BECKHOFF New Automation Technology

Manual | EN

TS5050

TwinCAT 2 | MC Camming

Supplement | Motion

2025-07-03 | Version: 1.4

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled 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 notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO® , EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

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

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are 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 technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

ℹ This information includes, for example:
recommendations for action, assistance or further information on the product.

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

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 TcMC2_Camming library contains function blocks for handling cam plates. Two types of cam plates are supported.

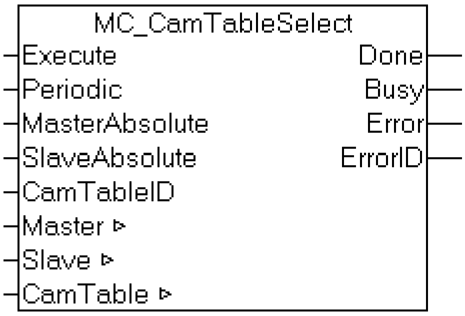
One option is a cam plate in the form of a 2-column table containing master and slave positions (standard table). The master column defines interpolation points via the travel path of the master, ascending from a minimum position value to a maximum value. The associated slave position is determined from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

3 Cam plates

3.1 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 [▶ 12] is sufficient.

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
Periodic	Periodic is TRUE if the cam plate is repeatedly cyclically.
MasterAbsolute	At the moment not in use.
SlaveAbsolute	At the moment not in use.
CamTableID	ID of the cam plate used for coupling

Outputs

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

Done	Becomes TRUE, if the cam plate was created successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

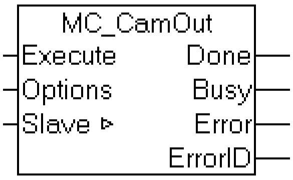
```
VAR_IN_OUT
    Master    : AXIS_REF;
    Slave     : AXIS_REF;
    CamTable  : MC_CAM_REF;
END_VAR
```

Master	Axis data structure of the master - currently not used.

Slave	Axis data structure of the slave - currently not used.
CamTable	The data structure of type MC_CAM_REF [▶ 45] describes the data storage for the cam plate in the PLC

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

3.2 MC_CamOut



The function block *MC_CamOut* deactivates a master-slave coupling.

NOTICE

If a slave axis is uncoupled during the movement, it is not automatically stopped, but reaches a continuous velocity with which it will continue to travel endlessly. The axis can be stopped with a Stop command.

NOTICE

Call during movement

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:

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
Options	Currently not implemented

Outputs

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

Done	Becomes TRUE, if the axis was successfully uncoupled.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number

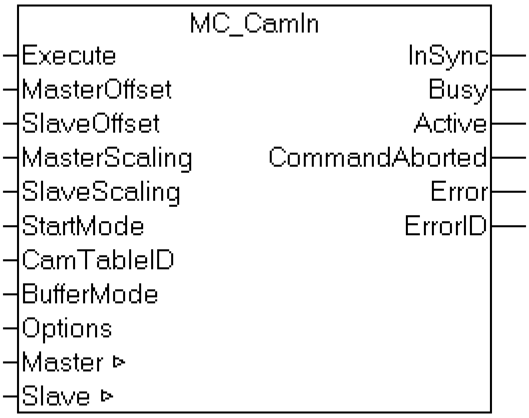
Inputs/outputs

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

Slave	Slave axis data structure.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

3.3 MC_CamIn



The function block *MC_CamIn* activates master-slave coupling with a certain cam plate. In addition it is possible to switch to a new cam plate in coupled state. The switching rules, in particular the time or position, can be specified.

The status flag Axis.Status.CamTableQueued can be used to check whether a cam plate is queued for switchover.

Important:

Further information on coupling with cam plates [▸ 14]

ActivationMode [▸ 46] (coupling or switching of cam plates)

StartMode [▸ 60]

ScalingMode [▸ 49]

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*
MasterOffset	Offset to the master position of the cam plate
SlaveOffset	Offset to the slave position of the cam plate
MasterScaling	Scaling of the master position of the cam plate
SlaveScaling	Scaling of the slaveposition of the cam plate
StartMode	StartMode [▸ 60] determines whether the cam plate position is interpreted absolute or relative to the coupling position. *StartMode* can be relative or absolute for master (X coordinate) and slave (Y coordinate).
CamTableID	ID [▸ 44] of the cam plate used for coupling
BufferMode	currently not implemented
Options	Data structure [▸ 61] with further coupling and switching options:

	ActivationMode	ActivationMode [▶ 46] specifies the switching time or position at which cam plate coupling or switchover takes place. *ActivationMode* can also be specified when a slave is coupled for the first time.
	ActivationPosition	Optional master position at which a cam plate is switched, depending on the *ActivationMode*. (not required for first coupling.) If *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
	MasterScalingMode :	Optional Scaling mode [▶ 49] for the master position of the cam plate
	SlaveScalingMode	Optional Scaling mode [▶ 49] for the Slave position of the cam plate
	InterpolationType	Interpolation type [▶ 53] for position tables. Not required for motion functions.

Outputs

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

InSync	Becomes TRUE, if the coupling was successful and the cam plate is active.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *InSync*, *CommandAborted* or *Error*, is set.
Active	Active indicates that the command is executed. For cam plate switching *Active* becomes TRUE, if the coupling command was executed successfully but the cam plate is still queued. If the cam plate is activated depending on the *ActivationMode*, *Active* becomes FALSE and *InSync* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Master	Master axis data structure.
Slave	Slave axis data structure.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

3.4 MC_CaminAppendix

The function block MC_CamIn [▸ 12] 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 synchronised with the cam plate, but it will jump if it is not already at the table position.

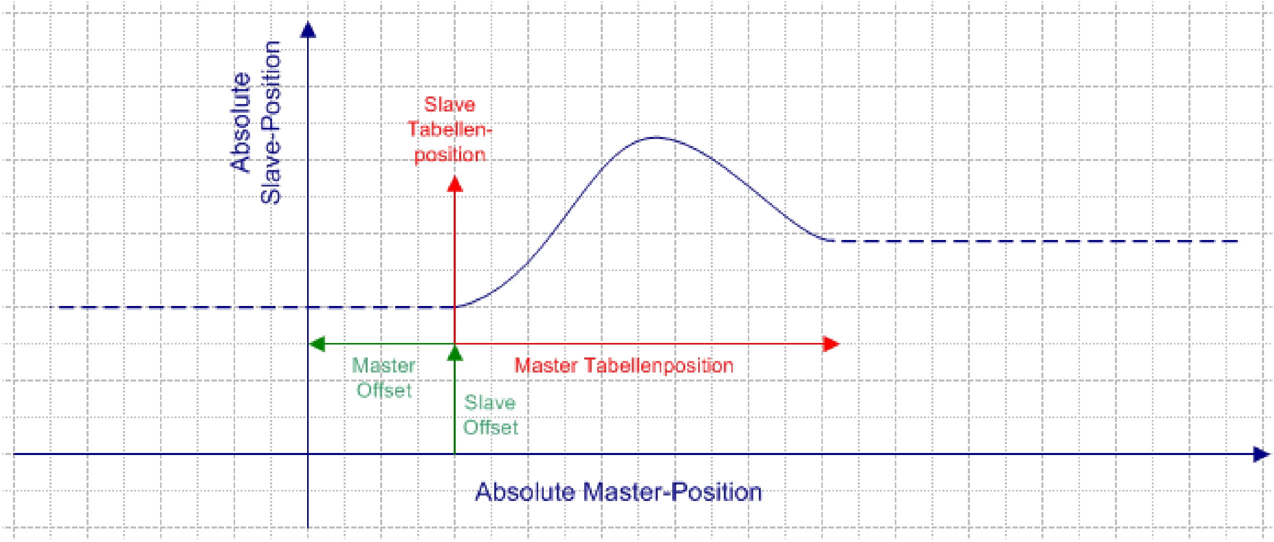
In practice the question arises what position the slave should be in prior to the coupling, and how this is calculated. The following figures illustrate the procedure.

NOTICE

- 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 [▸ 36] . The 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.

MasterCamTablePosition := (MasterPosition + MasterOffset) / MasterScaling;

The master table position is used as input parameter for the function block MC_ReadCamTableSlaveDynamics [▸ 36]. The result is converted to an absolute slave position with slave offset and scaling, if necessary.

