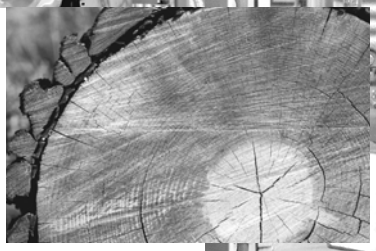
SEW EURODRIVE

MPLCTec.._MX Libraries for MOVI-PLC®
GearMotion, CamMotion, CamSwitch

Edition 01/2009
16751612 / EN

Manual



SEW EURODRIVE

1 General Notes

1.1 Structure of the safety notes

The safety notes in this manual are designed as follows:

Pictogram	SIGNAL WORD
	Type and source of danger. Possible consequence(s) if the safety notes are disregarded. • Measure(s) to prevent the danger.

Pictogram	Signal word	Meaning	Consequences if disregarded
Example:  General danger  Specific danger, e.g. electric shock	DANGER	Imminent danger	Severe or fatal injuries
	WARNING	Possible dangerous situation	Severe or fatal injuries
	CAUTION	Possible dangerous situation	Minor injuries
	STOP	Possible damage to property	Damage to the drive system or its environment
	TIP	Useful information or tip. Simplifies the handling of the drive system.	

1.2 Right to claim under warranty

A requirement of fault-free operation and fulfillment of any rights to claim under limited warranty is that you adhere to the information in the MOVI-PLC® documentation. Therefore, read the manual before you start operating the device!

Make sure that the manual is available to persons responsible for the plant and its operation, as well as to persons who work independently on the device. You must also ensure that the documentation is legible.

1.3 Exclusion of liability

You must comply with the information contained in the MOVI-PLC® documentation to ensure safe operation of the MOVI-PLC® controller and to achieve the specified product characteristics and performance requirements. SEW-EURODRIVE assumes no liability for injury to persons or damage to equipment or property resulting from non-observance of the operating instructions. In such cases, any liability for defects is excluded.

1.4 Other applicable documentation

- Installation and startup only by trained personnel observing the relevant accident prevention regulations and the following documents:
  - "MOVIAXIS® MX Multi-Axis Servo Inverter" operating instructions
  - "MOVI-PLC® *advanced* DH.41B" manual
  - "MPLCMotion_MC07 and MPLCMotion_MM Libraries for MOVI-PLC®" manual.
  - "Libraries for MOVI-PLC® – Fault Codes" manual
  - "MOVI-PLC® Programming in the PLC Editor" system manual
- Read through this manual carefully before you commence installation and startup of the MOVI-PLC® controller.
- As a prerequisite to fault-free operation and fulfillment of warranty claims, you must adhere to the information in the documentation.

1.5 General safety notes for bus systems

You are now in possession of a communication system that lets you adapt MOVI-PLC® controllers and MOVIAXIS® multi-axis servo inverters to the particulars of your specific system. As with all bus systems, there is a danger of invisible, external (as far as the inverter is concerned) modifications to the parameters which give rise to changes in the unit behavior. This may result in unexpected (not uncontrolled) system behavior.

1.6 Safety functions

MOVI-PLC® may not execute any safety functions.

For safety applications, ensure that the information in the following publication is observed.

- Safe disconnection for MOVIAXIS®

Use only those components in safety applications that were explicitly designed and delivered for this purpose by SEW-EURODRIVE.

1.7 Hoist applications

- Hoist applications can only be implemented with MOVI-PLC® under the following conditions:
  - A hoist startup must be performed.
- MOVI-PLC® is not designed for use as a safety device in hoist applications.

  Use monitoring systems or mechanical protection devices as safety equipment to avoid possible damage to property or injury to people.

2 MPLCTecGearMotion_MX

2.1 Introduction

The MOVIAXIS® multi-axis servo inverter offers comprehensive technology functions, for example:

- Electronic gear unit,
- Cams,
- Touch probe,
- Event control,
- Cam control.

The "Motion Technology Editor" startup assistant of MotionStudio enables configuration of all technology functions. Basic settings are configured this way. However, certain values or settings often have to be changed dynamically during a process, or various functions must be combined and run in sequence.

The function modules of the "MPLCTecGearMotion_MX" library set the general Gear parameters and configure other technology function that are required for the respective Gear function.

This means that in-depth familiarization with the individual parameters of the respective technology functions is not necessary. Configuration with the Technology Editor is not necessary, because the "MPLCTecGearMotion_MX" library performs all relevant settings.

Advantages of the "MPLCTecGearMotion_MX" library:

- Programming in accordance with IEC 61131, which means customer-specific applications can be implemented easily.
- Library for controlling the Gear technology function. This allows for simple configuration and control of the gear functionality without detailed knowledge of the individual technology function.
- Central control of several synchronous axes.
- Additional functions, such as specifying master values via a virtual master encoder, are controlled centrally using the MOVI-PLC®.
- Different startup cycle events and types can be selected.
- Additional offset travel can be activated during the startup cycle and/or in synchronous operation.
- Different offset start events and types can be selected.

Additional notes are can be found in the following documentation:

- For general notes on the operating principle of the libraries, refer to the publication "MPLCMotion_MDX and MPLCMotion_MX libraries for MOVI-PLC®".
- "MOVIAXIS® MX Multi-Axis Servo Inverter, Electronic Gear Unit Technology Function" manual.

2.2 Areas of application

The MPLCTecGearMotion_MX library is suited for all applications that require synchronized axis movements.

Application examples

- Flying saw, e.g. to cut continuous material
- Synchronous material handling, e.g. several conveyor belts are synchronized to one master encoder
- Multi-axis hoists/trolleys
- Filling station in the beverage industry
- Packaging technology, e.g. FFS machines

Characteristics

- Up to 64 (with MOVI-PLC® *advanced*) motor axes can be operated synchronously
- Various configuration options can be set easily with the respective function modules
- Various master encoder sources can be set
- Different startup cycle types can be selected
- Different offset types can be selected
- Simple offset switch-in during the startup cycle or in synchronous operation
- Various master/slave combinations can be implemented; several synchronous operation masters can be configured

2.3 Project planning

2.3.1 Prerequisites

- To use the MPLCTecGearMotion library, you need a MOVI-PLC® controller in technology version T1 or higher.
- The MPLCMotion_MX library must be integrated. Also adhere to the prerequisites in the "MPLCMotion_MDX and MPLCMotion_MX Libraries for MOVI-PLC®" manual.

2.3.2 Basic principle and notes

- Configuring the relevant technology functions of the axis using IEC function modules.
- The startup cycle and synchronization are performed with the Gear technology function. After that, the unit switches to linear cam function automatically. This enables offset travel during synchronous operation.
- Fast, interrupt-controlled startup cycle through use of the touch probe function and event control (is automatically configured correspondingly).
- Time-dependent offset realized by cam profile generator superimposition. Option for relative or absolute offset.
- Position-dependent offset through superimposition of a second, appropriately configured cam. Offset start configured as interrupt-controlled, after defined master distance or directly from the PLC program.
- Interrupt or counter-controlled startup cycle and offset processes can be repeated automatically using the AutoRestart function.
- The "MC_LinkTecGear_MX" function module is absolutely essential as a central data interface to the Gear technology function.
- The Technology Editor of MotionStudio is not necessary, but for diagnostic purposes, an online connection of the monitor can be established with the sample project "MOVI-PLC Defaultprojekt.Tec-Function". The sample project may not be downloaded, otherwise settings made by the library would be overwritten.

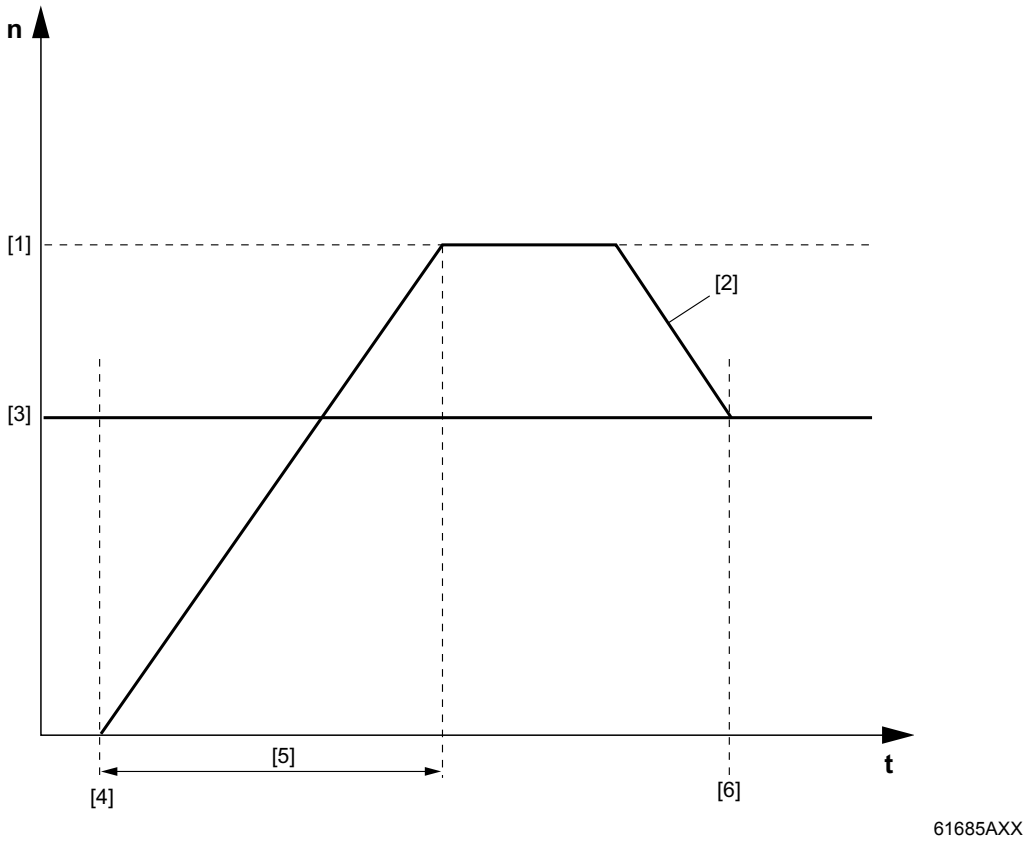
2.3.3 Different startup cycle times

You can define whether the startup cycle is to be performed time-related or position-dependent.

Time-related startup cycle

Time-related startup cycle is selected with the "MC_PrepareGearInTimeBased_MX" module.

Time-related startup cycle means the slave synchronizes to the master using the set velocity *GearSpeed* and *GearAcceleration/Deceleration*.



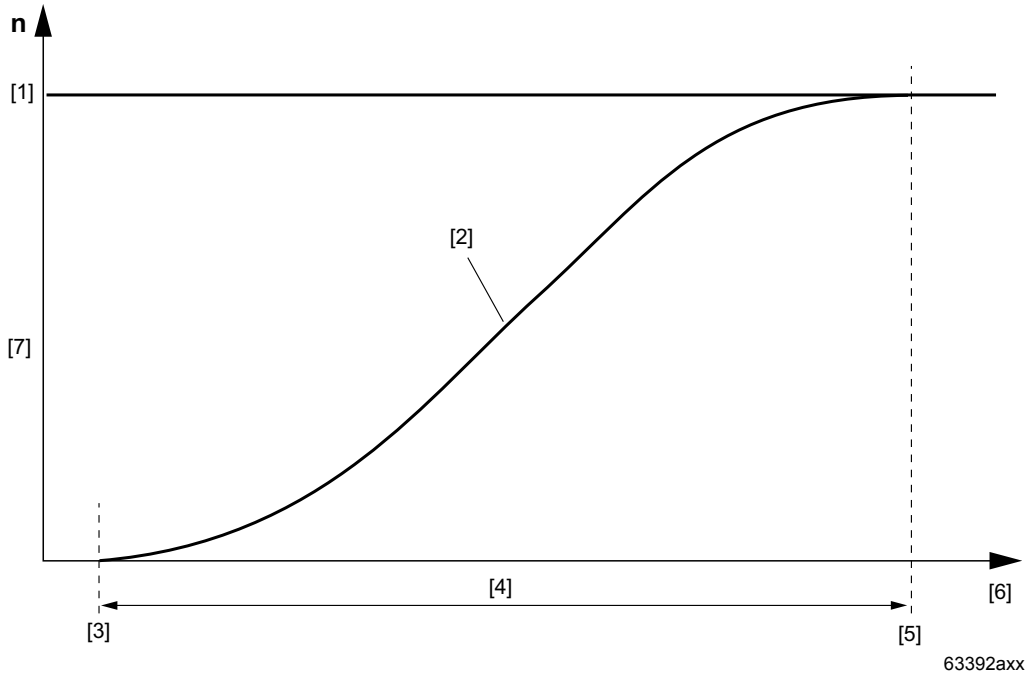
[1] Velocity set with MC_SetGearDynamics_MX
[2] Velocity of the slave
[3] Velocity of the master
[4] Start of the time-related startup cycle
[5] Ramp set with MC_SetGearDynamics_MX (based on 3000 rpm)
[6] Slave is in sync with the master

Position-dependent startup cycle

A position-dependent startup cycle is selected with the "MC_PrepareGearInDistanceBased_MX" module.

Position-dependent startup cycle means the slave synchronizes to the master within a master distance that can be set. The distance that the slave travels is set with *GearInSlaveDistance*

The position-dependent startup cycle is direction-sensitive. This means that if the master distance for the startup cycle, *GearInMasterDistance*, is specified as a positive value, the master must also turn in a positive direction.



[1] Velocity of the master
[2] Velocity of the slave
[3] Start of the position-dependent startup cycle
[4] Master travel for the startup cycle
[5] Slave is in sync with the master
[6] Position of the master
[7] Slave travel for the startup cycle

	TIPS
	The startup cycle profile is calculated with a fifth-degree polynomial. This is why the startup cycle distance of the slave should be roughly half of the scaled master distance. Otherwise, the slave can overshoot or change the direction of rotation briefly during the startup cycle.

GearInSlaveDistance = 0.5 × GearInMasterDistance × Numerator/Denominator

2.3.4 Offset cycle

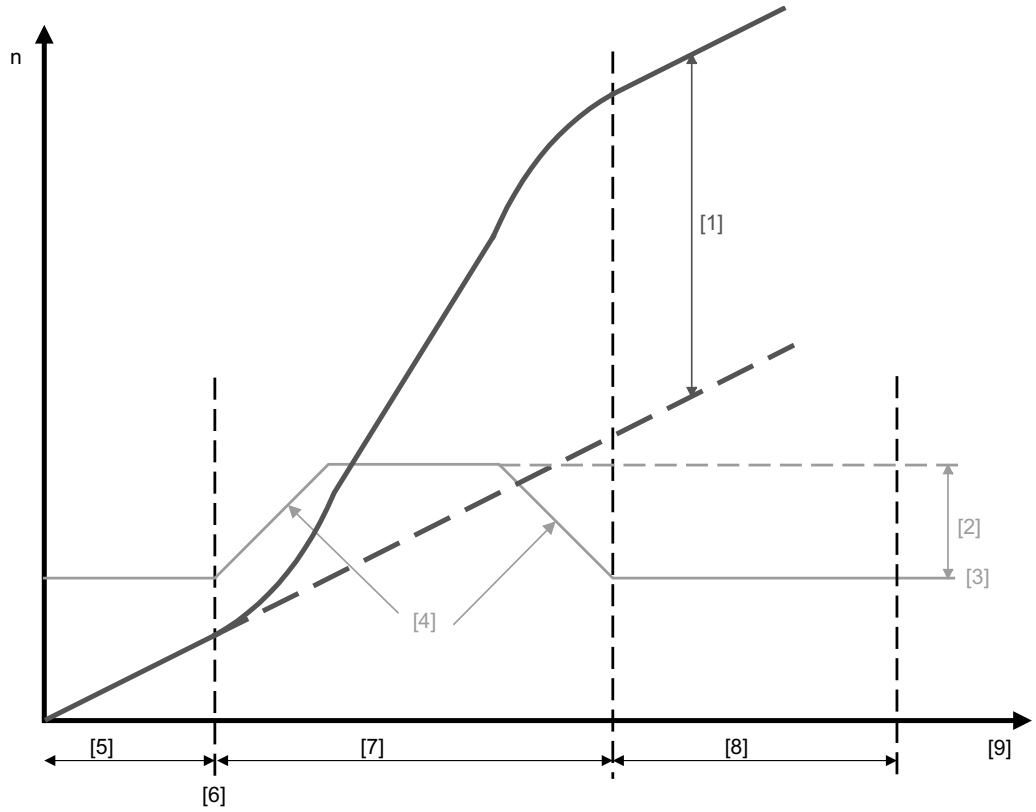
An offset travel can be activated during synchronous operation after a successfully completed startup cycle. Relative or absolute travel can be selected with the *OffsetMode* input:

OffsetMode = MX_GEAR_OFFSET_RELATIVE

Positioning is superimposed relative to the synchronous movement with the master. A permanent phase shift can be implemented in this way.

OffsetMode = MX_GEAR_OFFSET_ABSOLUTE

The *OffsetDistance* is traveled to absolutely with this setting. The reference value is set to zero upon changing to Cam (FCB16); the following position values refer to this value. An offset can be performed with this, for example, and with a second offset travel with *Distance* = 0, the phase shift can be cancelled again.



[1]	OffsetPosition	[6]	Start offset
[2]	OffsetSpeed	[7]	Offset travel
[3]	n slave	[8]	Synchronous
[4]	Offset Acceleration / Deceleration	[9]	Master position
[5]	Synchronous		

2.3.5 Different startup cycle and offset start events

In addition to the type of the startup cycle, the "PrepareGearIn..." modules also define which event is to start the synchronization process.

Independent of whether the startup cycle is time- or position-dependent, the startup cycle event is set with the *GearInEvent* input at the respective "Prepare" module.

Possible startup cycle events:

- MX_GEAR_DIRECT

  The startup cycle is directly started with the "MC_GearInDirect_MX" function module.

- Interrupt-controlled startup cycle events:

  MX_GEAR_INTERRUPT_DI01_POS_FLAG

  MX_GEAR_INTERRUPT_DI01_NEG_FLAG

  MX_GEAR_INTERRUPT_DI01

  ...

  MX_GEAR_INTERRUPT_DI08_POS_FLAG

  MX_GEAR_INTERRUPT_DI08_NEG_FLAG

  MX_GEAR_INTERRUPT_DI08

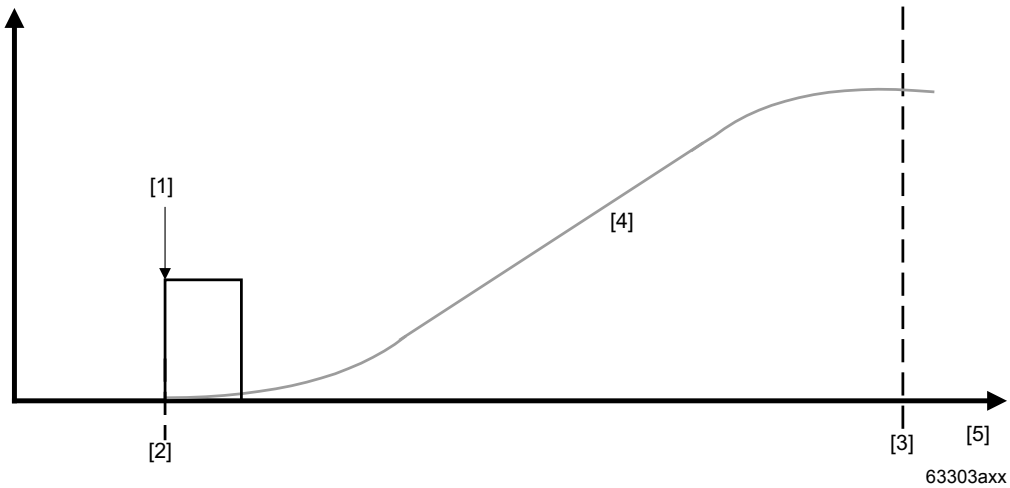
  MX_GEAR_INTERRUPT_C_TRACK_ENC1

  MX_GEAR_INTERRUPT_C_TRACK_ENC2

  MX_GEAR_INTERRUPT_C_TRACK_ENC3

  The startup cycle is started with an interrupt event triggered by a signal change at the MOVIAXIS® input DI01–DI08, with a rising/falling edge (pos/neg flag) or a signal change at DI01–DI08/C track.

Example: *GearInEvent* = MX_GEAR_INTERRUPT_DI02_POS_FLAG



[1] Interrupt DI02 (PosFlag)

[2] Startup cycle start

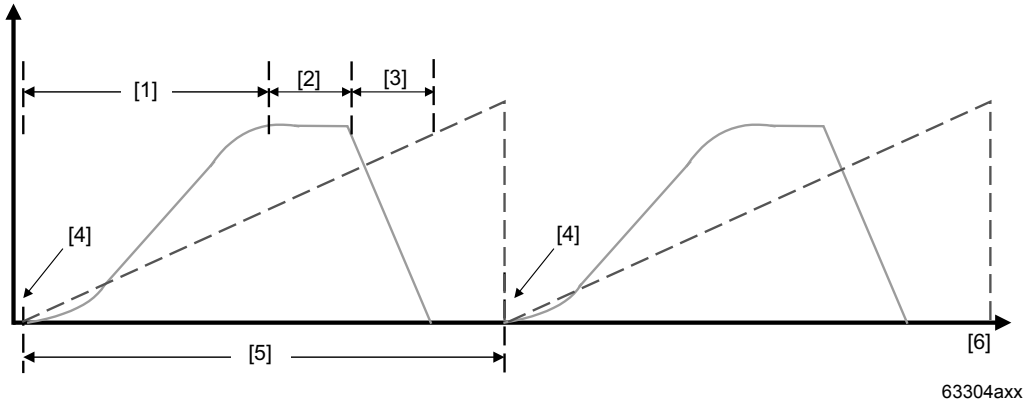
[3] Slave is in sync

[4] n slave

[5] Master position

- MX_GEAR_DISTANCE_COUNTER

  With this setting, the startup cycle is started according to the specified master distance. The current counter of the master distance is not deleted.



