Die Position bezieht sich auf die unskalierte Kurvenscheibe. Falls sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert 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_CamIn [▶ 13] 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 [▶ 66] = MC_STARTMODE_RELATIVE oder MC_STARTMODE_MASTERREL_SLAVEABS verwendet werden. Die Position bezieht sich auf die unskalierte Kurvenscheibe. Falls sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert 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 [▶ 66] = MC_STARTMODE_RELATIVE oder MC_STARTMODE_MASTERREL_SLAVEABS verwendet werden.
MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION	Die Kurvenscheibenkopplung wird an einer definierten absoluten Position der Master-Achse durchgeführt wenn der Master die Position in positiver Fahrtrichtung überfährt. (Dieser Modus ist ein Sonderfall von MC_CAMACTIVATION_ATMASTERAXISPOS, der ein sicheres Aktivieren im unmittelbaren Nahbereich um die aktuelle Position auch bei sehr kleinen Geschwindigkeiten mit kurzzeitigen Fahrtrichtungswechseln (Rauschen) gewährleistet)
MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION	Die Kurvenscheibenkopplung wird an einer definierten absoluten Position der Master-Achse durchgeführt wenn der Master die Position in negativer Fahrtrichtung überfährt. (Dieser Modus ist ein Sonderfall von MC_CAMACTIVATION_ATMASTERAXISPOS, der ein sicheres Aktivieren im unmittelbaren Nahbereich um die aktuelle Position auch bei sehr kleinen Geschwindigkeiten mit kurzzeitigen Fahrtrichtungswechseln (Rauschen) gewährleistet)

Umschalten von Kurvenscheiben

Mit dem Funktionsbausteinen MC_CamIn [▶ 13] 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. Die Position bezieht sich auf die unskalierte Kurvenscheibe. Falls sich die Position in der Anwendung auf die skalierte Kurvenscheibe bezieht, so kann sie vor Aufruf des Funktionsbausteins durch die MasterSkalierung *MasterScaling* dividiert werden.
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.
MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION	Die Kurvenscheibenumschaltung wird an einer definierten absoluten Position der Master-Achse durchgeführt wenn der Master die Position in positiver Fahrtrichtung überfährt.

	(Dieser Modus ist ein Sonderfall von MC_CAMACTIVATION_ATMASTERAXISPOS, der ein sicheres Aktivieren im unmittelbaren Nahbereich um die aktuelle Position auch bei sehr kleinen Geschwindigkeiten mit kurzzeitigen Fahrtrichtungswechseln (Rauschen) gewährleistet)
MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION	Die Kurvenscheibenumschaltung wird an einer definierten absoluten Position der Master-Achse durchgeführt wenn der Master die Position in negativer Fahrtrichtung überfährt. (Dieser Modus ist ein Sonderfall von MC_CAMACTIVATION_ATMASTERAXISPOS, der ein sicheres Aktivieren im unmittelbaren Nahbereich um die aktuelle Position auch bei sehr kleinen Geschwindigkeiten mit kurzzeitigen Fahrtrichtungswechseln (Rauschen) gewährleistet)

7.4 Datentyp MC_CamScalingMode

```
TYPE MC_CamScalingMode :
(
   (* user defines scaling parameters -scaling and -offset *)
   MC_CAMSCALING_USERDEFINED,

   (* offset is calculated automatically for best result *)
   MC_CAMSCALING_AUTOOFFSET,

   (* no modification accepted *)
   MC_CAMSCALING_OFF
);
END_TYPE
```

Typ und Umfang der Skalierung einer Kurvenscheibenkopplung mit dem Funktionsbaustein MC_CamScaling [▶ 20].

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 ein 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 durchgeführt werden soll.