SlaveCamTablePosition := ReadSlaveDynamics.SlavePosition;

SlavePosition := (SlaveCamTablePosition * SlaveScaling) + SlaveOffset;

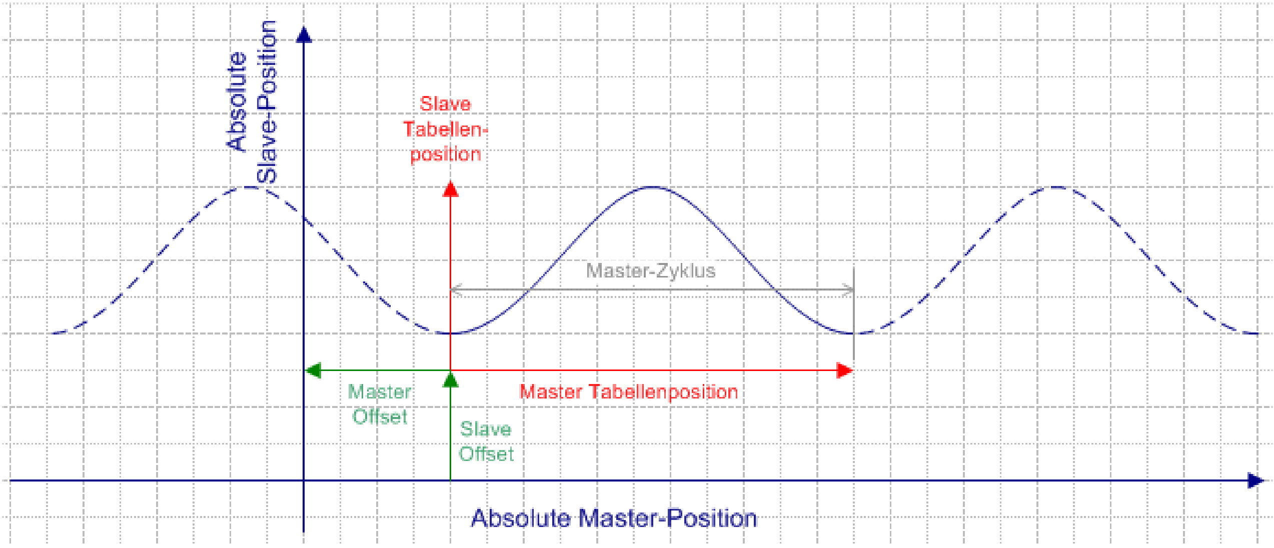
The slave is moved to this position prior to the coupling. Alternatively, the master may be moved to a position that corresponds to the current slave position. However, generally this position cannot be determined from the cam plate, since the cam plate may be ambiguous.

NOTICE

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 lift

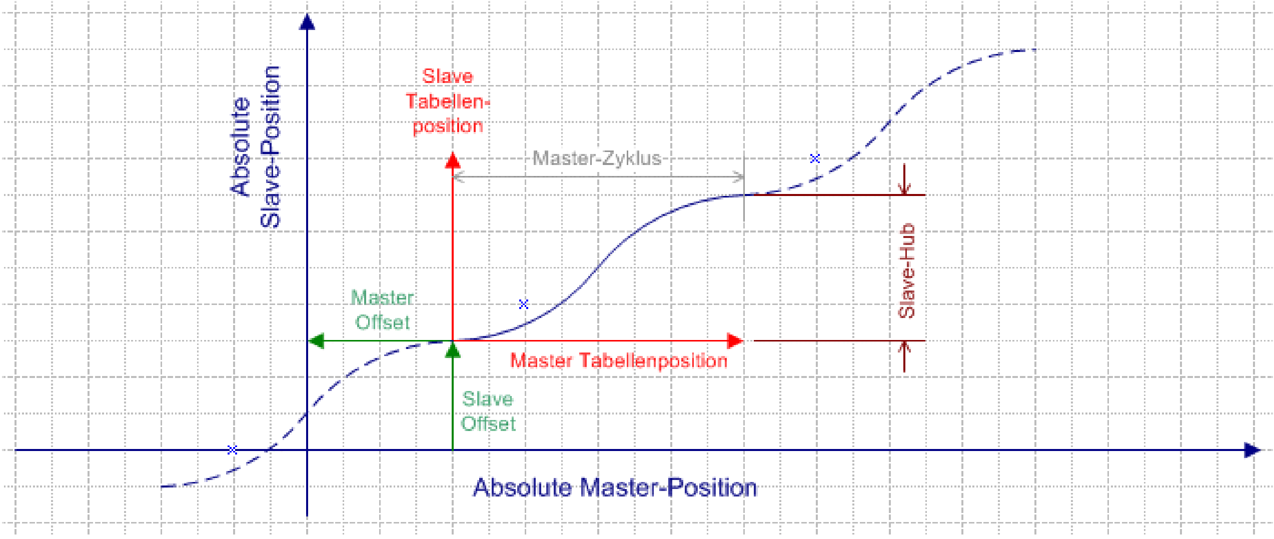
A cyclic cam plate without lift is characterised by the fact that the slave start and end positions in the table are identical. The slave therefore moves cyclically within a defined range, without changing its position permanently in a particular direction.



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 lift

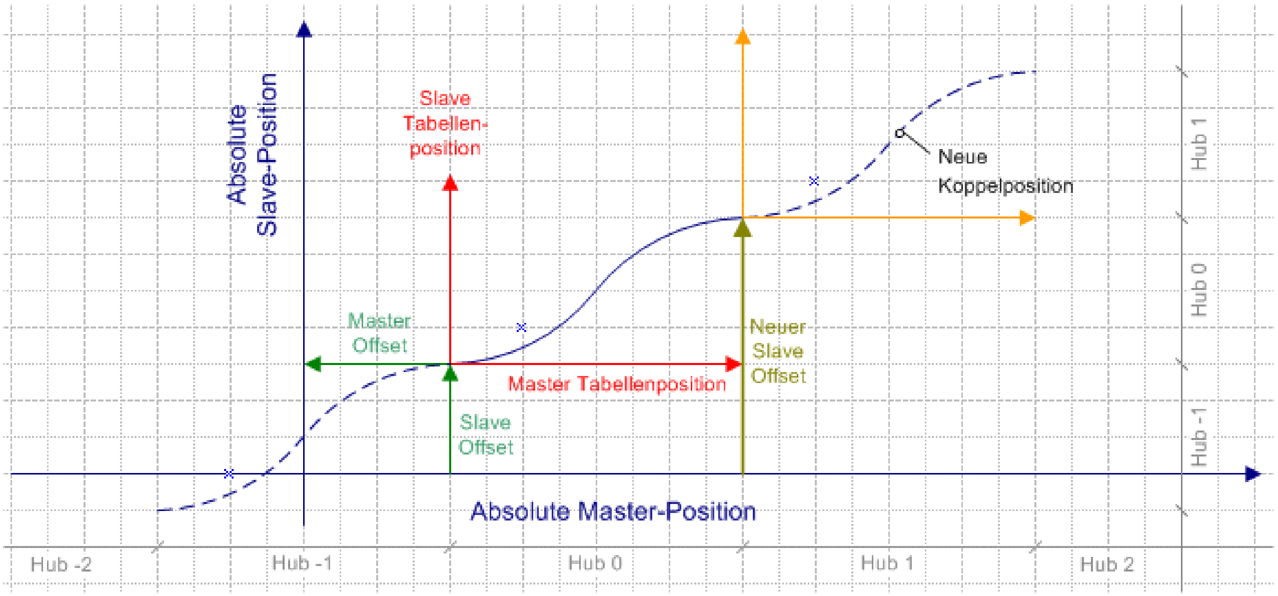
The lift of a cyclic cam plate is the difference between the last and the first table position of the slave.



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 lift 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 lift

If a slave is coupled to a cam plate with lift, the coupling is always done in the basic cycle (red coordinate system), i.e. without added lifting distances. If the slave is uncoupled after a few cycles and then re-coupled, the slave position returns to the basic cycle. If necessary, this behaviour has to be taken into account and compensated by re-calculating the slave offset.



MasterCamTablePos := (MasterPosition + MasterOffset) / MasterScaling;

The master table position is used as input parameter for the function block MC_ReadCamTableSlaveDynamics [▸ 36]. The result is converted to an absolute slave position with slave offset and scaling, if necessary. In addition, the number of pending lifts must be calculated and added to the slave position.