[1] Startup cycle
[2] Synchronous
[3] Stop cycle
[4] Startup cycle start
[5] DistanceForStartGearIn
[6] Master position

- MX_GEAR_DISTANCE_COUNTER_RESET_COUNT

  The startup cycle is similar to MX_GEAR_DISTANCE_COUNTER mode, but the counter of the master position is deleted when calling up "MC_PrepareGearIn....".

OffsetEvent

The description of the startup cycle events applies analogously for the various options to start offset travel.

2.3.6 Configuration process

The modules of the MPLCTecGearMotion_MX library are divided into four groups:

MX_Gear_Config

- MC_SetGearConfig_MX

MX_Gear_Main

- MC_LinkTecGear_MX
- MC_GearClearLag_MX

MX_Gear_Parameter

- MC_GearInControl_MX
- MC_PrepareGearInTimeBased_MX
- MC_PrepareGearInDistanceBased_MX
- MC_PrepareOffsetDistanceBased_MX

MX_Gear_SingleAxis

- MC_GearInDirect_MX
- MC_GearOffsetDirectDistance_MX
- MC_GearOffsetDirectTime_MX

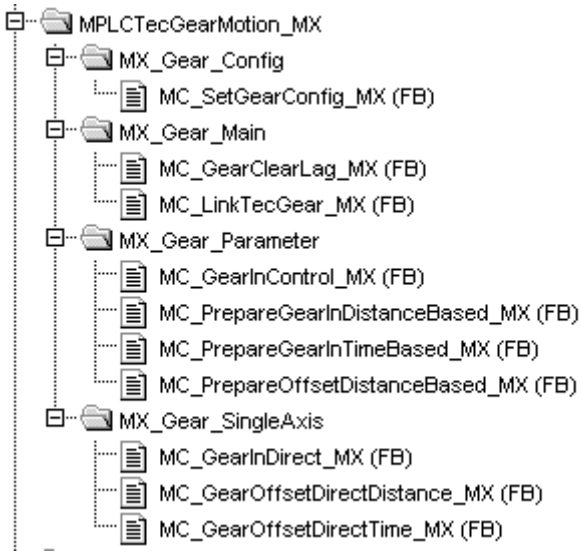
Procedure

- Integrate the "MPLCTecGearMotion_MX" library with the library manager into the project in addition to the "MPLCMotion_MX" library.
- The "MC_LinkTecGear_MX" function module constitutes the interface between technology function and axis and must therefore be called cyclically in the program. Once the "MC_ConnectAxis_MX" module has established the connection to the axis (Done = True), *LinkTecGear* activates the technology functions of the MOVIAXIS® necessary for the Gear function. In addition to that, the necessary settings are made in the PDO Editor. A receive object, for example, is set up in the In buffer 5 for receiving the master position via system bus.
- If the master position is transmitted via the system bus, the bus should be synchronized. The required SBus synchronization telegram is configured using the "MC_SetSync_CAN" function module (part of the MPLCCommunication_CAN library). If the virtual encoder of the controller is used as master, the SBus will be automatically synchronized by calling the "MC_LinkTecAxis_Virtual" function module (part of the MPLCTecVirtualEncoder library). The required basic settings in the parameter tree under communication are automatically made during initialization of *LinkTecGear*.
- The "MC_SetGearConfig_MX" function module is used to set the general synchronous operation parameters. This module must be called with a rising edge on the Execute input each time the MOVI-PLC® controller is restarted or if you want to change the general synchronous operation parameters.

- The Prepare modules are used to
  - Define the type of startup cycle,
  - Define how the offset is to be started,
  - Configure which event is to trigger this,
  - Set the respective parameters.
- The startup cycle is then performed using "MC_GearInDirect_MX" or another configured event.
- During synchronous operation, an offset travel can be started with a rising edge at the Execute input of "MC_GearOffsetDirectDistance_MX" or "MC_GearOffsetDirectTime_MX".

2.4 Overview of MPLCTecGearMotion_MX library

2.4.1 Short overview of the MPLCTecGearMotion_MX library



63305axx

MC_SetGearConfig_MX

This module is used for basic configuration of synchronous operation. Before the startup cycle can be performed, the synchronous operation parameters must have been transferred via this module.

MC_LinkTecGear_MX

This function module has different functions:

- Configuring the MOVIAXIS® technology functions and activating the functions necessary for the Gear technology function.
- Configuring the MOVIAXIS® process data interface.
- Display of the status of the Gear technology function and the initialization state.

MC_GearClearLag_MX

This function module can be used for deleting the difference between the Gear setpoint (conditioned master signal) and Gear actual value (current slave position).

MC_GearInControl_MX

This module offers additional control functions for interrupt- or counter-controlled startup cycle events. The startup cycle can be enabled or inhibited repeatedly.

MC_PrepareGearInTimeBased_MX

This Prepare module prepares the time-controlled startup cycle. The required dynamics parameters and the startup cycle event are defined and configured.

MC_PrepareGearInDistanceBased_MX

This Prepare module is used to select and configure the position-dependent startup cycle. The required master/slave startup cycle distances and the startup cycle event are defined and configured.

MC_PrepareOffsetDistanceBased_MX

This function module prepares a position-dependent offset travel. The master/slave offset distances and the offset start event are configured.

MC_GearInDirect_MX

The function module starts a direct startup cycle with a rising edge at the Execute input. The synchronization process uses the parameters set by *PrepareGearIn*.

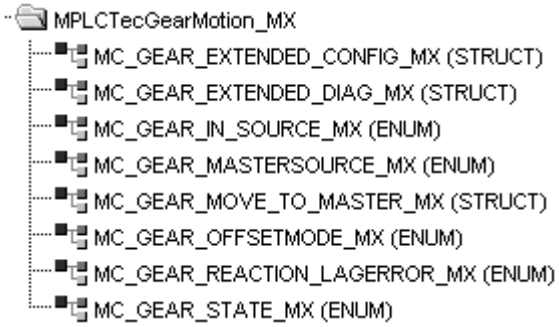
MC_GearOffsetDirectDistance_MX

This module starts a position-dependent offset travel with the set values *OffsetMasterDistance/OffsetSlaveDistance*.

MC_GearOffsetDirectTime_MX

This module activates a positioning sequence that is superimposed on the synchronous operation with the transferred parameters. Depending on "OffsetMode", a relative or absolute offset is realized ("Phasing").

2.4.2 Data types of the MPLCTecGearMotion_MX library



63306AXX

Standard configuration

- **Data type MC_GEAR_IN_SOURCE_MX**

  With this type, the start event for the startup cycle or offset travel is defined in the "PrepareGearIn" and "PrepareOffset" modules. As interrupt-controlled events, edge changes at the digital inputs 02/03 or at the C track of one of the three encoder inputs can be used. The (master) distance-counter-controlled modes are different in so far as they also offer the option to delete the current master distance counter when selecting the respective"Prepare" function module.

```
TYPE MC_GEAR_IN_SOURCE_MX :
    (
        MX_GEAR_DIRECT,
        MX_GEAR_INTERRUPT_DI01_POS_FLAG
        MX_GEAR_INTERRUPT_DI01_NEG_FLAG
        MX_GEAR_INTERRUPT_DI01
        MX_GEAR_INTERRUPT_DI02_POS_FLAG
        MX_GEAR_INTERRUPT_DI02_NEG_FLAG
        MX_GEAR_INTERRUPT_DI02
        MX_GEAR_INTERRUPT_DI03_POS_FLAG
        MX_GEAR_INTERRUPT_DI03_NEG_FLAG
        MX_GEAR_INTERRUPT_DI03
        MX_GEAR_INTERRUPT_DI04_POS_FLAG
        MX_GEAR_INTERRUPT_DI04_NEG_FLAG
        MX_GEAR_INTERRUPT_DI04
        MX_GEAR_INTERRUPT_DI05_POS_FLAG
        MX_GEAR_INTERRUPT_DI05_NEG_FLAG
        MX_GEAR_INTERRUPT_DI05
        MX_GEAR_INTERRUPT_DI06_POS_FLAG
        MX_GEAR_INTERRUPT_DI06_NEG_FLAG
        MX_GEAR_INTERRUPT_DI06
        MX_GEAR_INTERRUPT_DI07_POS_FLAG
        MX_GEAR_INTERRUPT_DI07_NEG_FLAG
        MX_GEAR_INTERRUPT_DI07
        MX_GEAR_INTERRUPT_DI08_POS_FLAG
        MX_GEAR_INTERRUPT_DI08_NEG_FLAG
        MX_GEAR_INTERRUPT_DI08
        MX_GEAR_INTERRUPT_C_TRACK_ENC1
        MX_GEAR_INTERRUPT_C_TRACK_ENC2
        MX_GEAR_INTERRUPT_C_TRACK_ENC3
        MX_GEAR_DISTANCE_COUNTER
        MX_GEAR_DISTANCE_COUNTER_RESET_COUNT
    );
END_TYPE
```

63307AXX

- **Data type MC_GEAR_MASTERSOURCE_MX**

  This data type defines the source of the master encoder signal. Either of the three MOVIAXIS® encoder inputs can be selected (linear or modulo). The virtual encoder of the axis (linear or modulo) can also be selected. In addition, the master signal can be received via SBus, e.g from the MOVI-PLC® controller or from a different axis. Two CAN objects are available for selection.

```
TYPE MC_GEAR_MASTERSOURCE_MX :
    (
        MX_GEAR_SYSTEMPOS_ENCODER1,
        MX_GEAR_SYSTEMPOS_ENCODER2,
        MX_GEAR_SYSTEMPOS_ENCODER3,
        MX_GEAR_MODULOPOS_ENCODER1,
        MX_GEAR_MODULOPOS_ENCODER2,
        MX_GEAR_MODULOPOS_ENCODER3,
        MX_GEAR_MODULO_VIRTUAL_ENCODER_MX,
        MX_GEAR_LINEAR_VIRTUAL_ENCODER_MX,
        MX_GEAR_RECEIVE_PDO_1,
        MX_GEAR_RECEIVE_PDO_2
    );
END_TYPE
```

63308axx

- **Data type MC_GEAR_STATE_MX**

  Displays the current synchronous operation state.

```
TYPE MC_GEAR_STATE_MX :
    (
        MX_GEAR_OUTGEAR,              (* Synchronous state : Axis is geared out *)
        MX_GEAR_ENGAGING_GEAR_IN,     (* Synchronous state : Axis engaging to gear *)
        MX_GEAR_IN_GEAR,              (* Synchronous state : Axis is in gear *)
        MX_GEAR_OFFSET_GEAR,          (* Synchronous state : Offset move while in gear *)
        MX_GEAR_MOVE_TO_MASTER,       (* Synchronous state : Slave moves to master after int
        MX_GEAR_NOTLINKED             (* Synchronous state : Axis not linked *)
    );
END_TYPE
```

63309AXX

- Data type MC_GEAR_OFFSETMODE_MX

  Defines whether an absolute or relative offset is to be performed.

```
TYPE MC_GEAR_OFFSETMODE_MX :
    (
        MX_GEAR_OFFSET_RELATIVE,      (* Relative Offset move *)
        MX_GEAR_OFFSET_ABSOLUTE       (* Absolute Offset move *)
    );
END_TYPE
```

63310axx

Extended configuration

The technology functions of MOVIAXIS® offer numerous setting options, filters and compensation functions that are not required for most standard applications. To keep the modules simple while providing an option for access, these parameters are integrated in the "MC_GEAR_EXTENDED_CONFIG_MX" input structure.

This structure is used both by the "MC_LinkTecGear_MX" module and by "MC_SetGearConfig_MX".

- **ExtConfig of MC_LinkTecGear_MX**

```
TYPE MC_GEAR_EXTENDED_CONFIG_MX :
STRUCT
(* ############################## Extended LinkTecGear Configurations ################################*)

    UseTecEditorConfig: BOOL;                (* based on the PLCdefault TecEditor project additional
                                                Tec configurations can be made, e.g. CAM2/3, Touchprobe, etc.*)
    TecFunctionsUserUnitSlave: BOOL;         (* True = Tec values of the slave are scaled in user units
                                                  False = Tec values are in system units (unscaled)*)

    stInitialParameter_MX : MC_INITIALPARAMETER;
                                             (*#####################################################################
                                             With this struct several user defined parameters can be written to the axis during the
                                             initialization of MC_LinkTecGear_MX. DHx11B : 10 parameters, DHx41B 60 parameters

                                             Add the following code to your program to initialise the struct :

                                             Example: write DDB variable 2577 and 3420:

                                                 stInitialParameter_MX.aIndex[1] := 20220;    (* Index of DDB Variable 2577 *)
                                                 stInitialParameter_MX.aSubIndex[1] := 18;    (* SubIndex of DDB Variable 2577 *)
                                                 stInitialParameter_MX.aData[1] := 10;        (* Data to be written to DDB Variable 2577*)

                                                 stInitialParameter_MX.aIndex[2] := 20226;    (* Index of DDB Variable 3420 *)
                                                 stInitialParameter_MX.aSubIndex[2] := 93;    (* SubIndex of DDB Variable 3420 *)
                                                 stInitialParameter_MX.aData[2] := 20;        (* Data to be written to DDB Variable 3420*)

                                             If there are more than 10 parameters to write, declare NUMOF_INITIALPARAMETER
                                             in your program.

                                                 VAR_GLOBAL CONSTANT
                                                     NUMOF_INITIALPARAMETER : UDINT := 20;
                                                 END_VAR

                                             #####################################################################*)
```

63311axx

The structure variable *UseTecEditorConfig* allows for an extended configuration with the "TecEditor" using the library modules in parallel to the Gear configuration. The touch probe function can be set here, for example, or another curve block for superimposition can be configured.

Instead of an extended configuration, this input also enables Gear configuration using the TecEditor wizard. The "MC_SetGearConfig_MX" is not required then.

	TIPS
	Procedure • Since DDB ranges might shift, the IEC program should first be stopped with reset (cold) to avoid error messages. • It is highly recommended that you use the TecEditor project "MOVI-PLCDefaultprojekt.TecFunction" as a basis. Now you can modify or extend this project accordingly. It is essential not to deactivate the technology functions that are activated in the basic project. • Download modified TecEditor project. • Once the variable *UseTecEditorConfig* = True has been set, the program can be started.

The *TecFunctionsUserUnitSlave* variable effects that the technology values and data that refer to the slave are scaled in user-defined units. The required scaling factors are the numerator/denominator values with accordingly adjusted decimal places.

	TIPS
	Note: Values relating to the master are not scaled.

"MC_PrepareGearInDistanceBased_MX", for example:

DistanceForStartGearIn: Master distance after which startup cycle is to begin, as a consequence, the values are not scaled.

GearInMasterDistance: Master distance for the startup cycle, as a consequence, values are not scaled.

GearInSlaveDistance: Startup cycle slave distance, scaled in user-defined units (if selected with *TecFunctionsUserUnitSlave* = True).

Input structure "sInitialParameter":

With this structure, additional, freely selectable parameters can be set on the axis during initialization of the"MC_LinkTecGear_MX" (for DH.41B: max. 60 parameters).

Each parameter that is to be written must be correctly defined with index/subindex/data.

	TIPS
	Note: It is not checked whether the data is permitted for the respective parameter, or whether a parameter, which has already been configured for a certain function, is overwritten. You have to make sure that there are no overlaps, otherwise the Gear function will be affected.

- ExtConfig of MC_SetGearConfig_MX

```
(*################################ Extended SetGearConfig configurations ############################*)

    MoveToMaster: MC_GEAR_MOVE_TO_MASTER_MX;
                                    (*#################################################################
                                    With this struct it can be activated and adjusted that the slave moves to the master
                                    after State 16 was interrupted by Inhibit or Safety Stop:
                                    MoveToMaster.Activate_Function : True = activates the move to master function
                                                                     False = move to master function deactivated, if slave
                                                                             or master has moved more than size of
                                                                             lag window while disabled, drive trips with
                                                                             lag error when state changes again to 16.
                                    MoveToMaster.Slave_Speed : Speed setpoint for moving to the master
                                    MoveToMaster.Slave_Acceleration : Acceleration setpoint  for moving to the master
                                    MoveToMaster.Slave_Deceleration : Deceleration setpoint  for moving to the master *)

    MSignal_InterpolationTime: DINT;        (* InterpolationTime: range 1...60 resolution [0.5ms] (0.5....30ms)*)
    MSignal_DelayTime: DINT;                (* PositionDelayTime in [µs] *)
    AverageFilterTime: DINT;                (* AverageFilterTime: range 1...60 resolution [0.5ms] (0.5....30ms)*)
    CompFilterTime: DINT;                   (* CompFilterTime: range 1...60 resolution [0.5ms] (0.5....30ms)*)
    DeathTimeCorr_ON: BOOL;                 (* Activation of the death time correction: false = OFF, true = ON*)
    SpeedCorr_ON: BOOL;                     (* Activation of the velocity related correction: false = OFF, true = ON*)
    AccelerationCorr_ON: BOOL;              (* Activation of the acceleration related correction: false = OFF, true = ON*)

    ReactionLagError: MC_GEAR_REACTION_LAGERROR_MX ;
                                    (* defines the reaction after lag detection. Possible selection:
                                        NO_RESPONSE
                                        DISPLAY_FAULT
                                        IMMEDIATE_STOP_WARNING (default)
                                        EMERGENCY_STOP_WARNING
                                        APPL_STOP_WARNING
                                        SYST_STOP_WARNING *)
END_STRUCT
END_TYPE
```

63330axx

Activating and setting filters and compensation functions:

The extended configuration of the "MC_SetGearConfig_MX" function module includes variables that are used to set additional filters for conditioning the master signal.

In the default setting, the filters are deactivated. When using the virtual encoder of MOVI-PLC®, the *MSignal_InterpolationTime* value is set to 5 ms.

In addition, compensation means for velocity and acceleration compensation can be activated. For detailed information on the function and effects of these filters and compensations measures, refer to the MOVIAXIS® documentation "MOVIAXIS® MX Multi-Axis Servo Inverter, Electronic Gear Unit" manual.

Setting the ReactionLagError:

This defines the response of the drive when the *LagErrorWindow* is exceeded. If nothing different is specified here, a stop at the application limit (waiting) is performed.

Additional function MoveToMaster:

Activating the controller inhibit or setting "Safe stop" during state "16" (i. e. *GearState* = InGear) and moving away the master or slave in this state usually triggers a lag error as soon as the state goes back to "16". If this is unwanted, positioning of the slave to the master position can be configured by activating the "MoveToMaster" function. Once the controller inhibit or safe stop is revoked, the slave travels to the master with the specified values for velocity and acceleration/deceleration.

- **Extended Diagnostics**

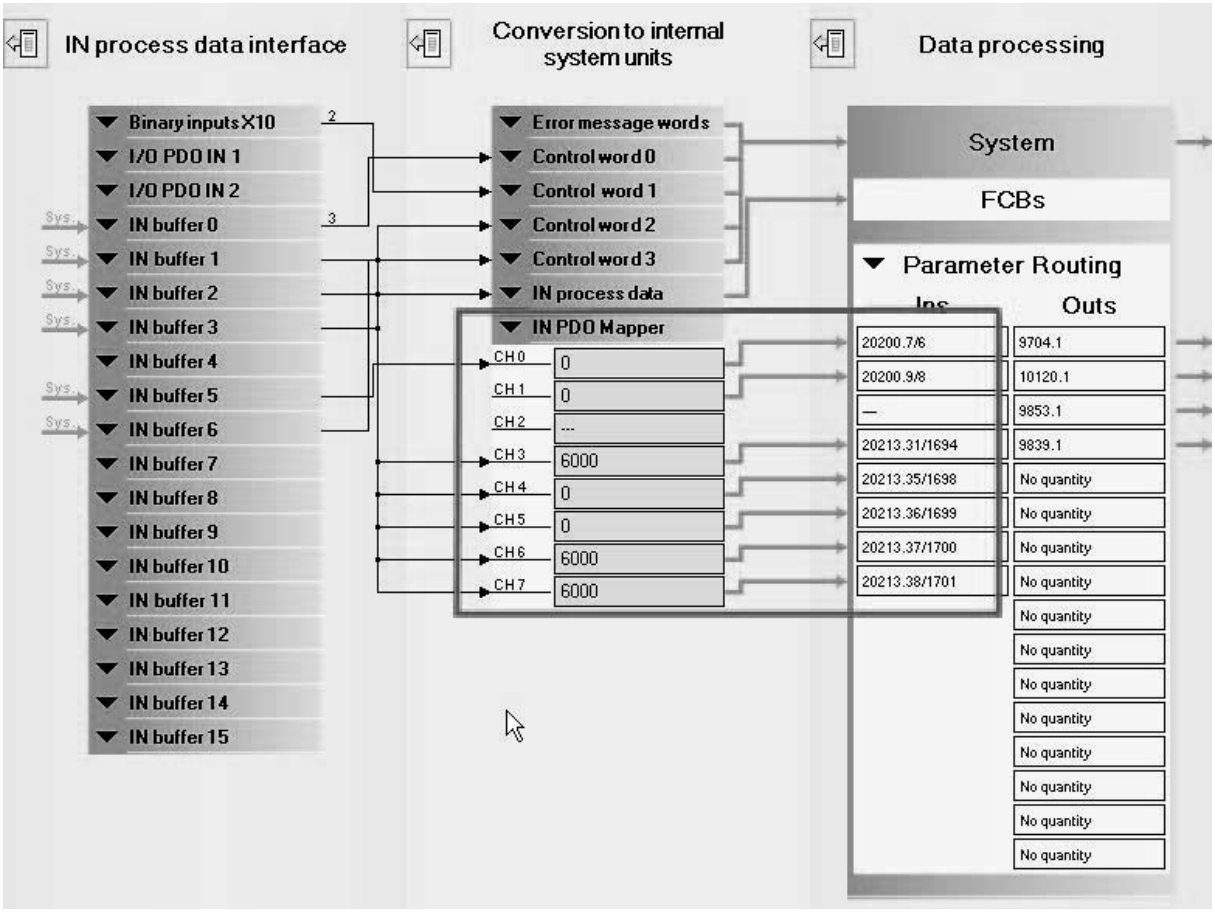
  The output structure "MC_GEAR_EXTENDED_DIAG_MX" is designed as additional diagnostic information of "MC_LinkTecGear_MX".

