BECKHOFF New Automation Technology

Manual | EN

TwinCAT 2

NC Camming

TwinCAT Supplement

2022-11-23 | Version: 1.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

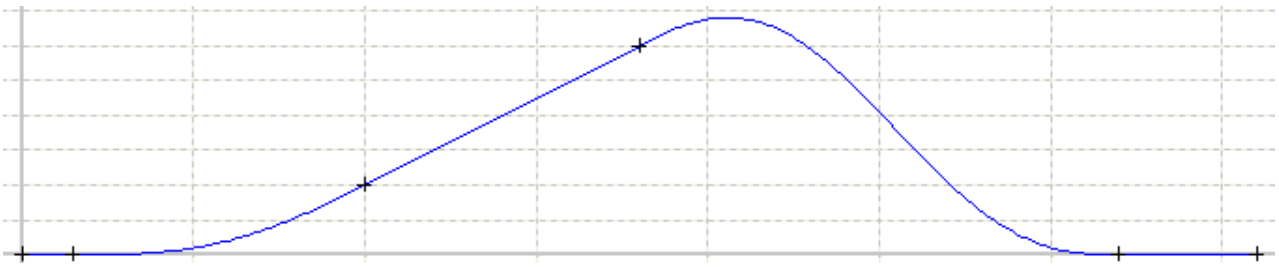
Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

Traditionally, non-linear coupling of electrical **master** and **slave axes** is referred to as a cam disc. TwinCAT offers different options for this type of axis coupling. One option are classic position tables, where a curve is described by a large number of points, i.e. a master position with corresponding slave position. Interpolation is used between these points. This type of cam has been tried and tested in the past, although it has the disadvantage that it is difficult to modify at run time.

As the next step in this direction, TwinCAT offers so-called **motion functions (MF)**, which describe a cam in a different way. A motion function usually has a small number of reference points, between which a mathematical function, e.g. a polynomial, is used to describe the curve. The slave positions are calculated at run time from the mathematical description of the cam. With motion functions it is much easier to modify the function, because the curve can be modified by manipulating a interpolation point.



TwinCAT offers a **cam design editor** that can be used for both types of cam plate, i.e. position tables and motion functions. A **PLC library** facilitates programming of cams.

Cams with motion functions

Cams with position tables [▶ 18]

Cam PLC library

Example program for cams with position tables

Example program for cams with motion functions

Cam design editor

Also see about this

- Camshaft axis - Motion Functions [▶ 9]
- TcNcCamming [▶ 36]

3 Camshaft axis - Motion Functions

A **motion function (MF)** describes a cam disc via mathematical functions. It sub-divides the curve into appropriate **segments** (sections), for which different **motion laws**, i.e. special mathematical functions, can be used (for cam examples see: Cam design tool examples) The motion laws for mechanical cams are defined in VDI guideline 2143 and other documents. The electronic cam in TwinCAT use these functions, among others. The motion functions realise these motion functions directly in the real-time driver of the NC. Unlike classic table couplings that only transfer discrete steps (scatter plots) in the form of larger data quantities to the NC, the complete information is stored in the NC in very compact form. Problems originating from data discretisation (position reference points) in the table are thus eliminated.

The realisation of motion laws in the NC has a further crucial advantage: A **motion diagram**, i.e. the complete description of the motion of a slave axis, can now simply and clearly be defined and modified from the PLC. Associated **PLCfunction blocks** make the application of this functionality very convenient. Users can influence not only the complete motion description, but also individual segments or sub-sections.

In order to ensure that the drive system can actually implement a cam in practice, the system calculates **characteristic values** (such as maximum and minimum position values, velocity and acceleration etc.), which the user has to analyse. The resulting dynamic limit values ultimately depend on the motion of the master and relate to constant master velocity. The characteristic values are thus calculated with the idealised assumption of constant master velocity.
In addition, the mean velocity and the effective acceleration are calculated. These values may be used, for example, for calculating the effective torque or the operating point P_A (n_m ; M_{eff}) in the torque/speed diagram of the motor. The PLC can access the current characteristic values of the NC via function blocks.

In the cam design tool **TwinCAT cam design editor** the decision whether to use classic table couplings (scatter plot) or motion functions can be configured via an associated selection. Subsequently, either the position tables or the motion function points are generated when the configuration is activated. If motion functions are used, these points can subsequently be modified individually by the PLC.

It is possible to modify individual values or complete sections of the motion functions online according to associated rules also online, i.e. while the cam is active. Very flexible cams can thus be realised.

3.1 Motion functions - Definition of a point

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

The information contained in the cam design tool table is sufficient for defining the motion in the NC. However, closer inspection of this MF table reveals the presence of redundant data. Because the motion is described in segments (sections), for motion diagrams with simple interrelationships the end point of a section is identical to the starting point of the next segment. The more complex point types offered by the cam design tool, such as slide point, are not yet implemented. In addition, users want to be able to deactivate individual points in a particular motion diagram (**MOTIONPOINTTYPE_IGNORE**, referred to as **IGNORE** below) at a later stage. These requirements lead a description that in addition to the starting point of a segment, including the point information (velocity, acceleration, point type), also contains the segment information (function type, symmetry value).

Point structure

PointIndex	UINT32	Point index
FunctionType	UINT16	Function type [▶ 10]

PointType	UINT16	Point type [▶ 10]
RelIndexNextPoint	INT32	Relative index of the end point (default: 0, subsequently corresponds to 1)
MasterPos	REAL64	Master position
SlavePos	REAL64	Slave position at this reference point
SlaveVelo	REAL64	Slave velocity at this reference point
SlaveAcc	REAL64	Slave acceleration at this reference point
SlaveJerk/Symmetry	REAL64	Slave jerk at this reference point or symmetry value of the segments for rest in rest motion laws

In this structure, a relative index is used to refer to the point index of the end point of this segment. In order to keep the definition simple for motion diagrams with simple interrelationships, the IGNORE points are indeed ignored completely. The relative point index is therefore automatically adjusted internally.

The default value of the relative point index may therefore be zero, although for a standard list with simple link the value should be one. The user therefore does not have to update this information. The possible point types of the cam design tool therefore include the IGNORE point.

Point types

MOTIONPOINTTYPE_IGNORE	0x0000	Ignore point	Ignored point
MOTIONPOINTTYPE_REST	0x0001	Restpoint	Rest point
MOTIONPOINTTYPE_VELOCITY	0x0002	Velocitypoint	Velocity point
MOTIONPOINTTYPE_TURN	0x0004	Turnpoint	Reversal point
MOTIONPOINTTYPE_MOTION	0x0008	Motionpoint	Movement point

Since no points can be added while the MF is active, the IGNORE point type enables associated points to be included. These can be activated online at a later stage by specifying the associated values (point type not equal IGNORE).

The master position has to be either strictly monotonic rising or falling. Otherwise it is rejected with an associated error message.

3.2 Function types

So far, only some of the function types offered by the TwinCAT cam design tool have been implemented:

Function type	No	Description	Boundary conditions
POLYNOM1	1	1st order polynomial, straight line or synchronized motion	VV
POLYNOM3	3	3rd order polynomial	RR
POLYNOM5	5	5th order polynomial	RR

POLYNOM8	8	8th order polynomial	RR
SINELINE	10	Sine line	RR
MODSINELINE	11	Modified sine line	RR
BESTEHORN	12	Bestehorn	RR
ACCTRAPEZ	13	Acceleration trapezoid	RR
POLYNOM5_MM	15	5th order polynomial with boundary value adaptation	MM
SINE_LINE_COMBI	16	Sine line combination	RR
HARMONIC_COMBI_RT	17	Harmonic combination	RT
HARMONIC_COMBI_TR	18	Harmonic combination	TR
HARMONIC_COMBI_VT	19	Harmonic combination	VT
HARMONIC_COMBI_TV	20	Harmonic combination	TV
ACCTRAPEZ_RT	21	Acceleration trapezoid	RT
ACCTRAPEZ_TR	22	Acceleration trapezoid	TR
MODSINELINE_VV	23	Modified sine line	VV
STEPFUNCTION	99	Step function	RR

The following abbreviations are used for motion function boundary conditions:

R for rest (velocity = zero, acceleration = zero),

V for velocity (acceleration = zero),

T for turn (velocity = zero), and

M for motion (no conditions for velocity or acceleration).

The boundary condition RT, for example, means that the starting point of the segment meets the condition R for rest (velocity = zero, acceleration = zero) and the end point meets the condition T for turn (velocity = zero).

POLYNOMIAL1 realizes a straight line or a synchronized motion. Since in POLYNOMIAL1 the velocity is defined unambiguously by the master and slave values at the segment boundaries, the velocity value does not have to be transferred, or the specified value (e.g. zero) will be overwritten by the value calculated internally. The only motion law without restrictions on boundary conditions is currently POLYNOMIAL5_MM. However, here too appropriate boundary parameters should be provided. If the adjacent segments of a POLYNOMIAL5_MM segment are function type POLYNOMIAL1 segments or meet the boundary condition R for rest, the boundary parameters are overwritten by the internal velocity values of POLYNOMIAL1 or the rest values (velocity = zero, acceleration = zero). The Cam design tool examples illustrate the options and the motion laws to be used for designing cams.

Boundary values are required for "non rest in rest" types. One exception is synchronised motion (polynomial1), for which the current velocities are calculated automatically from the current point positions. The internal coefficients of the adjacent segments are updated accordingly for the calculation.

When the motion function is selected in the cam design tool, all function types that are not realised in the NC are highlighted in red in the table. They are made available by the cam design tool as 5th order polynomial with boundary value adaptation.

3.3 Programming via the PLC

The TwinCAT cam design tool is used for designing a basic cam with the maximum number of points required. This is then automatically available when the TwinCAT system starts up, or it is loaded into the NC at the push of a button (download). Via PLC function blocks, individual points can now be ignored or their values manipulated. The data currently available in the NC can be visualised in cam design tool online mode

or loaded into the tool via upload. The data can thus be checked graphically and numerically. An additional option is for the NC to process the data as a table for discrete master positions including their derivatives. These data can then be used by the PLC or relayed to a visualisation software. The visualisation can display the point data including derivatives without having to calculate them.

Since the cam data are used exactly as defined in the motion functions, it is important that the functionality to be made available to the end user (operator) is considered during the design phase. Another consideration is how to check the input from the user interface to ensure that only sensible cam can be generated. Calculating characteristic values [▶ 14] can aid this process. For graphic control of the motion functions in the user interface, the NC can make the values for position, velocity and acceleration available in the form of a table (scatter plot).

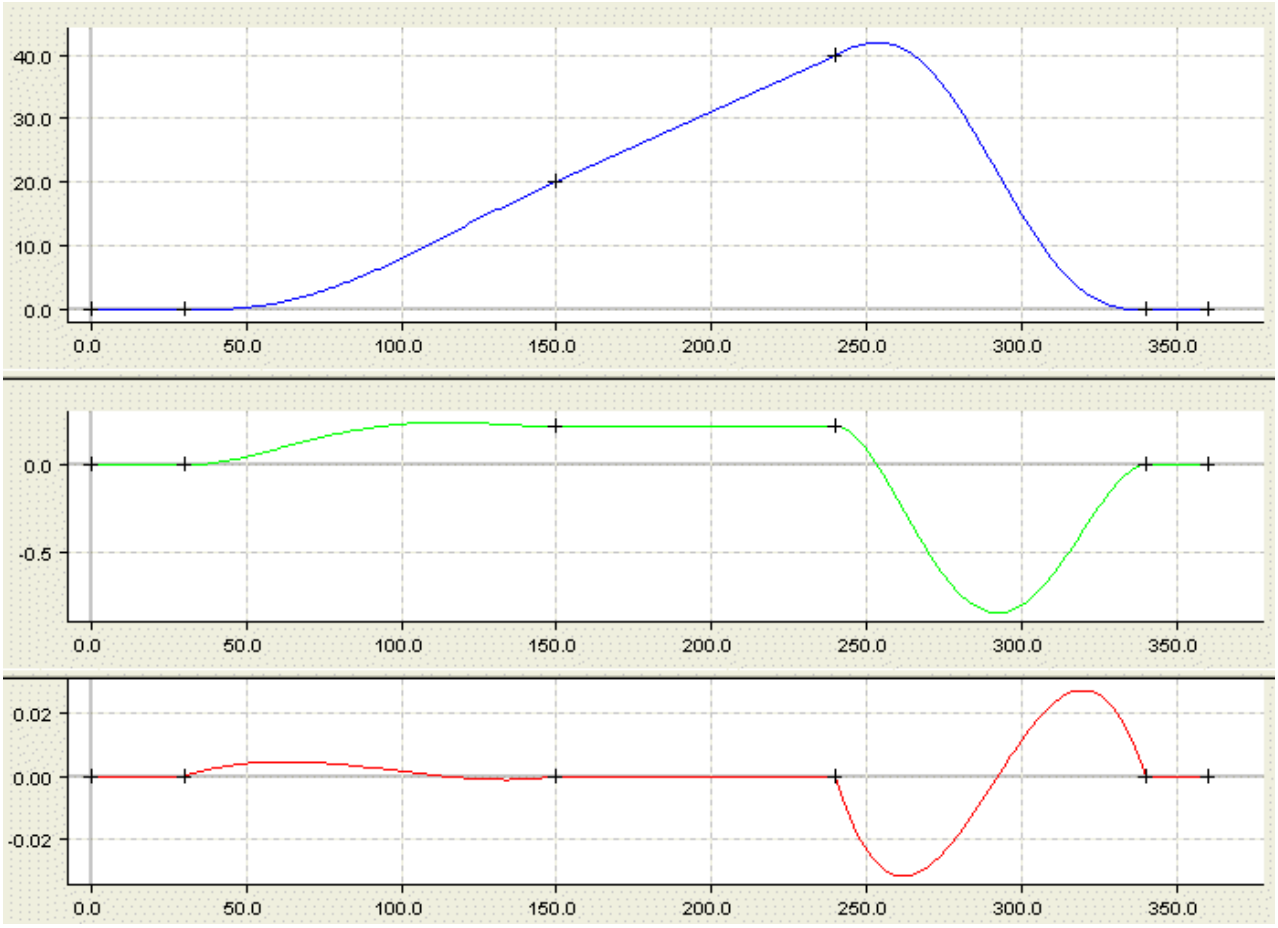
Example for an application:

In a transfer device for a press system, the transfer motion is defined by the lifting angle of the press. A motion coupled with this lifting angle is entered in the user interface of the transfer device. This information is processed by the PLC together with the internally saved mechanical specification. The PLC transfers the cam data to the NC for moving the axes. For the visualization, the PLC retrieves the cam data (position, velocity and acceleration) in tabular form from the NC and makes it available to the user interface.

For an optimization calculation, the PLC reads the characteristic values [▶ 14] (particularly average speed and effective acceleration) for the current cam and calculates the operating point P_A (n_m ; M_{Eff}) in the torque/speed diagram of the respective motor. The maximum permissible speed of the transfer device can thus be determined iteratively.

Example for a cam :

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



The above cam is to be realised as a motion function:

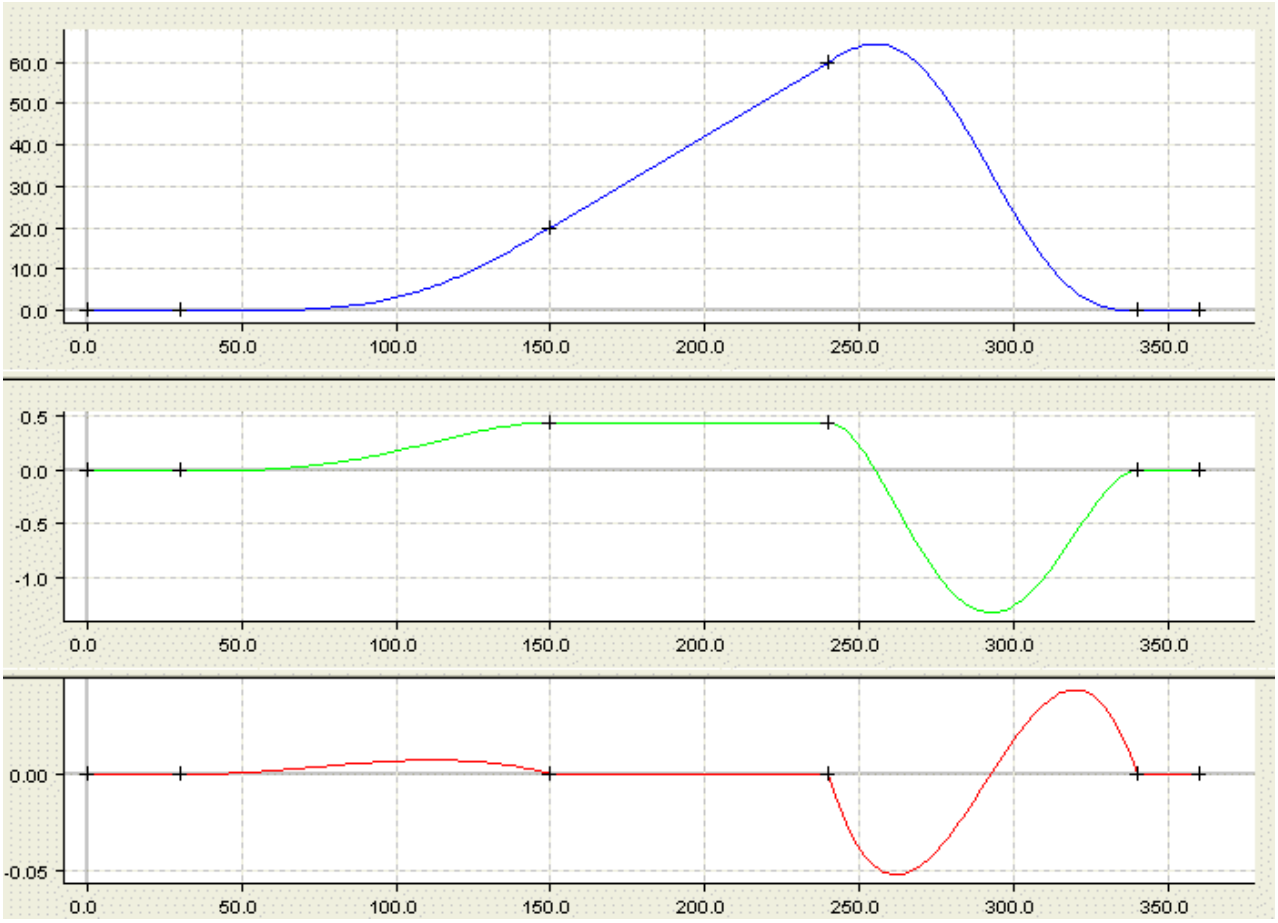
Point Index	Function Type	Point Type	RelIndex NextPoint	Master Pos	Slave Pos	Slave Velo	Slave Acc	Slave Jerk
1	POLYNOM1	REST	1 or 0	0	0	0	0	0
2	POLYNOM5_MM	REST	1 or 0	30	0	0	0	0
3	POLYNOM1	VELOCITY	1 or 0	150	20	0	0	0
4	POLYNOM5_MM	VELOCITY	1 or 0	240	40	0	0	0
5	POLYNOM1	REST	1 or 0	340	0	0	0	0
6	POLYNOM1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

In this case, the velocity does not have to be specified, since it is calculated internally (see function types [▶ 10]). The relative index can also be set to zero for all values, since the diagram only consists of points with a simple relationship. The index only has to be set unequal zero for complex point definitions such as slide points. The master position must be monotonic rising.