SlaveCamTablePosition := ReadSlaveDynamics.SlavePosition;

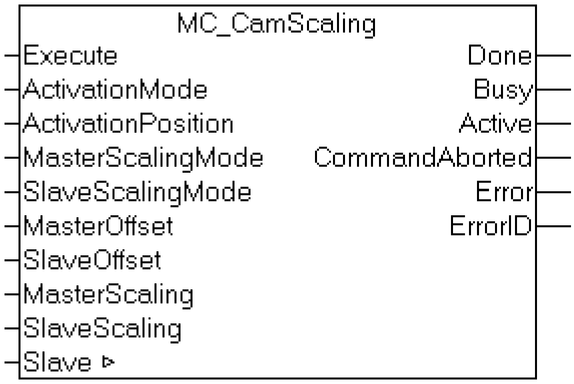
Lift number := MODTURNS((SlavePosition - SlaveOffset), SlaveHub);

NewSlaveOffset := SlaveOffset + (SlaveHub * lift number);

SlavePosition := (SlaveCamTablePosition * SlaveScaling) + NewSlaveOffset;

The Autooffset [▸ 49] function can simplify the calculation of offsets, particularly for switching of cam plates.

3.5 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, however 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.

The status flag Axis.Status.CamScalingPending (AXIS_REF) can be used to check whether a scaling procedure is queued.