Autooffset

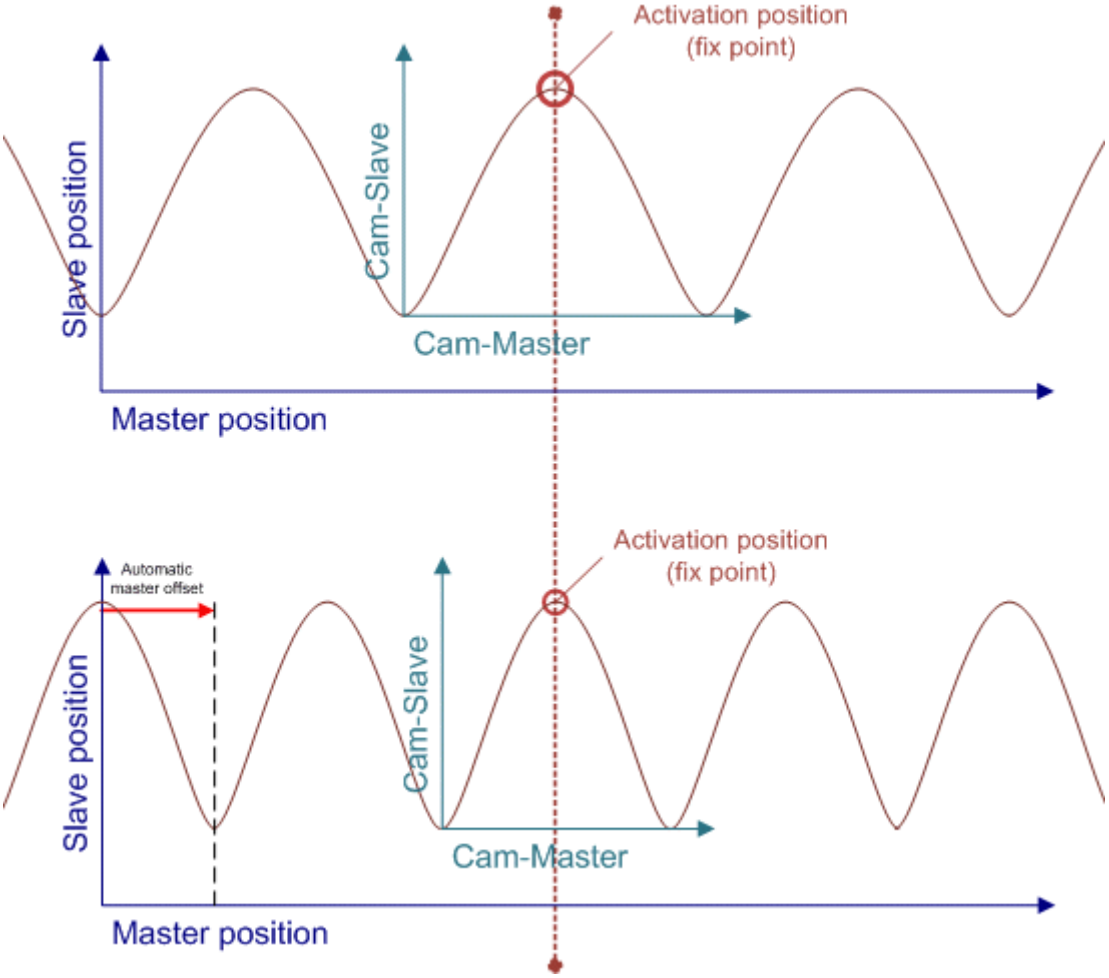
Der *Autooffset* Modus sorgt für eine automatische Anpassung eines Kurvenscheibenoffsets. *Autooffset* kann unabhängig für die Master- oder die Slave-Achse einer Kurvenscheibe angewendet werden und wirkt sowohl bei der Umschaltung als auch bei der Skalierung von Kurvenscheiben. Die Funktion arbeitet 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 setzt grundsätzlich voraus, dass bereits eine Kurvenscheibe in der angesprochenen Achskopplung als Bezug existiert und ist daher beim erstmaligen Koppeln nicht möglich. *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 [▶ 51] = *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.

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

		Switching of Cam Tables			
	Master-ScalingMode:	**without** Master-AutoOffset		**with** Master-AutoOffset	
	StartMode:	Absolute	Relative	Absolute	Relative
Activation Mode	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.

Slave-Autooffset kann in Verbindung mit einem beliebigen MC_StartMode [▶ 66] verwendet werden und justiert auf jeden Fall die Kurvenscheibe so, dass es keinen Sprung in der Slave-Position gibt.