Modification 1:

The complete cam can be changed by writing a single point:

4	POLYNOM5_MM	VELOCITY	1	240	60	0	0	0



Modification 2:

or the complete cam can be set to zero by ignoring the two relevant points, i.e. all slave values are zero.

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

These changes are currently only activated after coupling, although in future activation according to associated rules will be possible online.

Further cam examples can be found in: Cam design tool examples (note that example 4 cannot be realised with motion functions at this stage.)

3.4 Characteristic values

Through internal calculation of the *characteristic values*, the risk of violating certain physical axis limits by false input of values can be eliminated.

The parameters described in the table are calculated by the NC and can be read with a PLC FB or via ADS. The dynamic parameters can only be calculated for a master that moves with constant velocity. The velocity of the master is also included in the structure. By default, the values are calculated for velocity 1.0. These values are also called standardised values. Conversion to a different constant master velocity is implemented in the NC and can be called up via a PLC FB. The re-scaled values are stored in the same structure. The relationship between the master velocity and the characteristic values is identified in the first variable called master velocity.

For a motion function with constant maximum acceleration, the position of the master at maximum acceleration (fMPosAtSAccMax) is ambiguous. The same applies to the other master positions at maximum or minimum values. Therefore, these have to be treated with caution. The same applies to parameters derived from the master positions (fSVeloAtSAccMin, fSVeloAtSAccMax).

The resulting dynamic limit values ultimately depend on the motion of the master. These calculated values are therefore only accurate for motion with constant master velocity.

No	Names	Description		Data type
Master				
1.	fMasterVeloNom	master nominal velocity (normed = 1.0)	1	REAL64
Start Point				
2.	fMasterPosStart	master start position	2	REAL64
3.	fSlavePosStart	slave start position	3	REAL64
4.	fSlaveVeloStart	slave start velocity	4	REAL64
5.	fSlaveAccStart	slave start acceleration	5	REAL64
6.	fSlaveJerkStart	slave start jerk	6	REAL64
End Point				
7.	fMasterPosEnd	master end position	7	REAL64
8.	fSlavePosEnd	slave end position	8	REAL64
9.	fSlaveVeloEnd	slave end velocity	9	REAL64
10.	fSlaveAccEnd	slave end acceleration	10	REAL64
11.	fSlaveJerkEnd	slave end jerk	11	REAL64
Minimal Slave Position				
12.	fMPosAtSPosMin	master position at slave minimum position	12	REAL64
13.	fSlavePosMin	slave minimum position	13	REAL64
Minimal Slave Velocity				
14.	fMPosAtSVeloMin	master position at slave minimum velocity	14	REAL64
15.	fSlaveVeloMin	slave minimum velocity	15	REAL64
Minimal Slave Acceleration				
16.	fMPosAtSAccMin	master pos. at slave minimum acceleration	16	REAL64
17.	fSlaveAccMin	slave minimum acceleration	17	REAL64
18.	fSVeloAtSAccMin	slave velocity at slave min. acceleration	18	REAL64
Minimal Slave Jerk/Dyn.Mom				
19.	fSlaveJerkMin	slave minimum jerk	19	REAL64
20.	fSlaveDynMomMin	slave minimum dynamic momentum	20	REAL64 **NOT SUPPORTED**

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

3.5 Online modification

The motion functions not only offer the advantage that they can be defined via a function block from the PLC, they can also be modified at run time. Example: modification of the slave values of a turning point for adjusting the lift. More comprehensive modifications for the start-up curve of a complex cyclic motion (without rest or constant velocity) can also be implemented.

The online change of the **motion function (MF)** follows certain rules. It may be activated from a certain axis position (AtMasterAxisPos), immediately (Instantaneous) or from the next cycle. The current activation rule applies until a new rule is transferred. The command for online changes of MF points can be used to modify single points or coherent ranges of several points. One bit of the cyclic interface is allocated for synchronisation. Within a program, the activation rule has to be defined (via a function block), and the modified values of the motion function points have to be transferred. This transfer is done with the same ADS commands or PLC function blocks that are used to define the motion function on system start-up.

ActivationMode	
Instantaneous (Default)	0
AtMasterCamPos	1
AtMasterAxisPos	2
NextCycle	3
AsSoonAsPossible (former default)	5

Off	6
DeleteQueuedData	7

The **Instantaneous** activation mode activates the transferred motion function point values with the next tick. Obviously, this functionality should only be used if the new values have no direct effect on the current slave axis values. A change will thus affect another segment without interacting with the current segment. The option **AsSoonAsPossible** is used to check which segments were not modified. The change is only realised when the slave is in such a segment. The two modes **AtMasterAxisPos** and **AtMasterCamPos** differ in that **AtMasterAxisPos** uses the axis position as reference, while **AtMasterCamPos** relates to the position in the motion diagram. In a cyclic motion from 0 to 360, axis position 180 is crossed only once, while in the motion diagram position 180 is crossed during each cycle. If the axis position has already been crossed, an activation command with **AtMasterAxisPos** mode will lead to an error. In **AtMasterCamPos** mode, the activation command is executed in the next cycle from the specified position. **NextCycle** is used to set the activation position to the end or the start of the cycle. In **Off** mode, modification commands are not executed. **DeleteQueuedData**deletes point data that have already been transferred but not yet activated. The current cam is, of course, not affected.

A convenient way of implementing changes is the option of setting SlaveScalingMode to **AutoOffset** . In this case, the slave offset is automatically corrected such that jumps in slave position are avoided, although jumps are possible in the velocity or the acceleration. If, for example, the activation position is in a segment with constant velocity and the modified segment has the same velocity, the change can be implemented without problem.

SlaveScalingMode	
UserDefined (Default)	0
AutoOffset	1
Off	2

The memory management for the motion function points does not permit subsequent addition of points. An option to ignore points (IGNORE) has therefore been implemented. In order to make searching for the current segment efficient, the master positions had to be stored as a monotonic increasing sequence. This means that even points to be activated at a later stage have to be located at the correct master position. When starting the system, the operator should therefore be clear about the maximum number of points required.

NOTE

Online modification offers a wide range of options for implementing changes. The downside is that changes may lead to situations that can cause axis errors or other problems. This powerful tool should therefore be used with caution and only after appropriate simulation tests at the respective machine have been carried out.

4 Camshaft axis - position tables

A slave axis whose set position values can be found from the set position values of another axis by means of a pre-calculated table is known as a **camshaft axis (table slave axis)**.

The table contains the slave positions associated with the respective master positions, and is linear interpolated. The master positions in the table can be equidistantly monotonically ascending, or simply monotonically ascending. That tables can be processed linearly or cyclically. It is possible to give an offset to the master position and/or the slave position. These offsets can be changed on-line. The tables can be processed absolutely, or relatively in relation to the coupling position (of master and/or slave).

Interfaces

Interfaces [▶ 18]

The interfaces consist of the **System Manager**, **PLC blocks** and the **cyclic NC-PLC axis interface**.

Table types

Table types [▶ 19]

There are four types of table, differing in the table contents and in the way they are processed.

- Equidistant tables in which the master position rises in equidistant steps:
  - **Equidistant linear tables**, i.e. tables that are worked through from the table edge in accordance with the table contents.
  - **Equidistant cyclic tables**, i.e. tables that are worked through from the table edge and periodically restarted.
- Non-equidistant tables, in which the master position rises in strictly monotonic steps that are not necessarily equidistant:
  - **Non-equidistant linear tables**, i.e. tables that are worked through from the edge of the table in accordance with the table contents.
  - **Non-equidistant cyclic tables**, i.e. tables that are worked through from the edge of the table and are periodically restarted.

Processing Modes

Processing Modes [▶ 21]

There are master offsets and/or slave offsets, different reference systems (relative/absolute - in reference to the coupling position), and the possibility of **linear** or **cyclic** processing.

Procedure

Procedure [▶ 24]

The procedure consists of the coupling preparations (moving the axes to the coupling positions) coupling, starting the master axis, possible on-line changing of the offset(s), braking the slave axis (directly via the master, or indirectly via the table) and the uncoupling.

4.1 Interfaces

The user can operate the camshaft axis by means of the System Manager or by PLC blocks. Furthermore, a cyclic NC-PLC interface is available (in the System Manager or via the PLC) that contains the **axis process image** and enables access to various variables.

System Manager

The System Manager provides access to all the functionalities of the camshaft axis:

- Logical coupling, stating the master axis and setting the coupling table.
- Uncoupling.

PLC Blocks

The PLC Library offers access to the complete slave axis functionality

- Logical coupling, stating the master axis and setting the coupling table.
- Stop.

NC-PLC-Interface

The cyclic NC-PLC axis interface contains the axis process image and particularly access to the coupling information (coupling flag).

4.2 Table types

There are four types of table, differing in the table contents and in the way they are processed.

- Equidistant tables in which the master position rises in equidistant steps:
  - **Equidistant linear tables**, i.e. tables that are worked through from the table edge in accordance with the table contents.
  - **Equidistant cyclic tables**, i.e. tables that are worked through from the table edge and periodically restarted.
- Non-equidistant tables, in which the master position rises in strictly monotonic steps that are not necessarily equidistant:
  - **Non-equidistant linear tables**, i.e. tables that are worked through from the table edge in accordance with the table contents.
  - **Non-equidistant cyclic tables**, i.e. tables that are worked through from the table edge and periodically restarted.

The multi-table slaves only operate with non-equidistant tables.

Nature and source of the danger

- Non equidistant tables are based on a linear search algorithm that is started and re-initialized by bisection.
- Equidistant tables are also non-equidistant tables.
- Each table can be processed linearly or cyclically.

General Table Conventions

- Tables only contain binary data.
- A table consists of a header (the first line) and the table data (the remaining lines).
- The header contains two numbers of type unsigned short. The first column contains the number of lines (without header), while the second column contains the number of columns (for table-slave tables this is always 2). There are no separating characters between the data.
- Apart from the header, the table only contains data of type double.
- The first column (with the exception of the header line) contains the master positions, while the second column contains the associated slave positions (both in mm). There are no separating characters between the data.

The table(s) must be adapted to the desired dynamics. At dynamically critical locations where the master axis is moving slowly, the table must contain enough data to provide accurate resolution.

Table Notation

The table *T* contains *N* lines, each line containing two items of data, the master position **T***[k][0]* and the associated slave position. **T***[k][1], 0 ≤ k < N*. Notation:
T*[k,0]* master value, *0 ≤ k < N*,
T*[k,1]* slave value, *0 ≤ k < N*,
T*(y)* interpolated slave position value from the interval **T***[k,1]* and **T***[k+1,1]*, where **T***[k,0] < y ≤***T***[k+1,0]* for non-equidistant tables, and **T***[k,0] ≤ y <* **T***[k+1,0]* for equidistant tables.

Cyclic Tables (Cyclic Processing)

Cyclic tables are characterised by the following additional parameters.

- A **master period**: it is understood that the period is not negative, and begins at *0.0*. Tables extend as agreed from **T***[0][0] = 0.0* to **T***[N-1][0]* = periods.
- A **slave difference per master period** (which can have any arithmetic sign) that indicates how many mm the slave position changes in each period. If a slave is itself periodic, then this difference is *0.0*. The following applies: **T***[N-1][1]* - **T***[0][1]* = slave difference per master period.

Evaluation of Linear Tables

- In the case of a table overflow/underflow in the master position , the table value at the relevant edge of the table is output as the slave position value.
- It should particularly be noted that if the master position is outside the range of the table when coupling takes place, the slave axis will jump in the first step to the value at the relevant table edge. In general this will generate a following error.
- The current slave position is determined by linear interpolation.
- The current slave velocity is, if possible, calculated from the master's set velocity with the aid of the table and the chain rule (this does assume that the master set velocity is sufficiently smooth). Otherwise, the slave velocity is calculated by numerical differentiation (and is correspondingly coarse when multiple interpolation is needed between a pair of lines).
- The current slave acceleration is always calculated by numerical differentiation, and is correspondingly coarse.
- **Evaluation of Cyclic Tables**
  - The table has a cycle counter that is incremented in the case of table overflow and decremented in the case of table underflow.
  - In the case of table overflow/underflow, the slave difference per master period, multiplied by the cycle counter, is added to the slave position.
  - The current slave position is determined by linear interpolation.
  - The current slave velocity is, if possible, calculated from the master's set velocity with the aid of the table and the chain rule (this does assume that the master set velocity is sufficiently smooth). Otherwise, the slave velocity is calculated by numerical differentiation (and is correspondingly coarse when multiple interpolation is needed between a pair of lines).
  - The current slave acceleration is always calculated by numerical differentiation, and is correspondingly coarse.

    Interfaces

    PLC documentation link->outside

    ASCII->bin link->outside

    Table read link->outside

Evaluation of cycling tables

- The table uses a cycle counter, which increments with table overflow and dcremtnes wit a table undeflow
- The position oft he slave results as a sum of the product of the slave difference in each master period the cycle counter value
- The current determintation of the slave position is computed with a linear interpolation
- The current slave velocity will be derived, if the derivation conidtions are met. Otherwise a numerical procedure will be used to determine the current slave velocity
- The current slave acceleration will be determined in a numerical procedure

4.3 Reference systems:

Cam plates can be processed linearly or cyclically with coupled master/slave axes. They can be scaled on both the master and slave side and are fitted with a position offset. The reference system of a cam plate can thus be interpreted as absolute or relative to the coupling position of the axes.

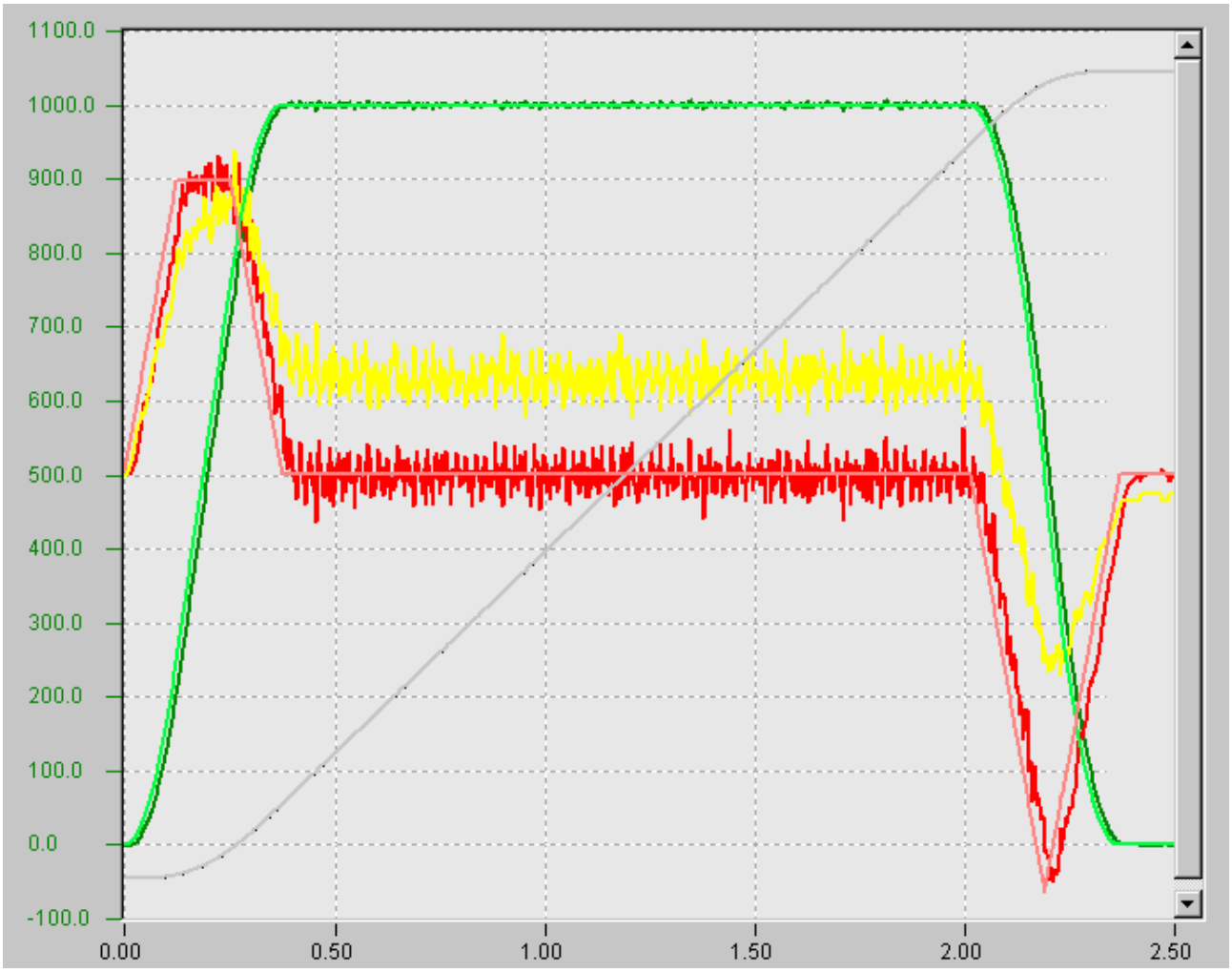


Fig. 1: Linear processing

Offsets

It is possible to provide a slave offset as well as a master offset:

- The master offset (*MO*) is added to the master position **before** any other operations.
- The slave offset (*SO*) is added to the slave position **after** any other operations.

Scaling

There is a slave scaling, with which all the slave positions are multiplied. Furthermore there is a master scaling, with which all the master positions are divided. The scalings are usually preset with 1.0 through the coupling of the slaves; however you can deviate from these values as you like (master scaling may not be 0.0). It is also possible to change the scaling on-line, but this should only be done moderately, because - even when the axes are dynamically stationary - a following error is created. With cyclic processing (see below) the slave difference (slave stroke) per master period is also multiplied by the slave scaling.

- The master scaling (*MS*) is used as a **divisor**.
- The slave scaling (*SS*) is used as a **multiplier**.

Cyclic processing

The rule for cyclic processing is that the modulo operations are performed at the lowest level, i.e. immediately before the table is evaluated.

- The table has a **cycle counter** (Z_m), that is incremented in the case of table overflow and decremented in the case of table underflow.
- The master position is given in **absolute** terms, not modulo.
- In the case of table overflow/underflow, the **slave difference (slave stroke) per master period** (*SH*), multiplied by the cycle counter, is added to the slave position.