Inputs

```
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 a rising edge at input *Execute*.
ActivationMode	ActivationMode [▶ 46] specifies the scaling time and position.
ActivationPosition	Master position at which a cam plate is scaled, depending on the ActivationMode [▶ 46]. If *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
MasterScalingMode	optional Scaling mode [▶ 49] for the master position of the cam plate
SlaveScalingMode	optional Scaling mode [▶ 49] for the slave position of the cam plate
MasterOffset	Offset to the master position of the cam plate
SlaveOffset	Offset to the slave position of the cam plate
MasterScaling	Scaling of the master position of the cam plate
SlaveScaling	Scaling of the slave position of the cam plate

Outputs

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

Done	Becomes TRUE if scaling was successful.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. If *Busy* becomes FALSE again, the function block is ready for a new job. At the same time one of the outputs, *Done*, *CommandAborted* or *Error*, is set.
Active	Active indicates that the command is executed. When the scaling was done depending on *ActivationMode*, *Active* becomes FALSE and *Done* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

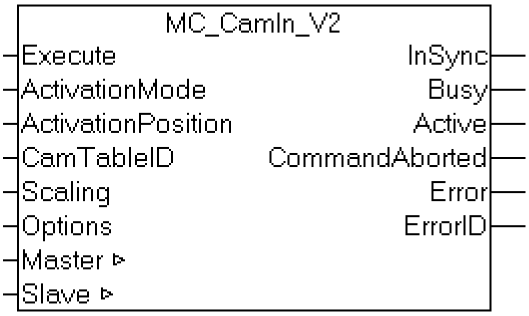
```
VAR_IN_OUT
   Slave                  : AXIS_REF;
END_VAR
```

Slave	Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

4 Multi cam plates

4.1 MC_CamIn_V2



MC_CamIn_V2 is a further development of the function block MC_CamIn [▸ 12] and is able to operate with several superimposed cam plates (Multi-Cam). When MC_CamIn_V2 is first called it creates a master/slave coupling with the cam plate. Subsequent calls during runtime can be used to superimpose additional cam plates for the same slave axis or remove them again. The switching rules, in particular the time or position, can be specified.

MC_CamIn_V2 can only be used as an alternative to *MC_CamIn*. The two function blocks cannot be used together for the same slave axis. For addition, replacement and removal of cam plates the function blocks MC_CamAdd [▸ 21], MC_CamExchange [▸ 23] and MC_CamRemove [▸ 25] are available as alternatives. All operations can also be carried out with *MC_CamIn_V2*.

The status flag Axis.Status.CamTableQueued (AXIS_REF) can be used to check whether a cam plate is queued for addition or switchover.

The block can be used from a runtime version TwinCAT 2.11 R2

Important:

ActivationMode [▸ 46] (time and/or position from which the operation takes place)

CamOperationMode [▸ 62] (adding, switching or removal of superimposed cam plates)

ScalingMode [▸ 49]

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
ActivationMode	The ActivationMode i [▸ 46]s used to specify the time and/or position at which the cam plate coupling or switchover is to take place. *ActivationMode* can also be specified when a slave is coupled for the first time.
ActivationPosition	Optional master position at which a cam plate is switched, depending on the *ActivationMode*. (not required for first coupling.) If *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
CamTableID	ID [▸ 44] of the cam plate used for the coupling

Scaling	Optional scaling parameters [▶ 63] for the cam plate	
Options	Data structure with further coupling and switching options:	
	Interpolation type	Interpolation type [▶ 53] for position tables. Not required for motion functions.
	CamOperationMode	The CamOperationMode [▶ 62] defines the way the specified cam plate (*CamTableID*) has to act in the coupled system. Cam plates can be added, switched or removed.
	ReferenceCamTableID	Optional ID of a cam plate that is already active in the coupling. This ID is specially required for operations that would otherwise be ambiguous, e.g. replacement of certain cam plates in multi-couplings. In unambiguous operations the value can remain 0.

Outputs

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

InSync	Becomes TRUE if the cam plate operation was completed successfully. In operations with activation position InSync only becomes TRUE after the actual activation.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *InSync*, *CommandAborted* or *Error* is set.
Active	Active indicates that the command is executed. *Active* becomes TRUE if the command was issued successfully but the operation is still queued. If the cam plate is activated depending on the *ActivationMode* , *Active* becomes FALSE and *InSync* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

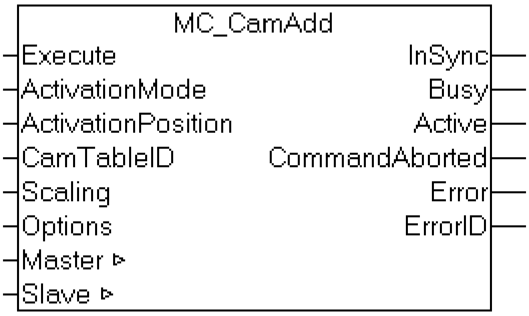
Inputs/outputs

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

Master	Master axis data structure.
Slave	Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

4.2 MC_CamAdd



MC_CamAdd adds a cam plate to a multi-cam coupling. The cam plate coupling is initially created with MC_CamIn_V2 [▶ 19].

Alternatively, a cam plate can be added with MC_CamIn_V2.

The status flag *Axis.Status.CamTableQueued* (AXIS_REF)can be used to check whether a cam plate is queued for addition or switchover.

The block can be used from a runtime version TwinCAT 2.11 R2

Important :
ActivationMode [▶ 46] (time and/or position from which the operation takes place)

CamOperationMode [▶ 62] (adding, switching or removal of superimposed cam plates)

ScalingMode [▶ 49]

Inputs

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

<table>
<tr><td><b>Execute</b></td><td colspan="2">The command is executed with a rising edge at input <i>Execute</i>.</td></tr>
<tr><td><b>ActivationMode</b></td><td colspan="2">The ActivationMode [▶ 46] is used to specify the time and/or position at which the cam plate coupling or switchover is to take place.
<i>ActivationMode</i> can also be specified when a slave is coupled for the first time.</td></tr>
<tr><td><b>ActivationPosition</b></td><td colspan="2">Optional master position at which a cam plate is switched, depending on the <i>ActivationMode</i>.
(not required for first coupling.)
If <i>ActivationMode</i> MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the <i>MasterScaling</i> before the function block is called.</td></tr>
<tr><td><b>CamTableID</b></td><td colspan="2">ID [▶ 44] of the cam plate used for the coupling</td></tr>
<tr><td><b>Scaling</b></td><td colspan="2">Optional scaling parameters [▶ 63] for the cam plate</td></tr>
<tr><td rowspan="3"><b>Options</b></td><td colspan="2">Data structure with further coupling and switching options:</td></tr>
<tr><td><b>Interpolation type</b></td><td>Interpolation type [▶ 53] for position tables. Not required for motion functions.</td></tr>
<tr><td><b>CamOperationMode</b></td><td>For <i>MC_CamAdd</i> theCamOperationMode [▶ 62] is preallocated with <i>CAMOPERATIONMODE_ADDITIVE.</i></td></tr>
</table>

	ReferenceCamTableID	Optional ID of a cam plate that is already active in the coupling. This ID is only required for operations that would otherwise be ambiguous, e.g. automatic offset adjustment in relation to an existing cam plate (Master AutoOffset).

Outputs

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

InSync	Becomes TRUE if the cam plate operation was completed successfully. In operations with activation position InSync only becomes TRUE after the actual activation.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *InSync*, *CommandAborted* or *Error* is set.
Active	Active indicates that the command is executed. *Active* becomes TRUE if the command was issued successfully but the operation is still queued. If the cam plate is activated depending on the *ActivationMode* , *Active* becomes FALSE and *InSync* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number

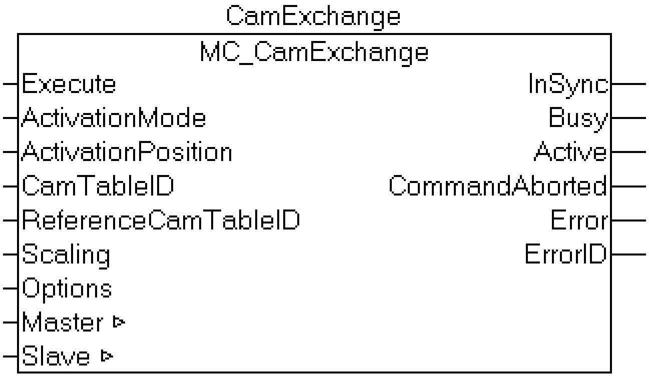
Inputs/outputs

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

Master	Master axis data structure.
Slave	Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

4.3 MC_CamExchange



MC_CamExchange exchanges a cam plate in a multi-cam coupling. The cam plate coupling is initially created with MC_CamIn_V2 [▸ 19].

Alternatively, a cam plate can be exchanged with MC_CamIn_V2.

The status flag *Axis.Status.CamTableQueued* (AXIS_REF)can be used to check whether a cam plate is queued for addition or switchover.

The block can be used from a runtime version TwinCAT 2.11 R2

Important:

ActivationMode [▸ 46] (time and/or position from which the operation takes place)

CamOperationMode [▸ 62] (adding, switching or removal of superimposed cam plates)

ScalingMode [▸ 49]

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
ActivationMode	The ActivationMode [▸ 46] is used to specify the time and/or position at which the cam plate coupling or switchover is to take place. ActivationMode [▸ 46] can also be specified when a slave is coupled for the first time.
ActivationPosition	Optional master position at which a cam plate is switched, depending on the ActivationMode [▸ 46]. (not required for first coupling.) If ActivationMode [▸ 46] MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
CamTableID	ID [▸ 44] of the cam plate used for the coupling
ReferenceCamTableID	Optional ID of a cam plate that is already active in the coupling. This ID is specially required for operations that would otherwise be ambiguous, e.g. replacement of certain cam plates in multi-couplings. In unambiguous operations the value can remain 0.

<table>
<tr><td><b>Scaling</b></td><td colspan="2">Optional scaling parameters [▶ 63] for the cam plate</td></tr>
<tr><td rowspan="4"><b>Options</b></td><td colspan="2">Data structure with further coupling and switching options:</td></tr>
<tr><td><b>Interpolation type</b></td><td>Interpolation type [▶ 53] for position tables. Not required for motion functions.</td></tr>
<tr><td><b>CamOperationMode</b></td><td>For<i>MC_CamExchange</i> the CamOperationMode [▶ 62] is preallocated with <i>CAMOPERATIONMODE_EXCHANGE</i>.</td></tr>
<tr><td><b>ReferenceCamTableID</b></td><td>is preallocated with the value of input ReferenceCamTableID</td></tr>
</table>

Outputs

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

InSync	Becomes TRUE if the cam plate operation was completed successfully. In operations with activation position InSync only becomes TRUE after the actual activation.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *InSync*, *CommandAborted* or *Error* is set.
Active	Active indicates that the command is executed. *Active* becomes TRUE if the command was issued successfully but the operation is still queued. If the cam plate is activated depending on the ActivationMode [▶ 46] , *Active* becomes FALSE and *InSync* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

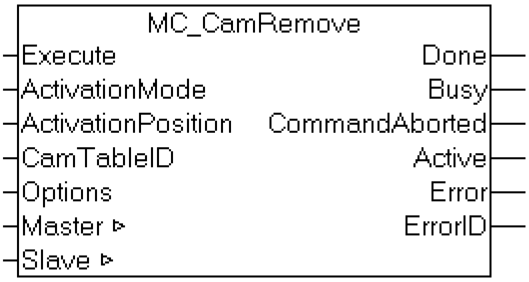
Inputs/outputs

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

Master	Master axis data structure.
Slave	axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

4.4 MC_CamRemove



MC_CamRemove removes a cam plate from a multi-cam coupling. The cam plate coupling is initially created with MC_CamIn_V2 [▶ 19].

Alternatively, a cam plate can be removed with MC_CamIn_V2.

The block can be used from a runtime version TwinCAT 2.11 R2

Important :

ActivationMode [▶ 46] (time and/or position from which the operation takes place)

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
ActivationMode	The ActivationMode [▶ 46] is used to specify the time and/or position at which the cam plate coupling or switchover is to take place. ActivationMode [▶ 46] can also be specified when a slave is coupled for the first time.
ActivationPosition	Optional master position at which a cam plate is switched, depending on the ActivationMod [▶ 46]*e*. (not required for first coupling.) If ActivationMode [▶ 46] *MC_CAMACTIVATION_ATMASTERCAMPOS* is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
CamTableID	ID [▶ 44] of the cam plate that is removed from the coupled system.
Options	not used

Outputs

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

Done	Becomes TRUE if the cam plate operation was completed successfully. In operations with activation position Done only becomes TRUE after the actual deactivation.

Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *InSync*, *CommandAborted* or *Error* is set.
Active	Active indicates that the command is executed. *Active* becomes TRUE if the command was issued successfully but the operation is still queued. If the cam plate is activated depending on the ActivationMode [▶ 46], *Active* becomes FALSE and *InSync* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

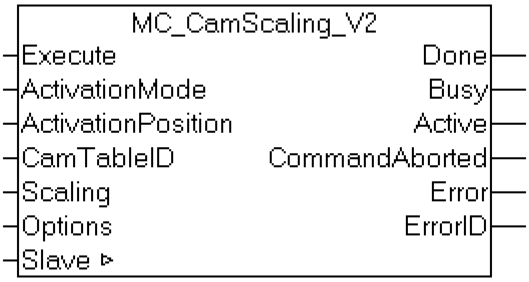
Inputs/outputs

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

Master	Master axis data structure.
Slave	Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

4.5 MC_CamScaling_V2



A cam plate coupling can be scaled with the function block *MC_CamScaling_V2*. The raw table data of the cam plate are not affected, however 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.

The status flag Axis.Status.CamcalingPending (AXIS_REF) can be used to check whether a scaling procedure is queued.

The block can be used from a runtime version TwinCAT 2.11 R2

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
ActivationMode	ActivationMode [▶ 46] specified the scaling time and position.
ActivationPosition	Master position at which a cam plate is scaled, depending on the ActivationMode [▶ 46] If ActivationMode [▶ 46] MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
CamTableID	ID [▶ 44] of the cam plate that is scaled.
Scaling	Scaling data such as mode, offset and scaling factor
Options	not used

Outputs

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

Done	Becomes TRUE if scaling was successful.

Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done*, *CommandAborted* or *Error*, is set.
Active	Active indicates that the command is executed. When the scaling was done depending on ActivationMode [▸ 46], *Active* becomes FALSE and *Done* is set.
CommandAborted	Becomes TRUE, if the command could not be fully executed.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

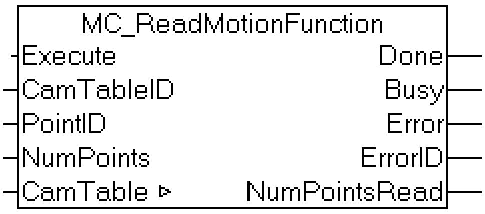
```
VAR_IN_OUT
    Slave     : AXIS_REF;
END_VAR
```

Slave	Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

5 Motion functions

5.1 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 [▶ 45].

Inputs

```
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 [▶ 44] of the loaded table.
PointID	Point ID [▶ 55] 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*.

Outputs

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

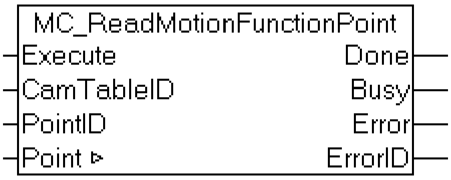
Done	Becomes TRUE, if the data were read successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.
NumPointsRead	The number of points that were actually read. The number may be less or equal *NumPoints*.

Inputs/outputs

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

CamTable	Reference to the table [▶ 45] (structure).

5.2 MC_ReadMotionFunctionPoint



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

Inputs

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

Execute	The command is executed with rising edge.
CamTableID	ID [▶ 44] of the loaded table.
PointID	Point ID [▶ 55] of the first point of the motion function to be read.