  This is still in preparation and not yet implemented.

2.4.3 PDO configuration

In addition to the configuration made by "MC_ConnectAxis_MX", the "MC_LinkTecGear_MX" module also performs some settings at the process data interface.

Even if an adaptation of the PDO configuration is not necessary for standard applications, the following describes which Gear-specific settings are made by "MC_LinkTecGear_MX".



63331axx

The IN PDO mapper is configured for transferring parameters for the time-related offset (realized with the profile generator of the cam).

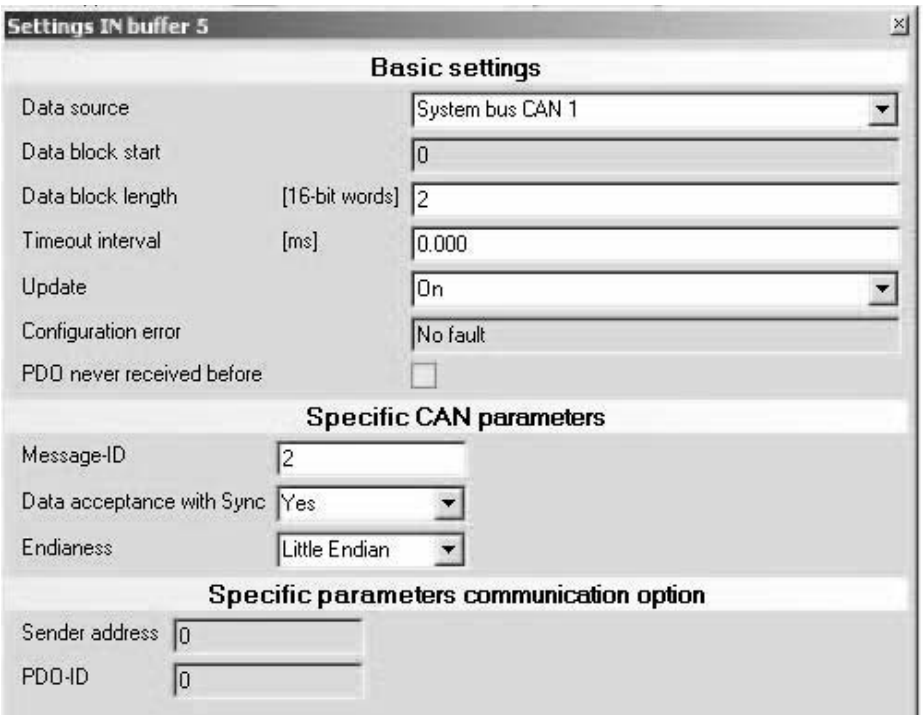
In addition, channel 0 establishes a connection from IN buffer 5 to the respective technology variable (DDB) for receiving the master position via CAN bus. When selecting "MX_GEAR_RECEIVE_PDO_1" as master source in the "MC_SetGearConfig_MX" module, this object is set as master value for the Gear function.

Channel 1 is configured in such a way that a technology variable is set as target which represents the master source as "MX_GEAR_RECEIVE_PDO_2" of "MC_SetGearConfig_MX". The user can freely select and configure the source for channel 1 within IN buffer 11-15.

The channels of the IN PDO mapper are assigned as follows:

- Channel 0: Master position of IN buffer 5
- Channel 1: Master position, configurable, IN buffer 11-15
- Channel 2: Reserved
- Channel 3: OffsetDistance
- Channel 4 / 5: OffsetSpeed
- Channel 6: OffsetAcceleration
- Channel 7: OffsetDeceleration

Settings in the IN buffer 5



63332axx

The IN buffer 5 is configured for receiving the master position via CAN bus. For this purpose, "MC_LinkTecGear_MX" sets the synchronized data transfer and the data format/data block length.

The data source and the message ID can be set via the "MC_InitialConfig_MX" function module (from the MPLCMotion_MX library).

Without *InitialConfig*, "System bus CAN1" and message ID 2 are selected as standard.

The "MC_InitialConfig_MX" module offers the following options, also via the "ExtendedConfiguration_MX" input structure:

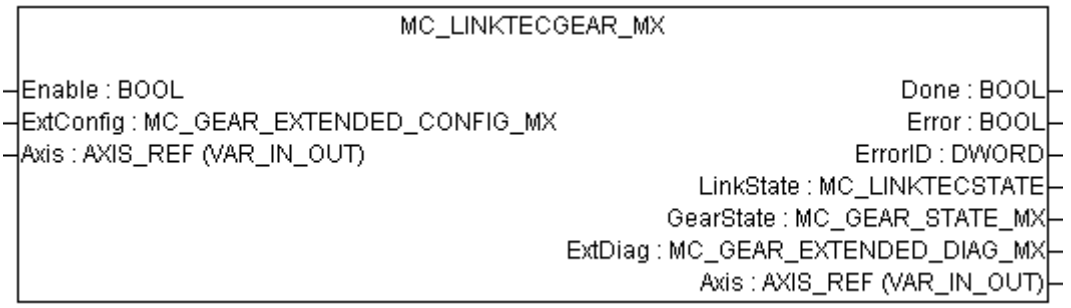
- The "SyncSource" structure variable can be used to set which synchronized CAN bus is to be used as data source for IN buffer 5. You have the following selection options:
  - MX_SYNCSOURCE_NONE
  - MX_SYNCSOURCE_CAN1
  - MX_SYNCSOURCE_CAN2
  - MX_SYNCSOURCE_COMOPTION
- The message ID is configured with the *ReceiveID* variable with the following selection options:
  - MX_VIRTUAL_ENCODER_ID1 (ID 2)
  - MX_VIRTUAL_ENCODER_ID2 (ID 3)
  - MX_PDO_ID1 (ID 4)
  - MX_PDO_ID2 (ID 5)
  - MX_PDO_ID3 (ID 6)
  - MX_PDO_ID4 (ID 7)
  - MX_PDO_ID5 (ID 8)
  - MX_PDO_ID6 (ID 9)
  - MX_PDO_ID7 (ID 10)
  - MX_PDO_ID8 (ID15)
  - MX_PDO_ID9 (ID 16)
  - MX_PDO_ID10 (ID 17)
  - MX_PDO_ID11 (ID 18)
  - MX_PDO_ID12 (ID 23)

Other Gear-specific settings

- IN buffer 6 is configured for receiving control word 3. The bits of this control word are set for controlling internal functions within the technology function.
- Out buffer 4 contains the status word 3.It provides status information of the respective technology functions which is required by the modules of the "MPLCTecGearMotion_MX" library.

2.5 The function modules of the MPLCTecGearMotion_MX library

2.5.1 MC_LinktecGear_MX



63333axx

Description:

The "MC_LinkTecGear_MX" function module is the logical interface between the axis and the Gear technology function. Once "ConnectAxis" has established the cyclical communication with the axis and after the unit has been enabled, the individual MOVIAXIS® technology functions are first configured. The ones necessary for Gear are activated. The PDO interface (IN-PDO 5 for receiving the master position via SBus, PDO mapping for offset values) is also configured and the communication parameters for SBus synchronization are set.

In cyclical operation, "MC_LinkTecGear_MX" ensures the changeover from Gear to a linear curve as soon as the axis is in sync (necessary to realize the offset travel option).

Additionally, various status signals are available as outputs.

Important:

The "MC_LinkTecGear_MX" module is absolutely essential for using the synchronous operation function. It must be called cyclically in the program.

If *LinkState* = NotLinked, none of the function modules of the "MPLCTecGearMotion_MX" can be executed, a corresponding error message is issued.

Prerequisite:

MOVI-PLC® advanced in technology version T1 or higher.

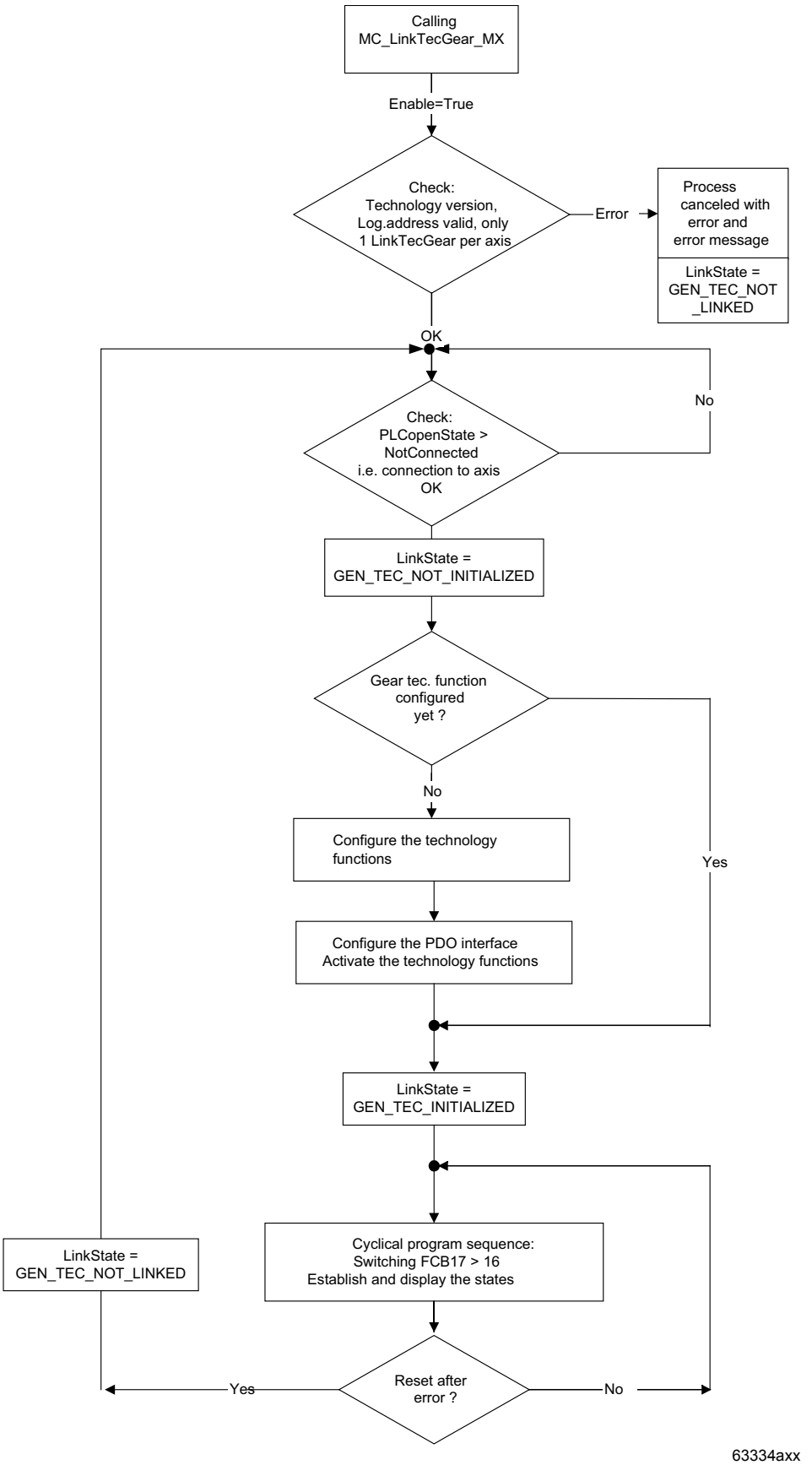
Inputs:

Name	Type	Meaning
Enable	BOOL	The module is processed as long as Enable „true“. Initialization is performed with a rising edge.
ExtConfig	MC_GEAR_EXTENDED_CONFIG_MX	Extended configuration, see page 22
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

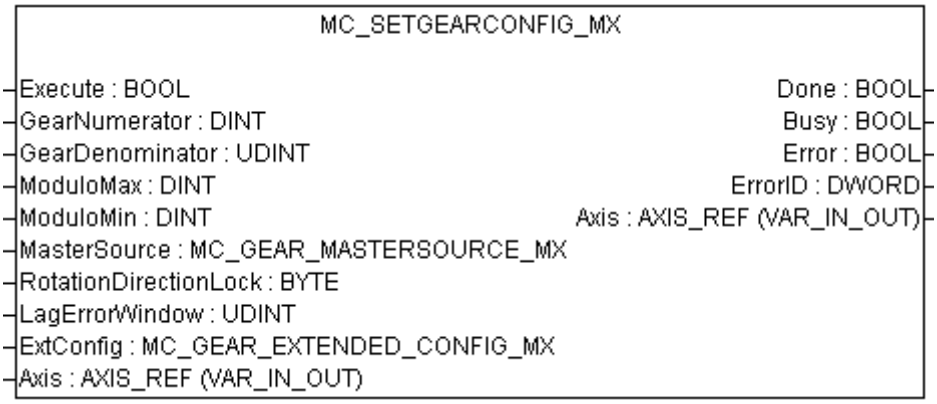
Outputs:

Name	Type	Meaning
Done	BOOL	Initialization successfully completed, connection with Gear technology function established
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54
LinkState	MC_LINKTECSTATE	Current status:GEN_TEC_NOTLINKED(no connection)GEN_TEC_NOTINITIALISED(ho initialization)GEN_TEC_INITIALISED(Gear technology function initialized)
GearState	MC_GEAR_STATE_MX	Current Gear status:MX_GEAR_OUTGEAR(disengaged)MX_GEAR_ENGAGING_GEAR_IN(engaged)MX_GEAR_IN_GEAR(synchronous operation)MX_GEAR_OFFSET_GEAR(offset operation) MX_GEAR_MOVE_TO_MASTER Slave moves to the master to EMERGENCY STOP or controller inhibit MX_GEAR_NOTLINKED(no valid Gear state, e.g. with LinkState = NotLinked)
ExtDiag	MC_GEAR_EXTENDED_DIAG_MX	Prepared for additional diagnostic information

MC_LinktecGear_MX: initialization



2.5.2 MC_SetGearConfig_MX



63335axx

Description:

This module sets basic parameters of the Gear function and transfers them to the respective address. The set values are adopted and sent with the rising edge on the Execute input.

The *MasterSource* input specifies the source of the master encoder signal. Here, you can choose between "real" encoders that are connected to the axis or "virtual" master signals sent via SBus (see page 20).

If the master signal is transferred in modulo format, the ModuloMax/ModuloMin inputs should be set to the respective limit values of the master signal. This avoids setpoint jumps when the modulo range overflows. The setting of these values is irrelevant for a linear master signal.

At the *ExtConfig* input, a structure can be transferred for setting various filters and compensation measures. This can be used, for example, to filter a "noisy" master signal. These filters are set to suitable default values, which means that this input is usually not connected.

Prerequisite:

"MC_LinkTecGear_MX" has set Gear technology function, *LinkState* may not be GEN_TEC_NOTLINKED.

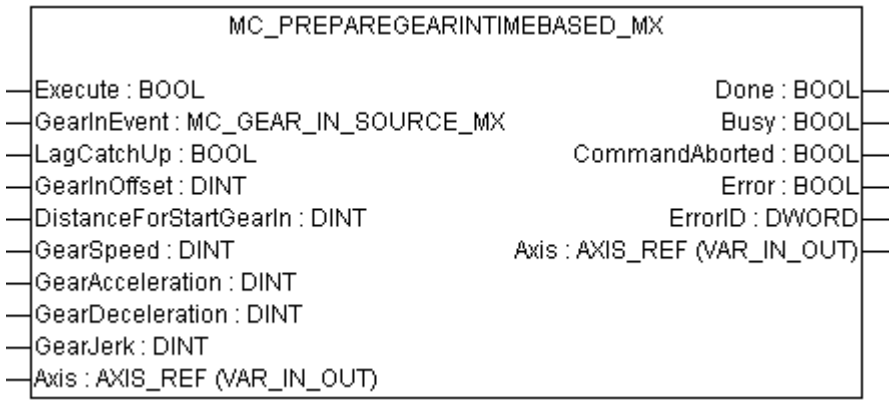
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module with a rising edge
GearNumerator	DINT	Numerator value for scaling the master signal
GearDenominator	UDINT	Denominator value for scaling the master signal
ModuloMax	DINT	Maximum modulo value of master signal
ModuloMin	DINT	Minimum modulo value of master signal
MasterSource	MC_GEAR_MASTERSOURCE_MX	Source of master position MX_GEAR_SYSTEMPOS_ENCODER1MX_GEAR_SYSTEMPOS_ENCODER2MX_GEAR_SYSTEMPOS_ENCODER3MX_GEAR_MODULOPOS_ENCODER1MX_GEAR_MODULOPOS_ENCODER2MX_GEAR_MODULOPOS_ENCODER3MX_GEAR_MODULO_VIRTUAL_ENCODER_MXMX_GEAR_LINEAR_VIRTUAL_ENCODER_MXMX_GEAR_RECEIVE_PDO_1MX_GEAR_RECEIVE_PDO_2
RotationDirectionLock	BYTE	Direction of rotation lock 0 = both directions enabled 1 = only CCW enabled 2 = only CW enabled
LagErrorWindow	UDINT	Lag error window in [inc]
ExtConfig	MC_GEAR_EXTENDED_CONFIG_MX	Extended configuration:Filter settings, compensation measures, lag error response, „MoveToMaster“ function, see page 22
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	All parameters have been transferred successfully
Busy	BOOL	Parameters are being transferred
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.3 MC_PrepareGearInTimeBased_MX



63336axx

Description:

This module sets the velocity, acceleration, deceleration and jerk for time-controlled startup cycles.

In addition, the *LagCatchUp* input can be used to select whether a lag error during the startup cycle is to be compensated or not. An additional offset travel sequence can be activated during the startup cycle with the *GearInOffset* value specifying the distance.

The startup cycle event is defined with *GearInEvent*. When "MX_GEAR_DISTANCE_COUNTER" is set, the value of the *DistanceForStartGearIn* input sets the master value once the startup cycle begins.

Important:

The *DistanceForStartGearIn* value of the master distance for the startup cycle is direction-sensitive, i.e. if the master turns CCW, this value must be a negative value. In case of extremely small values, the event is only safely detected when the master stands still. The master distance might change so fast that the sampling time is longer than the time in which the value is reached.

Important:

If the "LagCatchUp" function is activated and reference travel has been performed before the startup cycle, it might be necessary for the slave to catch up during the startup cycle even if previously there was no difference between the master and slave position.

Reason: Due to the reference travel, the slave has a new current position, which means the master/slave difference, which must be made up for, changes as well.

If this is not desired, the catching-up process can be deactivated before the next startup cycle by calling up "MC_PrepareGearInTimeBased_MX" again with *LagCatchUp* = False. The master/slave difference can also be deleted with the "MC_GearClearLag_MX" module.

Prerequisite:

"MC_LinkTecGear_MX" has set Gear technology function, *LinkState* may not be GEN_TEC_NOTLINKED.

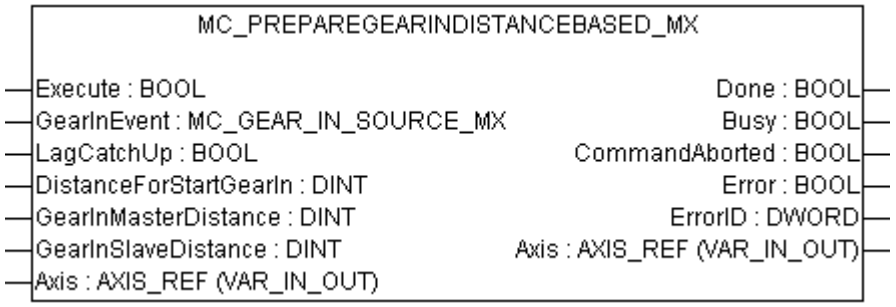
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
GearInEvent	MC_GEAR_IN_SOURCE_MX	Possible startup cycle events are:MX_GEAR_DIRECTMX_GEAR_INTERRUPT_DI01_POS_FLAGMX_GEAR_INTERRUPT_DI01_NEG_FLAGMX_GEAR_INTERRUPT_DI01 ...MX_GEAR_INTERRUPT_DI08_POS_FLAGMX_GEAR_INTERRUPT_DI08_NEG_FLAGMX_GEAR_INTERRUPT_DI08MX_GEAR_INTERRUPT_C_TRACK_ENC1MX_GEAR_INTERRUPT_C_TRACK_ENC2MX_GEAR_INTERRUPT_C_TRACK_ENC3MX_GEAR_DISTANCE_COUNTERMX_GEAR_DISTANCE_COUNTER_RESET_COUNT Explanation: DIRECT = direct startup cycle POS_FLAG= positive edge NEG_FLAG= negative edge only the input = both edges DI01 of MOVIAXISDI01 = input DI02 of MOVIAXISDI02 = input DI03 of MOVIAXISDI03 = input DI04 of MOVIAXISDI04 = input DI05 of MOVIAXISDI05 = input DI06 of MOVIAXISDI06 = input DI07 of MOVIAXISDI07 = input DI08 of MOVIAXISDI08 = input C_GEAR_ENC1 = C track encoder 1 C_GEAR_ENC2 = C track encoder 2 C_GEAR_ENC3 = C track encoder 3
LagCatchUp	BOOL	True: Lag distance is compensated False: Lag distance is not compensated (startup cycle relative to master position)
GearInOffset	DINT	Additional offset distance during the startup cycle
DistanceForStartGearIn	DINT	Only for mode Distance_Counter: Master distance after which the startup cycle is initiated
GearSpeed	DINT	Maximum startup cycle speed [rpm]
GearAcceleration	DINT	Startup cycle acceleration in [rpm/s]
GearDeceleration	DINT	Startup cycle deceleration in [rpm/s]
GearJerk	DINT	Startup cycle jerk in [rpm/s²]
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	All parameters have been transferred successfully
Busy	BOOL	Parameters are being transferred
Command Aborted	BOOL	Module was aborted by another instance of a PrepareGearIn function module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.4 MC_PrepareGearInDistanceBased_MX



63337axx

Description:

This module prepares the position-dependent startup cycle. The slave synchronizes with the master within the *GearInMasterDistance* distance, covering the distance *GearInSlaveDistance*.