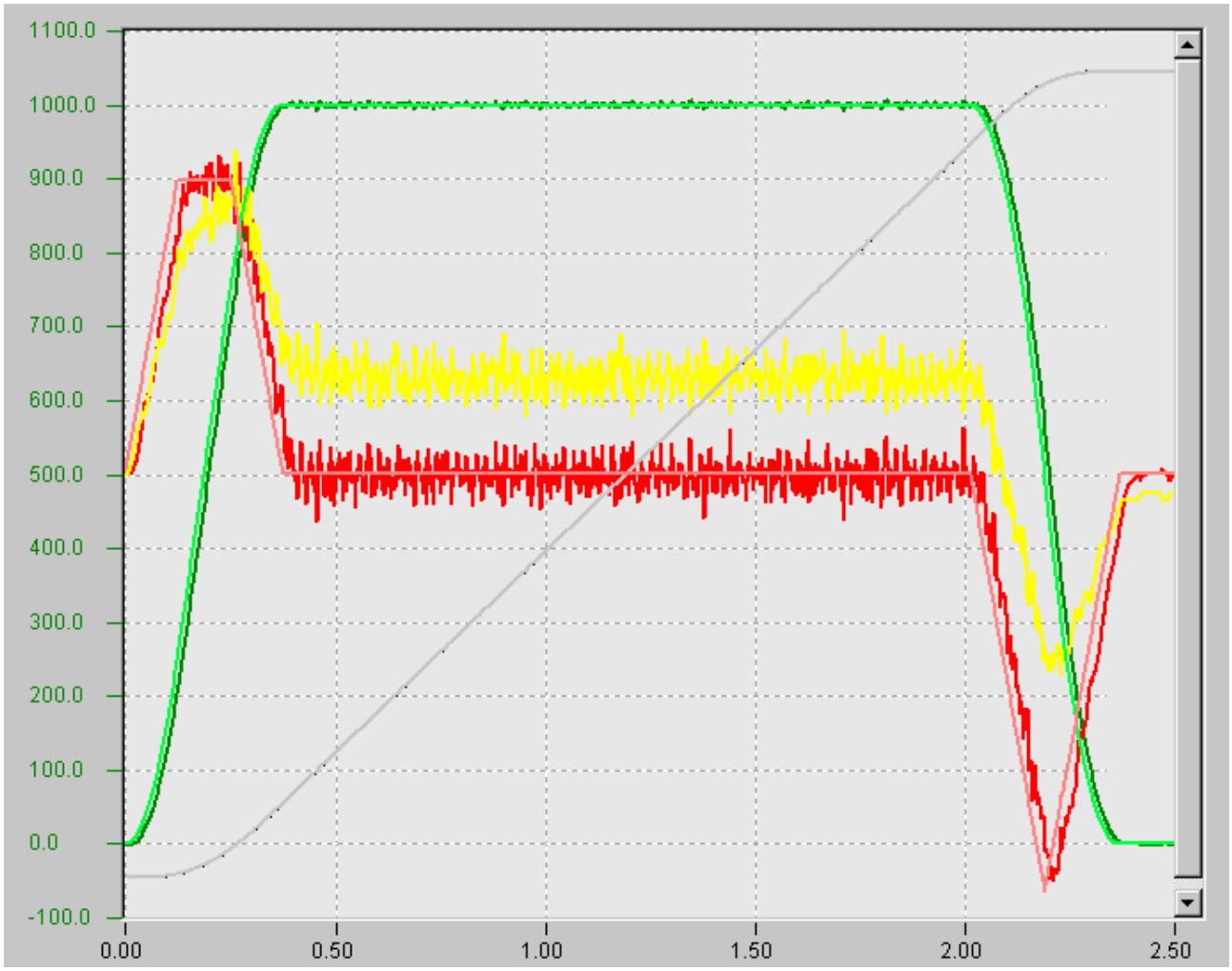


Fig. 2: Cyclic processing

Description

1. P_m Absolute master position
2. P_{mc} Master coupling position (master start position), only effective for *relative reference system*!

3. MO Master offset
4. MS Master scaling
5. P_s Absolute slave position
6. P_{sc} Slave coupling position (slave start position), only effective for *relative reference system*!
7. SO Slave offset
8. SS Slave scaling
9. *T (•)* Table
10. mod[•] Modulo calculation with master period (example for a master period of 360 deg: mod[40] = 40, mod[400] = 40, mod[-20] = 340)
11. Z_m Master cycle counter
12. SH Slave difference (slave stroke) per master period

Reference systems

The position of a slave axis results from a calculation regulations from the data of the cam plate table and the actual master position. If a relative reference system is selected for master or slave, then the associated position column of the table should begin with *0.0*. "Relative" always relates to the coupling position of the respective axis. The calculation regulations for combinations of relative and absolute reference are explained below.
Often it is necessary for a real application to change the formulas so that the slave offset (*SO*) or the master offset (*MO*) results.

Reference system for linear tables

General case: Offsets not 0.0 and scalings not 1.0

- Master absolute and slave absolute: $P = SO\ (\ (MO + P\ ) / MS\ )_s + \mathbf{T}_m \cdot SS$
- Master relative and slave absolute: $P = SO\ (\ (MO + P - P\ ) / MS\ )_s + \mathbf{T}_{mmc} \cdot SS$
- Master absolute and slave relative: $P = P + SO + (\ (MO + P\ ) / MS\ )_{ssc}\mathbf{T}_m \cdot SS$
- Master relative and slave relative: $P = P + SO + (\ (MO + P - P\ ) / MS\ )_{ssc}\mathbf{T}_{mmc} \cdot SS$

Simplified special case 1: Offsets not 0.0 and scalings are 1.0

- Master absolute and slave absolute: $P = SO\ (\ (MO + P\ )\ )_s + \mathbf{T}_m$
- Master relative and slave absolute: $P = SO\ (\ (MO + P - P\ )\ )_s + \mathbf{T}_{mmc}$
- Master absolute and slave relative: $P = P + SO + (\ (MO + P\ )\ )_{ssc}\mathbf{T}_m$
- Master relative and slave relative: $P = P + SO + (\ (MO + P - P\ )\ )_{ssc}\mathbf{T}_{mmc}$

Simplified special case 2: Offsets are 0.0 and scalings not 1.0

- Master absolute and slave absolute: $P = (\ P\ )_s\mathbf{T}_m$
- Master relative and slave absolute: $P = (\ P - P\ )_s\mathbf{T}_{mmc}$
- Master absolute and slave relative: $P = P + (\ P\ )_{ssc}\mathbf{T}_m$
- Master relative and slave relative: $P = P + (\ P - P\ )_{ssc}\mathbf{T}_{mmc}$

Reference system for cyclic tables

In the case of table overflow/underflow, the slave difference (slave stroke) per master period, multiplied by the cycle counter, is added to the slave position.

General case: Offsets and slave stroke not 0.0 and scalings not 1.0

- Master absolute and slave absolute: $P = SO\ (\ mod[\ (MO + P\ ) / MS\ ]\ )_s + \mathbf{T}_m \cdot SS + SH \cdot Z_M$
- Master relative and slave absolute: $P = SO\ (\ mod[\ (MO + P - P\ ) / MS\ ]\ )_s + \mathbf{T}_{mmc} \cdot SS + SH \cdot Z_M$
- Master absolute and slave relative: $P = P + SO + (\ mod[\ (MO + P\ ) / MS\ ]\ )_{ssc}\mathbf{T}_m \cdot SS + SH \cdot Z_M$
- Master relative and slave relative: $P = P + SO + (\ \ mod[\ (MO + P - P\ ) / MS\ ]\ \ )_{ssc}\mathbf{T}_{mmc} \cdot SS + SH \cdot Z_M$

Simplified special case 1: Offsets and slave stroke not 0.0, scalings are 1.0

- Master absolute and slave absolute: $P = SO\ (\ mod[\ (MO + P\ )\ ]\ ) + SH_s + \mathbf{T}_m \cdot Z_M$
- Master relative and slave absolute: $P = SO\ (\ mod[\ (MO + P - P\ )\ ]\ )_s + \mathbf{T}_{mmc} + SH \cdot Z_M$
- Master absolute and slave relative: $P = P + SO + (\ mod[\ (MO + P\ )\ ]\ ) + SH_{ssc}\mathbf{T}_m \cdot Z_M$
- Master relative and slave relative: $P = P + SO + (\ \ mod[\ (MO + P - P\ )\ ]\ \ ) + SH_{ssc}\mathbf{T}_{mmc} \cdot Z_M$

Simplified special case 2: Offsets and slave stroke are 0.0, scalings are 1.0

- Master absolute and slave absolute: $P = (\ mod[\ P\ ]\ )_s\mathbf{T}_m$
- Master relative and slave absolute: $P = (\ mod[\ P - P\ ]\ )_s\mathbf{T}_{mmc}$
- Master absolute and slave relative: $P = P + (\ mod[\ P\ ]\ )_{ssc}\mathbf{T}_m$
- Master relative and slave relative: $P = P + (\ mod[\ P - P\ ]\ )_{ssc}\mathbf{T}_{mmc}$

Interfaces

4.4 Procedure

The procedure consists of the coupling preparations (moving the axes to the coupling positions) coupling, starting the master axis, possible on-line changing of the offset(s), braking the slave axis (directly via the master, or indirectly via the table) and the uncoupling.

Preparation

1. The master axis is moved in PTP operation to the coupling position (=start position).
2. The slave position associated with this master position is found from the table. Reading the table only yields the table value. The user must calculate from this using any offsets and processing modes (relative/absolute) that may have been provided with the coupling.
3. The slave axis is moved in PTP mode to the coupling position (=start position).

Coupling/Start

The coupling is made off-line (slave and master are stationary).

The slave then starts when the master starts, and when the table contents require it (the slave position is not constant).

Online

The table master-slave coupling is a **linear path controller**.

The **offsets** can be changed on-line. Only moderate changes should be made to the offset during motion, because a following error is created.

In the case of table overflow/underflow of cyclic tables, the slave difference per master period, multiplied by the cycle counter, is added to the slave position.

There is no separate stop for the slave axis.

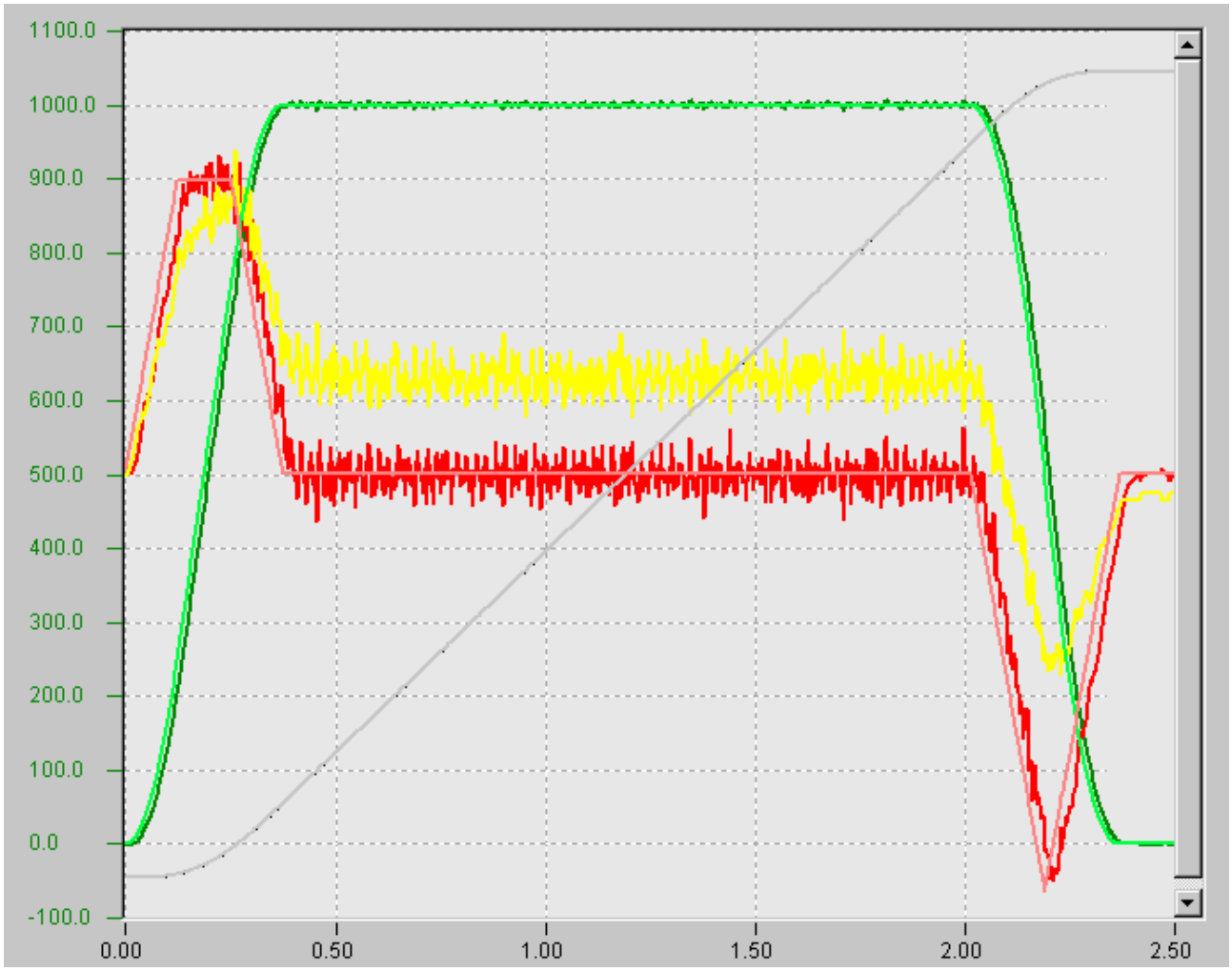


Fig. 3: On-line offset changes

Error Handling

There is **no explicit error handling** if the top or bottom of linear tables is passed. The slave axis remains instantly stationary at the relevant table edge, creating (depending on the dynamic state) a following error.

Stopping/Uncoupling

The slave axis either stops when the master stops (linear path control) or when required by the table contents (the slave position is constant over a wide range of master positions).

Uncoupling can take place off-line (master and slave are stationary) or semi-off-line (only the slave is stationary).

Interfaces

- **Table reading**
  - 1. PLC Blocks
- **Coupling**
  - 1. System Manager
  - 2. PLC Blocks
- **Uncoupling**
  - 1. System Manager
  - 2. PLC Blocks

4.5 Online modification of the scaling or the offset

Online changes of the scaling or the offset of cams (position tables and motion function) follow certain rules. It may be activated from a certain axis position (AtMasterAxisPos), immediately (Instantaneous) or from the next cycle. The activation rule (ActvivationMode, ActivationPosition, MasterScalingMode or SlaveScalingMode) and the parameters MasterOffset, SlaveOffset, MasterScaling and SlaveScaling are transferred with the online change command or the function block (**MC_CamScaling**).

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

The **Instantaneous** activation mode activates the transferred values with the next tick. The two modes **AtMasterAxisPos** and **AtMasterCamPos** differ in that **AtMasterAxisPos** uses the axis position as reference, while **AtMasterCamPos** relates to the position in the motion diagram. In a cyclic motion from 0 to 360, axis position 180 is crossed only once, while in the motion diagram position 180 is crossed during each cycle. If the axis position has already been crossed, an activation command with **AtMasterAxisPos** mode will lead to an error. In **AtMasterCamPos** mode, the activation command is executed in the next cycle from the specified position. **NextCycle** is used to set the activation position to the end or the start of the cycle.

A convenient way of implementing changes is the option of setting SlaveScalingMode or the Master ScalingMode to **AutoOffset** . In this case, the slave or master offset is automatically corrected such that jumps in slave position are avoided, although jumps are possible in the velocity or the acceleration.

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

Examples:

At the start of a cycle, the cam is at rest (velocity and acceleration are equal zero). The slave motion can be stopped with the values:

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

The coupling remains intact. With the associated command with SlaveScaling =1.0, the motion can subsequently be reactivated.

The cam has a constant velocity range between 200 and 250 degrees. With the values:

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

changes the MasterOffset by an absolute value of 10. SlaveScalingMode = AutoOffset automatically corrects the slave such that jumps in position are avoided. The result corresponds to instantaneous phasing.

NOTE

Online modification offers a wide range of options for implementing changes. The downside is that changes may lead to situations that can cause axis errors or other problems. This powerful tool should therefore be used with caution and only after appropriate simulation tests at the respective machine have been carried out.

4.6 Multitable axes

A slave axis whose set position values can be found from the set position values of another axis by means of a number of pre-calculated tables (serial and/or parallel) is known as a **multi-table slave axis**.

The tables contain the slave positions corresponding to each particular master position, along with the scaling that consists of the master offset, slave offset, master slope and slave slope, permitting the table to be **scaled** in any desired way. The usual case involves a normalised start table, a normalised working table and a normalised stop table (profile tables) that are worked through in sequence one after another. On top of this there are correction tables that are used in parallel with the profile tables, and which permit on-line position corrections to be made for the slave axis.

Interfaces and Configuration

Interfaces and Configuration [▶ 27]

The interfaces consist of the **System Manager**, **PLC blocks** and the **cyclic NC-PLC axis interface**.

Table types

Table types [▶ 19]

There are two types of table, differing in the way they are processed. These are non-equidistant tables, in which the master position rises in strictly monotonic steps that are not necessarily equidistant:

- **Non-equidistant linear tables**, i.e. tables that are worked through from the edge of the table in accordance with the table contents.
- **Non-equidistant cyclic tables**, i.e. tables that are worked through from the edge of the table and are periodically restarted.

Processing Modes

Processing Modes [▶ 30]

For every table associated with a multi-table axis there is a **scaling** consisting of the master offset, master slope, slave offset, slave slope and table weighting, with which the table can be similarly scaled to any desired magnitude in both co-ordinates. This makes it possible to restrict the system to a small number of **normalised tables**. Processing may take place in a linear or cyclic manner.

Procedure

Procedure [▶ 31]

The sequence of events consists of the coupling preparations (moving the axes to the coupling positions) coupling, starting the master axis, parallel table change, possible serial **on-line activation of correction tables**, braking the slave axis (directly via the master, or indirectly via the table(s)) and uncoupling.

4.6.1 Interfaces and Configuration

The user can operate the multi-table slave axis by means of the System Manager or by PLC blocks. Furthermore, a cyclic NC-PLC interface is available (in the System Manager or via the PLC) that contains the **axis process image** and enables access to various variables.

System Manager

The System Manager offers access to all the functionalities of the multi-table slave axes:

- Logical coupling, stating the master axis and the coupling tables.
- Stop.

PLC Blocks

The PLC Library offers access to the complete slave axis functionality

- Logical coupling, stating the master axis and the coupling tables.
- Stop.

NC-PLC-Interface

The cyclic NC-PLC axis interface contains the axis process image and particularly access to the coupling information (coupling flag).

4.6.2 Table types

There are four types of table, differing in the table contents and in the way they are processed.

- Equidistant tables in which the master position rises in equidistant steps:
  - **Equidistant linear tables**, i.e. tables that are worked through from the table edge in accordance with the table contents.
  - **Equidistant cyclic tables**, i.e. tables that are worked through from the table edge and periodically restarted.
- Non-equidistant tables, in which the master position rises in strictly monotonic steps that are not necessarily equidistant:
  - **Non-equidistant linear tables**, i.e. tables that are worked through from the table edge in accordance with the table contents.
  - **Non-equidistant cyclic tables**, i.e. tables that are worked through from the table edge and periodically restarted.

The multi-table slaves only operate with non-equidistant tables.

- Non equidistant tables are based on a linear search algorithm that is started and re-initialized by bisection.
- Equidistant tables are also non-equidistant tables
- Each table can be processed linearly or cyclically.

General Table Conventions

- Tables only contain binary data.
- A table consists of a header (the first line) and the table data (the remaining lines).
- The header contains two numbers of type unsigned short. The first column contains the number of lines (without header), while the second column contains the number of columns (for table-slave tables this is always 2). There are no separating characters between the data.
- Apart from the header, the table only contains data of type double.
- The first column (with the exception of the header line) contains the master positions, while the second column contains the associated slave positions (both in mm). There are no separating characters between the data.

The table(s) must be adapted to the desired dynamics. At dynamically critical locations where the master axis is moving slowly, the table must contain enough data to provide accurate resolution.

Table Notation

The table *T* contains *N* lines, each line containing two items of data, the master position **T***[k][0]* and the associated slave position. **T***[k][1], 0 ≤ k < N*. Notation:
T*[k,0]* master value, *0 ≤ k < N*,
T*[k,1]* slave value, *0 ≤ k < N*,
T*(y)* interpolated slave position value from the interval **T***[k,1]* and **T***[k+1,1]*, where **T***[k,0] < y ≤***T***[k+1,0]* for non-equidistant tables, and **T***[k,0] ≤ y <* **T***[k+1,0]* for equidistant tables.