Outputs

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

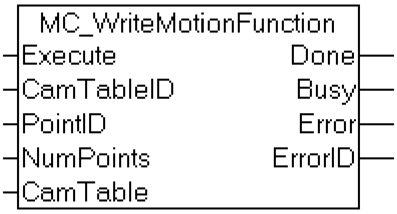
Done	Becomes TRUE, if the data were read successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Point	Data structure [▶ 54] containing the data of a motion function interpolation point

5.3 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 [▶ 45].

The function block MC_SetCamOnlineChangeMode [▶ 33] 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 status flag *Axis.Status.CamDataQueued* (AXIS_REF) can be used to check whether data have been queued (i.e. written but not yet activated).

Inputs

```
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 [▶ 44] of the loaded table.
PointID	Point ID [▶ 55] of the first point of the motion function to be written.
NumPoints	Number of motion function points to be written.

Outputs

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

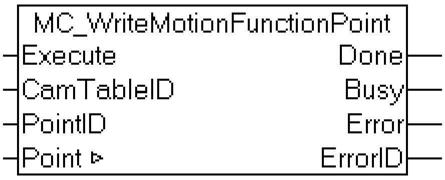
Done	Becomes TRUE, if the data were read successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

CamTable	Reference to the table [▶ 45] (structure). The start address of the table data structure (CamTable.pArray) indicates the first point to be written.

5.4 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 [▶ 33] 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 status flag *Axis.Status.CamDataQueued* (AXIS_REF)can be used to check whether data have been queued (i.e. written but not yet activated).

Inputs

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

Execute	The command is executed with rising edge.
CamTableID	ID [▶ 44] of the loaded table.
PointID	Point ID [▶ 55] of the first point of the motion function to be written.

Outputs

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

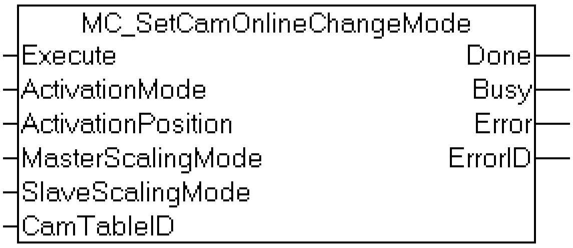
Done	Becomes TRUE, if the data were written successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Point	Data structure containing the data of a motion function interpolation point [▶ 54]

5.5 MC_SetCamOnlineChangeMode



The function block *MC_SetCamOnlineChangeMode* specifies the mode for write access to cam plate data.

Cam plate can be modified at run time via the PLC (see MC_WriteMotionFunction [▸ 31],MC_WriteMotionFunctionPoint [▸ 32]). The function block *MC_SetCamOnlineChangeMode* is used to specify when and how these changes take effect. The set mode affects all subsequent write operations. It is therefore not necessary to call the block before each write access.

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

Inputs

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

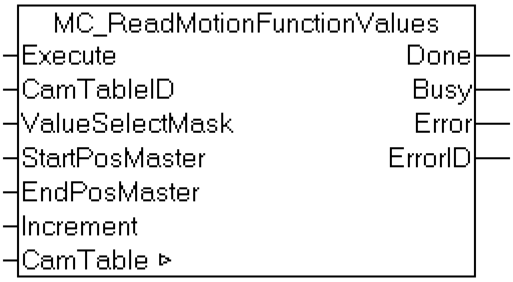
Execute	The command is executed with rising edge.
ActivationMode	Defines when and how scaling takes place . (MC_CamActivationMode [▸ 46])
ActivationPosition	Optional master position at which scaling is carried out (depending on *ActivationMode*). If *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS is used, the position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
MasterScalingMode :	Type of master scaling. (MC_CamScalingMode [▸ 49])
SlaveScalingMode	Type of slave scaling. (MC_CamScalingMode [▸ 49])
CamTableID	Table ID [▸ 44].

Outputs

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

Done	Becomes TRUE when the function has been successfully executed.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

5.6 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.

This function can be used for visualising a motion function, for example. The complete curve is digitised with a parameterisable step size. The data determined in this way are easier to display then a motion function.

Inputs

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

Execute	The command is executed with rising edge.
CamTableID	ID [▶ 44] of the loaded table (motion function type).
ValueSelectMask	The selection mask can be used to specify which data are to be interpolated and returned. The value is formed through addition of individual values of data type MC_ValueSelectType [▶ 59]. The number of columns of the data structure described by CamTable [▶ 45] must match the number of columns defined by ValueSelectMask [▶ 59]. 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

Outputs

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

Done	Becomes TRUE, if the data were read successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

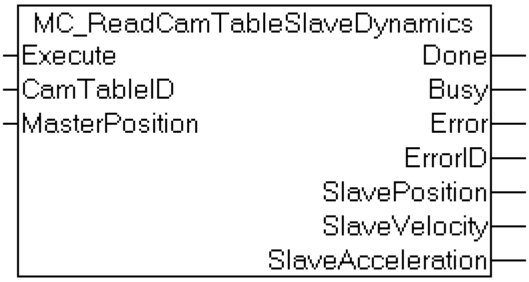
Inputs/outputs

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

CamTable	Reference to the table [▶ 45] (structure).

6 Status

6.1 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 [▸ 59], not all dynamic parameters can be determined. The following overview shows the expected result:

MC_TABLETYPE_MOTIONFUNCTION : slave position, velocity and acceleration are determined.

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

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

Inputs

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

Execute	The command is executed with rising edge.
CamTableID	Table ID [▸ 44].
MasterPosition	Master position within the table for which the slave dynamics is to be determined.

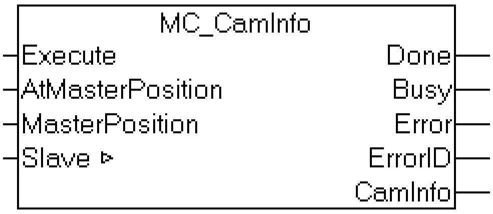
Outputs

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

Done	Becomes TRUE, if the command was executed successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
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	If the error output is set, this parameter supplies the error number.

6.2 MC_CamInfo



The *MC_CamInfo* function block 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 the state is determined with reference to the quoted master position instead of the current state. The data obtained is placed into the *CamInfo* data structure.

NOTICE

If the coupled group of axes 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.

Inputs

```
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.