In addition, the *LagCatchUp* input can be used to select whether a lag error during the startup cycle is to be compensated or not.

The startup cycle event is defined with *GearInEvent*. When "MX_GEAR_DISTANCE_COUNTER" is set, the value of the *DistanceForStartGearIn* input sets the master value once the startup cycle begins.

Important:

The value of the master distance *DistanceForStartGearIn* or *GearInMasterDistance* for the startup cycle (startup cycle master distance) is direction-sensitive, i. e. if the master turns CCW, the values must be negative values.

MX_GEAR_DISTANCE_COUNTER:

In case of extremely small values, the event is only safely detected when the master stands still. The master distance might change so fast that the sampling time is longer than the time in which *DistanceForStartGearIn* is reached.

The startup cycle profile is calculated with a fifth-degree polynomial.

This is why *GearInSlaveDistance* should be roughly half of the scaled master distance. Otherwise, the slave can overshoot or change the direction of rotation briefly during the startup cycle (see also page 11).

Important:

If the "LagCatchUp" function is activated and reference travel has been performed before the startup cycle, it might be necessary for the slave to catch up during the startup cycle even if previously there was no difference between the master and slave position.

Reason: Due to the reference travel, the slave has a new current position, which means the master/slave difference, which must be made up for, changes as well.

If this is not desired, the catching-up process can be deactivated before the next startup cycle by calling up "MC_PrepareGearInTimeBased_MX" again with *LagCatchUp* = False. The master/slave difference can also be deleted with the "MC_GearClearLag_MX" module.

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

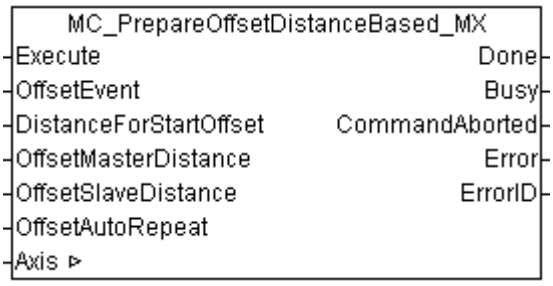
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
GearInEvent	MC_GEAR_IN_SOURCE_MX	Possible startup cycle events are: MX_GEAR_DIRECT MX_GEAR_INTERRUPT_DI01_POS_FLAG MX_GEAR_INTERRUPT_DI01_NEG_FLAG MX_GEAR_INTERRUPT_DI01 ... MX_GEAR_INTERRUPT_DI08_POS_FLAG MX_GEAR_INTERRUPT_DI08_NEG_FLAG MX_GEAR_INTERRUPT_DI08 MX_GEAR_INTERRUPT_C_TRACK_ENC1 MX_GEAR_INTERRUPT_C_TRACK_ENC2 MX_GEAR_INTERRUPT_C_TRACK_ENC3 MX_GEAR_DISTANCE_COUNTER MX_GEAR_DISTANCE_COUNTER_RESET_COUNT Explanation: DIRECT = direct startup cycle POS_FLAG= positive edge NEG_FLAG= negative edge only the input = both edges DI01 of MOVIAXISDI01 = input DI02 of MOVIAXISDI02 = input DI03 of MOVIAXISDI03 = input DI04 of MOVIAXISDI04 = input DI05 of MOVIAXISDI05 = input DI06 of MOVIAXISDI06 = input DI07 of MOVIAXISDI07 = input DI08 of MOVIAXISDI08 = input C_GEAR_ENC1 = C track encoder 1 C_GEAR_ENC2 = C track encoder 2 C_GEAR_ENC3 = C track encoder 3
LagCatchUp	BOOL	True: Lag distance is compensated False: Lag distance is not compensated (startup cycle relative to master position)
DistanceForStartGearIn	DINT	Only for mode Distance_Counter: Master distance after which the startup cycle is initiated
GearInMasterDistance	DINT	Master distance within which the startup cycle is initiated
GearInSlaveDistance	DINT	Slave distance during the startup cycle
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Startup cycle has been prepared successfully
Busy	BOOL	Parameters are being transferred
Command Aborted	BOOL	Module was aborted by another instance of a PrepareGearIn function module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.5 MC_PrepareOffsetDistanceBased_MX



63340axx

Description:

This module prepares a position-dependent offset travel. The *OffsetMasterDistance* specifies the master distance which is available for offset travel, *OffsetSlaveDistance* the additional slave distance.

The offset start event is defined with *OffsetEvent*. When "MX_GEAR_DISTANCE_COUNTER" is set, the value of the *DistanceForStartOffset* input sets the master value once the offset sequence begins. The *OffsetAutoRepeat* input selects automatic, repeated activation of an interrupt- or counter-controlled offset mode.

Important:

The value of the master distance *DistanceForStartOffset* or *OffsetMasterDistance* for the offset start (offset master distance) is direction-sensitive, i. e. if the master turns CCW, the values must be negative values.

MX_GEAR_DISTANCE_COUNTER:

In case of extremely small values, the event is only safely detected when the master stands still. The master distance might change so fast that the sampling time is longer than the time in which *DistanceForStartOffset* is reached.

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

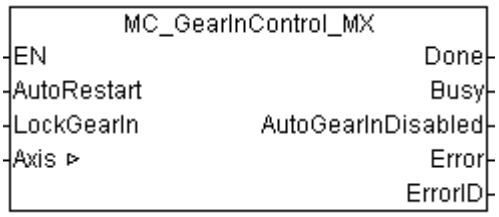
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
OffsetEvent	MC_GEAR_IN_SOURCE_MX	Possible offset start events are: MX_GEAR_DIRECT MX_GEAR_INTERRUPT_DI01_POS_FLAG MX_GEAR_INTERRUPT_DI01_NEG_FLAG MX_GEAR_INTERRUPT_DI01 ... MX_GEAR_INTERRUPT_DI08_POS_FLAG MX_GEAR_INTERRUPT_DI08_NEG_FLAG MX_GEAR_INTERRUPT_DI08 MX_GEAR_INTERRUPT_C_TRACK_ENC1 MX_GEAR_INTERRUPT_C_TRACK_ENC2 MX_GEAR_INTERRUPT_C_TRACK_ENC3 MX_GEAR_DISTANCE_COUNTER Explanation: DIRECT = direct startup cycle POS_FLAG= positive edge NEG_FLAG= negative edge only the input = both edges DI01 of MOVIAXISDI01 = input DI02 of MOVIAXISDI02 = input DI03 of MOVIAXISDI03 = input DI04 of MOVIAXISDI04 = input DI05 of MOVIAXISDI05 = input DI06 of MOVIAXISDI06 = input DI07 of MOVIAXISDI07 = input DI08 of MOVIAXISDI08 = input C_GEAR_ENC1 = C track encoder 1 C_GEAR_ENC2 = C track encoder 2 C_GEAR_ENC3 = C track encoder 3
DistanceForStartOffset	DINT	Only for mode Distance_Counter: Master distance after which the offset is initiated
OffsetMasterDistance	DINT	Master distance within which the offset is initiated.
OffsetSlaveDistance	DINT	Additional slave distance during offset travel
OffsetAutoRepeat	BOOL	Repeat function of the interrupt-/counter-controlled offset modes
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Startup cycle has been prepared successfully
Busy	BOOL	Parameters are being transferred
Command Aborted	BOOL	Module was aborted by another instance of a PrepareOffset function module or by OffsetDirectDistance
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.6 MC_GearInControl_MX



63341axx

Description:

The "MC_GearInControl_MX" module offers additional control functions for the interrupt- or counter-controlled startup cycle events. The *AutoRestart* input enables repeated startup cycles without repeated preparation using the "Prepare-GearIn" function modules. Interrupt-/counter-controlled startup cycles can be inhibited by setting the *LockGearIn* input to 'True'. Startup cycles are also inhibited after an axis fault (Error stop). In this case, interrupt-/counter-controlled startup cycles must be activated again with the "PrepareGearIn" modules. The *AutoGearInDisabled* output displays whether startup cycles are disabled.

Important:

Only interrupt-/counter-controlled startup cycles can be inhibited, startup cycles using "MC_GearInDirect_MX" are still possible.

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

Procedure:

To enable activation of the "AutoRestart" function, the respective "PrepareGearIn" function module must be used once to prepare the startup cycle event. If *AutoRestart* is deactivated, interrupt-/counter-controlled startup cycles are inhibited until they are prepared again with the "PrepareGearIn"function module.

With *LockGearIn* = True, interrupt-/counter-controlled startup cycles are inhibited. They can be enabled again by revoking the inhibit (*LockGearIn* = False) without calling the "PrepareGearIn" modules again.

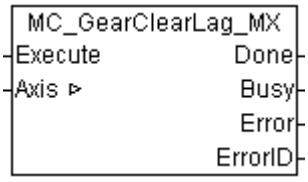
Inputs:

Name	Type	Meaning
EN	BOOL	The module is processed as long as this input = True.
AutoRestart	BOOL	True: Automatic repeated startup cycles activated False : Startup cycles deactivated
LockGearIn	BOOL	True: Startup cycles inhibited False: Startup cycles enabled
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Required function set
Busy	BOOL	Parameters are being transferred
AutoGearInDisabled	BOOL	Startup cycle disabled
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.7 MC_GearClearLag_MX



63342axx

Description:

With this function module, a rising edge at the execute input equates the Gear setpoint (conditioned master position) to the Gear actual value (current slave position), deleting the master-slave difference (once).

This can be necessary if the function "Catch up lag error" (*LagCatchUp* input at "PrepareGearIn" FB) is selected for the startup cycle, but the lag error should not be caught up for whatever reason (e.g. after reference travel has been performed).

If lag error catch up is deactivated (*LagCatchUp* = False at "PrepareGearIn" FB), the startup cycle is performed relative to the current position, i. e. "GearClearLag" is not necessary.

Important:

To execute this, the module requires 2 program cycles. We therefore recommend that you only activate this function during standstill of the master/slave.

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

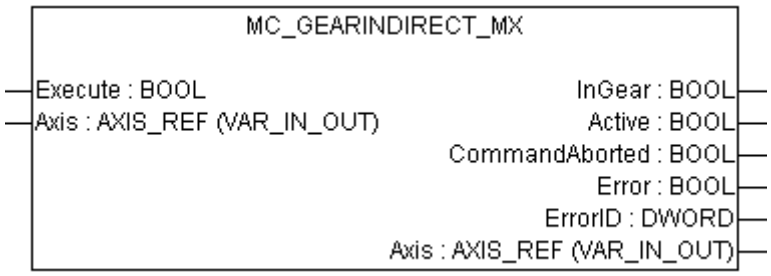
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Masterslave difference deleted
Busy	BOOL	Parameters are being transferred
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.8 MC_GearInDirect_MX



63343axx

Description:

A rising edge on the Execute input causes a direct startup cycle with the parameters set in *PrepareGearIn*.

The module interrupts "Move" and also "PrepareGearIn" modules that are still active.

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

GearState = MX_GEAR_OUTGEAR

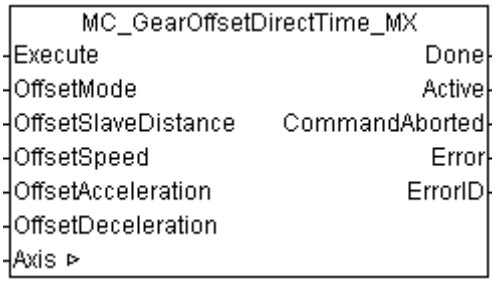
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the startup cycle
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
InGear	BOOL	Startup cycle successful, axis is in sync with the master
Active	BOOL	Axis is in the startup cycle state
Command Aborted	BOOL	Function module was interrupted by calling up another Move module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.9 MC_GearOffsetDirectTime_MX



63345axx

Description:

This module starts a time-related offset travel with the set values. The axis changes to synchronous operation again after successful offset.

Setting the *OffsetMode* determines whether *OffsetSlaveDistance* is to be used as relative or absolute value in relation to the master. In case of absolute travel, the reference point is set to zero once; all other offset values refer to this standard value (as long as MOVIAXIS® state "16" is displayed).

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

The axis is in synchronous operation (*GearState* = In_Gear).

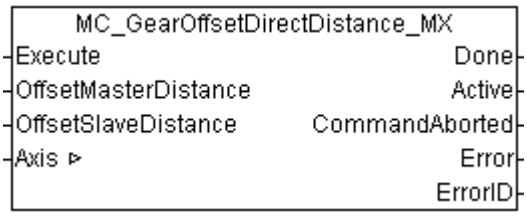
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
OffsetMode	MC_GEAR_OFFSETMODE_MX	Offset mode, set-ting:MX_GEAR_OFFSET_RELATIVEMX_GEAR_OFFSET_ABSOLUTE
OffsetDistance	DINT	Offset distance that the slave is to travel
OffsetSpeed	DINT	Maximum speed for offset travel in [rpm]
OffsetAcceleration	DINT	Acceleration for offset travel in [rpm/s]
OffsetDeceleration	DINT	Deceleration for offset travel in [rpm/s]
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Offset was processed, axis is synchronous again
Active	BOOL	Offset processing ongoing.
Command Aborted	BOOL	Function module was interrupted by calling up another Move module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.5.10 MC_GearOffsetDirectDistance_MX



63346axx

Description:

This module starts a position-dependent offset travel with the set values for *OffsetMasterDistance* and *OffsetSlaveDistance*. The axis changes to synchronous operation again after successful offset.

Prerequisite:

"MC_LinkTecGear_MX" has set the Gear technology function, *LinkState* may not be "GEN_TEC_NOTLINKED".

The axis is in synchronous operation (*GearState* = In_Gear).

Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
OffsetMasterDistance	DINT	Master distance within which the offset is initiated.
OffsetSlaveDistance	DINT	Additional slave distance during offset travel
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Offset was processed, axis is synchronous again
Active	BOOL	Offset processing ongoing.
Command Aborted	BOOL	Function module was interrupted by calling up another Move module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 54

2.6 Examples

2.6.1 1. Master is virtual encoder, direct, time-dependent startup cycle

As an example, we will use the "AxisControl_MX_Technology" sample project for demonstrating simple synchronous operation with direct, time-dependent startup cycles without lag error catch up.

The virtual encoder of the "MPLCTecVirtualEncoder" library will be used as master.

Comment:

Only those parts of the program will be explained here that are relevant for the Gear function. The "AxisControl...." sample projects are described in detail in the relevant documentation for these examples.

Solution:

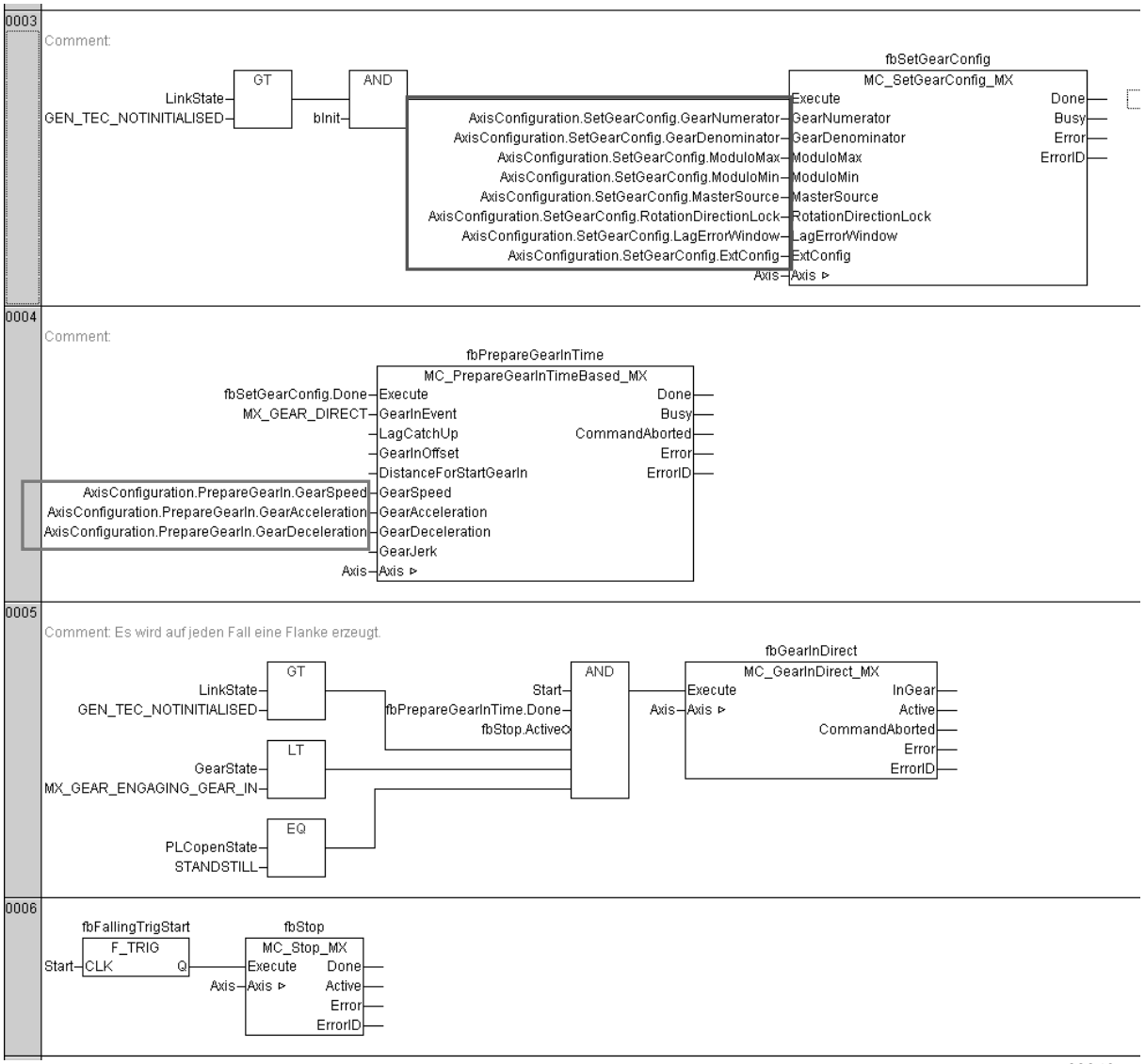
1. In the initialization routine, the input structures are configured for setting the Gear *parameters for "MC_SetGearConfig_MX" and the startup cycle values for "MC_PrepareGearInTimeBased_MX":

```
IF NOT bInit THEN
    (*##############################################
        Initialisation of AxisControlVirtual
    ##############################################*)

    (* Configuration virtual encoder *)
    gVirtualEncoderConfiguration.StartRisingEdge := FALSE;

    gVirtualEncoderConfiguration.LinkTecAxisVirtual.CanNode := SBUS_NODE_1;   (* SBUS_NODE_1
                                                                                 SBUS_NODE_2 *)

    gVirtualEncoderConfiguration.LinkTecAxisVirtual.SendObjectID := MDX_VIRTUAL_ENCODER_ID1;
                                                                  (* name of the message  which is used to transmit
                                                                      MDX_VIRTUAL_ENCODER_ID1
                                                                      MDX_VIRTUAL_ENCODER_ID2 *)
    (*##############################################
        Initialisation of AxisControl axis 3 (gear motion)
    ##############################################*)

    (* general axis control parameters *)
    gAxisConfiguration_MX[3].StartRisingEdge := FALSE;

    (* configuration modulo parameter --> only necessary if modulo mode is used *)
    gAxisConfiguration_MX[3].SetModuloParameters.Overflow          := 1;
    gAxisConfiguration_MX[3].SetModuloParameters.Underflow         := 1;


    (* Configuration gear parameters *)
    gAxisConfiguration_MX[3].SetGearConfig.GearNumerator      := 1;   (* Numerator for scaling of the master increments *)
    gAxisConfiguration_MX[3].SetGearConfig.GearDenominator    := 1;   (* Denominator for scaling of the master increments *)
    gAxisConfiguration_MX[3].SetGearConfig.ModuloMax          := 0;   (* Modulo maximum Value of the master position*)
    gAxisConfiguration_MX[3].SetGearConfig.ModuloMin          := 0;   (* Modulo minimum Value of the master position*)
    gAxisConfiguration_MX[3].SetGearConfig.MasterSource       := MX_GEAR_RECEIVE_PDO_1;
    gAxisConfiguration_MX[3].SetGearConfig.LagErrorWindow := 10000;     (* lag error window in [inc] *)
    gAxisConfiguration_MX[3].SetGearConfig.RotationDirectionLock :=0;   (* 0 = both directions, 1 = only CW enabled, 2 = only CCW enabled *)

    gAxisConfiguration_MX[3].PrepareGearIn.GearSpeed := 1000;           (* gear in speed in [rpm] *)
    gAxisConfiguration_MX[3].PrepareGearIn.GearAcceleration := 10000;   (* gear in acceleration in [rpm/s] *)
    gAxisConfiguration_MX[3].PrepareGearIn.GearDeceleration := 10000;   (* gear in deceleration in [rpm/s] *)

    gAxisConfiguration_MX[3].SetGearConfig.ExtConfig.MSignal_InterpolationTime := 10;   (* Interpolation time: range 1...60 means 0.5....30ms filter time*)
    gAxisConfiguration_MX[3].SetGearConfig.ExtConfig.AverageFilterTime := 10;  (* AverageFilter: range 1...60 means 0.5....30ms filter time*)

    bInit:=TRUE;

END_IF;
```

63347axx

2. When calling up the "Gearing" axis mode, the basic Gear parameters are set first, then the values relevant for the startup cycle. "MC_GearInDirect_MX" is used for the startup cycle when everything has been prepared and when *Start* = True.