Cyclic Tables (Cyclic Processing)

Cyclic tables are characterised by the following additional parameters.

- A **master period**: it is understood that the period is not negative, and begins at *0.0*. Tables extend as agreed from **T***[0][0] = 0.0* to **T***[N-1][0]* = periods.
- A **slave difference per master period** (which can have any arithmetic sign) that indicates how many mm the slave position changes in each period. If a slave is itself periodic, then this difference is *0.0*. The following applies: **T***[N-1][1]* - **T***[0][1]* = slave difference per master period.

Evaluation of Linear Tables

- In the case of a table overflow/underflow in the master position , the table value at the relevant edge of the table is output as the slave position value.
- It should particularly be noted that if the master position is outside the range of the table when coupling takes place, the slave axis will jump in the first step to the value at the relevant table edge. In general this will generate a following error.
- The current slave position is determined by linear interpolation.
- The current slave velocity is, if possible, calculated from the master's set velocity with the aid of the table and the chain rule (this does assume that the master set velocity is sufficiently smooth). Otherwise, the slave velocity is calculated by numerical differentiation (and is correspondingly coarse when multiple interpolation is needed between a pair of lines).
- The current slave acceleration is always calculated by numerical differentiation, and is correspondingly coarse.
- **Evaluation of Cyclic Tables**
  - The table has a cycle counter that is incremented in the case of table overflow and decremented in the case of table underflow.
  - In the case of table overflow/underflow, the slave difference per master period, multiplied by the cycle counter, is added to the slave position.
  - The current slave position is determined by linear interpolation.
  - The current slave velocity is, if possible, calculated from the master's set velocity with the aid of the table and the chain rule (this does assume that the master set velocity is sufficiently smooth). Otherwise, the slave velocity is calculated by numerical differentiation (and is correspondingly coarse when multiple interpolation is needed between a pair of lines).
  - The current slave acceleration is always calculated by numerical differentiation, and is correspondingly coarse.

    Interfaces

    PLC documentation link->outside

    ASCII->bin link->outside

    Table read link->outside

Evaluation of cycling tables

- The table uses a cycle counter, which increments with table overflow and decrements with a table underflow
- The position of the slave results as a sum of the product of the slave difference in each master period the cycle counter value
- The current determination of the slave position is computed with a linear interpolation

- The current slave velocity will be derived, if the derivation conditions are met. Otherwise a numerical procedure will be used to determine the current slave velocity
- The current slave acceleration will be determined in a numerical procedure

4.6.3 Processing Modes

For every table associated with a multi-table slave axis there is a **scaling** consisting of the master offset, master slope, slave offset, slave slope and table weighting, with which the table can be similarly scaled to any desired magnitude in both co-ordinates. This makes it possible to restrict the system to a small number of **normalised tables**. Processing may take place in a linear or cyclic manner.

Reference System

The scaled axis positions are always stated absolutely.

Cyclic Processing

The rule for cyclic processing is that the modulo operations are performed at the **lowest level**, i.e. immediately before the table is evaluated.

- The table has a **cycle counter** that is incremented in the case of table overflow and decremented in the case of table underflow.
- The master position is given in absolute terms, not modulo.
- In the case of table overflow/underflow, the slave difference per master period, multiplied by the cycle counter, is added to the slave position.

Table types

It is usual for the total travel of a multi-table slave axis to be divided into a number of segments. Individual (normalised) tables are assigned to these (**profile tables**): start table, working table(s), stop table. These tables are worked through in sequence. On top of these there are **correction tables**, only valid over restricted regions, processed in parallel with the profile tables. The profile tables can be processed in a linear or cyclic mode; the correction tables are always processed in linear mode.

Scaling

Table 1: Scaling Parameters for a Table

Scaling parameter	Meaning and boundary conditions
RangeLow RL	Lower limit for the table's range of validity (*RL* < *RH*)
RangeLow RH	Upper limit for the table's range of validity (*RH* > *RL*)
Table weighting W	Weighting with which the table is entered into a parallel convex combination of tables (*W* > *0.0*)
Master-Offset MO	Offset to the master position *[0][0]***T**
Master slope MS	Scaling factor of the table's master position: *[k][0]* <- *MS* × *[k][0]***TT**
Slave-Offset SO	Offset to the slave position *[0][1]***T**
Slave slope SS	Scaling factor of the table's slave position: *[k][1]* <- *SS* × *[k][1]***TT**

It is not the purpose of on-line setting of the scaling to change the parameters while the table is active, but to set the parameters before the table is made active. **Only the correction tables are to be used to correct an active table.** (The parameters of these correction tables must also not be changed while the table is active.)

Notes on scaling correction tables:

- RangeLow = *-DBL_MAX* is automatically set by the NC.

- RangeHigh = *-DBK_MAX* is automatically set by the NC.
- Table weighting is set by the user (usually *1.0*).
- The master offset is automatically set by the NC.
- Master slope, slave offset and slave slope are set by the user.

Interfaces

4.6.4 Procedure

The sequence of events consists of the coupling preparations (moving the axes to the coupling positions) coupling, starting the master axis, parallel table change, possible serial **on-line activation of correction tables**, braking the slave axis (directly via the master, or indirectly via the table(s)) and uncoupling.

Preparation

1. The master axis is moved in PTP operation to the coupling position (=start position).
2. The slave position associated with this master position is found from the table(s). Reading the table only yields the table value. The user must calculate from this using any offsets, slopes or weightings that may have been provided with the coupling, along with the planned composition of the table.
3. The slave axis is moved in PTP mode to the coupling position (=start position).

Coupling/Start

The coupling is made off-line (slave and master are stationary).

The slave then starts when the master starts, and when the table contents require it (the slave position is not constant).

Online: General

The table master-slave coupling is a **linear path controller**.

In the case of table overflow/underflow of cyclic tables, the slave difference per master period, multiplied by the cycle counter, is added to the slave position.

There is no separate stop for the slave axis.

Online: Table Change and Table Composition

Changing the correction table is prepared by the PLC to the extent that the table scaling for each table is set at an early enough stage by the PLC. The scaling parameters include specification of the validity intervals (in terms of absolute positions) for each table. The actual table change is performed by the NC.

If a number of tables (several profile tables or profile tables and correction tables) are active at the same time, the slave position is calculated as follows (the index *i* refers in each case to the *i*-th active table):

P_mi current absolute master position,

SO_i slave offset of the *i*-th table,

SS_i slave slope of the *i*-th table,

MO_i master offset of the *i*-th table,

MS_i master slope of the *i*-th table,

W_i weighting of the *i*-th table.

Table 2: Determination of the slave position using a number of parallel tables

Scaling parameter	Meaning and boundary conditions
one linear table	P_si(P_m) = SO_i + SS_i × **T**_i(MO_i + MS_i × P_m)

Scaling parameter	Meaning and boundary conditions
one cyclic table	P_si(P_m) = SO_i + SS_i × **T**_i([MO_i + MS_i × P_m]mod(•))
all profile tables	P_s = Σ_i W_i × P_si(P_m)

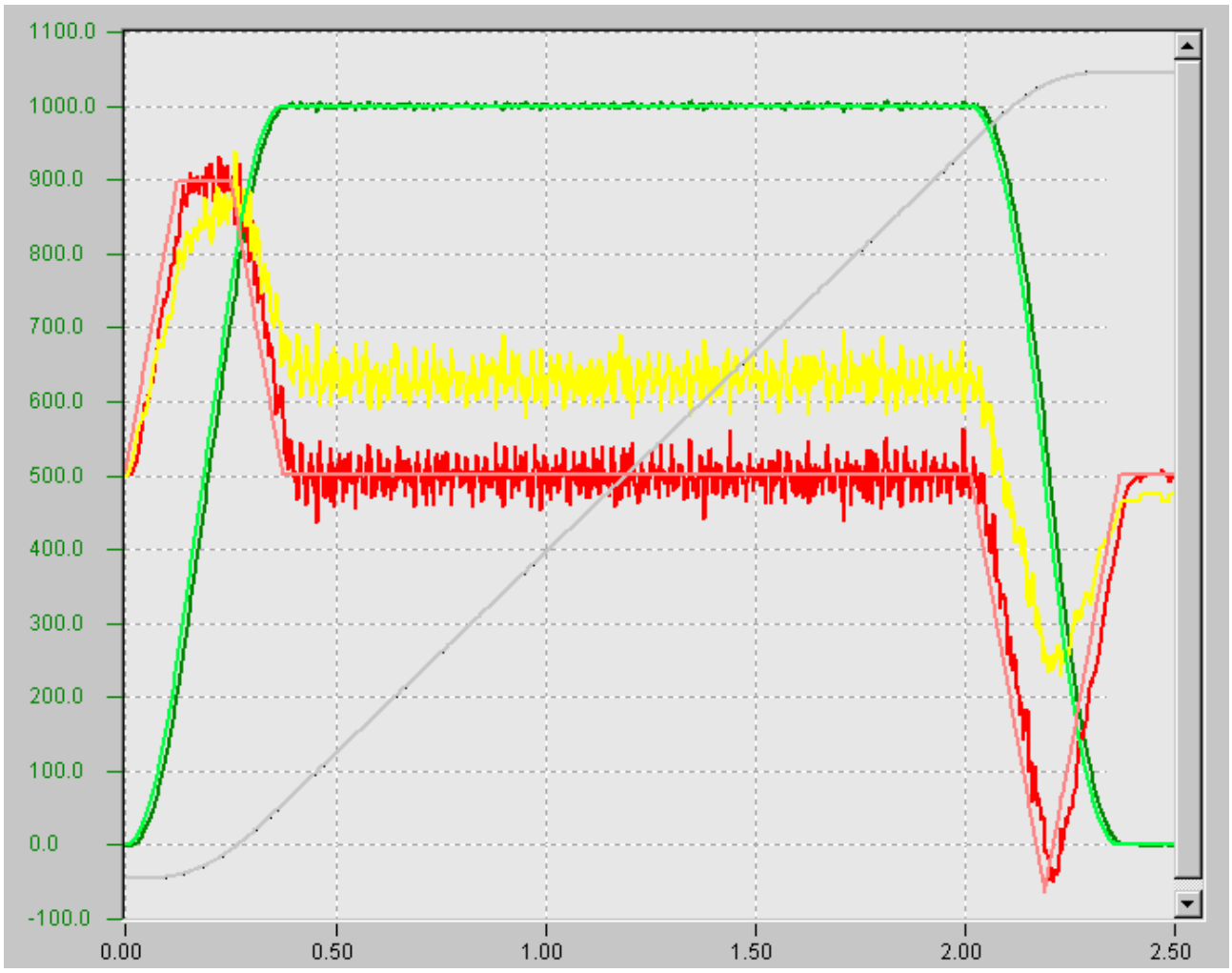


Fig. 4: Linear serial processing

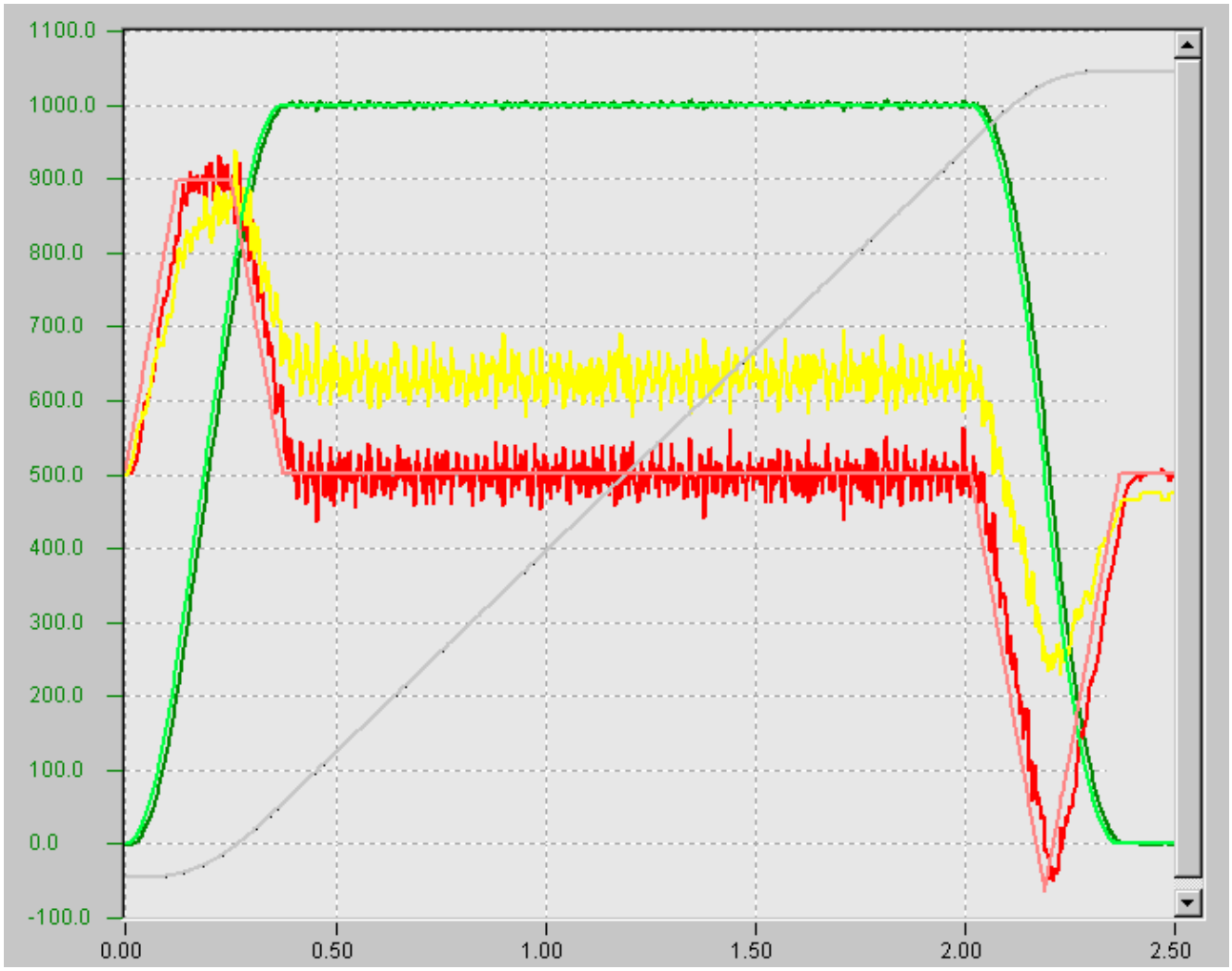


Fig. 5: Linear/cyclic/linear serial processing

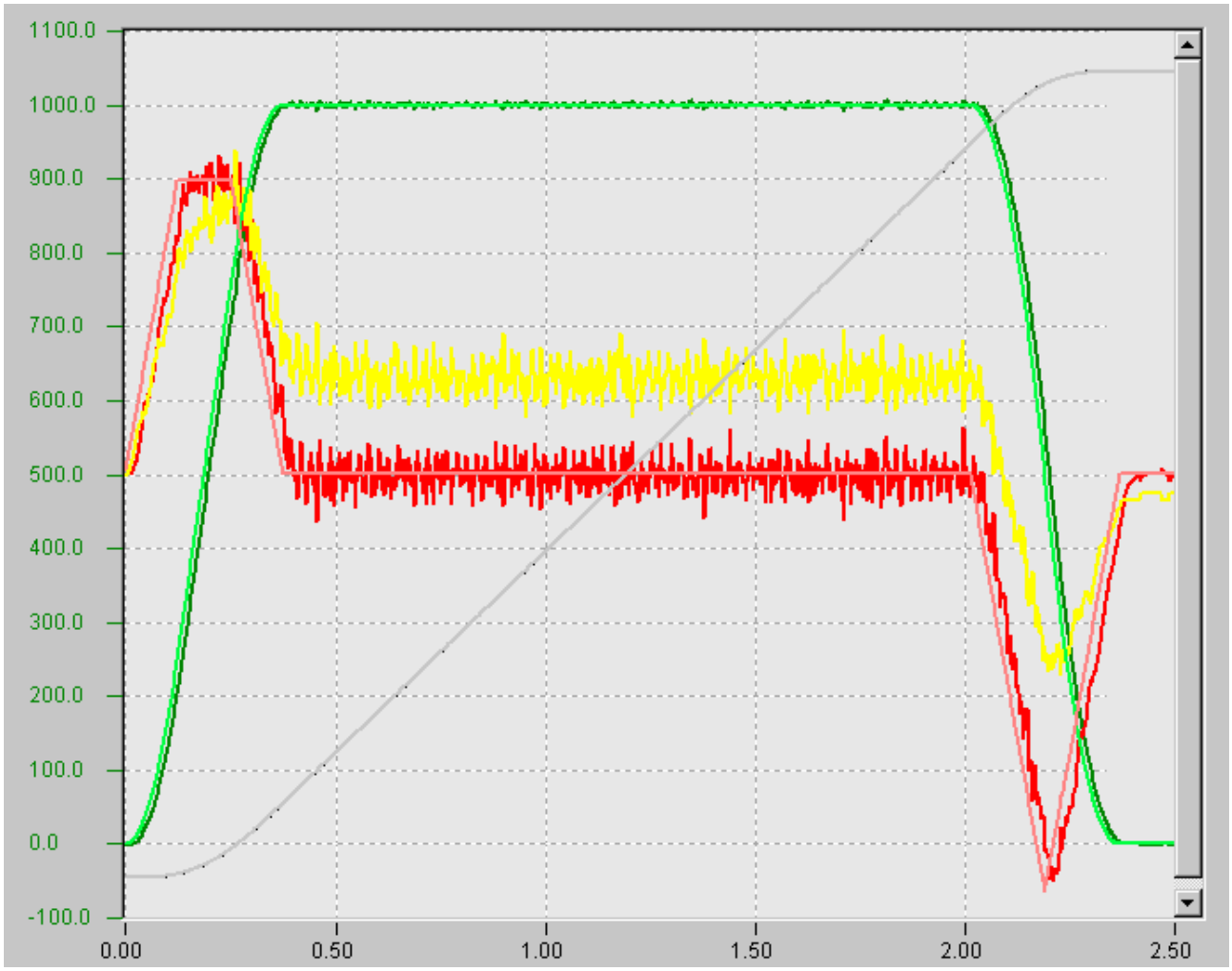


Fig. 6: TCNCSlaveMultiTabularSequence3

Linear/cyclic/linear serial processing with parallel correction

Online: Deactivating Cyclic Tables

The end of cyclic processing of a table is requested through the PLC by the user, and takes place at the end of the relevant cyclic table.

Online: Activation of Correction Tables

Correction tables are processed in parallel with the profile tables.

There are two possibilities for the activation of correction tables:

- Instantaneous activation: The user scales and activates the correction tables, they are then instantaneously activated in the NC.
- Activation by transcending an activation position: the correction tables are scaled and logically activated by the user befrore transcending the activation position. The actvation position is a parameter of the logical activation. After transcending the activation position the correction is automatically activated by the NC.

Correction tables are deactivate themselves automatically when the correction has been worked through.

Notes on **scaling correction tables**:

- RangeLow = *-DBL_MAX* is automatically set by the NC.
- RangeHigh = *DBK_MAX* is automatically set by the NC.
- Table weighting is set by the user (usually *1.0*).
- The master offset is automatically set by the NC.
- Master slope, slave offset and slave slope are set by the user.