Outputs

```
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	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.
CamInfo	The CamInfo data structure [▶ 52] contains all the data determined about the cam plate coupling.

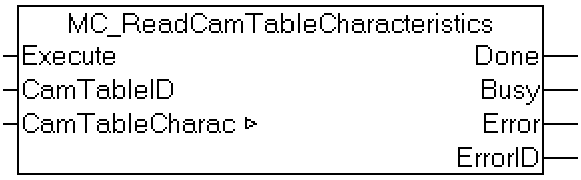
Inputs/outputs

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

Slave	Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

6.3 MC_ReadCamTableCharacteristics



The function block *MC_ReadCamTableCharacteristics* is used to calculate and read the characteristic parameters of a motion function. This includes minimum and maximum values of position, velocity, acceleration and jerk.

Inputs

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

Execute	The command is executed with rising edge.
CamTableID	Table ID [▶ 44]

Outputs

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

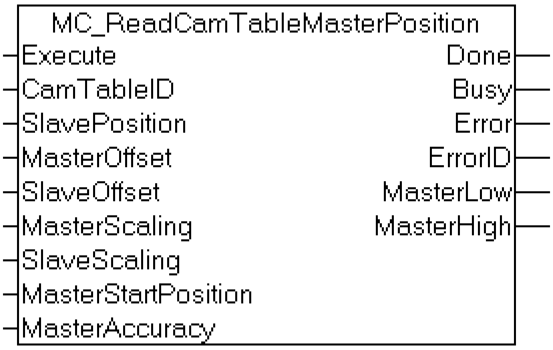
Done	Becomes TRUE, if the calculation was carried out successfully.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

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

6.4 MC_ReadCamTableMasterPosition



The function block*MC_ReadCamTableMasterPosition* can be used to calculate the master position for a given slave position. While the slave position for a given master position must be unique, the inverse is not true. In order to limit the number of master output options for the function block, for a given master position (*MasterStartPosition*) the smaller (*MasterLow*) and larger master position (*MasterHigh)* for the slave value is output.

For example, for the cam plate in Fig. 1, for a slave value of 80 and a master start value of 180, the output values are 225 for *MasterHigh* and 135 for *MasterLow*. If the cam plate is cyclic, for a master start value of 90 in addition to the *MasterHigh* of 135 a *MasterLow* of -135 is calculated. In the linear cam plate case (non-cyclic) only the value *MasterHigh* is shown as valid in Fig. 2.

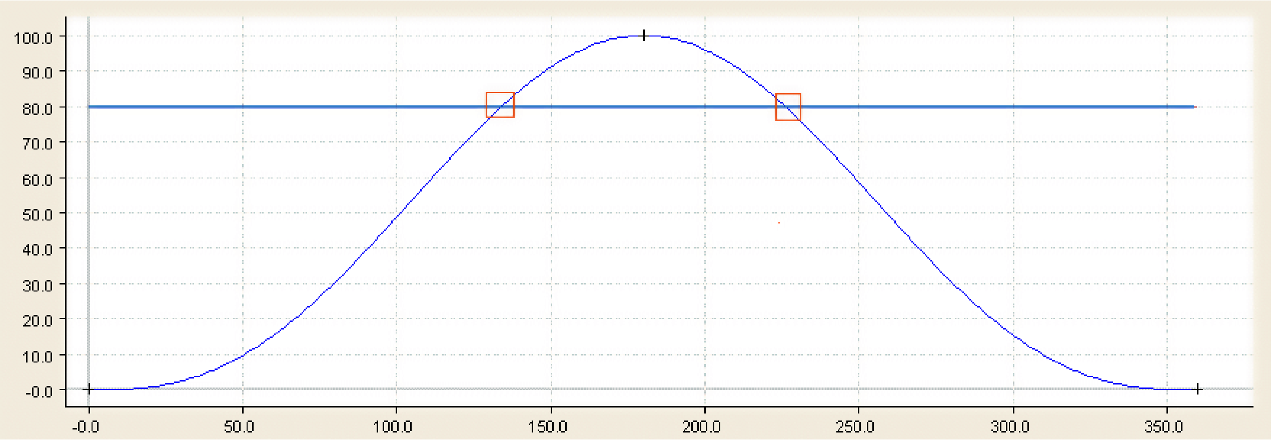


Fig. 1

In cyclic cam plates with hub the master position can not only lie in one of the cycles adjacent to the StartMasterpos, but several cycles further, depending on the slave position.

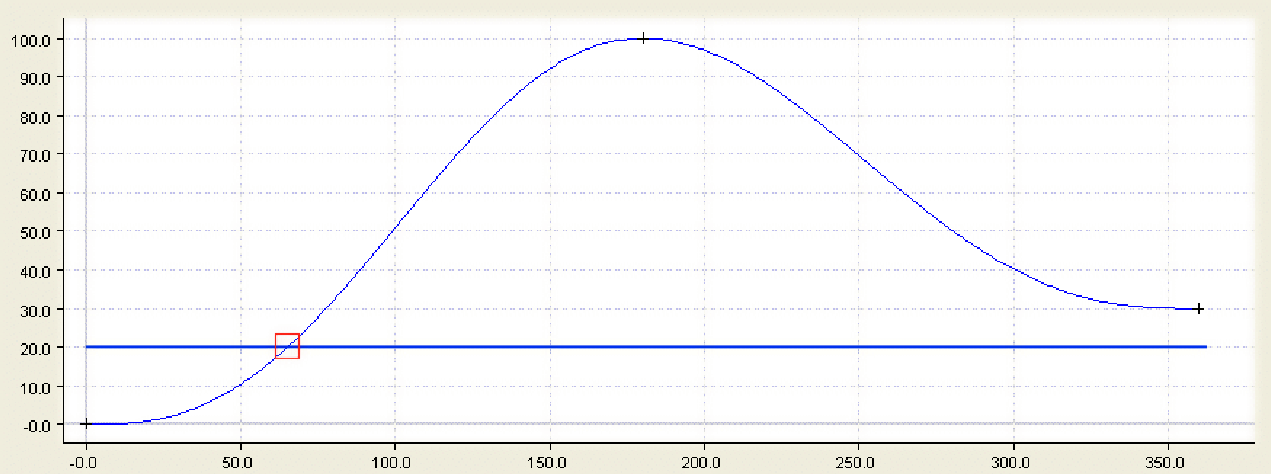


Fig. 2

The master position is calculated with numeric algorithms, the precision of which can be set via the variable *MasterAccuracy*.

Inputs

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

Execute	The command is executed with a rising edge at input *Execute*.
CamTableID	ID [▶ 44] of the cam plate for which the calculation is carried out
SlavePosition	The slave position for which the master position is sought
MasterOffset	Offset to the master position of the cam plate
SlaveOffset	Offset to the slave position of the cam plate
MasterScaling	Scaling of the master position of the cam plate
SlaveScaling	Scaling of the slave position of the cam plate
MasterStartPosition	Start position of the master
MasterAccuracy	Precision for the calculation

Outputs

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

Done	Becomes TRUE, if the coupling was successful and the cam plate is active.
Busy	The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. When *Busy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, *Done* or *Error*, is set.

Error	Becomes TRUE, as soon as an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.
MasterLow	Master position is smaller than the MasterStartPosition in the data structure ST_CamMasterData [▶ 62]
MasterHigh	Master position is smaller than the MasterStartPosition in the data structure ST_CamMasterData [▶ 62]

7 Data type

7.1 Data type MC_CAM_ID

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

Type definition for the tables ID.

7.2 Data type MC_CAM_REF

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

The data structure *MC_CAM_REF* describes the data memory of a cam plate in a further PLC variable (array).

The first parameter *pArray* is a pointer to a data structure containing the cam plate 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).

Table 1: Example 1: Position table structure description

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. The address can be assigned with the ADR function.
	Example:
	Table1 : ARRAY[0..360, 0..1] OF LREAL;
	pArray := ADR(Table1);

ArraySize	Storage capacity of the two-dimensional array, which can be determined with the SIZEOF function. Example: ArraySize := SIZEOF(Table1);
TableType	The table type [▶ 59] is *MC_TABLETYPE_EQUIDISTANT*, if the master positions have the same distance, or *MC_TABLETYPE_NONEQUIDISTANT* it the distance is variable.
NoOfRows	The number or rows corresponds to the number of table points.
NoOfColumns	The number of columns is 2.

Table 2: Example 2: Structure description of a motion function

pArray	Address of a one-dimensional array of type MC_MotionFunctionPoint. [▶ 54] Each array element contains a description of a cam plate interpolation point.
	Example:
	MotionFunction : ARRAY[1..10] OF MC_MotionFunctionPoint;
	pArray := ADR(MotionFunction);

ArraySize	Storage capacity of the one-dimensional array, which can be determined with the SIZEOF function. Example: ArraySize := SIZEOF(MotionFunction);
TableType	The table type is *MC_TABLETYPE_MOTIONFUNCTION*.
NoOfRows	The number or rows corresponds to the number of table points.
NoOfColumns	The number of columns is 1.

7.3 Data type MC_CamActivationMode

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

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

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

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

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

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

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

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

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

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

MC_CamActivationMode specifies the timing and type of change for a cam plate. Changes can be affected through scaling, modification of the cam plate data, or switching of cam plates.

The following modes are possible:

Scaling of cam plates

Cam plates can be scaled with the function block MC_CamScaling [▶ 17]. The following activation modes are available.

MC_CAMACTIVATION_INSTANTANEOUS	Scaling takes effect immediately.
MC_CAMACTIVATION_ATMASTERCAMPOS	Scaling takes effect at a certain cam plate position (master position within the cam plate). The scaling command must be issued ahead of this position. The position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
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 is carried out. This can be used to limit scaling to one axis (master or slave), for example.

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

MC_SetCamOnlineChangeMode [▶ 33] is used to specify when modified cam plate data become active (see also MC_WriteMotionFunction [▶ 31] and MC_WriteMotionFunctionPoint [▶ 32]).

In both cases the following modes are possible:

MC_CAMACTIVATION_INSTANTANEOUS	The change takes effect immediately.
MC_CAMACTIVATION_ATMASTERCAMPOS	The change takes effect at a certain cam plate position (master position within the cam plate). The command must be issued ahead of this position. The position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