For ending synchronous operation, a falling edge at Start performs a stop at the application limits, the drive status changes to "26". Once the drive has stopped, "PLCopenState" goes to "Standstill".



63349axx

2.6.2 2. MOVIAXIS® is the master, startup cycle with interrupt DI02

A MOVIAXIS® unit is to be configured for a Gear slave MOVIAXIS®. The startup cycle is to be performed with a rising edge at input DI02.

To take load of the CAN1 system bus which handles general communication, the position of encoder 1 of the master axis is sent via the synchronized CAN2 bus.

Here, too, the "AxisControl_MX_Technology" sample project is the basis. It is modified according to the requirements.

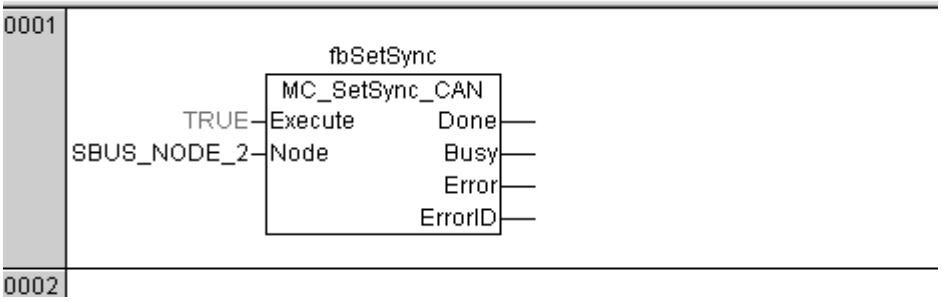
Comment:

Only those parts of the program will be explained here that are relevant for the Gear function. The "AxisControl...." sample projects are described in detail in the relevant documentation for these examples.

Solution:

Since the master position is to be sent via CAN2, the CAN2 of MOVI-PLC® (X32) is connected to the CAN2 of the MOVIAXIS® axes (X12) and a cyclical synchronization telegram is configured using "MC_SetSync_CAN".

```
0013 PROGRAM PLC_PRG_Technology
0014 VAR
0015     fbAxisControlA1: AxisControl_MX;          (* Instance of axis control *)
0016     fbAxisControlA2: AxisControl_MX;          (* Instance of axis control *)
0017     fbAxisControlA3: AxisControlGear_MX;      (* Instance of axis control *)
0018     fbAxisConfiguration_MX: Configuration_MX_Technology;
0019
0020
0021     fbSetSync: MC_SetSync_CAN;
0022 END_VAR
```



63350axx

The necessary settings are made in the initialization part, which are then sent from the relevant modules to the drives in the cyclical part of the program.

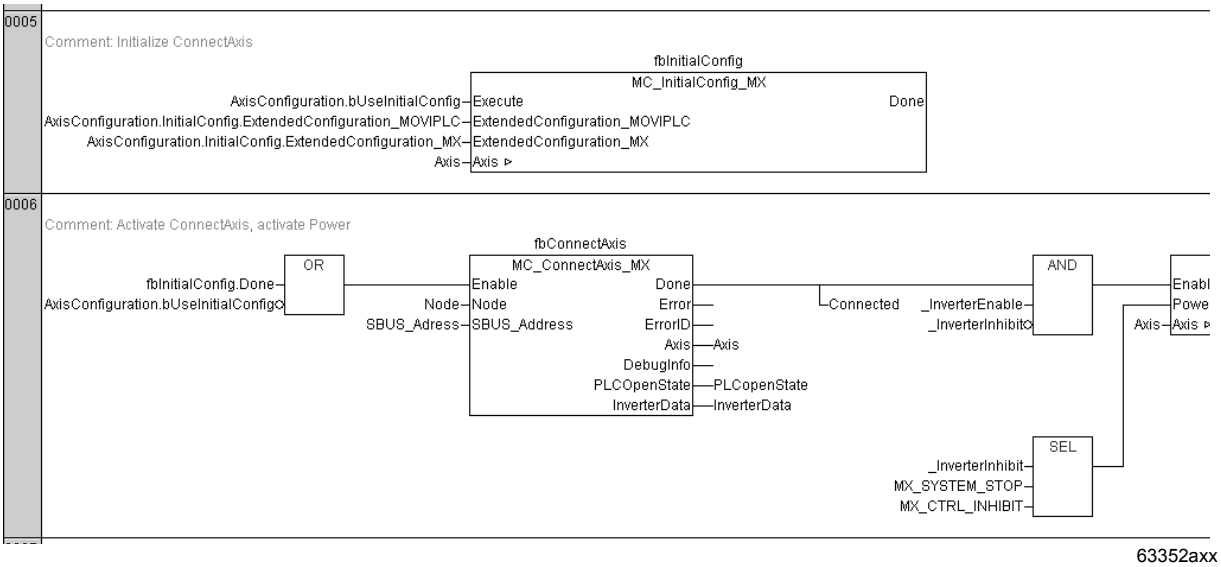
To create a cyclical send object with the required position for the master axis and to configure the respective identifier as receive object for the slave, the process data configuration of both drives is adjusted with the "MC_InitialConfig_MX" module.

```
IF NOT bInit THEN
    (*##########################################
        Initialisation of AxisControlVirtual
    ##########################################*)

    (* Configuration virtual encoder *)
    gVirtualEncoderConfiguration.StartRisingEdge := FALSE;

    gVirtualEncoderConfiguration.LinkTecAxisVirtual.CanNode := SBUS_NODE_1;   (* SBUS_NODE_1
                                                                                 SBUS_NODE_2 *)

    gVirtualEncoderConfiguration.LinkTecAxisVirtual.SendObjectID := MDX_VIRTUAL_ENCODER_ID1;
                                                                    (* name of the message  which is used to transmit
                                                                       MDX_VIRTUAL_ENCODER_ID1
                                                                       MDX_VIRTUAL_ENCODER_ID2 *)


    (*##########################################
        Initialisation of AxisControl axis 1 (single axis)
    ##########################################*)

    (* general axis control parameters *)
    gAxisConfiguration_MX[1].StartRisingEdge := FALSE;

    (* configuration modulo parameter --> only necessary if modulo mode is used *)
    gAxisConfiguration_MX[1].SetModuloParameters.Overflow              := 1;
    gAxisConfiguration_MX[1].SetModuloParameters.Underflow             := 1;

    (* Extended Configuration by use of InitialConfig for send the encoder 1 position via CAN2 with ID  "MX_PDO_ID1" *)
    gAxisConfiguration_MX[1].bUseInitialConfig                                  := TRUE;
    gAxisConfiguration_MX[1].InitialConfig.ExtendedConfiguration_MX.SendSource  := MX_SEND_ENCODER_1_SYS;
    gAxisConfiguration_MX[1].InitialConfig.ExtendedConfiguration_MX.SendID      := MX_PDO_ID1;
    gAxisConfiguration_MX[1].InitialConfig.ExtendedConfiguration_MX.SyncSource  := MX_SYNCSOURCE_CAN2;

    (*##########################################
        Initialisation of AxisControl axis 3 (gear motion)
    ##########################################*)

    (* general axis control parameters *)
    gAxisConfiguration_MX[3].StartRisingEdge := FALSE;

    (* configuration modulo parameter --> only necessary if modulo mode is used *)
    gAxisConfiguration_MX[3].SetModuloParameters.Overflow              := 1;
    gAxisConfiguration_MX[3].SetModuloParameters.Underflow             := 1;

    (* Extended Configuration by use of InitialConfig for receive data with ID "MX_PDO_ID1" via CAN2 *)
    gAxisConfiguration_MX[3].bUseInitialConfig                      := TRUE;
    gAxisConfiguration_MX[3].InitialConfig.ExtendedConfiguration_MX.ReceiveID    := MX_PDO_ID1;
    gAxisConfiguration_MX[3].InitialConfig.ExtendedConfiguration_MX.SyncSource   := MX_SYNCSOURCE_CAN2;

    (* Configuration gear parameters *)
    gAxisConfiguration_MX[3].SetGearConfig.GearNumerator      := 1;   (* Numerator for scaling of the master increments *)
    gAxisConfiguration_MX[3].SetGearConfig.GearDenominator    := 1;   (* Denominator for scaling of the master increments *)
    gAxisConfiguration_MX[3].SetGearConfig.ModuloMax          := 0;   (* Modulo maximum Value of the master position*)
    gAxisConfiguration_MX[3].SetGearConfig.ModuloMin          := 0;   (* Modulo minimum Value of the master position*)
    gAxisConfiguration_MX[3].SetGearConfig.MasterSource       := MX_GEAR_RECEIVE_PDO_1;

    gAxisConfiguration_MX[3].SetGearConfig.LagErrorWindow := 10000;      (* lag error window in [inc] *)
    gAxisConfiguration_MX[3].SetGearConfig.RotationDirectionLock :=0;    (* 0 = both directions, 1 = only CW enabled, 2 = only CCW enabled *)
    gAxisConfiguration_MX[3].PrepareGearIn.GearSpeed := 1000;            (* gear in speed in [rpm] *)
    gAxisConfiguration_MX[3].PrepareGearIn.GearAcceleration := 10000;    (* gear in acceleration in [rpm/s] *)
    gAxisConfiguration_MX[3].PrepareGearIn.GearDeceleration := 10000;    (* gear in deceleration in [rpm/s] *)

    gAxisConfiguration_MX[3].SetGearConfig.ExtConfig.MSignal_InterpolationTime := 10;    (* Interpolation time: range 1...60 means 0.5....30ms filter time*)
    gAxisConfiguration_MX[3].SetGearConfig.ExtConfig.AverageFilterTime := 10;  (* AverageFilter: range 1...60 means 0.5....30ms filter time*)

    bInit:=TRUE;

END_IF;
```

63351axx

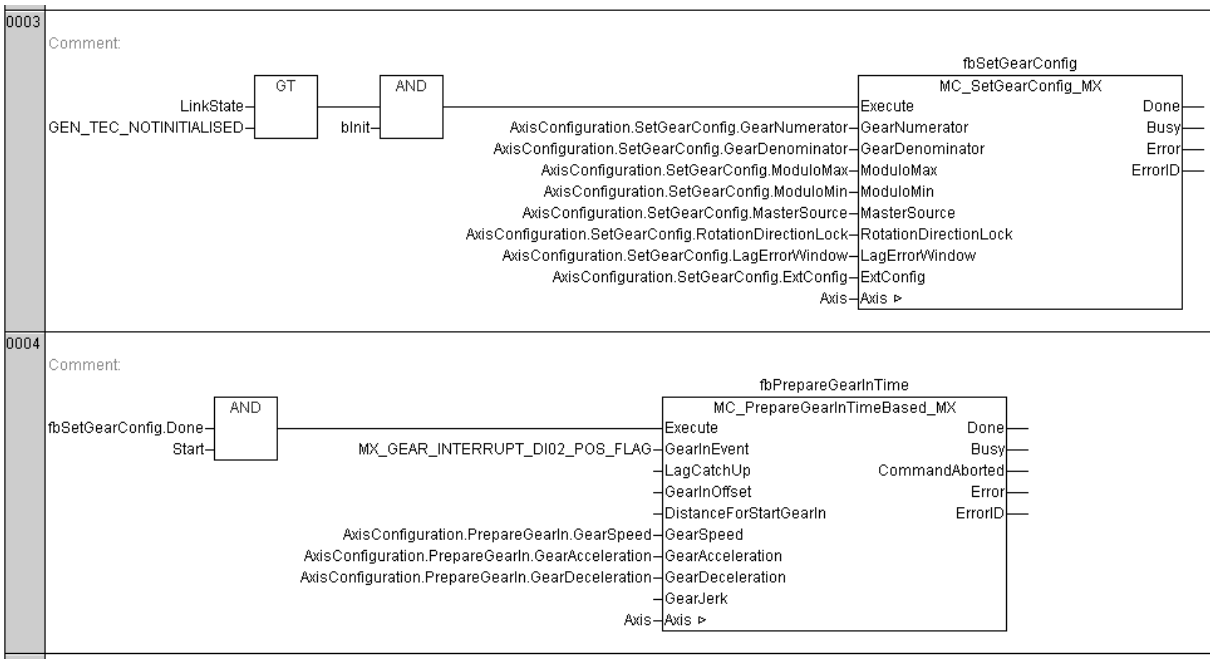
Initialization part for configuring the send/receive objects for master and slave.

InitialConfig must be called up before *ConnectAxis*. The process data configuration is then set during the boot sequence of *ConnectAxis*.



63352axx

In the "AxisModeGear" module, the *GearInEvent* of "MC_PrepareGearInTimeBased_MX" is set accordingly and sent when *Start* = True. The interrupt-controlled startup cycle is now prepared. When the event occurs, the slave performs a startup cycle. A falling edge at *Start* triggers a stop at the application limit.



63353axx

2.6.3 3. Position-dependent offset, activated after a defined master distance, automatic repetition

In this example, a position-dependent offset is added to example 2.

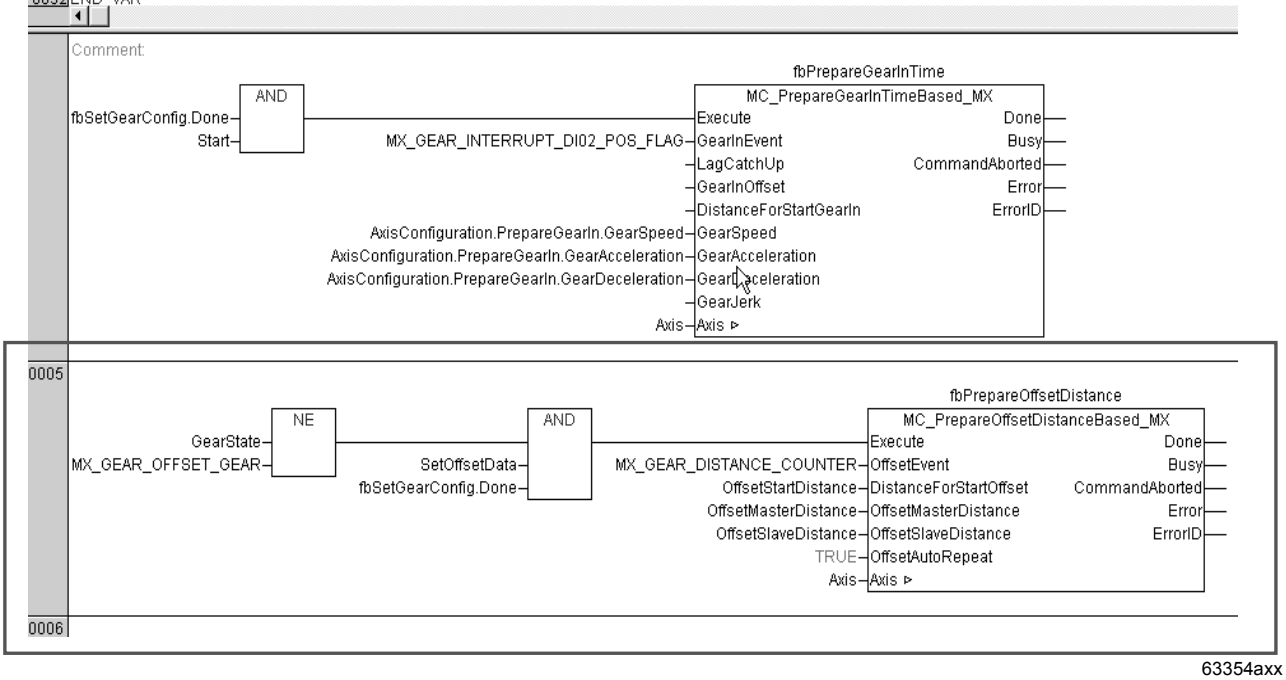
The master/slave distance for the offset and the master distance after offset travel has been started can be determined as an input at AxisControl. The values are transferred by setting the *SetOffsetData* input.

Solution:

First, the "AxisModeGear_MX" module is expanded accordingly:

PrepareOffsetDistance is integrated with the relevant configuration of the event. The setpoints for offset master/slave distance and *DistanceForStartOffset* as well as *SetOffsetData* are implemented as inputs of AxisModeGear. The "PrepareOffset" function module may not be performed during offset travel. It is therefore still inhibited.

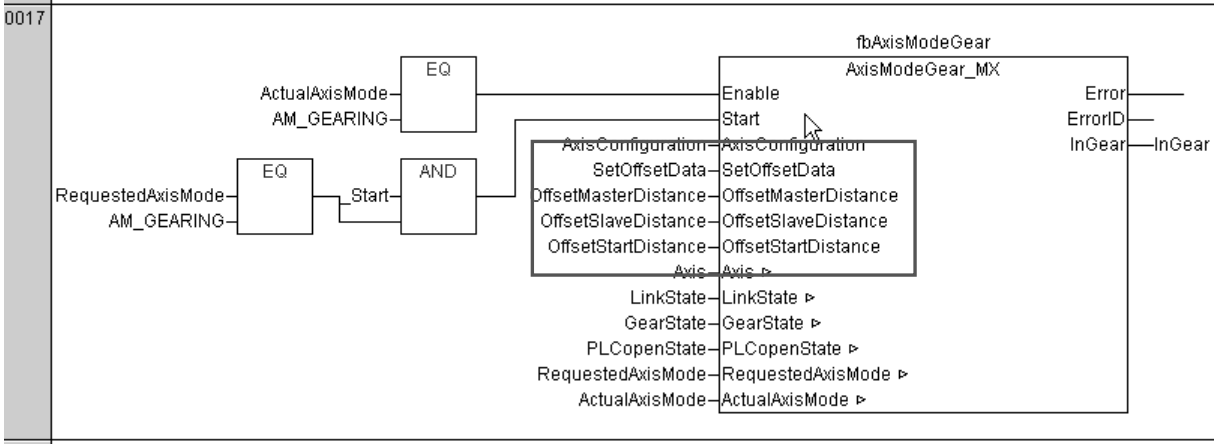
```
0013 FUNCTION_BLOCK AxisModeGear_MX
0014
0015 VAR
0016     fbSetGearConfig: MC_SetGearConfig_MX;
0017     fbGearInDirect: MC_GearInDirect_MX;
0018     bInit: BOOL;
0019     fbStop: MC_Stop_MX;
0020     fbFallingTrigStart: F_TRIG;
0021     fbPrepareGearInTime: MC_PrepareGearInTimeBased_MX;
0022     fbPrepareOffsetDistance: MC_PrepareOffsetDistanceBased_MX;
0023 END_VAR
0024 VAR_INPUT
0025     Enable: BOOL;
0026     Start: BOOL;
0027     AxisConfiguration: ST_AxisConfiguration_MX;
0028     SetOffsetData: BOOL;
0029     OffsetMasterDistance: DINT;
0030     OffsetSlaveDistance: DINT;
0031     OffsetStartDistance: DINT;
0032 END_VAR
```



63354axx

AxisModeGear is called up in *AxisControlGear*, which is why the latter must also be adapted accordingly. Here, too, the additional setting options for the offset are implemented as inputs.

```
0064
0065 END_VAR
0066 VAR_INPUT
0067     HControl: BOOL;
0068     Node: UINT;
0069     SBUS_Adress: UINT;
0070     InverterInhibit: BOOL;
0071     InverterEnable: BOOL;
0072     Reset: BOOL;
0073     AxisMode: ENUM_AXISMODE;
0074     Start: BOOL;
0075     JogPos: BOOL;
0076     JogNeg: BOOL;
0077     Position: DINT;
0078     Velocity: DINT;
0079     Acceleration: DINT;
0080     Deceleration: DINT;
0081     ModuloMode: MC_MODULOMODE_MX;
0082     AxisConfiguration : ST_AxisConfiguration_MX;
0083     SetOffsetData: BOOL;
0084     OffsetStartDistance: DINT;
0085     OffsetMasterDistance: DINT;
0086     OffsetSlaveDistance: DINT;
0087 END_VAR
0088 VAR_OUTPUT
```



63355axx

These setting options are also added to the "ST_AxisInterfaceInType_MX" structure so that the offset configuration can also be set via the input interface.

```
TYPE ST_AxisInterfaceInType_MX :
STRUCT
    HControl: BOOL;                 (* FALSE: fbAxisControl is controlled by the input variables TRUE: fbAxisCor
    InverterInhibit: BOOL;          (* TRUE: inverter is in state inverter inhibit *)
    InverterEnable: BOOL;           (* TRUE: inverter is enabled (if no inverter inhibit is set, no digital input is pr
    Reset: BOOL;                    (* rising edge resets and inverter fault *)
    AxisMode: ENUM_AXISMODE;        (* choose the requested axis mode
                                        AM_DEFAULT,                         (* 0 *)
                                        AM_VELOCITY,                        (* 1 *)
                                        AM_POSITIONING,                     (* 2 *)
                                        AM_POSITIONINGMODULO,               (* 3 *)
                                        AM_POSITIONINGRELATIVE,             (* 4 *)
                                        AM_JOG_5,                           (* 5 *)
                                        AM_JOG_A,                           (* 6 *)
                                        AM_HOMING,                          (* 7 *)
                                        AM_CAMING,                          (* 8 *)
                                        AM_GEARING,                         (* 9 *)
                                        AM_POSITIONINGMODULO_RELATIVE,      (* 10 *)
                                        AM_USER_SELECTION                   (* 11 *) *)

    Start: BOOL;                    (* starts a movement in the choosen axis mode (except jogging modes *)
    JogPos: BOOL;                   (* jog axis in positv direction *)
    JogNeg: BOOL;                   (* jog axis in negativ direction *)
    Position: DINT;                 (* target position for positioning axis modes (4096 incr. / motor rotation)*)
    Velocity: DINT;                 (* velocity for all movements (rpm) *)
    Acceleration: DINT;             (* ramp up time ( rpm/s ) for all movements *)
    Deceleration: DINT;             (* ramp down time ( rpm/s ) for all movements *)
    ModuloMode: MC_MODULOMODE_MX;   (* modulo mode for the axis mode modulo
                                        MX_SHORT,           Shortest way to target
                                        MX_CW,              Clockwise direction to target
                                        MX_CCW              Conterclockwise direction to target*)
    SetOffsetData: BOOL;            (* Sets the selected offset data *)
    OffsetStartDistance: DINT;      (* MasterDistance the Offset will be started *)
    OffsetMasterDistance: DINT;     (* Master distance within the offset will be executed *)
    OffsetSlaveDistance: DINT;      (* additional Slave distance during the offset move*)
END_STRUCT
END_TYPE
```