Error Handling

There is **no explicit error handling** if the top or bottom of linear tables is passed. The slave axis position value remains instantly stationary at the relevant table edge, creating (depending on the dynamic state and the table composition) a following error.

Stopping/Uncoupling

The slave axis either stops when the master stops (linear path control) or when the table contents require it (the slave position is constant over a wide range of master positions).

Uncoupling can take place off-line (master and slave are stationary) or semi-off-line (only the slave is stationary).

Interfaces

- **Table reading**
  - 1. PLC Blocks
- **Set scaling**
  - 1. PLC Blocks
- **Coupling**
  - 1. System Manager
  - 2. PLC Blocks
- **Uncoupling**
  - 1. System Manager
  - 2. PLC Blocks

5 TcNcCamming

In many applications it is necessary to synchronise two or more axes. Axes can be coupled together in the TwinCAT NC PTP. A master axis is then actively controlled, and the position of one or more coupled slave axes is synchronously controlled by the NC.

The simplest type of coupling is linear coupling with a fixed ratio of transmission (an electronic gearbox).

Some applications require a more complex coupling of master and slave, one which can not be described by a simple mathematical formula. Such a dependency can be described by means of a table that specifies an associated slave position for every master position.

The TwinCAT NC PTP offers the possibility of coupling a slave axis to a master axis by means of a table (electronic cam plate) Here the table contains a certain number of prescribed reference points, and the NC interpolates position and speed between them.

The TcNcCamming library contains function blocks for handling cam plates. Two types of cam plates are supported.

One option is a cam plate in the form of a 2-column table containing master and slave positions (standard table). The master column defines interpolation points via the travel path of the master, ascending from a minimum position value to a maximum value. The associated slave position is determined from the second column using the interpolation points of the table. Values between these points are interpolated.

Another option is to define a cam plate as a so-called motion function. A motion function is a single-column table of interpolation points. Each interpolation point not only contains a position, but a complete description of the shape of the curve within a section (segment) of the cam plate. In addition to the master and slave position at the start of the segment, the shape of the function for example is specified up to the next interpolation point in the form of a mathematical function. Using this procedure, a motion function requires only very few interpolation points. Despite this, each point between the interpolation points is precisely defined through the mathematical function, and there are no uncertainties due to interpolation.

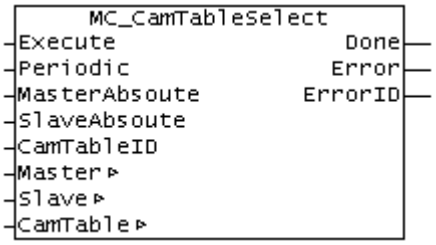
Unlike a standard table, the points of a motion function can be manipulated at run time. The system ensures that a manipulation only becomes effective once an alteration has no direct influence on the slave. Position jumps are thus avoided.

With the extension of the cam plate functionality towards motion functions, PLCopen-compliant function blocks were introduced. It is recommended to use these blocks (names starting with *MC...*) instead of the older blocks (names starting with *CamTable...*). However, the older blocks still remain functional.

The TwinCAT PLC library **TcNcCamming.lib**, available as an additional product, provides easy management of table-based axis coupling. Example programs related to cam plates and motion functions use this library.

5.1 PLCopen Blocks and Motion Functions

5.1.1 MC_CamTableSelect



With the function block MC_CamTableSelect, a table can be specified and loaded into the NC. The block creates a new table and simultaneously fills it with data provided by the PLC.

MC_CamTableSelect does not have to be used, if a table created with the TwinCAT cam plate editor is to be used. In this case, simple coupling with MC_CamIn is sufficient.

VAR_INPUT

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

Execute : The command is executed with rising edge.

Periodic : TRUE = periodic execution, FALSE = no periodic execution

MasterAbsolute : TRUE = master absolute, FALSE = master relative

SlaveAbsolute : TRUE = slave absolute, FALSE = slave relative.

CamTableID : ID of the loaded table [▶ 54].

VAR_OUTPUT

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

Done : Becomes TRUE, if the function was executed successfully.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

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

Master : Axis structure of the master.

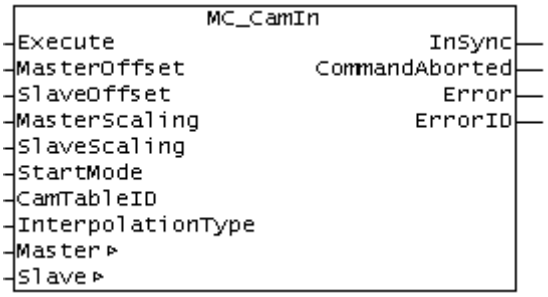
Slave : Axis structure of the slave.

CamTable : Reference to the table [▶ 54] (structure).

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.2 MC_CamIn



The function block MC_CamIn activates master-slave coupling with a certain table.

There is an extended function block MC_CamInExt [▶ 39] available.

Important: Detailed information about master-slave coupling with cam tables. [▶ 41]

VAR_INPUT

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

Execute : The command is executed with rising edge.

MasterOffset : Offset for the master positions.

SlaveOffset : Offset for the slave positions.

MasterScaling : Factor for master positions (default 1.0).

SlaveScaling : Factor for slave positions (default 1.0).

StartMode : Starting mode [▶ 55] for coupled motion. Valid values are START_ABSOLUTE und START_RELATIVE.

CamTableID : Table ID [▶ 54].

InterpolationType : Interpolation type of the table data. The interpolation type is not used for motion functions.

VAR_OUTPUT

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

InSync : Becomes TRUE, if the coupling was successful.

CommandAborted : Becomes TRUE, if the coupling could not be carried out.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

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

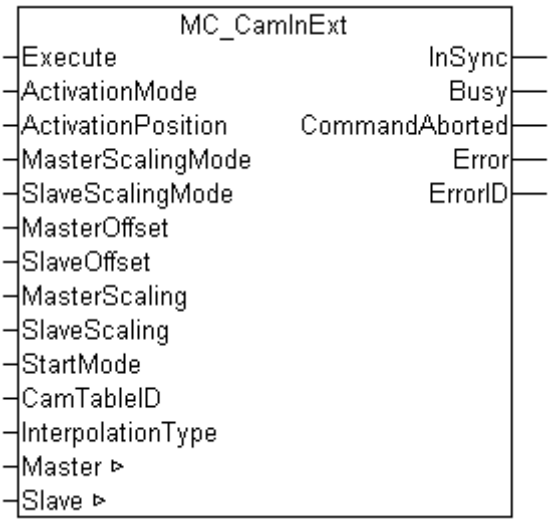
Master : Axis structure of the master.

Slave : Axis structure of the slave.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.3 MC_CamInExt



The function block *MC_CamInExt* activates master-slave coupling with a certain cam plate. In contrast to the standard block (MC_CamIn [▶ 38]), *MC_CamInExt* enables switching to a new cam plate in coupled state. The switching rules, in particular the time or position, can be specified.

To use the extended capabilities of *MC_CamInExt*, the block must be used even for the initial coupling. If an axis is coupled with *MC_CamIn*, it will not be possible to change cam plates later on.

Important:

Detailed information about master-slave coupling with cam tables. [▶ 41]

ActivationMode [▶ 60] (coupling and switching of cam plates)

StartMode [▶ 55]

ScalingMode [▶ 63]

VAR_INPUT

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

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

Execute : The command is executed with rising edge.

ActivationMode : ActivationMode [▶ 60] specifies the switching time or position (only if coupling is already active)

ActivationPosition : Activation position required for position-based activation modes.

MasterScalingMode : Specifies the cam plate scaling type for master scaling (MC_CamScalingMode [▶ 63]).

SlaveScalingMode : Specifies the cam plate scaling type for slave scaling (MC_CamScalingMode [▶ 63]).

MasterOffset : Offset for the master positions.

SlaveOffset : Offset for the slave positions.

MasterScaling : Factor for master positions (default 1.0).

SlaveScaling : Factor for slave positions (default 1.0).

StartMode : Starting mode [▶ 55] for coupled motion.

CamTableID : cam plate ID.

InterpolationType : Interpolation type [▶ 56] of the cam plate data.

VAR_OUTPUT

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

InSync : Becomes TRUE, if the coupling was successful.

Busy : Becomes TRUE, as soon as the function block gets busy and becomes FALSE when the operation is finished and the block can be triggered again

CommandAborted : Becomes TRUE, if the coupling could not be carried out.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

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

Master : Axis structure of the master.

Slave : Axis structure of the slave.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10 Build 1251	PC (i386)	TcNcCamming.Lib

5.1.4 Axis coupling with cam plates

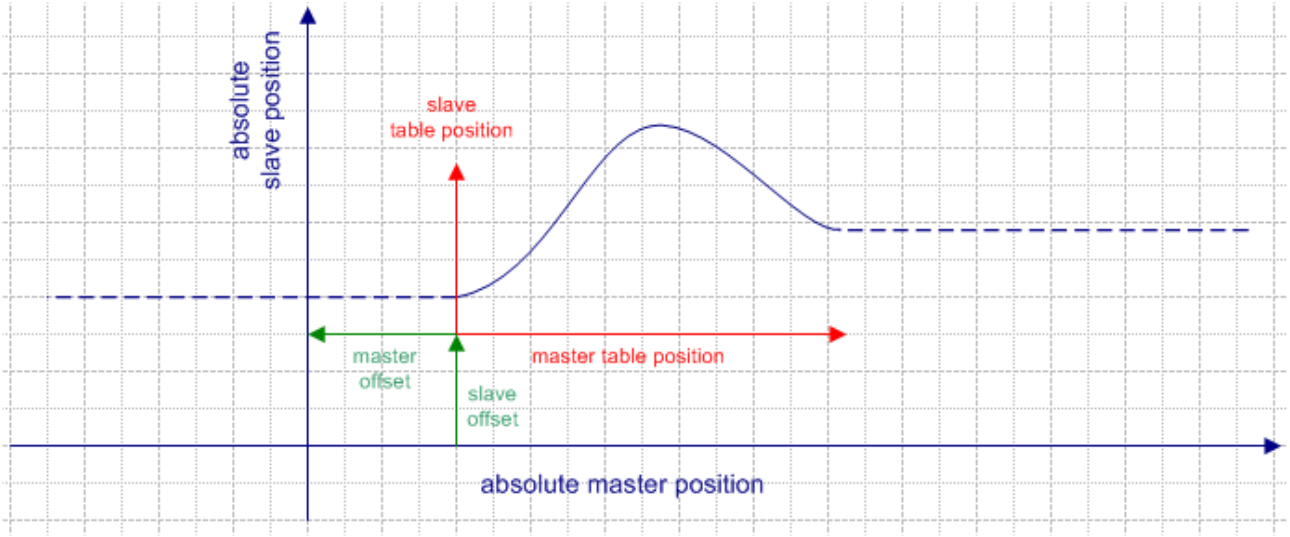
The function block MC_CamIn [▶ 38] can be used to establish a cam plate coupling (or table coupling) between a master axis and a slave axis. Note that prior to the coupling the slave axis has to be at a position defined by the cam plate. After the coupling and once the master has been started, the slave position is calculated directly from the cam plate. The slave axis is therefore not slowly synchronized with the cam plate, but it will jump if it is not already at the table position.

In practice the question arises what position the slave should be in prior to the coupling, and how this is calculated. The following figures illustrate the procedure.

- For all subsequent calculations only axis set positions are used. The actual positions are not used in the calculations, since they would lead to calculation errors, particularly with cyclic cam plates.
- Only absolute table couplings are considered. For relative couplings, the coupling position of the master or slave axis is considered in the calculations as an additional offset.

Linear cam plates

A linear cam plate is only defined via a limited master position range. Outside this range the slave position is defined by the first or last table position. The slave therefore stops at the table edges as soon as the master leaves the defined range.



The diagram shows that the absolute axis coordinate system (blue) does not have to be identical to the cam plate coordinate system (red). The cam plate coordinate system may be offset by a master offset or a slave offset. Scaling is also possible.

The slave position relating to a certain master position can be determined via the function block MC_ReadCamTableSlaveDynamics [▶ 45]. The function block refers to the raw table data, which means that offsets and scaling factors have to be considered via the PLC program itself. Initially, the master offset is added to the current master position. If the cam plate is to be scaled, it is divided by this scaling factor.

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

The master table position is used as input parameter for the function block MC_ReadCamTableSlaveDynamics [▶ 45]. The result is converted to an absolute slave position with slave offset and scaling, if necessary.

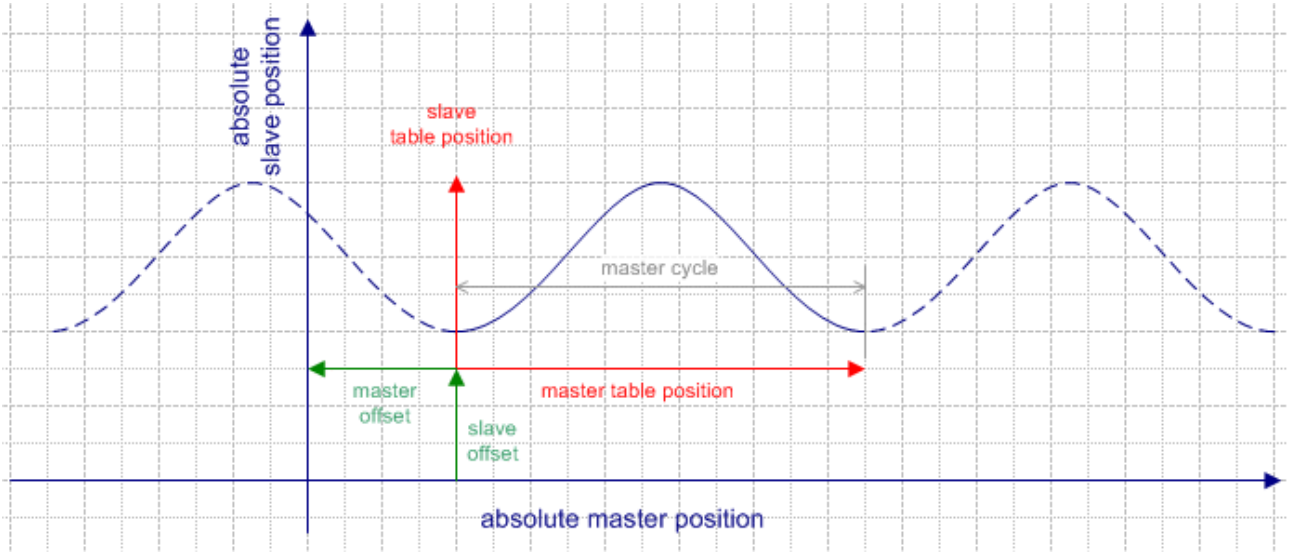
```
SlaveTablePosition := ReadSlaveDynamics.SlavePosition;

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

The slave is moved to this position prior to the coupling. Alternatively, the master may be moved to a position that corresponds to the current slave position. However, generally this position cannot be determined from the cam plate, since the cam plate may be ambiguous.

Since the master offset is added in the first formula, a positive offset leads to the cam plate coordinate system being shifted to the left in negative direction. Accordingly, the master offset in the diagram is negative. A positive slave offset leads to the cam plate coordinate system being shifted upwards in positive direction.

Cyclic cam plates without stroke

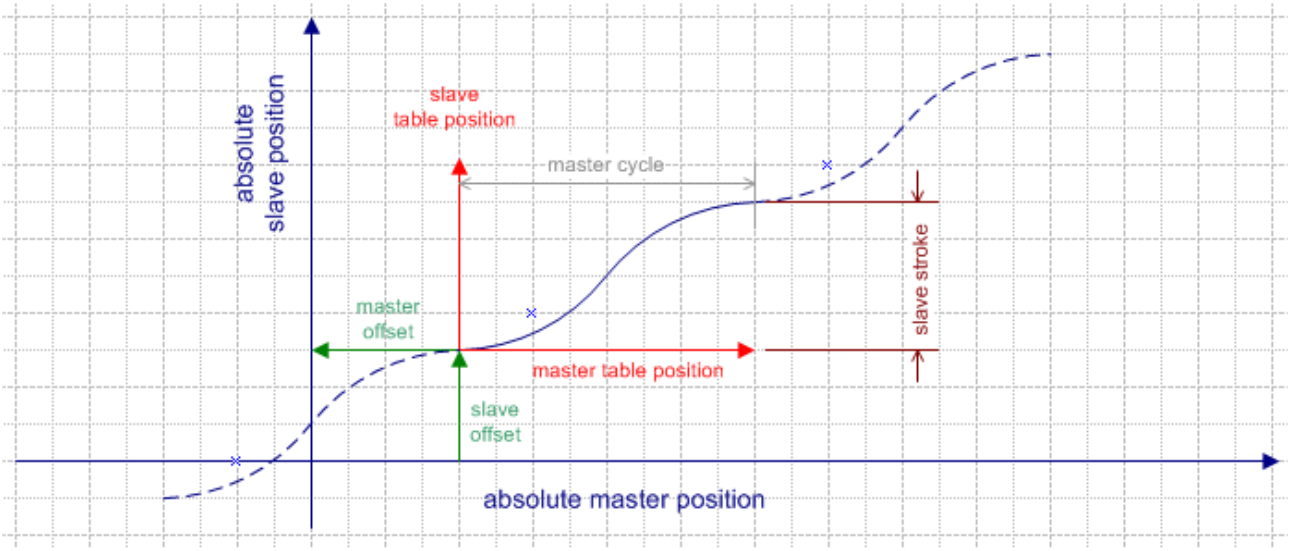
A cyclic cam plate without stroke is characterized by the fact that the slave start and end positions in the table are identical. The slave therefore moves cyclically within a defined range, without changing its position permanently in a particular direction.



For these cam plate types, master/slave coupling requires the same preparation as for a linear cam plate. The starting position of the slave can therefore be calculated as described above. It is not necessary to use the modulo position of the master for the calculation, since the absolute position is already correctly taken into account via the coupling command.

Cyclic cam plates with stroke

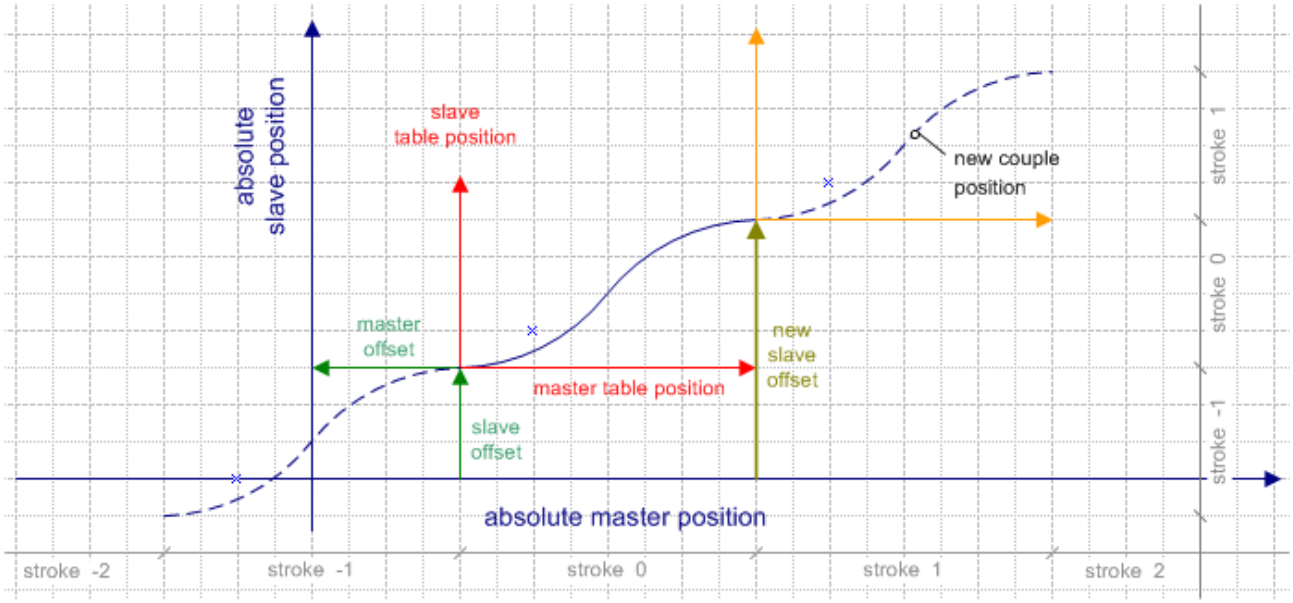
The stroke of a cyclic cam plate is the difference between the last and the first table position of the slave.



Such a cam plate is continued cyclically at the end of the table. The slave position does not jump back to the initial table value. Instead, the motion continues steadily. With each new cycle, the stroke is therefore added as an additional internal slave offset or subtracted if the motion is reversed.

Uncoupling and re-coupling for cyclic cam plates with stroke

If a slave is coupled to a cam plate with stroke, the coupling is always done in the basic cycle (red coordinate system). If the slave is uncoupled after a few cycles and then re-coupled, the slave position returns to the basic cycle. If necessary, this behavior has to be taken into account and compensated by re-calculating the slave offset.




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

The master table position is used as input parameter for the function block MC_ReadCamTableSlaveDynamics [▶ 45]. The result is converted to an absolute slave position with slave offset and scaling, if necessary. In addition, the number of pending strokes must be calculated and added to the slave position.

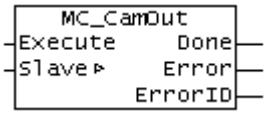
```
SlaveTablePosition := ReadSlaveDynamics.SlavePosition;
```