7.5 Datentyp MC_CamInfoData

```
TYPE MC_CamInfoData :
STRUCT
   Execute                      : BOOL;
   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 pos. of the first cam table point (raw, unscaled
cam table pos.) *)
   SlaveCamStartPos             : LREAL; (* Slave pos. of the first cam table point (raw, unscaled
cam table pos.) *)
   RawMasterPeriod              : LREAL; (* raw, unscaled difference between last and first cam
point *)
   RawSlaveStroke               : LREAL; (* raw, unscaled difference between last and first cam
point *)
   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; (* raw, unscaled cam table position of the master *)
   ActualSlaveCamPos            : LREAL; (* raw, unscaled 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
```

Die Datenstruktur *MC_CamInfoData* enthält Daten zum aktuellen Zustand einer Kurvenscheibenkopplung. Die Daten werden mit dem Funktionsbaustein MC_CamInfo [▶ 43] ermittelt.

Die Struktur enthält zum einen absolute Achspositionen, die sich auf das Master- oder Slave-Achskoordinatensystem beziehen. Zum anderen sind Kurvenscheibenpositionen enthalten, die sich auf das Kurvenscheibenkoordinatensystem beziehen (z. B. *ActualMasterCamPos* und *ActualSlaveCamPos*). Alle Cam-Positionen beziehen sich auf das unskalierte Kurvenscheibenkoordinatensystem und können bei Bedarf auf das skalierte Koordinatensystem umgerechnet werden. Eine Master-Cam-Position kann mit dem Master-Skalierungsfaktor *MasterScaling* multipliziert und eine Slave-Cam-Position mit *SlaveScaling* multipliziert werden.

Die Aktivierungspositionen *ActivationPos* werden je nach *ActivationMode* auf das Master-Achskoordinatensystem oder auf das Kurvenscheibenkoordinatensystem bezogen. Im letzten Fall wird eine unskalierte Kurvenscheibenpositionen angegeben.

7.6 Datentyp MC_InterpolationType

Interpolationsmodus für Positionstabellen (Kurvenscheiben). Positionstabellen bestehen aus einer Liste von Master- und Slave-Positionen zwischen denen auf verschiedene Weise interpoliert werden kann.

Der Interpolations-Typ wird nicht bei erweiterten Kurvenscheiben (Motion Functions) angewendet.

```
TYPE MC_InterpolationType :
(
   (* linear 2 point interpolation *)
   MC_INTERPOLATIONTYPE_LINEAR             := 0,

   (* no longer supported - 4 point interpolation (for equidistant tables only) *)
   MC_INTERPOLATIONTYPE_4POINT             := 1,

   (* spline interpolation (tangential or cyclic depending on table) *)
   MC_INTERPOLATIONTYPE_SPLINE             := 2,

   (* moving cubic spline interpolation with n sampling points ('local spline') *)
   MC_INTERPOLATIONTYPE_SLIDINGSPLINE      := 3
);
END_TYPE
```

7.7 Datentyp 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 dieser Stützstelle innerhalb der Motion Function. Der Punktindex aller Stützstellen muss streng monoton steigend, lückenlos und größer 0 sein.

FunctionType: Typ MC_MotionFunctionType [▶ 62] der mathematischen Funktion zwischen dieser und der nachfolgenden Stützstelle

PointType: Typ MC_MotionPointType [▶ 63] 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

7.8 Datentyp MC_MotionFunctionPoint_ID

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

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

7.9 Datentyp 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
```

Typdefinition für Motion Functions.

Typ beachten

Der im TwinCAT Cam Design Editor verwendete Motion Function-Typ *Automatic* entspricht *MOTIONFUNCTYPE_POLYNOM5_MM* und der Motion Function-Typ *Synchron* entspricht *MOTIONFUNCTYPE_POLYNOM1*

7.10 Datentyp 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
```

Typdefinition für den Tabellen Punkt.

⚠ GEFAHR

Lebensgefahr oder Gefahr von schweren Verletzungen oder Sachschäden durch unbeabsichtigte Bewegungen der Achse

Eine Verwendung von MOTIONPOINTTYPE_IGNORE für den ersten und letzten MotionFunctionPoint einer Tabellendefinition ist nicht erlaubt und darf nicht verwendet werden.