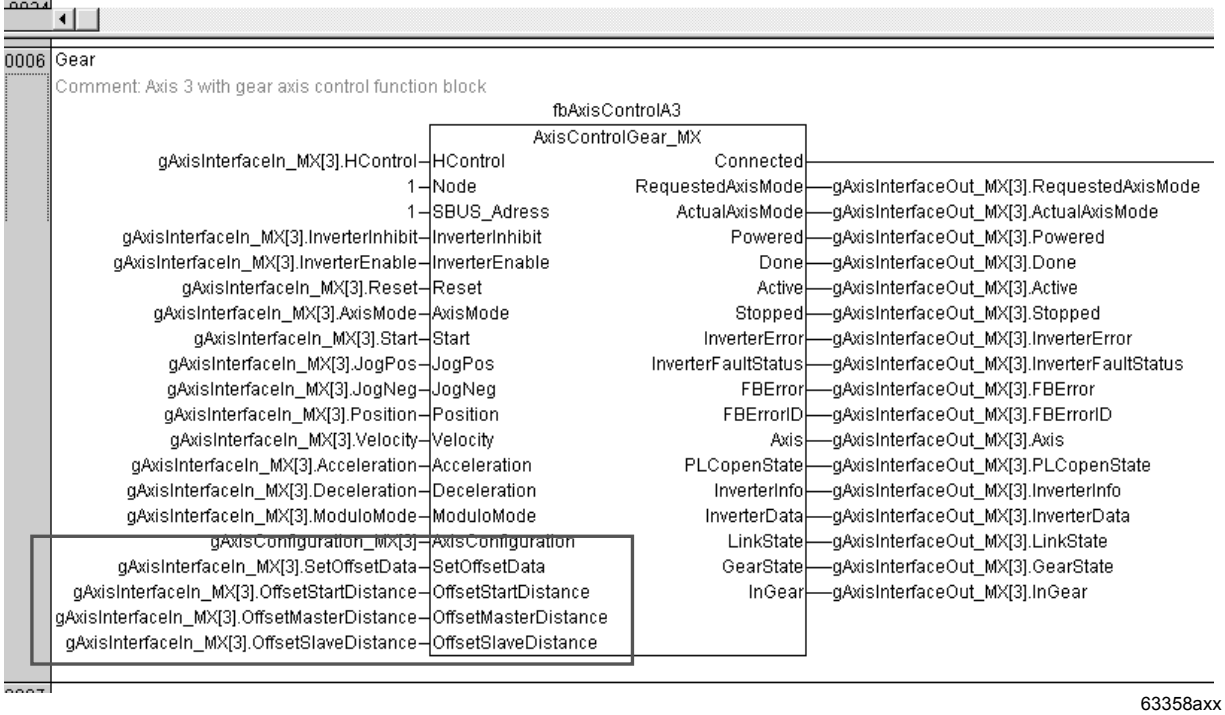
63556axx

Now, the additional inputs of "AxisControlGear" can be connected to the extended input structure in the main program.

Comment:

In this example, the additional inputs are not displayed via system variables. This is why for this function, there is no interface with the "ApplicationBuilder"user interface or a DOP operator panel.

```
0013 PROGRAM PLC_PRG_Technology
0014 VAR
0015     fbAxisControlA1: AxisControl_MX;          (* Instance of axis control *)
0016     fbAxisControlA2: AxisControl_MX;          (* Instance of axis control *)
0017     fbAxisControlA3: AxisControlGear_MX;      (* Instance of axis control *)
0018     fbAxisConfiguration_MX: Configuration_MX_Technology;
0019
0020
0021     fbSetSync: MC_SetSync_CAN;
0022 END_VAR
0023
```



63358axx

2.7 Appendix

2.7.1 Fault detection (ErrorID)

Fault code	Error designation	Problem
FA0010	E_IEC_GENERAL_INVALID_TECHNOLOGIE_OPTION	Requested technology function not supported
FA020...	General link tec error	
FA0200	E_TEC_GENERAL_MULTIPLE_TECLINKS	The LinkTec FB may not be called repeatedly.
FA0201	E_TEC_GENERAL_INVALID_LINKSTATE	Invalid LinkState for function call
FA0202	E_TEC_GENERAL_NOT_LINKED	LinkTec FB has no logical connection
FA0203	E_TEC_GENERAL_NOT_INITIALIZED	Technology function not yet active
FA0204[1)]	E_TEC_GENERAL_SERVICE_NOT_IMPLEMENTED	Requested service not supported
FA024…	Gear technology function error	
FA0240[1)]	E_TEC_GEAR_CLEAR_LAG_FAILED	Deleting the lag counter with MC_GearClearLag_MDX not successful
FA0241	E_TEC_GEAR_INVALID_GEAR_MODE	Function call in current Gear state not permitted
	IEC general error messages	
FA0070	E_IEC_PARAMETER_VALUE_OUT_OF_RANGE	Invalid entry value for parameter
FB0071	E_MDX_MOTIONBLOCK_LOG_ADR_NOT_INITIALIZED	MC_ConnectAxis_MDX has not yet assigned a log address
FB0072	E_MDX_MOTIONBLOCK_INVALID_LOG_ADR	Invalid log address
FB0073	E_MDX_MOTIONBLOCK_INVALID_STATE	Function call in current PLCopen state not permitted
FB0074[1)]	E_MDX_MOTIONBLOCK_INVALID_OPERATING_MODE	The requested technology function is not supported by the set MOVIDRIVE® operating mode

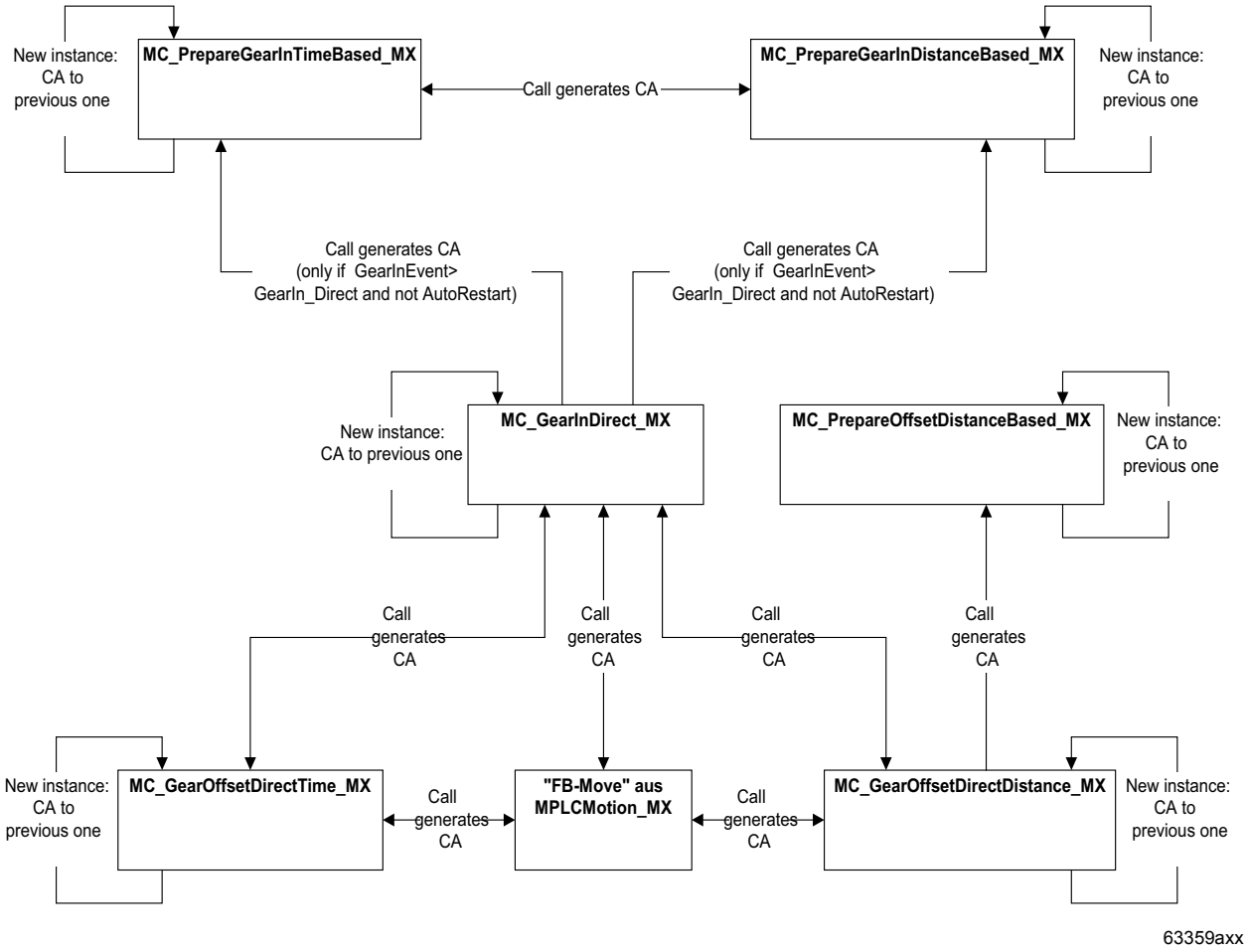
1) This error message is not relevant for MOVIAXIS® (only for MOVIDRIVE®)

2.7.2 Interrupting the modules (CommandAborted)

The "CommandAborted" output signal is used to signal repeated calls of a motion module or the call of a different motion module. The aborted command of the function module is no longer executed (see detailed information in the "MPLCMotion_MDX / MX library for MOVI-PLC®" manual).

Activated module	Generates CommandAborted at:
MC_GearInDirect_MX	- Instances of MC_GearInDirect_MX - MC_PrepareGearIn..... (if event<>direct and not AutoRestart) - “Move FB”
MC_GearOffsetDirectTime_MX	- Instances of MC_GearOffsetDirectTime_MX - MC_GearInDirect_MX- „Move FB“
MC_GearOffsetDirectDistance_MX	- Instances of MC_GearOffsetDirectDistance_MX - MC_PrepareOffsetDistanceBased_MX - MC_GearInDirect_MX- „Move FB“
MC_PrepareGearInTimeBased_MX	- Instances of MC_PrepareGearIn.....
MC_PrepareGearInDistanceBased_MX	- Instances of MC_PrepareGearIn.....
MC_PrepareOffsetDistanceBased_MX	- Instances of MC_PrepareOffsetDistanceBased_MX
“Move FB”	- MC_GearInDirect_MX - MC_GearOffsetDirectTime_MX - MC_GearOffsetDirectDistance_MX

Schematic representation of "CommandAborted":



Comment:

"Move FBs" of the MPLCMotion_MX library are:

- MC_MoveAbsolute_MX
- MC_MoveRelative_MX
- MC_MoveTargetPosition_MX
- MC_MoveTargetSpeed_MX
- etc.

3 MPLCTecCamMotion_MX

3.1 Introduction

The MOVIAXIS® multi-axis servo inverter offers comprehensive technology functions, for example:

- Electronic gear unit,
- Cams,
- Touch probe,
- Event control,
- Cam control.

The "Motion Technology Editor" startup assistant of MotionStudio enables configuration of all technology functions. Basic settings are configured this way. However, certain values or settings often have to be changed dynamically during a process, or various functions must be combined and run in sequence.

The function modules of the "MPLCTecCamMotion_MX" library set the general cam parameters and configure other technology function that are required for the respective cam function.

This means that in-depth familiarization with the individual parameters of the respective technology functions is not necessary. Configuration with the Technology Editor is not necessary, because the "MPLCTecCamMotion_MX" library performs all relevant settings.

Advantages of the "MPLCTecCamMotion_MX" library:

- Programming in accordance with IEC 61131, which means customer-specific applications can be implemented easily.
- Library for controlling the "Cam" technology function. This allows for simple configuration and control of the cam functionality without detailed knowledge of the individual technology function.
- Central control of several cam axes.
- Additional functions, such as specifying master values via a virtual master encoder, are controlled centrally using the MOVI-PLC®.
- Different startup cycle events and types can be selected.
- Additional offset travel can be activated in cam operation
- Simple curve change between several curves
- Up to 10 curves can be configured with transitions.
- Curve point-based calculation blocks for simple section-by-section movements.
- Curve profile can be set via scaling blocks.
- 10 curves can be edited with the Cam Editor. Curves can be down- and uploaded separately.

Additional notes are can be found in the following documentation:

- For general notes on the operating principle of the libraries, refer to the publication "MPLCMotion_MDX and MPLCMotion_MX libraries for MOVI-PLC®".
- "MOVIAXIS® MX Multi-Axis Servo Inverter" manual.

3.2 Areas of application

The "MPLCTecCamMotion_MX" library is suited for all applications that require synchronized axis movements.

Application examples

- Several axes synchronized with one virtual master.
- Numerous application options in packaging technology.
  - FFS machines, sealing and knife
  - Carton erector, stamping tool and carton transport synchronized with VE
  - Cartoner, strap inserter and carton transport synchronized
  - Banderoling machine, product adjustment
  - Synchronized turning, positioning, pushing
- Book trimming machine/book gluing machine.
- Flying saw, e.g. to cut continuous material.
- Synchronous material transport (e.g. several conveyor belts are synchronized to one master encoder).
- Multi-axis hoists/trolleys
- Filling station in the beverage industry.
- Any synchronized movements with a defined master-slave relation.

Characteristics

- Up to 64 (with MOVI-PLC® *advanced*) motor axes can be operated cam mode.
- Various configuration options can be set easily with the respective function modules.
- Various master encoder sources can be set.
- Different startup cycle types can be selected.
- Different offset types can be selected.
- Simple offset switch-in during the startup cycle or in synchronous operation.
- Various master/slave combinations can be implemented; several cam masters can be configured

3.3 Project planning

3.3.1 Prerequisites

- To use the MPLCTecCamMotion library, you need a MOVI-PLC® controller in technology version T1 or higher.
- The MPLCMotion_MX library must be integrated. Also adhere to the prerequisites in the "MPLCMotion_MDX and MPLCMotion_MX Libraries for MOVI-PLC®" manual.

3.3.2 Basic principle and notes

- Configuring the relevant technology functions of the axis using IEC function modules.
- The startup cycle and synchronization are performed with the "Cam" technology function. After that, the unit switches to linear cam function automatically. This enables offset travel during synchronous operation.
- Fast, interrupt-controlled startup cycle through use of the touch probe function and event control (is automatically configured correspondingly).
- The "MC_LinkTecCam_MX" function module is absolutely essential as a central data interface to the "Cam" technology function.
- The Technology Editor of MotionStudio is not necessary, but for diagnostic purposes, an online connection of the monitor can be established with the sample project "MOVI-PLC® Defaultprojekt.....Tec-Function". Do not download the sample project, otherwise settings made by the library would be overwritten.
- With the Cam Editor, you can define 10 curve sequences and download them to the MOVIAXIS®. When downloaded, the curve is stored with 512 control points. An upload will generate a table with 512 control points. For a better resolution, the control points are scaled up with the shift factor. The shift factor is determined automatically for all curves You can also determine the maximum shift factor. If a new curve results in a new shift factor, you cannot download the curve separately. You have to load all the curves.

3.3.3 Direct startup cycle

Use the "MC_CamInDirekt_MX" module to directly switch to cam mode. The selected curve is activated. With controller inhibit (emergency off), the axis is not aligned. If you want to align the axis after a controller inhibit you can activate the "Move-toMaster" function via the "MC_PrepareCamIn_MX" module.

You can use the "MC_CamTableSelect_MX" module to switch between individual curves with the switchover performed at the end of the curve.

3.3.4 Aligning and engaging the axis

You can use the "MC_CamInAdjust_MX" module to align the axis (FCB 9) and switch to cam mode. Use the *Adjustmode* input to determine as to how the axis is aligned. The selected curve is activated. After a controller inhibit (emergency off) the axis is aligned on the shortest way.

You can use the "MC_CamTableSelect_MX" module to switch between individual curves with the switchover performed at the end of the curve.

3.3.5 Startup cycle using interrupt

You can use the "MC_PrepareCamIn_MX" module to determine the "CamInEvent". Further you can activate the MovetoMaste function for the direct startup cycle.

Possible startup cycle events:

All MOVIAXIS® inputs as well as the C track signals of encoders 1 3 can be parameterized as interrupt event.

The startup cycle is started with an interrupt event triggered by a signal change at the MOVIAXIS® input DI02, with a rising/falling edge (pos/neg flag) or a signal change at DI02.

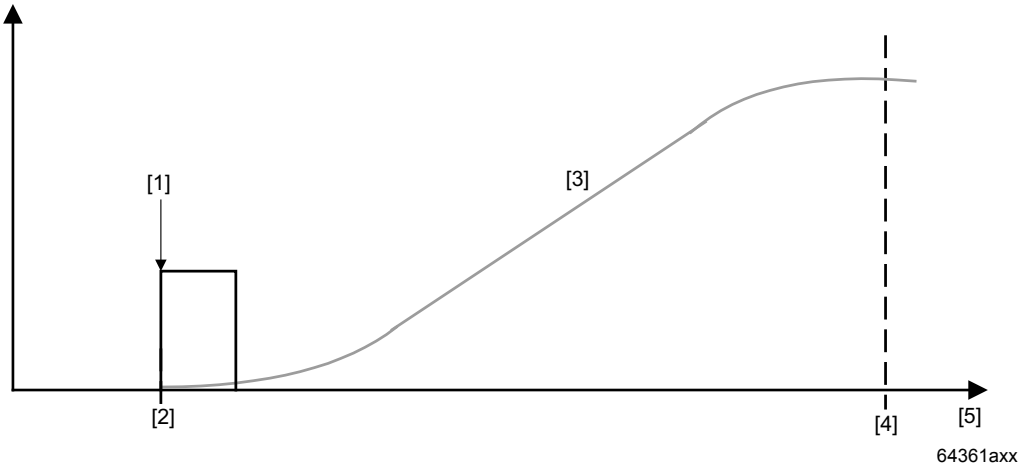
- MX_CAM_DIRECT

  The startup cycle is directly started with the "MC_CamInDirect_MX" function module.
- MX_CAM_INTERRUPT_DI01_POS_FLAG
- MX_CAM_INTERRUPT_DI01_NEG_FLAG
- MX_CAM_INTERRUPT_DI01

...

- MX_CAM_INTERRUPT_DI08_POS_FLAG
- MX_CAM_INTERRUPT_DI08_NEG_FLAG
- MX_CAM_INTERRUPT_DI08

- MX_CAM_INTERRUPT_C_TRACK_ENC1
- MX_CAM_INTERRUPT_C_TRACK_ENC2
- MX_CAM_INTERRUPT_C_TRACK_ENC3



[1] Interrupt DI02 (PosFlag)

[2] Startup cycle start

[3] n slave

[4] Slave is in sync

[5] Master position

3.3.6 Offset cycle

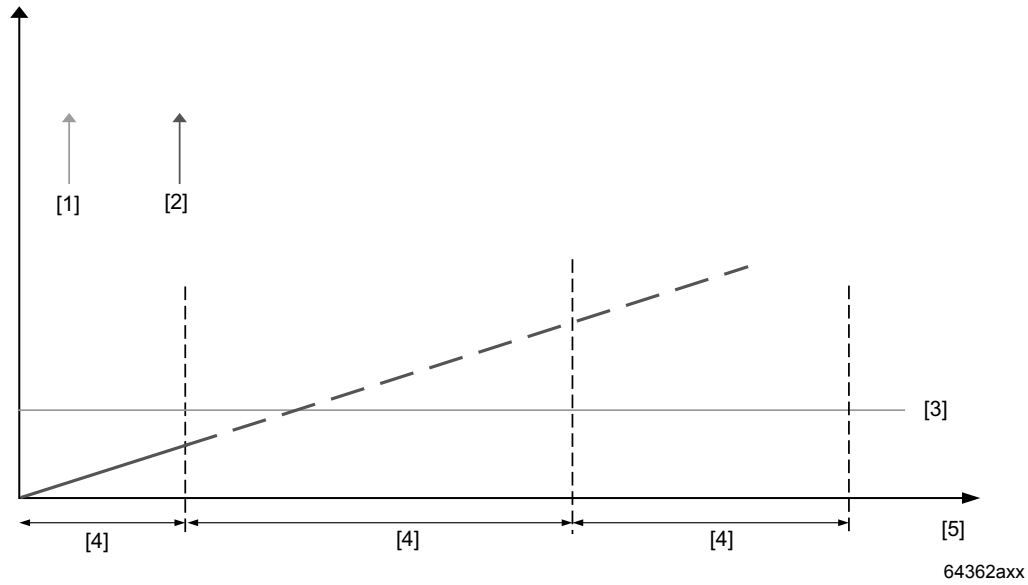
An offset travel can be activated during synchronous operation after a successfully completed startup cycle. With the "MC_OffsetDirekt_MX" module, the offset travel is overlapped depending on the position compared to the master distance. This means that the SlaveOffset is added to the current curve. Thus the master-slave relation can be trimmed.