```
NumberOfStrokes := MODTURNS( (SlavePosition - SlaveOffset), SlaveStroke );
```
(See MODTURNS documentation)

```
NewSlaveOffset := SlaveOffset + (SlaveStroke * NumberOfStrokes );
```

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

5.1.5 MC_CamOut



The function block MC_CamOut deactivates a master-slave coupling.

NOTE

If the Set Point Generator Type is set to '7 Phases (optimized)', the slave axis will reduce its acceleration to zero after it is being decoupled and it will then continue moving endless at constant velocity. The decoupled axis will not be positioned to any target position. The behavior is comparable to a move commanded by MC_MoveVelocity.

With TwinCAT 2.10 the set point generator type is selectable.

From TwinCAT 2.11 the setting is fixed to '7 Phases (optimized)'.

If a project is upgraded from TwinCAT 2.10 to TwinCAT 2.11, the behavior will be as described here.

After updating an existing application to TwinCAT 2.11, it might be necessary to adapt the PLC program.

VAR_INPUT

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

Execute : The command is executed with rising edge.

VAR_OUTPUT

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

Done : Becomes TRUE, if decoupling was successful.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

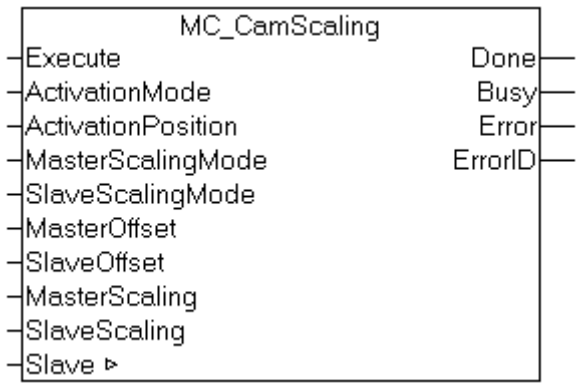
```
VAR_IN_OUT
    Slave : NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Axis structure of the slave.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.6 MC_CamScaling



A cam plate coupling can be scaled with the function block MC_CamScaling. The raw table data of the cam plate are not affected - the scaling refers to an existing master/slave coupling. The following parameters can be modified: scaling factors for master and slave, and offsets for the cam plate within the coordinate system.

Optionally, the modification will only take effect from a certain master position, enabling precise scaling during the motion. Caution when scaling during motion! The slave position at the time of scaling should only be affected slightly by the change.

VAR_INPUT

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

```
    SlaveScalingMode :   MC_CamScalingMode;
    MasterOffset :       LREAL;
    SlaveOffset :        LREAL;
    MasterScaling :      LREAL := 1.0;
    SlaveScaling :       LREAL := 1.0;
END_VAR
```

Execute : The command is executed with rising edge.

ActivationMode : defines when and how the cam table is scaled. (MC_CamActivationMode [▶ 60])

ActivationPosition : optional master position where the table is scaled (depending on ActivationMode).

MasterScalingMode : Type of master scaling. (MC_CamScalingMode [▶ 63])

SlaveScalingMode : Type of slave scaling. (MC_CamScalingMode [▶ 63])

MasterOffset : Offset for the master positions.

SlaveOffset : Offset for the slave positions.

MasterScaling : Factor for master positions (default 1.0).

SlaveScaling : Factor for slave positions (default 1.0).

VAR_OUTPUT

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

Done : Becomes TRUE if scaling was successful.

Busy : Becomes TRUE, as soon as the function block gets busy and becomes FALSE when the operation is finished and the block can be triggered again

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

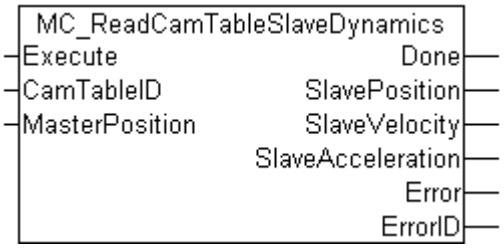
```
VAR_IN_OUT
    Slave :     NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9 Build 1001	PC (i386)	TcNcCamming.Lib

5.1.7 MC_ReadCamTableSlaveDynamics



The function block MC_ReadCamTableSlaveDynamics can be used to determine the slave dynamics at a certain point of a cam plate table. The function evaluates the raw table data. Any scaling of the cam plate is not taken into account.

For older cam plate table types [▶ 55], not all dynamic parameters can be determined. The following overview shows the expected result:

MC_TableType_TABULARTYPE_MOTIONFUNC : slave position, velocity and acceleration are determined.

MC_TableType_TABULARTYPE_EQUIDIST : slave position and velocity are determined. The acceleration is always 0.

MC_TableType_TABULARTYPE_NONEQUIDIST : the slave position is determined. Velocity and acceleration are always 0.

VAR_INPUT

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

Execute : The command is executed with rising edge.

CamTableID : Table ID [▶ 54].

MasterPosition : Master position within the table for which the slave dynamics is to be determined.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was executed successfully.

SlavePosition : Position of the slave within the cam plate table at the specified master position.

SlaveVelocity : Velocity of the slave within the cam plate table at the specified master position.

SlaveAcceleration : Acceleration of the slave within the cam plate table at the specified master position.

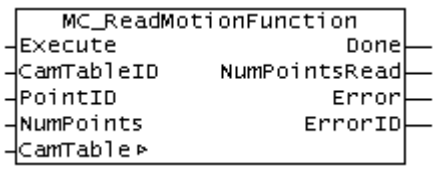
Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.8 MC_ReadMotionFunction



The function block MC_ReadMotionFunction can be used to read the data of a motion function. Either the complete function with all interpolation points or only a part can be read. The data are stored within the PLC in the structure described by CamTable.

VAR_INPUT

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

Execute : The command is executed with rising edge.

CamTableID : ID of the loaded table [▶ 54].

PointID : Point ID [▶ 57] of the first point of the motion function to be read.

NumPoints : Number of motion function points to be read. For reading all points, 0 can be specified here, in which case the number that is actually read is returned in the output variable *NumPointsRead*.

VAR_OUTPUT

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

Done : Becomes TRUE, if the data were read successfully.

NumPointsRead : The number of points that were actually read. The number may be less or equal *NumPoints*.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

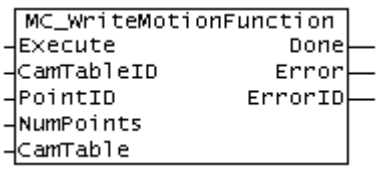
```
VAR_IN_OUT
    CamTable :      MC_CAM_REF;
ND_VAR
```

CamTable : Reference to the table [▶ 54] (structure).

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.9 MC_WriteMotionFunction



The function block *MC_WriteMotionFunction* can be used to write the data of a motion function into the NC. Either the complete function with all interpolation points or only a part can be written. First, the data are stored within the PLC in the structure described by *CamTable*.

The function block MC_SetCamOnlineChangeMode [▶ 50] can be used to specify when the data are read into the cam plate. If activation of the data is to be delayed until the master reaches a certain position, the system will initially queue the written data and activate them at the master position. The function AxisCamDataQueued can be used to check whether data have been queued (i.e. written but not yet activated).

VAR_INPUT

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

Execute: The command is executed with rising edge.

CamTableID: ID of the loaded table [▶ 54].

PointID: Point ID [▶ 57] of the first point of the motion function to be written.

NumPoints: Number of motion function points to be written.

VAR_OUTPUT

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

Done: Becomes TRUE, if the data were read successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: Supplies the error number when the Error output is set

VAR_IN_OUT

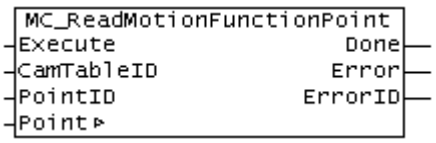
```
VAR_IN_OUT
    CamTable :      MC_CAM_REF;
ND_VAR
```

CamTable: Reference to the table [▶ 54] (structure). The start address CamTable.pArray must address to the first point to be written.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.10 MC_ReadMotionFunctionPoint



The function block MC_ReadMotionFunctionPoint can be used to read the data of a motion function interpolation point.

VAR_INPUT

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

Execute : The command is executed with rising edge.

CamTableID : ID of the loaded table [▶ 54].

PointID : Point ID [▶ 57] of the first point of the motion function to be read.

VAR_OUTPUT

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

Done : Becomes TRUE, if the data were read successfully.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

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

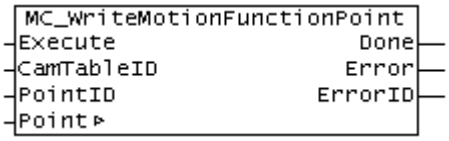
MC_MotionFunctionPoint

Point : Data structure [▶ 56] containing the data of a motion function interpolation point

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.11 MC_WriteMotionFunctionPoint



The function block *MC_WriteMotionFunctionPoint* can be used to write the data of a motion function interpolation point.

The function block MC_SetCamOnlineChangeMode [▶ 50] can be used to specify when the data are read into the cam plate. If activation of the data is to be delayed until the master reaches a certain position, the system will initially queue the written data and activate them at the master position. The function AxisCamDataQueued can be used to check whether data have been queued (i.e. written but not yet activated).

VAR_INPUT

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

Execute: The command is executed with rising edge.

CamTableID: ID of the loaded table [▶ 54].

PointID: Point ID [▶ 57] of the first point of the motion function to be written.

VAR_OUTPUT

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

Done: Becomes TRUE, if the data were written successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: Supplies the error number when the Error output is set

VAR_IN_OUT

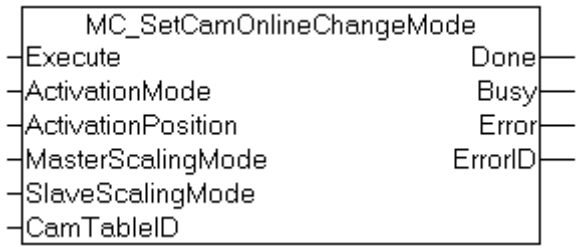
```
VAR_IN_OUT
    Point : MC_MotionFunctionPoint;
ND_VAR
```

Point: Data structure [▶ 56] containing the data of a motion function interpolation point

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.12 MC_SetCamOnlineChangeMode



Cam plate data can be modified at run time via the PLC (see MC_WriteMotionFunction [▶ 47], C_WMriteMotionFunctionPoint [▶ 49]). The function block MC_SetCamOnlineChangeMode is used to specify when and how these changes take effect.

This function specifies the activation mode for modifications but does not affect a change or change-over of cam plates.

VAR_INPUT

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

Execute : The command is executed with rising edge.

ActivationMode: defines when and how the cam table is scaled. (MC_CamActivationMode [▶ 60])

ActivationPosition: optional master position where the table is scaled (depending on ActivationMode).

MasterScalingMode: Type of master scaling. (MC_CamScalingMode [▶ 63])

SlaveScalingMode: Type of slave scaling. (MC_CamScalingMode [▶ 63])

CamTableID: Table ID [▶ 54].

VAR_OUTPUT

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

Done: Becomes TRUE if scaling was successful.

Busy: Becomes TRUE, as soon as the function block gets busy and becomes FALSE when the operation is finished and the block can be triggered again

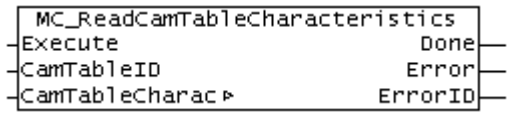
Error: Becomes TRUE, as soon as an error occurs.

ErrorID: Supplies the error number when the Error output is set

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9 Build 1001	PC (i386)	TcNcCamming.Lib

5.1.13 MC_ReadCamTableCharacteristics



The function block MC_ReadCamTableCharacteristics is used to calculate and read the characteristic parameters of a motion function.

VAR_INPUT

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

Execute : The command is executed with rising edge.

CamTableID : Table ID [▶ 54].

VAR_OUTPUT

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

Done : Becomes TRUE, if the calculation was carried out successfully.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

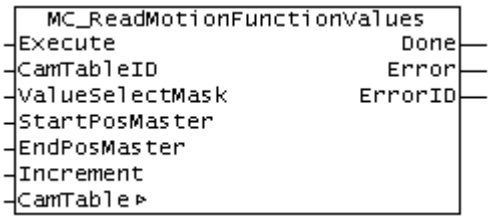
```
VAR_IN_OUT
    CamTableCharac : MC_TableCharacValues;
END_VAR
```

CamTableCharac: Data structure [▶ 58] with characteristic parameters of the motion function

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.7.0 & TwinCAT v2.8	PC (i386)	TcNcCamming.Lib

5.1.14 MC_ReadMotionFunctionValues



The function block MC_ReadMotionFunctionValues can be used to read the interpolated data of a motion function in the form of a table.

VAR_INPUT

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

MC_ValueSelectType MC_CAM_ID

Execute : The command is executed with rising edge.

CamTableID : ID of the loaded table [▶ 54] (motion function type).

ValueSelectMask : The selection mask [▶ 60] can be used to specify which data are to be interpolated and returned. The number of columns of the data structure described by CamTable must match the number of columns defined by ValueSelectMask. If, for example, only position data are read, the number of columns is 2 (master and slave position). With each further derivative (speed, acceleration, jerk), the number of columns increases by 1.

StartPosMaster : Position value of the master axis of the first interpolated point

EndPosMaster : Position value of the master axis of the last interpolated point

Increment : Interpolation step size

VAR_OUTPUT

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

Done : Becomes TRUE, if the data were read successfully.

Error : Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

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

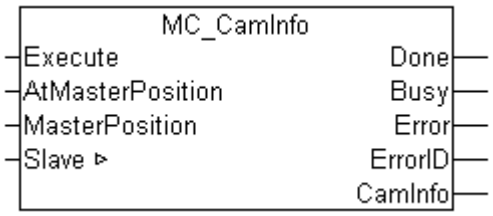
MC_CAM_REF [▶ 54]

CamTable : Reference to the table [▶ 54] (structure).

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.15 MC_CamInfo



The function block MC_CamInfo obtains data relating to the current state and current parameterization of a cam plate coupling. The command assumes that the slave axis is coupled by a cam plate. If the AtMasterPosition input is TRUE, not the current state but the state related to the specified master position is determined. The data obtained is placed into the CamInfo data structure.

If the coupled axis group gets into an error situation (e.g. emergency off), the function block will return the most recent valid state of the coupling. The function block must be called before decoupling the slave. The data that has been obtained can be used to restore the coupling to the original axis position.

VAR_INPUT

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

Execute: The command is executed with rising edge.

AtMasterPosition: If *AtMasterPosition* is TRUE the data is determined with reference to the quoted *MasterPosition*. Otherwise the data refers to the current master position.

MasterPosition : Master position to which the data that is determined refers. This input parameter is not necessary if *AtMasterPosition* is FALSE.

VAR_OUTPUT

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

Done: Becomes TRUE when the function has been successfully executed.

Busy: Becomes TRUE as soon as the function block is active and becomes FALSE again as soon as the function has been completed and is ready to be retriggered.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

CamInfo: The data structure CamInfo [▶ 65] contains data that has been obtained of the cam plate coupling.

VAR_IN_OUT

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

Slave: Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9, build 1026 onwards	PC (i386)	TcNcCamming.Lib

5.1.16 Data types

5.1.16.1 MC_CAM_ID

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

Type declaration for table ID.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.2 MC_CAM_REF

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

Table structure [▶ 55] definition. The first parameter *pArray* is a pointer to a data structure containing the table data. The data structure depends on the table type *nTableType*. The number of rows is entered in the component nNoOfRows, the number of columns in nNoOfCols (usually 1 or 2).

Example 1: Structure description of a point table

pArray: Address of a two-dimensional array. The first column contains an ascending list of master positions. The second column contains the associated slave positions.

nArraySize: Size of the array which can be calculated with SIZEOF(Array).

nTableType: The table type is TABULARTYPE_EQUIDIST, if the master positions have the same distance, or TABULARTYPE_NONEQUIDIST it the distance is variable.

nNoOfRows: The number or rows corresponds to the number of table points.

nNoOfCols: The number of columns is 2.

Example 2: Structure description of a motion function

pArray: Address of a one-dimensional array of type *MC_MotionFunctionPoint*. Each array element contains a description of a cam plate interpolation point.

nArraySize: Size of the array which can be calculated with SIZEOF(Array).

nTableType: The table type is TABULARTYPE_MOTIONFUNC.

nNoOfRows: The number or rows corresponds to the number of table points.

nNoOfCols: The number of columns is 1.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.3 MC_StartMode

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

StartMode is used for coupling with cam plates through MC_CamIn [▶ 38] or MC_CamInExt [▶ 39] and defines whether a cam plate is interpreted absolute (based on the origin of the axis coordinate system) or relative to the coupling position. The mode can be specified as absolute or relative separately for both coordinate axes.

With *StartModeabsolut* the cam plate coordinate system is congruent with the axis coordinate system and can be moved through an offset, if required (master or slave offset).

With *StartModerelativ* the origin of the cam plate coordinate system is at the axis position of the respective axis (master or slave) at the time of coupling or cam plate switching. In addition, the cam plate can be shifted by an offset.

The modes MC_STARTMODE_RELATIVE and MC_STARTMODE_MASTERREL_SLAVEABS cannot be used in conjunction with automatic master offset calculation (MC_CamScalingMode [▶ 63]), since this would cause a conflict.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.10 Build 1329	PC (i386)	TcNcCamming.Lib

5.1.16.4 MC_TableType

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

```
values
    (not imperative equidistant) *)
    MC_TableType_TABULARTYPE_NONEQUIDIST := 11,
    (* motion function calculated in runtime *)
    MC_TableType_TABULARTYPE_MOTIONFUNC := 22 );
END_TYPE
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.5 MC_InterpolationType

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

Interpolation type for table data.