7.11 Datentyp 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 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
```

Typdefinition für die charakteristischen Kenngrößen einer Motion Function.

7.12 Datentyp MC_TableErrorCodes

```
TYPE MC_TableErrorCodes :
(
   (* 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
```

7.13 Datentyp MC_TableType

```
TYPE MC_TableType :
(
   (* n*m tabular with equidistant ascending master values *)
   MC_TABLETYPE_EQUIDISTANT      := 10,

   (* n*m tabular with strictly monotone ascending master values (not imperative equidistant) *)
   MC_TABLETYPE_NONEQUIDISTANT   := 11,

   (* motion function calculated in runtime *)
   MC_TABLETYPE_MOTIONFUNCTION   := 22
);
END_TYPE
```

7.14 Datentyp 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
```

Typdefinition für den Zugriff auf Wertetabellen mit dem Funktionsbaustein MC_ReadMotionFunctionValues [▶ 39].

7.15 Datentyp MC_StartMode

```
TYPE MC_StartMode :
(
   MC_STARTMODE_ABSOLUTE            := 1,  (* cam table is absolute for master and slave *)
   MC_STARTMODE_RELATIVE            := 2,  (* cam table is relative for master and slave *)
   MC_STARTMODE_MASTERABS_SLAVEREL  := 3, (* cam table is absolute for master and relative for slave
*)
   MC_STARTMODE_MASTERREL_SLAVEABS  := 4  (* cam table is relative for master and absolute for slave
*)
);
END_TYPE
```

Der *StartMode* wird beim Koppeln mit Kurvenscheiben durch MC_CamIn [▶ 13] 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 [▶ 55]) verwendet werden, da es hier einen Widerspruch gibt.

7.16 Datentyp ST_CamInOptions

Daten vom Typ *ST_CamInOptions* können dem Funktionsbaustein MC_CamIn [▶ 13] optional übergeben werden.

```
TYPE ST_CamInOptions :
STRUCT
   (* ActivationMode defines when and where the cam table will be activated *)
   (* (only valid if slave is already coupled and cam table will be exchanged) *)
   ActivationMode          : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition      : LREAL;

   (* Scaling Modes enable, disable or define the way of scaling the cam table *)
   MasterScalingMode       : MC_CamScalingMode := MC_CAMSCALING_USERDEFINED;
   SlaveScalingMode        : MC_CamScalingMode := MC_CAMSCALING_USERDEFINED;

   (* InterpolationType is required for position tables only. *)
   (* MotionFunctions don't need an InterpolationType *)
   InterpolationType       : MC_InterpolationType := MC_InterpolationType_Linear;
END_STRUCT
END_TYPE
```

7.17 Datentyp CamMasterData

Daten vom Typ CamMasterData werden vom Funktionsbaustein MC_ReadCamTableMasterPosition [▶ 46] optional übergeben.

```
TYPE CamMasterData :
STRUCT
   Valid               : BOOL;  (* position information is valid *)
   MasterAxisPosition  : LREAL; (* absolute master axis position *)
   MasterCamPosition   : LREAL; (* local master cam position *)
   SlaveOffset         : LREAL; (* slave cam offset corresponding to the master position *)
END_STRUCT
END_TYPE
```

7.18 Datentyp MC_CamOperationMode

Der *CamOperationMode* wird zur Verwaltung von Kopplungen mit überlagerten Kurvenscheiben mit dem Funktionsbaustein MC_CamIn_V2 [▶ 22] verwendet (Multi-Cam). Kurvenscheiben lassen sich addieren, austauschen oder entfernen.

```
TYPE MC_CamOperationMode :
(
   CAMOPERATIONMODE_DEFAULT,  (* same as additive *)
   CAMOPERATIONMODE_ADDITIVE, (* additive cam in a multi cam scenario *)
   CAMOPERATIONMODE_EXCHANGE, (* exchange existing cam in a multi cam scenario *)
   CAMOPERATIONMODE_REMOVE    (* remove cam from a multi cam scenario *)
);
END_TYPE
```

7.19 Datentyp ST_CamScalingData

Die Struktur *ST_CamScalingData* enthält Informationen zum Skalieren einer Kurvenscheibe und wird mit dem Funktionsbaustein MC_CamIn_V2 [▶ 22] verwendet.

```
TYPE ST_CamScalingData :
STRUCT
   (* scaling of the X axis of the cam (master scaling) *)
   MasterScalingMode  : MC_CamScalingMode;
   MasterRelative     : BOOL;
   MasterOffset       : LREAL;
   MasterScaling      : LREAL := 1.0;

   (* scaling of the Y axis of the cam (slave scaling) *)
   SlaveScalingMode   : MC_CamScalingMode;
   SlaveRelative      : BOOL;
   SlaveOffset        : LREAL;
   SlaveScaling       : LREAL := 1.0;
END_STRUCT
END_TYPE
```