You can enter a negative or positive slave offset.

The offset is calculated with a polynom_5 type transition.

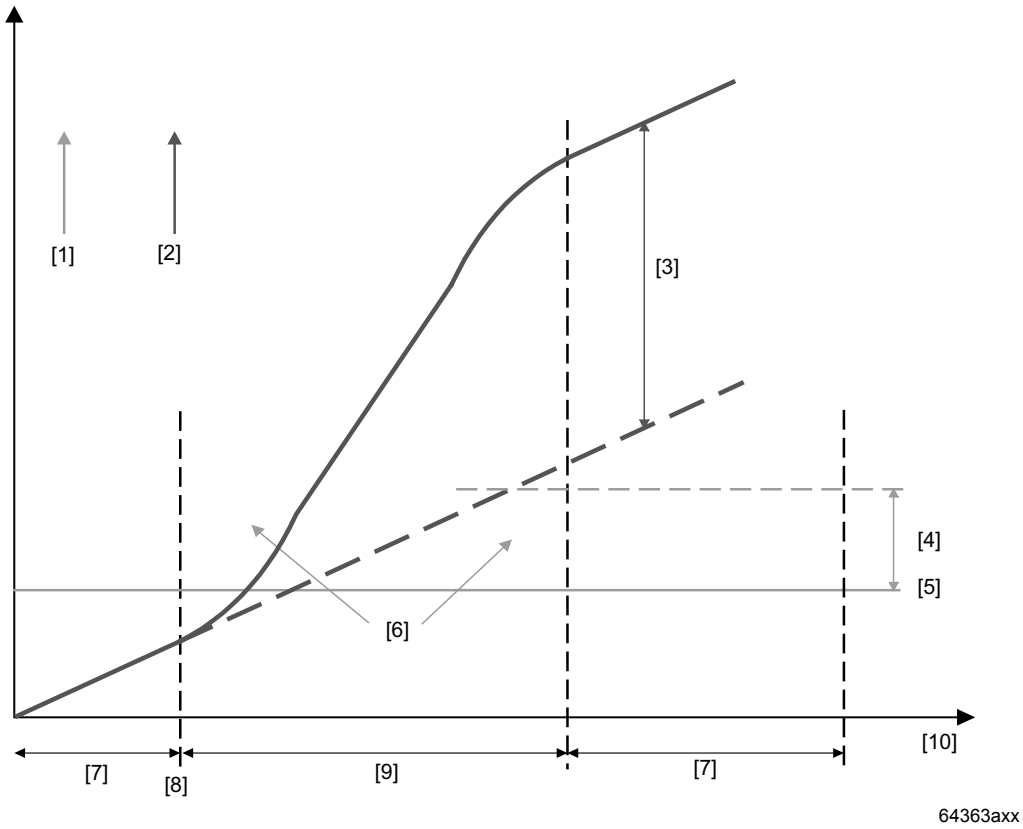
Example: The offset travel is displayed using two synchronous operation curves for better illustration.

Curve profile without offset



[1] Slave speed
[2] Slave position
[3] n slave
[4] Synchronous
[5] Master position

Curve profile with offset:



[1] Slave speed
[2] Slave position
[3] Slave offset
[4] Offset speed
[5] n slave
[6] Offset Acceleration/Deceleration
[7] Synchronous
[8] Start offset
[9] Master cycle of the offset travel
[10] Master position

3.3.7 Project planning

Procedure

- Integrate the MPLCTecCamMotion_MX library with the library manager into the project in addition to the MPLCMotion_MX library.
- The "MC_LinkTecCam_MX" function module constitutes the interface between technology function and axis and must therefore be called cyclically in the program. Once the "MC_ConnectAxis_MX" module has established the connection to the axis (Done = TRUE), "LinkTecCam" activates the technology functions of the MOVIAXIS® necessary for the Cam function. In addition to that, the necessary settings are made in the PDO Editor. A receive object, for example, is set up in the In buffer 5 for receiving the master position via system bus.
- If you want to determine the master position via the system bus, we recommend synchronizing the system bus. The required SBus synchronization telegram is configured using the "MC_SetSync_MX" function module (part of the MPLCMotion_MX library). If the virtual encoder of the controller is used as master, the SBus will be automatically synchronized by calling the "MC_LinkTecAxis_Virtual" function module (part of the MPLCTecVirtualEncoder library). The required basic settings can be found in the parameter tree under communication. The "LinkTecCam" is initialized automatically.
- The "MC_SetCamConfig_MX" function module is used to set the general parameters. This module must be called with a rising edge on the Execute input each time the MOVI-PLC® controller is restarted or if you want to change the general parameters.
- You can use the "MC_CamInDirect_MX" for a direct startup cycle. The axis has to be aligned by the application after an emergency off. If you want the axis to automatically move back the the curve point after an emergency off, you can use the *Move_to_Master* function (prepare). If you use the "MC_CamInAdjust_MX" for the startup cycle, the axis is is aligned first depending on the set Adjustmode (FCB 9). The compensation movement after an emergency off or a controller inhibit is performed with the profile generator when the unit is switched on. It is always carried out via the shortest way DX/DV.

3.3.8 Data location

Location of the curves

The maximum number of curves is 10.

Each curve has 512 points.

The library creates a GEAR and a WAIT curve.

Offset	Index/subindex	Description
0	20000.1	Start of GEAR curve
..		
4	20000.5	End of GEAR curve
5	20000.6	Start of WAIT curve
...		
9	20000.10	End of WAIT curve
10	20000.11	Start of curve 1
...		
521	20004.10	End of curve 1
522	20004.11	Start of curve 2
...		
1033	20008.10	End of curve 2
1034	20008.11	Start of curve 3
...		
1545	20012.10	End of curve 3
1546	20012.11	Start of curve 4
...		
2057	20016.10	End of curve 4
2058	20016.11	Start of curve 5
...		
2569	20020.10	End of curve 5
2570	20020.11	Start of curve 6
...		
3081	20024.10	End of curve 6
3082	20024.11	Start of curve 7
...		
3593	20028.10	End of curve 7
3594	20028.11	Start of curve 8
...		
4105	20032.10	End of curve 8
4106	20032.11	Start of curve 9
...		
4617	20036.10	End of curve 9
4618	20036.11	Start of curve 10
...		
5129	20040.10	End of curve 10

Storing the curve characteristics

Data interface between cam and CAM library.

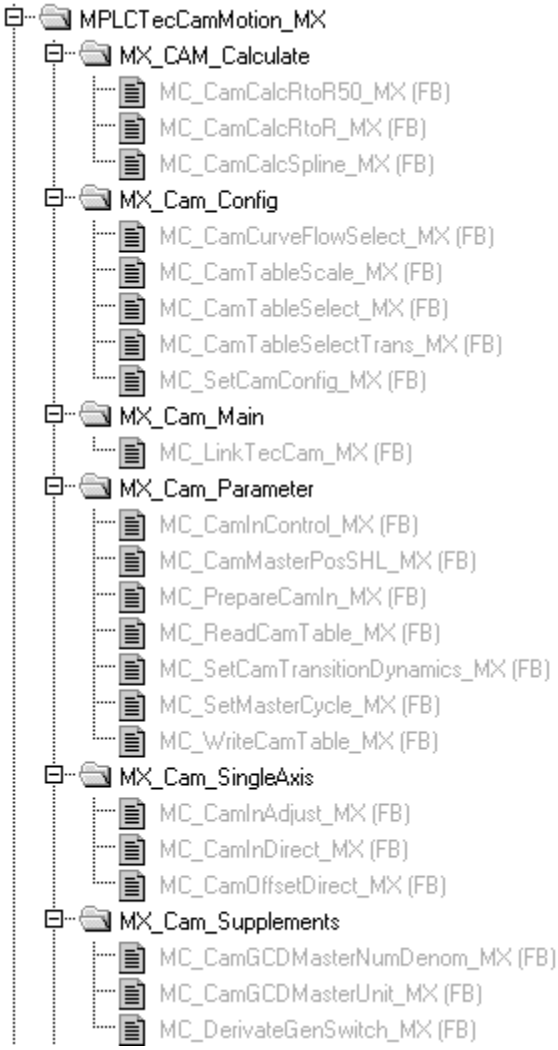
Offset	Index/subindex	Description
4095	20231.128	Version and subversion of the editor
4094	20231.127	Flags for loaded curves (1 bit per curve) bit 0 = collective bit
4093	20231.126	Number of curves
4092	20231.125	Number of control points (512)
4091	20231.124	Slave shift factor
4090	20231.123	Master shift factor
4089	20231.122	Master length curve 1
4088	20231.121	Master length curve 2
4087	20231.120	Master length curve 3
4086	20231.119	Master length curve 4
4085	20231.118	Master length curve 5
4084	20231.117	Master length curve 6
4083	20231.116	Master length curve 7
4082	20231.115	Master length curve 8
4081	20231.114	Master length curve 9
4080	20231.113	Master length curve 10

The cam editor describes the data interface (apart from master shift) and the "MC_LinkTecCam_MX" module copies the values to the corresponding locations in the DDB in the event of any changes via the editor.

You can use the "MC_CamMasterPosSHL_MX" module to adjust the master scaling factor. The curve parameters are written by the module automatically.

3.4 Overview of MPLCTecCamMotion_MX library

3.4.1 Short overview of the MPLCTecCamMotion_MX library



64364axx

MC_LinkTecCam_MX

The "MC_LinkTecCam_MX" function module has various functions:

- Setting up the technology functions of MOVIAXIS® and activating the functions required for Cam.
- Configuring the process data interface of the MOVIAXIS®
- Displaying the current state of the "Cam" technology function.
- Displaying the initialization state.

MC_SetCamConfig_MX

This module is used for basic configuration of synchronous operation. Before the startup cycle can be performed, the parameters must have been transferred via this module.

MC_CamInDirect_MX

The function module starts a direct startup cycle with a rising edge at the Execute input. The axis has to be aligned by the application after an emergency off. If you want the axis to automatically move back the the curve point after an emergency off, you can use the *Move_to_Master* function (prepare).

MC_CamInAdjust_MX

The function module starts a direct startup cycle with a rising edge at the Execute input. Depending on the set Adjustmode, the axis is aligned first (FCB9).

The compensation movement after an emergency off or a controller inhibit is performed with the profile generator when the unit is switched on. It is always carried out via the shortest way DX/DV.

MC_CamOffsetDirect_MX

With the transferred parameters, the "MC_CamOffsetDirect_MX" module activates a positioning travel superimposing the curve sequence. The slave distance is added to the master distance depending on the position.

MC_PrepareCamIn_MX

The "MC_PrepareCamIn_MX" function module can configure an interrupt event for the startup cycle. You can select/deselect the *Move_To_Master* function. You can enable or inhibit the interrupt function via the "MC_CamInControl_MX" module.

MC_CamInControl_MX

With this function module you can enable or inhibit the interrupt event. The automatic restart can be activated.

MC_CamTableSelect_MX

You can use the "MC_CamTableSelect_MX" module to select a curve (0 – 10). If a curve is activated, the switchover to the new curve is performed at the end of the activated one. The curve output is connected to the curve end, so that the curve is run through infinitely (curve 0 = waiting curve).

MC_CamTableSelectTrans_MX

You can use the "MC_CamTableSelectTrans_MX" module to select a curve (1 – 10) with a transition. If a curve is activated, the switchover to the new curve is performed at the end of the activated one. The curve output leads to a transition. The end of the transition is connected to the start of the curve, so that the curve and the transition are run through infinitely.

MC_CamCurveFlowSelect_MX

This module allows you to link up to 10 curves.

The curve outputs can be linked to one transition each.

MC_CamTableScale_MX

You can use the "MC_CamTableScale" moule to adjust the numerator and denominator of a curve.

MC_SetMasterCycle_MX

You can use the "MC_SetMasterCycle_MX" module to adjust the master cycle length for a curve.

MC_CAM_MasterPosSHL_MX

In order to achieve a better resolution, the master cycle can be scaled according to the master shift factor.

Example: Master cycle =10000 => master shift factor = 6, thus the master cycle is 10000 $\times 2^6$ = 64000.

The module scales the master cycles of all curves.

MC_SetCamTransitionDynamics_MX

This module allows you to adjust the dynamic parameters of the transition functions of the selected curve.

MC_ReadCamTable_MX

With this module you can read the 512 control points of the selected curve from the MOVIAXIS® and write them in an array.

MC_WriteCamTable_MX

You can use this module to write an array with 512 control points for the selected curve into the MOVIAXIS®.

MC_CamGCDMasterNumDenom_MX

This module determines the highest common factor and reduces the fractions accordingly. No parameters are written.

MC_CamGCDMasterUnit_MX

This module determines a numerator and a denominator from the master distance. No parameters are written.

MC_Derivate_Gen_Switch_MX

The library works with the CamFlow1 and CamFlow2 and the Modbus Position.

This module allows you to switch the source of the derivate generator.

MC_CamCalRtoR_MX

Calculation of section-by-section curves starting from 2 – 6 coordinate points. The curve always begins at 0 / 0 (X1 / Y1).

MC_CamCalRtoR50_MX

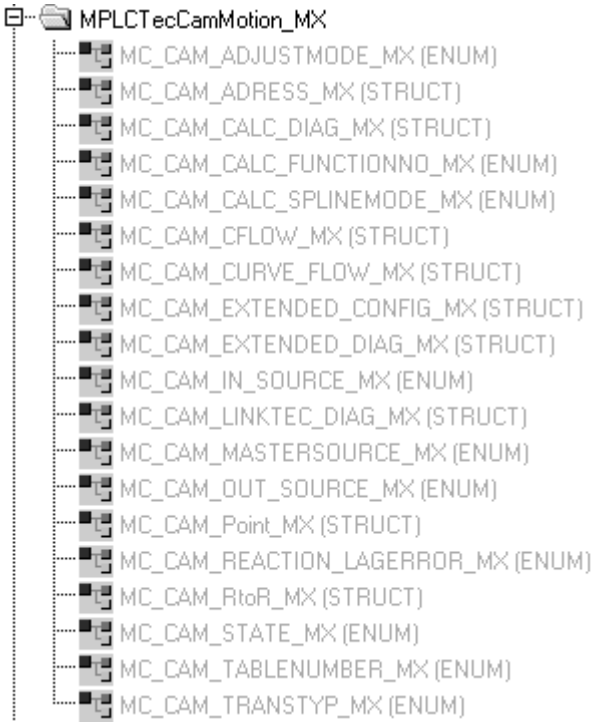
Calculation of section-by-section curves starting from 2 – 50 coordinate points.

Master offset and slave offset can be set.

MC_CamCalcSpline_MX

Calculation of a spline curve consisting of 3 – 29 coordinate points.

3.4.2 Data types of the MPLCTecCamMotion_MX library



64366axx

MC_CAM_ADJUSTMODE_MX

```
TYPE MC_CAM_ADJUSTMODE_MX :(
    MX_CAMINDIRECT      := 0,   (*cam in direct without adjust, init with XMasterPosZero*)
    MX_ADJUST           := 1    (*move the slave to XMasterPosAct *)
);
END_TYPE
```

64367axx

MC_CAM_ADRESS_MX

```
TYPE MC_CAM_ADRESS_MX :
STRUCT
    Variable_IO         : DINT := 3;      (*DDB Startadress Vatiable IO*)
    PosSetPointGen      : DINT := 24;     (*DDB Startadress PositionSetpointGenerator*)
    Gear                : DINT := 61;     (*DDB Startadress Gear*)
    SystemData          : DINT := 104;    (*DDB Startadress Systemdata*)
    VirtualEncoderMX    : DINT := 149;    (*DDB Startadress Virtual Encoder*)
    Messtaster_1        : DINT := 189;    (*DDB Startadress Messtaster 1*)
    Messtaster_2        : DINT := 226;    (*DDB Startadress Messtaster 2*)
    Messtaster_3        : DINT := 263;    (*DDB Startadress Messtaster 3*)
    Messtaster_4        : DINT := 300;    (*DDB Startadress Messtaster 4*)
    Databuffer_1        : DINT := 339;    (*DDB Startadress Databuffer 1*)
    Databuffer_2        : DINT := 430;    (*DDB Startadress Databuffer 2*)
    Databuffer_3        : DINT := 521;    (*DDB Startadress Databuffer 3*)
    Databuffer_4        : DINT := 612;    (*DDB Startadress Databuffer 4*)
    EventControl        : DINT := 705;    (*DDB Startadress Event Control*)
    Cam_Controller      : DINT:= 815;     (* 0*)     (*DDB Startadress Cam Control*)
    Cam_1               : DINT :=1401;    (* 819*)   (*DDB Startadress CAMFlow1 *)
    Cam_2               : DINT :=2021;    (*1439*)  (*DDB Startadress CAMFlow2 *)
    Cam_3               : DINT :=2173;    (*159;*)  (*DDB Startadress CAMFlow3 *)
    ProfGen             : DINT :=2271;    (*1689*)  (*DDB Startadress CAM Profilgenerator*)
    DerivateGen         : DINT :=2304;    (*1722*)  (*DDB Startadress CAM DerivateGenerators/MotorManager*)
    Motormangment       : DINT :=2317;    (*1735*)  (*DDB Startadress CAM Motormanagment*)
END_STRUCT
```

64368axx

MC_CAM_CALC_DIAG_MX

```
TYPE MC_CAM_CALC_DIAG_MX :
STRUCT
        (*parameter fbcamAnalyzer*)
        Vmax            :DINT;    (*maximum speed*)
        Vmin            :DINT;    (*minimum speed*)
        AbsVmax         :DINT;    (*absolut maximum speed*)
        Amax            :DINT;    (*maximum acceleration*)
        Amin            :DINT;    (*minimum acceleration*)
        AbsAmax         :DINT;    (*absolut maximum acceleration*)
        Mastercycle     :DINT;    (*mastercyclescale*)
        Shiftfaktor     :UINT;    (*Slave RightShift Faktor Default 10*)
        State           :BYTE;    (*state machine*)
        MasterShiftTab  :DINT;    (* the table point of  the shift masterposition only used by RtoR50*)
END_STRUCT
END_TYPE
```

64369axx

MC_CAM_CALC_FUNCTIONNO_MX

```
TYPE MC_CAM_CALC_FUNCTIONNO_MX :(
    MX_CAM_STRAIGHT                 :=0, (*straight line*)
    MX_CAM_POLYNOMIAL_5             :=1, (*polynomial 5th order *)
    MX_CAM_SINUS                    :=2, (*sine curve*)
    MX_CAM_POLYNOMIAL_3             :=3, (*polynomial 3rd order *)
    MX_CAM_BESTEHORN_SINOIDE        :=4, (*sinoide  Bestehorn*)
    MX_CAM_ACC_TRAPEZOID            :=5); (*modified trapezoid*)
END_TYPE
```

64370axx

MC_CAM_CALC_SPLINEMODE_MX

```
TYPE MC_CAM_CALC_SPLINEMODE_MX :(
    MX_CAM_Spline_0             :=0, (* Spline-Interpolation *)
    MX_CAM_Spline_B             :=1, (*Approximation   - Submode 0 - polynomial 3rd order
                                      - Submode 1- not used
                                     - Submode 2 - polynomial 2nd order
                                     - Submode 3 - polynomial 3rd  order
                                     - Submode 4 - polynomial 4th  order
                                     - Submode 5 - polynomial 5th  order
                                     - Submode 6 - polynomial 6th  order
                                     - Submode 7 - polynomial 7th  order*)
    MX_CAM_Spline_1             :=2 (* fast, piecewise Interpolation *)
    );
END_TYPE
```

64371axx

MC_CAM_CFLOW_MX

```
TYPE MC_CAM_CFLOW_MX :
STRUCT
    Curvenumber:      MC_CAM_TABLENUMBER_MX:=1;      (*number of the curve*)
    Nextcurve:        MC_CAM_TABLENUMBER_MX:=1;      (*next curve pos direction*)
    Transitionmaster: DINT;      (*transision during  ...master units*)
    YSlaveoffsetadd:  DINT;      (*slaveoffset between the curve included the difference off the old curve*)
END_STRUCT
END_TYPE
```

64372axx

MC_CAM_CURVE_FLOW_MX

```
TYPE MC_CAM_CURVE_FLOW_MX :
STRUCT
    Camsegment: ARRAY[1..10] OF MC_CAM_CFLOW_MX;
END_STRUCT
END_TYPE
```