Requirements

Development Environement	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.6 MC_MotionFunctionPoint

```
TYPE MC_MotionFunctionPoint :
STRUCT
    PointIndex    : MC_MotionFunctionPoint_ID;

    FunctionType      : MC_MotionFunctionType;

    PointType     : MC_MotionPointType;

    RelIndexNextPoint : MC_MotionFunctionPoint_ID;

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

The data structure *MC_MotionFunctionPoint* describes an interpolation point of a motion function. A motion function is a one-dimensional list (array) of type *MC_MotionFunctionPoint*.

PointIndex: absolute index [▶ 57] of this interpolation point within the motion function. The point index of all interpolation points must increase strictly monotonously and must have no gaps and be greater than 0.

FunctionType: type MC_MotionFunctionType [▶ 58] of the mathematical function between this and the subsequent interpolation point

PointType: type MC_MotionPointType [▶ 57] of this interpolation point.

RelIndexNextPoint: relative reference to the subsequent interpolation point (usually 1).

MasterPos: position of the master axis at this interpolation point

SlavePos: position of the slave axis at this interpolation point

SlaveVelo: velocity of the slave axis at this interpolation point

SlaveAcc: acceleration of the slave axis at this interpolation point

SlaveJerk: jerk of the slave axis at this interpolation point

Some motion laws (MC_MotionFunctionType) allow a symmetry setting. Since for all these functions the jerk at the boundary points is 0, the jerk parameter SlaveJerk can be used to set the symmetry in these cases. A value of 0 means a symmetry of 50%. For the following functions, the value can be set in the range +/- 0.5:

POLYNOM3
POLYNOM5
POLYNOM8
POLYNOM11
SINUSLINIE
MODSINUSLINIE
BESTEHORN
BESCHLTRAPEZ
SINUS_GERADE_KOMBI

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.7 MC_MotionFunctionPoint_ID

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

Type declaration for the point ID of the motion function points

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.8 MC_MotionPointType

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

Type declaration for table ID.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.9 MC_MotionFunctionType

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

Type declaration for motion functions

The motion function type Automatic used in the TwinCAT Cam Design Editor corresponds to MOTIONFUNCTYPE_POLYNOM5_MM.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.10 MC_TableCharacValues

```
TYPE MC_TableCharacValues :
STRUCT
    (* Master Velocity*)
    fMasterVeloNom : LREAL; (* 1. master nominal velocity (normed:
=> 1.0) *)
    (* characteristic slave data *)
    (*==========================*)
    (* Start of cam table *)
    fMasterPosStart : LREAL;    (* 2. master start position
*)
```

```
    fSlavePosStart : LREAL;     (* 3. slave start position *)
    fSlaveVeloStart : LREAL;    (* 4. slave start velocity *)
    fSlaveAccStart : LREAL;     (* 5. slave start acceleration
*)
    fSlaveJerkStart : LREAL;    (* 6. slave start jerk *)
    (* End of cam table*)
    fMasterPosEnd : LREAL;      (* 7. master end position *)
    fSlavePosEnd : LREAL;       (* 8. slave end position *)
    fSlaveVeloEnd : LREAL;      (* 9. slave end velocity *)
    fSlaveAccEnd : LREAL;       (* 10. slave end acceleration
*)
    fSlaveJerkEnd : LREAL;      (* 11. slave end jerk *)
    (* minimum slave position *)
    fMPosAtSPosMin : LREAL;     (* 12. master position AT slave
minimum position *)
    fSlavePosMin : LREAL;       (* 13. slave minimum position
*)
    (* minimum Slave velocity *)
    fMPosAtSVeloMin : LREAL;    (* 14. master position AT slave
minimum velocity *)
    fSlaveVeloMin : LREAL;      (* 15. slave minimum velocity
*)
    (* minimum slave acceleration *)
    fMPosAtSAccMin : LREAL;     (* 16. master position AT slave
minimum acceleration *)
    fSlaveAccMin : LREAL;       (* 17. slave minimum acceleration
*)
    fSVeloAtSAccMin : LREAL;    (* 18. slave velocity AT slave
minimum acceleration *)
    (* minimum slave jerk and dynamic momentum *)
    fSlaveJerkMin : LREAL;      (* 19. slave minimum jerk *)
    fSlaveDynMomMin : LREAL;    (* 20. slave minimum dynamic
momentum (NOT SUPPORTED YET !) *)
    (* maximum slave position *)
    fMPosAtSPosMax : LREAL;     (* 21. master position AT slave
maximum position *)
    fSlavePosMax : LREAL;       (* 22. slave maximum position
*)
    (* maximum Slave velocity *)
    fMPosAtSVeloMax : LREAL;    (* 23. master position AT slave
maximum velocity *)
    fSlaveVeloMax : LREAL;      (* 24. slave maximum velocity
*)
    (* maximum slave acceleration *)
    fMPosAtSAccMax : LREAL;     (* 25. master position AT slave
maximum acceleration *)
    fSlaveAccMax : LREAL;       (* 26. slave maximum acceleration
*)
    fSVeloAtSAccMax : LREAL;    (* 27. slave velocity AT slave
maximum acceleration *)
    (* maximum Slave slave jerk and dynamic momentum *)
    fSlaveJerkMax : LREAL;      (* 28. slave maximum jerk *)
    fSlaveDynMomMax : LREAL;    (* 29. slave maximum dynamic
momentum (NOT SUPPORTED YET !) *)
    (* mean and effective values *)
    fSlaveVeloMean : LREAL;     (* 30. slave mean absolute velocity
(NOT SUPPORTED YET !) *)
    fSlaveAccEff : LREAL;       (* 31. slave effective acceleration
(NOT SUPPORTED YET !) *)
    (* reserved space for future extension *)
    reserved : ARRAY[32..47] OF LREAL;
    (* organization structure of the cam table *)
    CamTableID : UDINT;
    NumberOfRows : UDINT;       (* number of cam table entries, e.
g. number of points *)
    NumberOfColumns : UDINT;    (* number of table columns,
typically 1 or 2 *)
    TableType : UINT;       (* MC_TableType *)
    Periodic : BOOL;
    reserved2 : ARRAY[1..121] OF BYTE;
END_STRUCT
END_TYPE
```

Type declaration for the characteristic values of a motion function.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.11 MC_ValueSelectType

```
TYPE MC_ValueSelectType :
(
    (* a bitmask can be created by adding the following values
*)
    MC_VALUETYPE_POSITION     := 1,
    MC_VALUETYPE_VELOCITY     := 2,
    MC_VALUETYPE_ACCELERATION := 4,
    MC_VALUETYPE_JERK     := 8
);
END_TYPE
```

Type declaration for access to value tables with the functionblock MC_ReadMotionFunctionValues.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.12 MC_TableErrorCodes

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

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcNcCamming.Lib

5.1.16.13 MC_CamActivationMode

```
TYPE
MC_CamActivationMode :

(

    MC_CAMACTIVATION_INSTANTANEOUS,     (* instantaneous change
*)

    MC_CAMACTIVATION_ATMASTERCAMPOS,    (* modify the data at a
defined master

                        position referring to
the cam tables

                    1      master position *)

    MC_CAMACTIVATION_ATMASTERAXISPOS,   (* modify the data at a
```

```
defined master

                        position referring to
the absolute

                        master axis position
*)

    MC_CAMACTIVATION_NEXTCYCLE,     (* modify the data at the
beginning of

                        the next cam table cycle
*)
    MC_CAMACTIVATION_NEXTCYCLEONCE,     (* not
yet implemented!

                        modify the data at the
beginning of

                        the next cam table
cycle, activation

                        is valid for one cycle
only *)

    MC_CAMACTIVATION_ASSOONASPOSSIBLE,  (* modify the data as soon
as the

                        cam table is in a safe
state to

                        change its data *)

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

    MC_CAMACTIVATION_DELETEQUEUEDDATA   (* delete all data which
was written

                        to modify the cam table
but is

                        still not activated
*)

);

END_TYPE
```

MC_CamActivationMode specifies the timing and type of change for a cam plate. Changes can be affected through scaling or through modification of the cam plate data. In either case, not all modes are available.

Following modes are available:

Scaling of cam plates

Cam plates can be scaled with the function block MC_CamScaling [▶ 44]. The following activation modes are available.

MC_CAMACTIVATION_INSTANTANEOUS	Scaling takes effect immediately.
MC_CAMACTIVATION_ATMASTERCAMPOS	Scaling takes effect at a certain master position within the cam plate. The scaling command must be issued ahead of this position.
MC_CAMACTIVATION_ATMASTERAXISPOS	Scaling takes effect at a certain absolute position of the master axis. The scaling command must be issued ahead of this position.
MC_CAMACTIVATION_NEXTCYCLE	For a cyclic cam plate, scaling takes effect at the transition to the next period.
MC_CAMACTIVATION_OFF	No scaling will be done

Setting the mode for changing a cam plate online (Writing of point data)

Function block MC_SetCamOnlineChangeMode [▶ 50] is used to specify when modified cam plate data become active. The PLC uses separate commands for writing the modified data (see WriteMotionFunction [▶ 47] and MC_WriteMotionFunctionPoint [▶ 49]).

MC_CAMACTIVATION_INSTANTANEOUS	The modified cam plate data take effect as soon as they are written by the PLC.
MC_CAMACTIVATION_ATMASTERCAMPOS	The modified cam plate data take effect at a certain master position within the cam plate. The command for writing the data must be issued ahead of this position.
MC_CAMACTIVATION_ATMASTERAXISPOS	The modified cam plate data take effect at a certain absolute master axis position. The command for writing the data must be issued ahead of this position.
MC_CAMACTIVATION_NEXTCYCLE	For a cyclic cam plate, the modified cam plate data take effect at the transition to the next period.
MC_CAMACTIVATION_ASSOONASPOSSIBLE	The modified cam plate data take effect as soon system dynamics allow.
MC_CAMACTIVATION_OFF	Changes in cam plate data are ignored.
MC_CAMACTIVATION_DELETEQUEUEDDATA	Queued cam plate data are deleted. Data are queued if the change was requested at a certain master position or at the end of the cycle, for example.

Coupling with cam plates

Axes can be coupled with cam plates using the function block MC_CamInExt [▶ 39] .*ActivationMode* defines the coupling position.

MC_CAMACTIVATION_INSTANTANEOUS	The cam plate gets active instantaneously.
MC_CAMACTIVATION_ATMASTERCAMPOS	The cam plate will be pending active until a certain master cam plate position is reached. The slave starts moving after the master reached the defined position. This *ActivationMode* cannot be used in combination with MC_StartMode [▶ 55] = MC_STARTMODE_RELATIVE or MC_STARTMODE_MASTERREL_SLAVEABS.
MC_CAMACTIVATION_ATMASTERAXISPOS	The cam plate will be pending active until a certain absolute master position is reached. The slave starts moving after the master reached the defined position.
MC_CAMACTIVATION_NEXTCYCLE	The cam plate will be pending active until the master reaches the next cam plate cycle. The slave starts moving after the master reached the defined position. This *ActivationMode* cannot be used in combination with MC_StartMode [▶ 55] = MC_STARTMODE_RELATIVE or MC_STARTMODE_MASTERREL_SLAVEABS.

Switching cam plates

Cam plates can be switched using MC_CamInExt [▶ 39]. The *ActivationMode* defines the exact master switching position.

MC_CAMACTIVATION_INSTANTANEOUS	Cam plates will be switched instantaneously.
MC_CAMACTIVATION_ATMASTERCAMPOS	Cam plates will be switched at a certain master cam plate position.
MC_CAMACTIVATION_ATMASTERAXISPOS	Cam plates will be switched at a certain absolute master position.
MC_CAMACTIVATION_NEXTCYCLE	Cam plates will be switched at the next cam plate cycle.
MC_CAMACTIVATION_DELETEQUEUEDDATA	A pending switching operation will be cancled.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10 Build 1329	PC (i386)	TcNcCamming.Lib

5.1.16.14 MC_CamScalingMode

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

Type and scope of the scaling of a cam plate coupling via function block MC_CamScaling [▶ 44].

MC_CAMSCALING_USERDEFINED: The scaling and offset are retained unchanged. The user has to calculate the scaling and offset such that a jump in the position is avoided.

MC_CAMSCALING_AUTOOFFSET: The scaling takes effect and the system adjusts the offset such that a jump in the position is avoided. Scaling should nevertheless occur during a phase with slave velocity 0, since otherwise a jump in velocity cannot be avoided.

MC_CAMSCALING_OFF: The scaling and offset are ignored. This mode is used when only slave scaling (i.e. without master scaling) is to be implemented.

Autooffset adjusts the cam plate offsets automatically. *Autooffset* can be used independently for master or slave and is available for switching and scaling of cam plates.

Autooffset

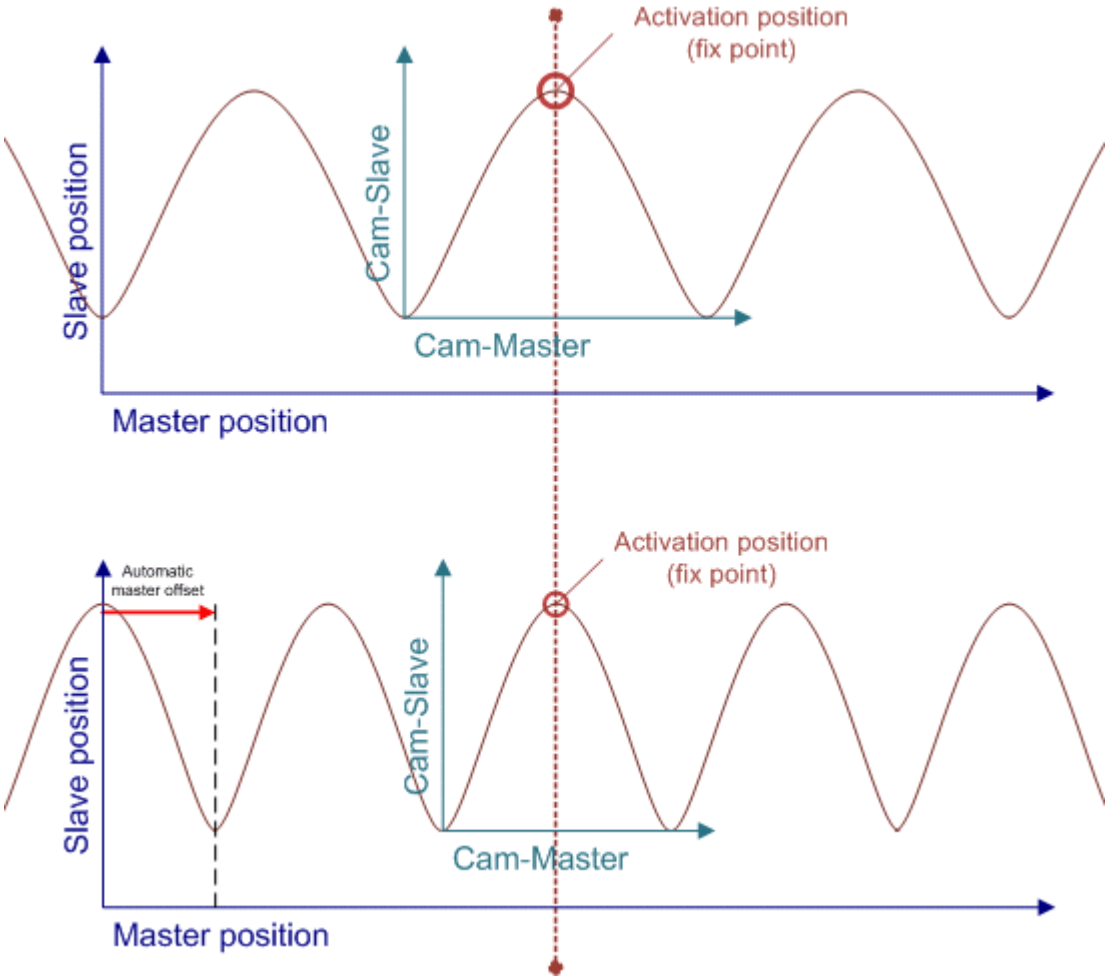
Autooffset adjusts the cam plate offsets automatically. *Autooffset* can be used independently for master or slave and is available for switching and scaling of cam plates.

Master-Autooffset

Master-Autooffset adjusts the master position of a cam plate when switching between cam plates with different cycle or when changing the *MasterScaling* of a cam plate. This function is useful to hold the relative position of a cam plate in the absolute axis coordinate system which depends on the master cycle of a cam plate.

Master-Autooffset calculates the master offset of a cam plate so that the position, relative to the cam plate, does not change. In case the master cycle is changed when switching cam plates or by changing the master scaling, the proportional position relative to the cam cycle will be fixed.

Example: A cam plate with a master cycle of 360° will be scaled by a factor of 2 to a new cycle of 720°. The scaling operation takes place at 90° of the cam plate, which means at 25% from the beginning of the cam cycle. After the scaling operation has been executed, the relative master cam plate position is still at 25% from the beginning of the new master cycle which is at 180° now.



When switching cam plates at the beginning or end of a cam plate cycle (see MC_CamActivationMode [▶ 60]*MC_CAMACTIVATION_NEXTCYCLE*), *Master-Autooffset* will adjust the cam plates to be executed in sequence.

Master-Autooffset cannot be used in combination with a relative coupling mode. There are even more restrictions when coupling master and slave initially because some combinations do not have a unique solution.

		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	—	—	—	—

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

Slave-Autooffset

Slave-Autooffset adjusts the slave offset of a cam plate so that the slave position does not change due to a switching or scaling operation. If *Master-Autooffset* and *Slave-Autooffset* are used at the same time, the master offset will be calculated first and the slave position will then be adjusted with a slave offset.

Requirements

Development Environment/th>	Target System	PLC Libraries to include
TTwinCAT v2.10 Build 1329	PC (i386)	TcNcCamming.Lib

5.1.16.15 MC_CamInfoData

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

    NumberOfRows          : UDINT; (* number of cam table
entries, e. g. number of points *)
    NumberOfColumns           : UDINT; (* number of table
columns, typically 1 or 2 *)
    MasterCamStartPos         : LREAL; (* Master position of
the first cam table point *)
    SlaveCamStartPos          : LREAL; (* Slave position of the
first cam table point *)
    RawMasterPeriod           : LREAL;
    RawSlaveStroke        : LREAL;
    MasterAxisCouplingPos     : LREAL; (* Master axis position
when slave has been coupled *)
    SlaveAxisCouplingPos      : LREAL; (* Slave axis position
when slave has been coupled *)
    MasterAbsolute        : BOOL; (* raw unscaled distance
from first to last master cam table position *)
    SlaveAbsolute         : BOOL; (* raw unscaled distance
from first to last slave cam table position *)
    MasterOffset          : LREAL; (* total master offset
*)
    SlaveOffset           : LREAL; (* total slave offset
*)
    MasterScaling         : LREAL; (* total master scaling
factor *)
    SlaveScaling          : LREAL; (* total slave scaling
factor *)
    SumOfSlaveStrokes         : LREAL; (* sum OF the slave
strokes up TO ActualMasterAxisPos *)
    SumOfSuperpositionDistance    : LREAL; (* sum of additional
moves through MC_MoveSuperimposed *)
    ActualMasterAxisPos       : LREAL; (* absolute set position
of the master axis *)
    ActualSlaveAxisPos        : LREAL; (* absolute set position
of the slave axis *)
    ActualMasterCamPos        : LREAL; (* absolute cam table
position of the master *)
    ActualSlaveCamPos         : LREAL; (* absolute cam table
position of the slave *)
```

```
    (* mode for the scaling of cam tables *)
    ScalingPending        : BOOL; (* a change is currently
pending *)
    ScalingActivationMode     : MC_CamActivationMode;

    ScalingActivationPos      : LREAL;
    ScalingMasterScalingMode      : MC_CamScalingMode;

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

    OnlineChangeActivationPos     : LREAL;
    OnlineChangeMasterScalingMode : MC_CamScalingMode;

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

    CamExchangeActivationPos      : LREAL;
    CamExchangeMasterScalingMode  : MC_CamScalingMode;

    CamExchangeSlaveScalingMode   : MC_CamScalingMode;
END_STRUCT
END_TYPE
```

MC_CamScalingMode [▶ 63], MC_CamActivationMode [▶ 60] MC_CAM_ID [▶ 54], MC_InterpolationType [▶ 56], MC_TableType [▶ 55]

The data structure *MC_CamInfoData* contains data referring to the current state of a cam plate coupling. The data can be determinded using the MC_CamInfo [▶ 53] function block.