MasterScalingMode	Skalierungsmodus [▶ 55] für die Masterposition der Kurvenscheibe
MasterRelative	Wenn TRUE, wirkt die Kurvenscheibe relativ zur aktuellen Master-Position zum Zeitpunkt der Aktivierung.
MasterOffset	Master-Offset zur Orientierung der Kurvenscheibe im Achs-Koordinatensystem. *MasterOffset* wirkt im absoluten Modus ab der Master-Achsposition 0 und im relativen Modus ab der aktuellen Position zum Zeitpunkt der Aktivierung.
MasterScaling	Skalierung der Master-Position der Kurvenscheibe. Default ist 1.0
SlaveScalingMode	Skalierungsmodus [▶ 55] für die Slave-Position der Kurvenscheibe
SlaveRelative	Wenn TRUE, wirkt die Kurvenscheibe relativ zur aktuellen Slave-Position zum Zeitpunkt der Aktivierung.
SlaveOffset	Slave-Offset zur Orientierung der Kurvenscheibe im Achs-Koordinatensystem. *SlaveOffset* wirkt im absoluten Modus ab der Slave-Achsposition 0 und im relativen Modus ab der aktuellen Position zum Zeitpunkt der Aktivierung.
SlaveScaling	Skalierung der Slave-Position der Kurvenscheibe. Default ist 1.0

8 Beispielprogramme

Elektronische Kurvenscheiben

Das Beispielprogramm koppelt eine Master- und eine Slave-Achse mittels Kurvenscheiben. Während der gekoppelten Fahrt werden Kurvenscheiben umgeschaltet, einzelne Stützstellen einer Kurvenscheibe geändert und Skalierungen der Kurvenscheibe vorgenommen.

Das Beispielprogramm benötigt die zusätzliche Bibliothek TcMC2_Camming.lib und läuft vollständig im Simulationsmodus. Der Ablauf kann im TwinCAT Scope View mit beiliegender Konfiguration verfolgt werden.

Zum Speichern des Beispielprogramms hier klicken:

https://infosys.beckhoff.com/content/1031/tcplclibmc2_camming/Resources/460479371.zip

Rotationsschneider mit Registerausgleich (Rotary Knife and Registration

Im Beispiel wird ein rotierendes Messer verwendet, um Folienabschnitte bestimmter Länge zu schneiden. Dazu muss das Messer mit seiner Umfangsgeschwindigkeit während des Schnitts mit der Bahn synchron sein. Da der Umfang des Messers üblicherweise nicht mit der Schnittlänge übereinstimmt, wird es nach dem Schnitt beschleunigt oder abgebremst. Registermarken dienen zur Synchronisierung des Schnittpunktes mit dem Material. Das Messer wird fortlaufend neu justiert, um kleine Längenunterschiede durch Temperaturunterschiede und Materialdehnung auszugleichen.

Um das rotierende Werkzeug mit dem Material zu synchronisieren, wird eine elektronische Kurvenscheibe verwendet. Die Kurvenscheibe wird als normierte Kurvenscheibe mit einer Produktlänge von 360° definiert. Über diese Strecke rotiert das Werkzeug ebenfalls um 360° um genau einen Bearbeitungsschritt durchzuführen. Der Schneidepunkt ist hier an der Position (0;0) definiert. Die Umfangsgeschwindigkeit wird in einem Bereich von 30° vor und nach diesem Schneidepunkt mit der Materialbahn synchronisiert, dieser Bereich ist durch den zweiten und dritten Punkt in der Kurvenscheibe festgelegt. Zwischen diesen Punkten, also außerhalb des Materialeingriffes, kann das Werkzeug beschleunigen um die aktuelle Produktlänge an den Werkzeugumfang anzupassen.

https://infosys.beckhoff.com/content/1031/tcplclibmc2_camming/Resources/460482315.zip

Trademark statements

Mehr Informationen:
www.beckhoff.com/ts5050

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