MC_CAMACTIVATION_ATMASTERAXISPOS	The change takes effect at a certain absolute position of the master axis. The command must be issued ahead of this position.
MC_CAMACTIVATION_NEXTCYCLE	For a cyclic cam plate, the change takes effect at the transition to the next period.
MC_CAMACTIVATION_ASSOONASPOSSIBLE	Modified cam plate data take effect as soon as 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

The function block MC_CamIn [▶ 12] can be used to couple axes with cam plates. *ActivationMode* can optionally be used to specify from which position the slave axis becomes active.

MC_CAMACTIVATION_INSTANTANEOUS	Cam plate coupling takes effect immediately, and the slave moves according to the cam plate data.
MC_CAMACTIVATION_ATMASTERCAMPOS	Cam plate coupling activation is suspended. The slave only moves from a defined cam plate position (master position within the cam plate) according to the cam plate data. In coupling mode, the *ActivationMode* cannot be used in conjunction with MC_StartMode [▶ 60] = MC_STARTMODE_RELATIVE or MC_STARTMODE_MASTERREL_SLAVEABS. The position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
MC_CAMACTIVATION_ATMASTERAXISPOS	Cam plate coupling activation is suspended. The slave only moves from a defined absolute position of the mster axis according to the cam plate data.
MC_CAMACTIVATION_NEXTCYCLE	Cam plate coupling activation is suspended. The slave moves from the next cycle transition (for cyclic cam plates). In coupling mode, the *ActivationMode* cannot be used in conjunction with MC_StartMode [▶ 60] = MC_STARTMODE_RELATIVE or MC_STARTMODE_MASTERREL_SLAVEABS.
MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION	The cam will be activated when the master overruns the defined position in positive direction
MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION	The cam will be activated when the master overruns the defined position in negative direction

Switching of cam plates

The function block MC_CamIn [▶ 12] can be used to switch between cam plates in coupled state. *ActivationMode* can be used to specify from which position the changeover takes place.

MC_CAMACTIVATION_INSTANTANEOUS	The cam plate is switched immediately, and the slave moves according to the new cam plate data.

MC_CAMACTIVATION_ATMASTERCAMPOS	Cam plate switching takes place at a defined cam plate position (master position within the cam plate). The position refers to the non-scaled cam plate. If the position in the application refers to the scaled cam plate, it can be divided by the *MasterScaling* before the function block is called.
MC_CAMACTIVATION_ATMASTERAXISPOS	Cam plate switching takes place at a defined absolute master axis position.
MC_CAMACTIVATION_NEXTCYCLE	For cyclic cam plates cam plate switching takes place at the next cycle transition. For linear cam plates the switchover takes place at the edges of a defined region.
MC_CAMACTIVATION_DELETEQUEUEDDATA	A suspended cam plate switching process that has not been activated is discarded.
MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION	The cam will be activated when the master overruns the defined position in positive direction
MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION	The cam will be activated when the master overruns the defined position in negative direction

7.4 Data type MC_CamScalingMode

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

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

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

Type and scope of the scaling of a cam plate coupling via function block MC_CamScaling [▶ 17].

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

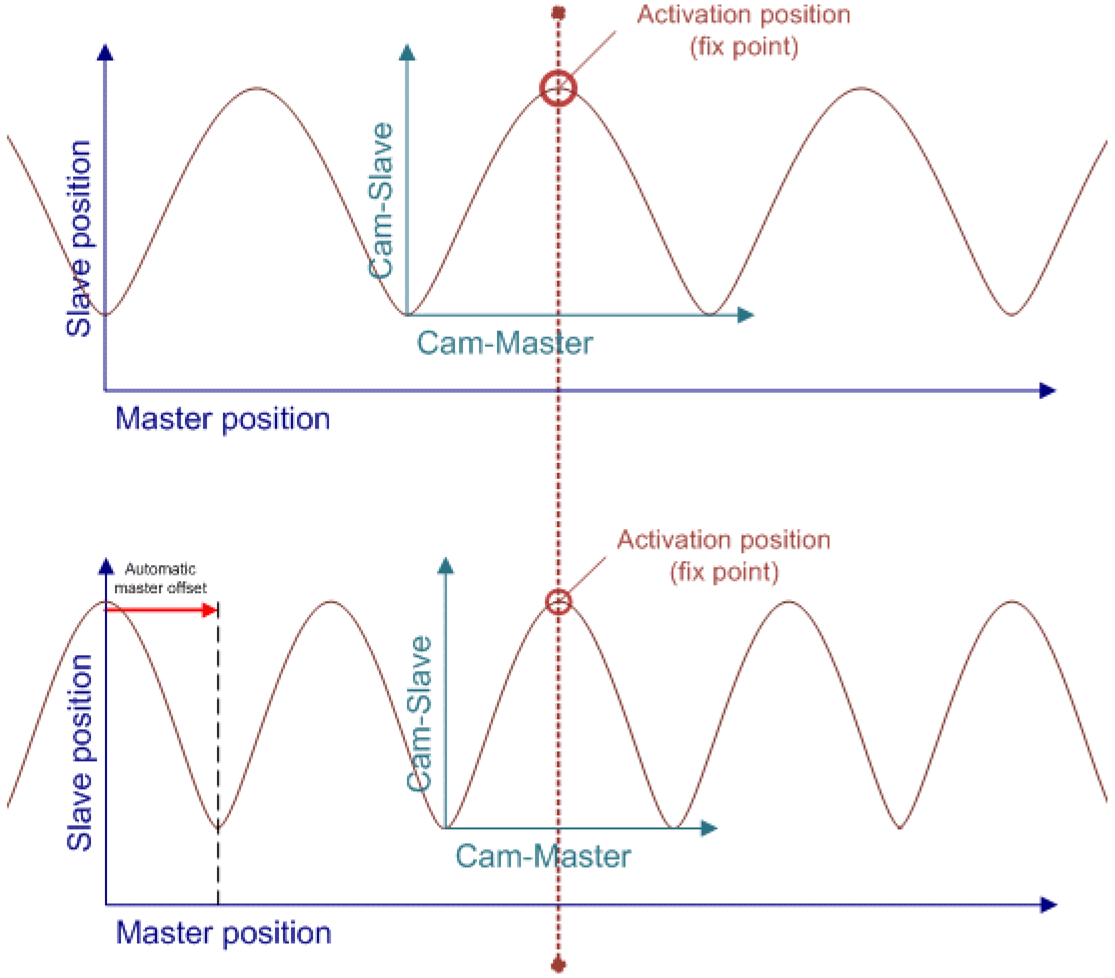
Autooffset mode ensures automatic adaptation of a cam plate offset. *Autooffset* can be used independently for the master or slave axis of a cam plate and affects both switchover and scaling of cam plates. The function operates based on the rules described below.

Master-Autooffset

Master-Autooffset Prevents discontinuity of the master position of the cam plate in the axis coordinate system during switching of cam plates with different master cycle or scaling of cam plates (master scaling). This function is required because the relative position of a cam plate in the axis coordinate system depends on the master cycle. If the master cycle is changed, e.g. through scaling, the position would change.

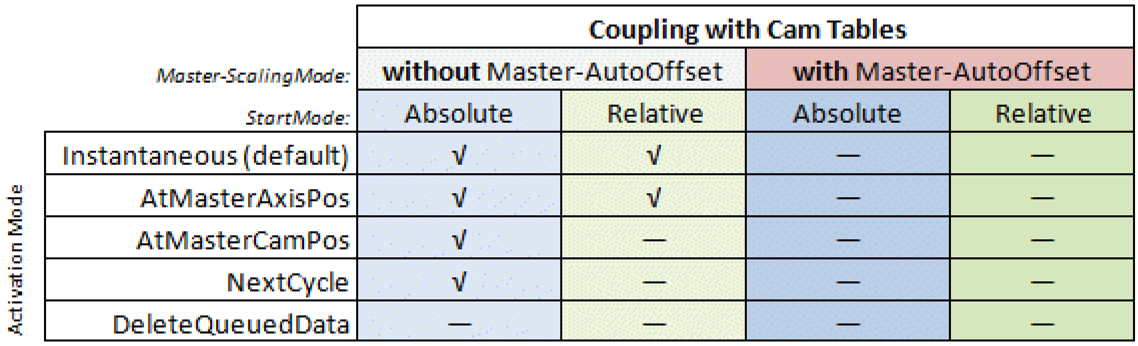
*Master-Autooffset*always needs an existing cam table to refer to and can therefore not be used with an initial coupling operation.*Master-Autooffset* determines the master offset of the cam plate such that the master position within the cam plate is maintained. For scaling or switchover to a cam plate with a different master cycle this means that the relative (percentage) position before and after the switchover is identical.

Example: A cam plate has master cycle of 360° and is scaled by a factor of 2 to 720°. Scaling takes place at the 90° position within the cam plate, i.e. at 25% of the start of a cycle. After the scaling the relative master position in the cam plate at 180° is therefore also 25% of the start of a cycle.

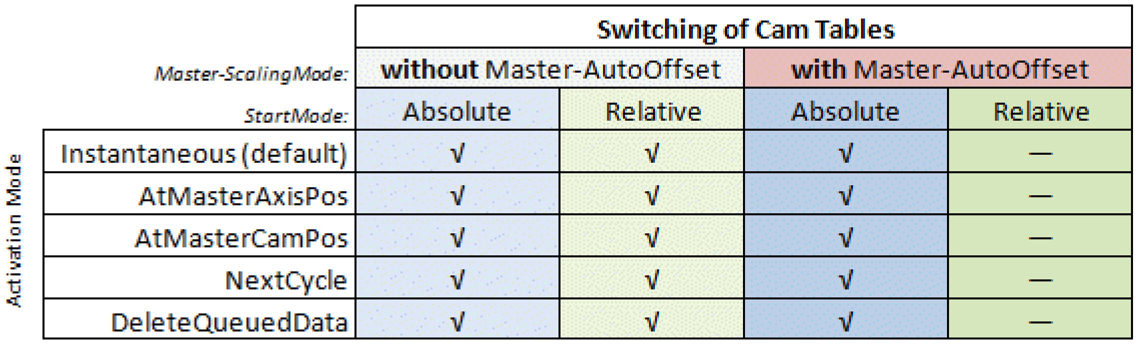


During a switchover at the edges of a cam plate (see MC_CamActivationMode [▸ 46]*MC_CAMACTIVATION_NEXTCYCLE*), Master-Autooffset ensures a seamless sequence of cam plates, both for cyclic and linear cam plates.

Master-Autooffset cannot be used for a cam plate with relative coupling or switching, since these functions are mutually exclusive. Further restrictions apply to initial coupling. These are shown in the following table.



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



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

Slave-Autooffset

Slave-Autooffset calculates a slave offset such that discontinuities in the slave position are avoided during cam plate switching or scaling. The slave offset is adjusted to ensure that the slave position is identical before and after the action.

If both *Master Autooffs*et and *Slave-Autooffset* are used for cam plate switching or scaling, the master offset is calculated first, followed by the slave offset.

Slave-Autooffset can be used with any MC_StartMode [▶ 60] and will always adjust the cam plate such that the slave position doesn't jump.

7.5 Data type MC_CamInfoData

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

   (* mode for the scaling of cam tables *)
   ScalingPending                : BOOL; (* a change is currently pending *)
   ScalingActivationMode         : MC_CamActivationMode;
   ScalingActivationPos          : LREAL;
   ScalingMasterScalingMode      : MC_CamScalingMode;
   ScalingSlaveScalingMode       : MC_CamScalingMode;

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

   (* mode for exchanging cam tables with MC_CamIn *)
   CamTableQueued                : BOOL; (* a change is currently pending *)
   CamExchangeCamTableID         : MC_CAM_ID;
   CamExchangeActivationMode     : MC_CamActivationMode;
   CamExchangeActivationPos      : LREAL;
   CamExchangeMasterScalingMode  : MC_CamScalingMode;
   CamExchangeSlaveScalingMode   : MC_CamScalingMode;
END_STRUCT
END_TYPE
```