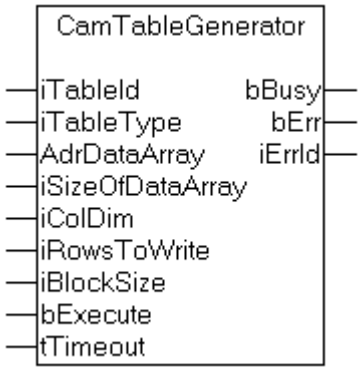
Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9 from build 1026	PC (i386)	TcNcCamming.Lib

5.2 Blocks for compatibility with exisiting programs

5.2.1 Table blocks

5.2.1.1 CamTableGenerator



CamTableGenerate creates a new table in the TwinCAT NC context and fills it with data. If any table exists with the same ID, it will be deleted.

Interface

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

iTableID: Table ID which will in future be used to address the table.

iTableType: The table type

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

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

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

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

AdrDataArray: The address of a data field containing position data for the master and slave axes. The first of the field's dimensions describes the table lines, and the second dimension describes the columns.

iSizeOfDataArray: Total size of the data field in bytes.

iColDim: Number of columns in the data field. This value must correspond to the actual size of the second field dimension.

iRowsToWrite: Number of table lines. This value may be less than or equal to the size of the first field dimension.

iBlockSize: The number of data bytes that the function block will transfer in a continuous block into the table in the NC context. If iBlockSize=0 a default value is used, and this is recommended in the majority of cases. Only if the internal processes are understood in detail should a figure for iBlockSize>0 be given.

bExecute: Edge-triggered signal for execution of the command.

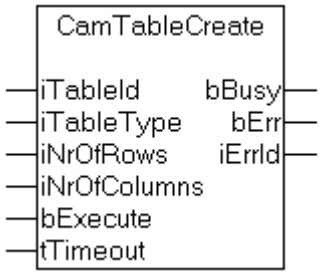
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.1.2 CamTableCreate



CamTableCreate creates a new table, initially empty, in the TwinCAT NC context.

Interface

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

iTableID: Table ID which will in future be used to address the table.

iTableType: The table type

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

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

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

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

iNrOfRows: Number of table lines.

iNrOfColumns: Number of table columns.

bExecute: Edge-triggered signal for execution of the command.

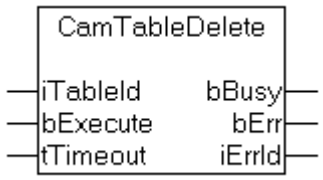
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.1.3 CamTableDelete



CamTableDelete deletes a table from the NC context.

Interface

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

iTableID: Table ID of the table that is to be deleted.

bExecute: Edge-triggered signal for execution of the command.

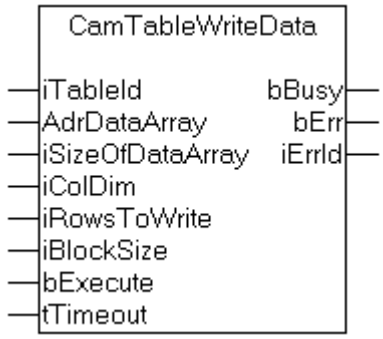
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.1.4 CamTableWriteData



CamTableWriteData writes table data from a PLC variable into a table in the NC context.

Interface

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

```
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID: Table ID of the table from which the data is to be read.

AdrDataArray: The address of a data field containing position data for the master and slave axes. The first of the field's dimensions describes the table lines, and the second dimension describes the columns.

iSizeOfDataArray: Total size of the data field in bytes.

iColDim: Number of columns in the data field. This value must correspond to the actual size of the second field dimension.

iRowsToWrite: Number of table lines. This value may be less than or equal to the size of the first field dimension.

iBlockSize: The number of data bytes that the function block will transfer in a continuous block into the table in the NC context. If iBlockSize=0 a default value is used, and this is recommended in the majority of cases. Only if the internal processes are understood in detail should a figure for iBlockSize>0 be given.

bExecute: Edge-triggered signal for execution of the command.

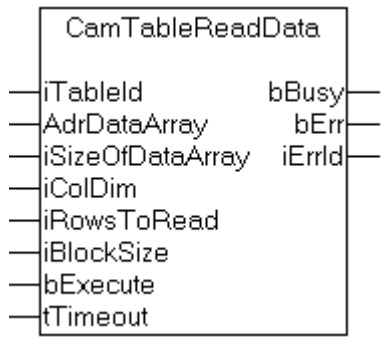
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.1.5 CamTableReadData



CamTableReadData reads the table data from a table in the NC context and places it into a PLC variable.

Interface

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

iTableID: Table ID of the table from which the data is to be read.

AdrDataArray: The address of a data field that is to receive position data for the master and slave axes. The first of the field's dimensions describes the table lines, and the second dimension describes the columns.

iSizeOfDataArray: Total size of the data field in bytes.

iColDim: Number of columns in the data field. This value must correspond to the actual size of the second field dimension.

iRowsToRead: Number of table lines. This value may be less than or equal to the size of the first field dimension.

iBlockSize: The number of data bytes that the function block will transfer in a continuous block into the table in the NC context. If iBlockSize=0 a default value is used, and this is recommended in the majority of cases. Only if the internal processes are understood in detail should a figure for iBlockSize>0 be given.

bExecute: Edge-triggered signal for execution of the command.

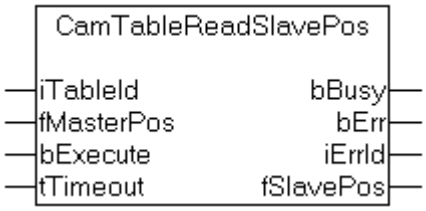
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.1.6 CamTableReadSlavePos



CamTableReadSlavePos reads the slave position defined by the table corresponding to the table position of a master axis. In some cases this will involve interpolation of the slave axis position data, if the master position does not correspond exactly to a table entry.

Interface

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

iTableID: Table ID which will in future be used to address the table.

fMasterPos: Position of the master axis.

bExecute: Edge-triggered signal for execution of the command.

tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

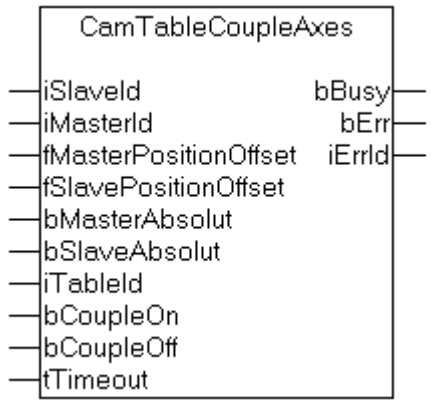
bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

fSlavePos: Position determined for the slave axis.

5.2.2 Blocks for camshaft axes

5.2.2.1 CamTableCoupleAxes



CamTableCoupleAxes couples a slave axis to a master axis by way of an existing table.

Interface

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

iSlaveId: Axis ID of the slave axis that is to be coupled.

iMasterId: Axis ID of the master axis that is to be coupled to.

fMasterPositionOffset: Master position offset in the table data.

fSlavePositionOffset: Slave position offset in the table data.

bMasterAbsolut: Table data for the master as absolute position values.

bSlaveAbsolut: Table data for the slave as absolute position values.

iTableId: Table ID of the table to be used for the coupling.

bCoupleOn: Edge-triggered signal for execution of the axis coupling.

bCoupleOff: Edge-triggered signal for uncoupling the axes.

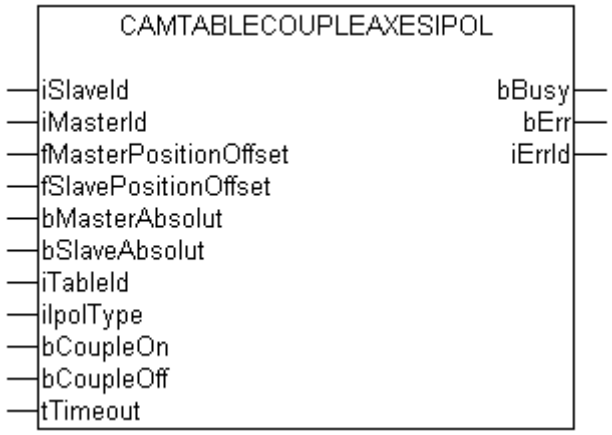
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bCoupleOn or bCoupleOff, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.2.2 CamTableCoupleAxesIpol



CamTableCoupleAxes couples a slave axis to a master axis by way of an existing table. Furthermore the type of interpolation between two table entries can be defined

Interface

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

iSlaveId: Axis ID of the slave axis that is to be coupled.

iMasterId: Axis ID of the master axis that is to be coupled to.

fMasterPositionOffset: Master position offset in the table data.

fSlavePositionOffset: Slave position offset in the table data.

bMasterAbsolut: Table data for the master as absolute position values.

bSlaveAbsolut: Table data for the slave as absolute position values.

iTableId: Table ID of the table to be used for the coupling.

iIpolType : Type of interpolation between two table entries. Valid values are:

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

bCoupleOn: Edge-triggered signal for execution of the axis coupling.

bCoupleOff: Edge-triggered signal for uncoupling the axes.

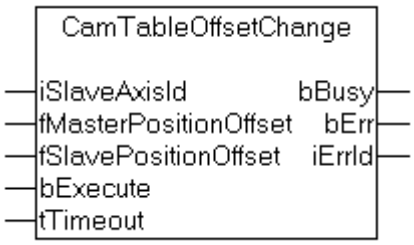
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bCoupleOn or bCoupleOff, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.2.3 CamTableOffsetChange



CamTableOffsetChange changes the master and slave offsets in the table online, i.e. while the master and slave axes are coupled through the table.

Interface

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

iSlaveAxisId: Axis Id of the slave axis. The corresponding master is known from the existing axis coupling.

fMasterPositionOffset: Position offset in the master position data.

fSlavePositionOffset: Position offset in the slave position data.

bExecute: Edge-triggered signal for execution of the command.

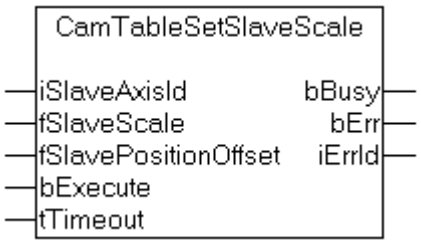
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.2.4 CamTableSetSlaveScale



CamTableSetSlaveScale scales a table's slave column for an existing master slave table coupling.

Interface

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

iSlaveId: Axis ID of the slave axis that is coupled to a master axis through a table.

fSlaveScale: Scaling factor for the table's slave column.

bExecute: Edge-triggered signal for execution of the command.

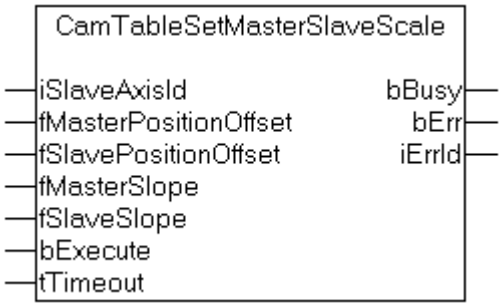
tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.2.5 CamTableSetMasterSlaveScale



CamTableSetMasterSlaveScale scales a table's master and slave columns for an existing master-slave table coupling.

Interface

```
VAR_INPUT
    iSlaveAxisId : UINT;
    fMasterPositionOffset : LREAL;
    fSlavePositionOffset : LREAL;
    fMasterSlope : LREAL;
```

```
    fSlaveSlope : LREAL;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr : BOOL;
    iErrId : UDINT;
END_VAR
```

iSlaveId: Axis ID of the slave axis that is coupled to a master axis through a table.

fMasterOffset : Master position offset for the table

fSlaveOffset : Slave position offset for the table

fMasterSlope : Master scaling for the table

fSlaveSlope : Slave scaling for the table

bExecute: Edge-triggered signal for execution of the command.

tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

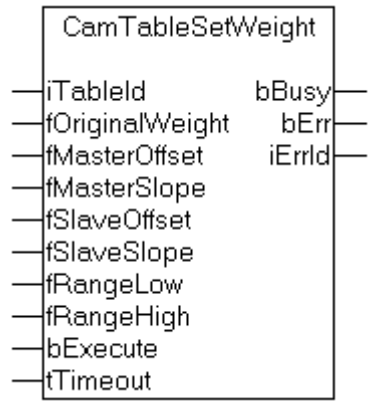
bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.3 Blocks for multitable axes

5.2.3.1 CamTableSetWeight

(only for multi-tables)



CamTableSetWeight sets the scaling parameter of a scalable *multi*-table.

Interface

```
VAR_INPUT
    iTableId    : INT;
    fOriginalWeight : LREAL;
    fMasterOffset   : LREAL;
    fMasterSlope    : LREAL;
    fSlaveOffset    : LREAL;
    fSlaveSlope     : LREAL;
    fRangeLow       : LREAL;
    fRangeHigh      : LREAL;
    bExecute    : BOOL;
    tTimeout    : TIME;
```

```
END_VAR
VAR_OUTPUT
    bBusy : BOOL;
    bErr  : BOOL;
    iErrId : UDINT;
END_VAR
```

iTableID: Table ID which will in future be used to address the table.

fOriginalWeight : Original table weighting
fMasterOffset : Master position offset for the table
fMasterSlope : Master scaling for the table
fSlaveOffset : Slave position offset for the table
fSlaveSlope : Slave scaling for the table
fRangeLow : lower range limit for which the table is valid
fRangeHigh : upper range limit for which the table is valid

bExecute: Edge-triggered signal for execution of the command.

tTimeout: ADS timeout (about 1 second).

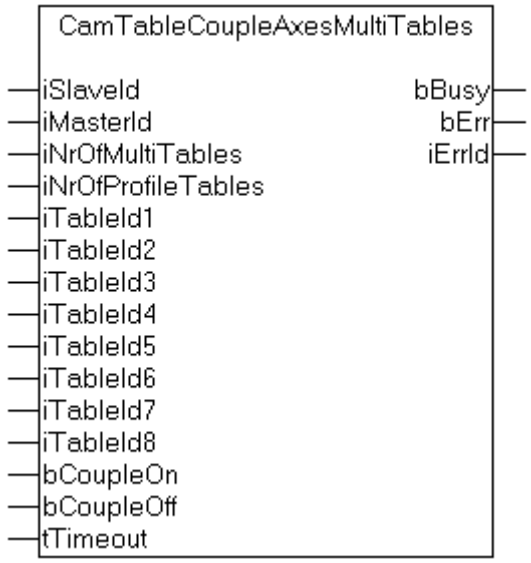
bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.3.2 CamTableCoupleAxesMultiTables

(only for multi-tables)



CamTableCoupleAxesMultiTables couples a slave axis to a master axis via a number of existing tables (*multi*-table coupling).

Interface

```
VAR_INPUT
    iSlaveId        : INT;
    iMasterId       : INT;
    iNrOfMultiTables    : INT;
    iNrOfProfileTables  : INT;
    iTableId1       : INT;
    iTableId2       : INT;
    iTableId3       : INT;
    iTableId4       : INT;
    iTableId5       : INT;
```

```
    iTableId6       : INT;
    iTableId7       : INT;
    iTableId8       : INT;
    bCoupleOn       : BOOL;
    bCoupleOff      : BOOL;
    tTimeout        : TIME;
END_VAR
VAR_OUTPUT
    bBusy           : BOOL;
    bErr        : BOOL;
    iErrId          : UDINT;
END_VAR
```

iSlaveId: Axis ID of the slave axis that is to be coupled.

iMasterId: Axis ID of the master axis that is to be coupled to.

iNrOfMultiTables : Total number of tables actually used (1 to 8)

iNrOfProfileTables : Number of profile tables (1 to iNrOfMultiTables). Other tables are correction tables that may be switched in.

iTableId1 : Table index for the first table

iTableId2 : Table index for the second table

iTableId3 : Table index for the third table

iTableId4 : Table index for the fourth table

iTableId5 : Table index for the fifth table

iTableId6 : Table index for the sixth table

iTableId7 : Table index for the seventh table

iTableId8 : Table index for the eight table

bCoupleOn: Edge-triggered signal for execution of the axis coupling.

bCoupleOff: Edge-triggered signal for uncoupling the axes.

tTimeout: ADS timeout (about 1 second).

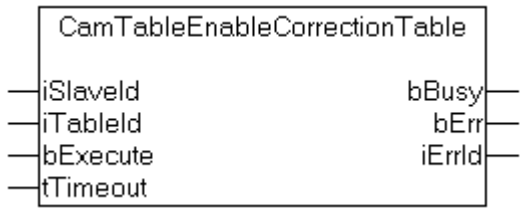
bBusy: Becomes TRUE with a rising edge at bCoupleOn or bCoupleOff, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.3.3 CamTableEnableCorrectionTable

(only for multi-tables)



CamTableEnableCorrectionTable activates a correction table in *multi*-table applications.

Interface

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

iSlaveAxisId: Axis Id of the slave axis. The corresponding master is known from the existing axis coupling.

iTableID: Table ID of the table from which the data is to be read.

bExecute: Edge-triggered signal for execution of the command.

tTimeout: ADS timeout (about 1 second).

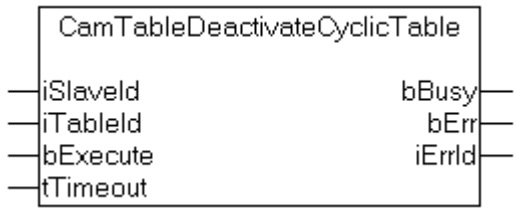
bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.2.3.4 CamTableDeactivateCyclicTable

(only for multi-tables)



CamTableDeactivateCyclicTable switches an active cyclical *multi*-table off at the end of the cycle.

Interface

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

iSlaveAxisId: Axis Id of the slave axis. The corresponding master is known from the existing axis coupling.

iTableID: Table ID of the table from which the data is to be read.

bExecute: Edge-triggered signal for execution of the command.

tTimeout: ADS timeout (about 1 second).

bBusy: Becomes TRUE with a rising edge at bExecute, and remains TRUE until the block has executed the command.

bErr: Becomes TRUE if an error occurs while executing the command.

bErrId: Error number (ADS or NC error number).

5.3 Get_TcNcCamming_Version

Get_TcNcCamming_Version determines the version number of the PLC library TcNcCamming.lib. The function returns the version number in a string.

Interface

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

bGet: Signal for execution of the command.

More Information:
www.beckhoff.com

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