The data structure *MC_CamInfoData* contains data on the current state of a cam plate coupling. The data are determined with the function block MC_CamInfo [▶ 38].

The structure contains absolute axis positions relating to the master or slave axis coordinate system. It also contains cam plate positions relating to the cam plate coordinate system refer (e.g. *ActualMasterCamPos* and *ActualSlaveCamPos*). All cam positions relate to the non-scaled cam plate coordinate system and can be converted to the scaled coordinate system, if required. A master cam position can be multiplied with the *MasterScaling* factor, a slave cam position can be multiplied with the *SlaveScaling* factor.

The activation positions (*ActivationPos*) relate to the master axis coordinate system or the cam plate coordinate system, depending on the *ActivationMode*. In the latter case non-scaled cam plate positions are specified.

7.6 Data type MC_InterpolationType

Interpolation mode for position tables (cam plates). Position tables consist of a list of master and slave positions between which interpolation can take place in different ways.

The interpolation type is not used for extended cam plates (motion functions).

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

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

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

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

7.7 Data type 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 a interpolation point of a motion function. A motion function is a one-dimensional list (array) of type *MC_MotionFunctionPoint*.

PointIndex: Absolute index 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 [▶ 56] 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

7.8 Data type MC_MotionFunctionPoint_ID

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

Type definition for the point IDs for a motion function.

7.9 Data type 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 definition for motion functions.

Consider type

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

7.10 Data type 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 definition for the tables point.

⚠ **DANGER**

Danger to life or risk of serious injury or damage to property due to unintentional movements of the axis

Using MOTIONPOINTTYPE_IGNORE for the first and last MotionFunctionPoint of a table definition is not allowed and must not be used.

7.11 Data type 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 definition for the characteristic parameters of a motion function.

7.12 Data type MC_TableErrorCodes

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

7.13 Data type MC_TableType

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

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

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

7.14 Data type 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 definition for access to value tables with the function block MC_ReadMotionFunctionValues [▶ 34].

7.15 Data type MC_StartMode

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

StartMode is used for coupling with cam plates through MC_CamIn [▶ 12] 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 *StartModeabsolute* 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 *StartModerelative* 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. The cam plate can additionally be moved through an offset.

The modes MC_STARTMODE_RELATIVE and MC_STARTMODE_MASTERREL_SLAVEABS cannot be used in conjunction with automatic master offset calculation (MC_CamScalingMode [▶ 49]), since this would cause a conflict.

7.16 Data type ST_CamInOptions

Data of type *ST_CamInOptions* can be transferred optionally to the function block MC_CamIn [▶ 12].

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

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

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

7.17 Data type CamMasterData

Data of type CamMasterData are optionally transferred by function block MC_ReadCamTableMasterPosition [▶ 41].

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

7.18 Data type MC_CamOperationMode

The *CamOperationMode* is used for managing couplings with superimposed cam plates with the function block MC_CamIn_V2 [▶ 19] (multi-cam). Cam plates can be added, switched or removed.

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

7.19 Data type ST_CamScalingData

The structure *ST_CamScalingData* contains information for scaling a cam plate and is used with the function block MC_CamIn_V2 [▶ 19].

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

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

MasterScalingMode	Scaling mode [▶ 49] for the master position of the cam plate
MasterRelative	If TRUE the cam plate operates relative to the current master position at the time of activation.
MasterOffset	Master offset for orientation of the cam plate in the axis coordinate system. *MasterOffset* takes effect in absolute mode from the master axis position 0 and in relative mode from the current position at the time of activation.
MasterScaling	Scaling of the master position of the cam plate. Default is 1.0
SlaveScalingMode	Scaling mode [▶ 49] for the slave position of the cam plate
SlaveRelative	If TRUE the cam plate operates relative to the current slave position at the time of activation.
SlaveOffset	Slave offset for orientation of the cam plate in the axis coordinate system. *SlaveOffset* takes effect in absolute mode from the slave axis position 0 and in relative mode from the current position at the time of activation.
SlaveScaling	Scaling of the slave position of the cam plate. Default is 1.0

8 Example programs

Electronic Cam Tables

The example program couples a master and a slave axis via cam plates. During the coupled movement cam plates are switched, individual sampling points of a cam plate are modified, and the cam plate is scaled.

The example program requires the additional TcMC2_Camming.lib library and operates fully in simulation mode. Progress can be monitored in TwinCAT Scope View with the configuration provided.

Click here to save the example program:

https://infosys.beckhoff.com/content/1033/tcplclibmc2_camming/Resources/460479371.zip

Rotary Knife and Registration

The sample uses a rotational knife to cut sheets of a defined length. For this purpose the circumference speed must be synchronized with the web while cutting a sheet. The rotation knife must the accelerator or deceleration since the circumference of the knife is different from the sheet length. Registration marks are used to synchronize with the product. The knife is continuously adjusted to compensate small differences caused by temperature or stretching.

An electronic cam table is used to synchronize the rotational knife with the material. The cam table is a normalized cam table defined with a length of 360°. The tool moves over a full turn while the cam table is executed and the cut position is defined as 0°. The circumference speed is synchronous from 270° to 30°. The range from 30° to 270° is used to adjust the operation to the actual distance of registration marks.

https://infosys.beckhoff.com/content/1033/tcplclibmc2_camming/Resources/460482315.zip

Trademark statements

More Information:
www.beckhoff.com/ts5050

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