64373axx

MC_CAM_EXTENDED_CONFIG_MX

```
TYPE MC_CAM_EXTENDED_CONFIG_MX :
STRUCT
(* Extended LinkTecCam Configurations *)
    UseTecEditorConfig:          BOOL;              (* based on the PLCdefault TecEditor project additional
                                                    Tec configurations can be made, e.g. CAM2/3, Touchprobe, etc.*)
    TecFunctionsUserUnitSlave: BOOL:=1;             (* True = Tec values are scaled in user units
                                                       False = Tec values are in system units (unscaled)*)
    TecFunctionCamExpant:        BOOL;              (*False =it is  only to compress a cam curve possible
                                                      True =it  is also allowed to expant a cam curve*)
    TecFunktionTransition:    MC_CAM_TRANSTYP_MX:=4;(*
                                    MX_CAM_TransPolynomial3:=1,  Transition with Ploynom 3
                                    MX_CAM_TransPolynomial5:=2,  Transition with Ploynom 5
                                    MX_CAM_LPG2:=3,              Transition with LPG2
                                    MX_CAM_LPG1:=4               Transition with LPG1
                                    *)
    Calc_SlaveShift_max          : DINT := 10;   (* limit of the calculate SlaveShift*)

    stInitialParameter_MX : MC_INITIALPARAMETER;
                                    (*#################################################################
                                    With this struct several user defined parameters can be written to the axis during the
                                    initialization of MC_ConnectAxis_MX. DHx11B : 10 parameters, DHx41B 60 parameters

                                    Add the following code to your program to initialise the struct :

                                    Example: write DDB variable 2577 and 3420:

                                        stInitialParameter_MX.aIndex[1] := 20220;    (* Index of DDB Variable 2577 *)
                                        stInitialParameter_MX.aSubIndex[1] := 18;    (* SubIndex of DDB Variable 2577 *)
                                        stInitialParameter_MX.aData[1] := 10;        (* Data to be written to DDB Variable 2577*)

                                        stInitialParameter_MX.aIndex[2] := 20226;    (* Index of DDB Variable 3420 *)
                                        stInitialParameter_MX.aSubIndex[2] := 93;    (* SubIndex of DDB Variable 3420 *)
                                        stInitialParameter_MX.aData[2] := 20;        (* Data to be written to DDB Variable 3420*)

                                    If there are more than 10 parameters to write, declare NUMOF_INITIALPARAMETER
                                    in your program.

                                        VAR_GLOBAL CONSTANT
                                            NUMOF_INITIALPARAMETER : UDINT := 20;
                                        END_VAR

                                    #################################################################*)

(*################################## Extended SetCamConfig configurations ##########################*)
(* Extended SetCamConfig configurations *)
    MSignal_InterpolationTime:  DINT;               (* InterpolationTime: range 1...60 resolution [0.5ms] (0.5....30ms)*)
    MSignal_DelayTime:          DINT;               (* PositionDelayTime in [µs] *)
    AverageFilterTime:          DINT;               (* AverageFilterTime: range 1...60 resolution [0.5ms] (0.5....30ms)*)
    CompFilterTime:             DINT;               (* CompFilterTime: range 1...60 resolution [0.5ms] (0.5....30ms)*)
    DeathTimeCorr_ON:           BOOL;               (* Activation of the death time correction: false = OFF, true = ON*)
    SpeedCorr_ON:               BOOL;               (* Activation of the velocity related correction: false = OFF, true = ON*)
    AccelerationCorr_ON:        BOOL;               (* Activation of the acceleration related correction: false = OFF, true = ON*)
    ReactionLagError:           MC_CAM_REACTION_LAGERROR_MX:=10;(* defines the reaction after lag detection. Possible se

                                    MX_CAM_NO_RESPONSE :=0,
                                    MX_CAM_DISPLAY_ONLY := 1,
                                    MX_CAM_STOP_APPL_LIMIT_AUTORESET_NO_HISTORY:=21,
                                    MX_CAM_STOP_APPL_LIMIT_AUTORESET:=17,
                                    MX_CAM_STOP_APPL_LIMIT_WAITING_NO_HISTORY:=13,
                                    MX_CAM_STOP_APPL_LIMIT_WAITING:=8,
                                    MX_CAM_STOP_APPL_LIMIT_LOCKED:=9,

                                    MX_CAM_IMMEDIATE_STOP_AUTORESET_NO_HISTORY:=22,
                                    MX_CAM_IMMEDIATE_STOP_AUTORESET:=18,
                                    MX_CAM_IMMEDIATE_STOP_WAITING_NO_HISTORY:=14,
                                    MX_CAM_IMMEDIATE_STOP_WAITING:=6,
                                    MX_CAM_IMMEDIATE_STOP_LOCKED:=3,

                                    MX_CAM_STOP_SYSTEM_LIMIT_AUTORESET_NO_HISTORY:=23,
                                    MX_CAM_STOP_SYSTEM_LIMIT_AUTORESET:=19,
                                    MX_CAM_STOP_SYSTEM_LIMIT_WAITING_NO_HISTORY:=15,
                                    MX_CAM_STOP_SYSTEM_LIMIT_WAITING:=10,
                                    MX_CAM_STOP_SYSTEM_LIMIT_LOCKED:=11,

                                    MX_CAM_INHIBIT_AUTORESET_NO_HISTORY:=24,
                                    MX_CAM_INHIBIT_AUTORESET:=20,
                                    MX_CAM_INHIBIT_WAITING_NO_HISTORY:=16,
                                    MX_CAM_INHIBIT_WAITING:=5,
                                    MX_CAM_INHIBIT_LOCKED:=2,
                                    MX_CAM_INHIBIT_LOCKED_OVERIDE:=25,

                                    MX_CAM_SYSTEM_INTERNAL_WAITING:=12,
                                    MX_CAM_STOP_WAITING:=7,
                                    MX_CAM_STOP_LOCKED:=4,
                                    MX_CAM_DISPLAY_ERRORHISTORY:=26
                                    *)
END_STRUCT
END_TYPE
```

64374axx

MC_CAM_EXTENDED_DIAG_MX

```
TYPE MC_CAM_EXTENDED_DIAG_MX :
STRUCT
    CamFlow1ActCurve            :UINT;      (*Cam Flow 1 status actual curve number *)
    CamFlow2ActCurve            :UINT;      (*Cam Flow 2 status actual curve number *)
    Modulo_underflow            :DINT;      (*Modulo limit underflow*)
    Modulo_overflow             :DINT;      (*Modulo limit overflow*)

    Calc_SlaveShift_max         : DINT;     (* limit of the calculate SlaveShift*)
    number_mastercycles         :UINT:=1;       (*number of mastercycles after tableselect*)
    CamEditor_VersionSubversion   : DINT;       (*version off the cam editor*)
    CurveChanged                :DINT; (*   Bit -0 one or more Curve are changed
                                               Bit -1 Curve 1 is changed
                                               Bit -2 Curve 2 is changed
                                               Bit -3 Curve 3 is changed
                                               Bit -4 Curve 4 is changed
                                               Bit -5 Curve 5 is changed
                                               Bit -6 Curve 6 is changed
                                               Bit -7 Curve 7 is changed
                                               Bit -8 Curve 8 is changed
                                               Bit -9 Curve 9 is changed
                                               Bit -10 Curve 10 is changed*)
    NumberofCurve               :DINT; (*number off designed curve*)
    Tablepoints                 :DINT; (*number of tablepoints*)
    SlaveShift                  :DINT; (*all curve points are shifted with the Slave Shift *)
    MasterShift                 :DINT; (*the master position is shifted *)
    Mastercycle_lenght :        ARRAY[1..10]  OF DINT;  (*Mastercyclelenght Curve 1 -10
                                                        Mastercycle_lenght_1        :DINT;  Mastercyclelenght Curve 1
                                                        Mastercycle_lenght_2        :DINT;  Mastercyclelenght Curve 2
                                                        Mastercycle_lenght_3        :DINT;  Mastercyclelenght Curve 3
                                                        Mastercycle_lenght_4        :DINT;  Mastercyclelenght Curve 4
                                                        Mastercycle_lenght_5        :DINT;  Mastercyclelenght Curve 5
                                                        Mastercycle_lenght_6        :DINT;  Mastercyclelenght Curve 6
                                                        Mastercycle_lenght_7        :DINT;  Mastercyclelenght Curve 7
                                                        Mastercycle_lenght_8        :DINT   Mastercyclelenght Curve 8
                                                        Mastercycle_lenght_9        :DINT;  Mastercyclelenght Curve 9
                                                        Mastercycle_lenght_10       :DINT;  Mastercyclelenght Curve 10*)
    Transition_mastercycle :    ARRAY [11..20] OF DINT;(*Mastercyclelenght Transition Block 11 -20
                                                        Transition_Mastercycle_11           :DINT;  Mastercyclelenght Curve 11
                                                        Transition_Mastercycle_12           :DINT;  Mastercyclelenght Curve 12
                                                        Transition_Mastercycle_13           :DINT;  Mastercyclelenght Curve 13
                                                        Transition_Mastercycle_14           :DINT;  Mastercyclelenght Curve 14
                                                        Transition_Mastercycle_15           :DINT;  Mastercyclelenght Curve 15
                                                        Transition_Mastercycle_16           :DINT;  Mastercyclelenght Curve 16
                                                        Transition_Mastercycle_17           :DINT;  Mastercyclelenght Curve 17
                                                        Transition_Mastercycle_18           :DINT   Mastercyclelenght Curve 18
                                                        Transition_Mastercycle_19           :DINT;  Mastercyclelenght Curve 19
                                                        Transition_Mastercycle_20           :DINT;  Mastercyclelenght Curve 20*)
    Slave_deviation:            ARRAY[1..10] OF DINT;   (*Slave deviation from the statpoint to the endpoint of a curve
                                                        Slave_deviation_1    - difference curve 1
                                                        Slave_deviation_2    - difference curve 2
                                                        Slave_deviation_3    - difference curve 3
                                                        Slave_deviation_4    - difference curve 4
                                                        Slave_deviation_5    - difference curve 5
                                                        Slave_deviation_6    - difference curve 6
                                                        Slave_deviation_7    - difference curve 7
                                                        Slave_deviation_8    - difference curve 8
                                                        Slave_deviation_9    - difference curve 9
                                                        Slave_deviation_10   - difference curve 10*)
    Slave_OffsetAdd:            ARRAY[1..10] OF DINT;   (*Slave deviation from the statpoint to the endpoint of a curve
                                                        Slave_OffsetAdd_1    - offset add difference curve 1
                                                        Slave_OffsetAdd_2    - offset add difference curve 2
                                                        Slave_OffsetAdd_3    - offset add difference curve 3
                                                        Slave_OffsetAdd_4    - offset add difference curve 4
                                                        Slave_OffsetAdd_5    - offset add difference curve 5
                                                        Slave_OffsetAdd_6    - offset add difference curve 6
                                                        Slave_OffsetAdd_7    - offset add difference curve 7
                                                        Slave_OffsetAdd_8    - offset add difference curve 8
                                                        Slave_OffsetAdd_9    - offset add difference curve 9
                                                        Slave_OffsetAdd_10   - offset add difference curve 10*)
    Cam_Adress                  :MC_CAM_ADRESS_MX;  (*Startadress DDB*)
END_STRUCT
END_TYPE
```

64375axx

MC_CAM_IN_SOURCE_DIAG_MX

```
TYPE MC_CAM_IN_SOURCE_MX :
    (
        MX_CAM_DIRECT,                                  (* CAM in direct with MC_CamInDirect_MX  *)
        MX_CAM_INTERRUPT_DI01_POS_FLAG:=16#30,          (* CamIn(Offset) by DI01 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI01_NEG_FLAG:=16#31,          (* CamIn(Offset) by DI01 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI01:=16#32,                   (* CamIn(Offset) by DI01 interrupt both edges *)
        MX_CAM_INTERRUPT_DI02_POS_FLAG:=16#40,          (* CamIn(Offset) by DI02 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI02_NEG_FLAG:=16#41,          (* CamIn(Offset) by DI02 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI02:=16#42,                   (* CamIn(Offset) by DI02 interrupt both edges *)
        MX_CAM_INTERRUPT_DI03_POS_FLAG:=16#50,          (* CamIn(Offset) by DI03 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI03_NEG_FLAG:=16#51,          (* CamIn(Offset) by DI03 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI03:=16#52,                   (* CamIn(Offset) by DI03 interrupt both edges *)
        MX_CAM_INTERRUPT_DI04_POS_FLAG:=16#60,          (* CamIn(Offset) by DI04 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI04_NEG_FLAG:=16#61,          (* CamIn(Offset) by DI04 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI04:=16#62,                   (* CamIn(Offset) by DI04 interrupt both edges *)
        MX_CAM_INTERRUPT_DI05_POS_FLAG:=16#70,          (* CamIn(Offset) by DI05 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI05_NEG_FLAG:=16#71,          (* CamIn(Offset) by DI05 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI05:=16#72,                   (* CamIn(Offset) by DI05 interrupt both edges *)
        MX_CAM_INTERRUPT_DI06_POS_FLAG:=16#80,          (* CamIn(Offset) by DI06 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI06_NEG_FLAG:=16#81,          (* CamIn(Offset) by DI06 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI06:=16#82,                   (* CamIn(Offset) by DI06 interrupt both edges *)
        MX_CAM_INTERRUPT_DI07_POS_FLAG:=16#90,          (* CamIn(Offset) by DI07 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI07_NEG_FLAG:=16#91,          (* CamIn(Offset) by DI07 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI07:=16#92,                   (* CamIn(Offset) by DI07 interrupt both edges *)
        MX_CAM_INTERRUPT_DI08_POS_FLAG:=16#A0,          (* CamIn(Offset) by DI08 interrupt rising edge *)
        MX_CAM_INTERRUPT_DI08_NEG_FLAG:=16#A1,          (* CamIn(Offset) by DI08 interrupt falling edge *)
        MX_CAM_INTERRUPT_DI08:=16#A2,                   (* CamIn(Offset) by DI08 interrupt both edges *)
        MX_CAM_INTERRUPT_C_TRACK_ENC1:=16#2,            (* CamIn(Offset) by C-track interrupt of encoder1 *)
        MX_CAM_INTERRUPT_C_TRACK_ENC2:=16#12,           (* CamIn(Offset) by C-track interrupt of encoder2 *)
        MX_CAM_INTERRUPT_C_TRACK_ENC3:=16#22,           (* CamIn(Offset) by C-track interrupt of encoder3 *)
        MX_CAM_DISTANCE_COUNTER:=16#01FF,               (* CamIn(Offset) by master distance without reset actual counter value*)
        MX_CAM_DISTANCE_COUNTER_RESET_COUNT:=16#02FF    (* CamIn(Offset) by master distance , reset actual counter value*)
    );
END_TYPE
```

64376axx

MC_CAM_LINKTEC_DIAG_MX

```
TYPE MC_CAM_LINKTEC_DIAG_MX :
STRUCT
    CamFlow1ActCurve        :UINT;    (*Cam Flow 1 status actual curve number *)
    CamFlow2ActCurve        :UINT;    (*Cam Flow 2 status actual curve number *)
```

64377axx

MC_CAM_MASTERSOURCE_MX

```
TYPE MC_CAM_MASTERSOURCE_MX :
    (
        MX_CAM_SYSTEMPOS_ENCODER1,          (* Master position is the linear position of encoder 1*)
        MX_CAM_SYSTEMPOS_ENCODER2,          (* Master position is the linear position of encoder 2*)
        MX_CAM_SYSTEMPOS_ENCODER3,          (* Master position is the linear position of encoder 3*)
        MX_CAM_MODULOPOS_ENCODER1,          (* Master position is the modulo position of encoder 1*)
        MX_CAM_MODULOPOS_ENCODER2,          (* Master position is the modulo position of encoder 2*)
        MX_CAM_MODULOPOS_ENCODER3,          (* Master position is the modulo position of encoder 3*)
        MX_CAM_MODULO_VIRTUAL_ENCODER_MX,   (* Master position is the modulo position of the MX virtual encoder *)
        MX_CAM_LINEAR_VIRTUAL_ENCODER_MX,   (* Master position is the linear position of the MX virtual encoder *)
        MX_CAM_RECEIVE_PDO_1,               (* Master position is the SBus receive objekt PDO1 *)
        MX_CAM_RECEIVE_PDO_2                (* Master position is the SBus receive objekt PDO2 *)
    );
END_TYPE
```

64378axx

MC_CAM_POINT_MX

```
TYPE MC_CAM_POINT_MX :
STRUCT
    X       : DINT;     (*masterpoints*)
    Y       : DINT;     (*slavepoints*)
END_STRUCT
END_TYPE
```

64379axx

MC_CAM_REACTION_LAGERROR_MX

```
TYPE MC_CAM_REACTION_LAGERROR_MX :
    (
    MX_CAM_NO_RESPONSE :=0,
    MX_CAM_DISPLAY_ONLY := 1,
    MX_CAM_STOP_APPL_LIMIT_AUTORESET_NO_HISTORY:=21,
    MX_CAM_STOP_APPL_LIMIT_AUTORESET:=17,
    MX_CAM_STOP_APPL_LIMIT_WAITING_NO_HISTORY:=13,
    MX_CAM_STOP_APPL_LIMIT_WAITING:=8,
    MX_CAM_STOP_APPL_LIMIT_LOCKED:=9,

    MX_CAM_IMMEDIATE_STOP_AUTORESET_NO_HISTORY:=22,
    MX_CAM_IMMEDIATE_STOP_AUTORESET:=18,
    MX_CAM_IMMEDIATE_STOP_WAITING_NO_HISTORY:=14,
    MX_CAM_IMMEDIATE_STOP_WAITING:=6,
    MX_CAM_IMMEDIATE_STOP_LOCKED:=3,

    MX_CAM_STOP_SYSTEM_LIMIT_AUTORESET_NO_HISTORY:=23,
    MX_CAM_STOP_SYSTEM_LIMIT_AUTORESET:=19,
    MX_CAM_STOP_SYSTEM_LIMIT_WAITING_NO_HISTORY:=15,
    MX_CAM_STOP_SYSTEM_LIMIT_WAITING:=10,
    MX_CAM_STOP_SYSTEM_LIMIT_LOCKED:=11,

    MX_CAM_INHIBIT_AUTORESET_NO_HISTORY:=24,
    MX_CAM_INHIBIT_AUTORESET:=20,
    MX_CAM_INHIBIT_WAITING_NO_HISTORY:=16,
    MX_CAM_INHIBIT_WAITING:=5,
    MX_CAM_INHIBIT_LOCKED:=2,
    MX_CAM_INHIBIT_LOCKED_OVERIDE:=25,

    MX_CAM_SYSTEM_INTERNAL_WAITING:=12,
    MX_CAM_STOP_WAITING:=7,
    MX_CAM_STOP_LOCKED:=4,
    MX_CAM_DISPLAY_ERRORHISTORY:=26
    );
END_TYPE
```

64380axx

MC_CAM_RtoR_MX

```
TYPE MC_CAM_RtoR_MX :
STRUCT
    Point: ARRAY[1..50] OF MC_CAM_Point_MX;
END_STRUCT
END_TYPE
```

64381axx

MC_CAM_STATE_MX

```
TYPE MC_CAM_STATE_MX :
    (
        MX_CAM_OUTCAM                   :=0,    (* Synchronous state : Axis is out caming  *)
        MX_CAM_INCAM                    :=1,    (* Synchronous state : Axis is in caming *)
        MX_CAM_INCAM_ADJUST             :=2,    (* Synchronous state : Axis is in caming *)
        MX_CAM_INCAM_OFFSET             :=3,    (* Synchronous state : Offset move while in cam direct *)
        MX_CAM_INCAM_ADJUST_OFFSET      :=4,    (* Synchronous state : Offset move while in cam adjust*)
        MX_CAM_MOVE_TO_MASTER           :=5,    (* Synchronous state : Axis is moving to Master after inhibit or safety stop init DX / DV *)
        MX_CAM_NOTLINKED                :=6     (* Synchronous state : Axis not linked *)
    );
END_TYPE
```

64382bxx

MC_CAM_TABLENUMBER_MX

```
TYPE MC_CAM_TABLENUMBER_MX :
(       CAM_WAIT_MX     :=0,        (*Curve Wait Startpoint  5 Lenght 5*)
        CAM_1_MX        :=1,        (*CamFLow 1 Curve  1 Startpoint    10 Lenght 512*)
        CAM_2_MX        :=2,        (*CamFLow 1 Curve  2 Startpoint   522  Lenght 512*)
        CAM_3_MX        :=3,        (*CamFLow 1 Curve  3 Startpoint 1034  Lenght 512*)
        CAM_4_MX        :=4,        (*CamFLow 1 Curve  4 Startpoint 1546  Lenght 512*)
        CAM_5_MX        :=5,        (*CamFLow 1 Curve  5 Startpoint 2058  Lenght 512*)
        CAM_6_MX        :=6,        (*CamFLow 1 Curve  6 Startpoint 2570  Lenght 512*)
        CAM_7_MX        :=7,        (*CamFLow 1 Curve  7 Startpoint 3082  Lenght 512*)
        CAM_8_MX        :=8,        (*CamFLow 1 Curve  8 Startpoint 3594  Lenght 512*)
        CAM_9_MX        :=9,        (*CamFLow 1 Curve  9 Startpoint 4106  Lenght 512*)
        CAM_10_MX       :=10,       (*CamFLow 1 Curve 10 Startpoint 2058  Lenght 512*)
        CAM_GEAR_MX     :=11        (*Curve GEAR  Startpoint    0 Lenght 5*)
);
END_TYPE
```

64383axx

MC_CAM_TRANSTYP_MX

```
TYPE MC_CAM_TRANSTYP_MX :(
    MX_CAM_TRANSPOLYNOMINAL3:=1,    (*Transition with Ploynom 3 / Ya,Va,Ye,Ve*)
    MX_CAM_TRANSPOLYNOMINAL5:=2,    (*Transition with Ploynom 5 /Ya,Va,Aa,Ye,Ve,Ae*)
    MX_CAM_LPG2:=3,                 (*Transition with LPG2 / Amax*)
    MX_CAM_LPG1:=4                  (*Transition with LPG1 / Amax,VmaxP,VmaxN*)
        );
END_TYPE
```

64384axx

3.4.3 Extended configuration

The technology functions of MOVIAXIS® offer numerous setting options, filters and compensation functions that are not required for most standard applications. To keep the modules simple while providing an option for access, these parameters are integrated in the "MC_CAM_EXTENDED_CONFIG_MX" input structure.

This structure is used both by the "MC_LinkTecCam_MX" module and by "MC_SetCamConfig_MX".

3.4.4 ExtConfig of MC_LinkTecCam_MX

UseTecEditorConfig

The structure variable *UseTecEditorConfig* allows for an extended configuration with the TecEditor using the library modules in parallel to the CAM configuration.

The touch probe function can be set here, for example, or another curve block for superimposition can be configured.

	TIPS
	Procedure • Since DDB ranges might shift, the IEC program should first be stopped with reset (cold) to avoid fault messages. • The "MOVI-PLCDefaultprojekt..... TecFunction" TecEditor project must be used as a basis. Now you can modify or extend this project accordingly. It is essential not to deactivate TecFunctions that are activated in the basic project. • Download modified TecEditor project. • Once the variable *UseTecEditorConfig* = TRUE has been set, the program can be started.

TecFunctionUserUnitSlave

The variable is initialized to TRUE and mans that all system variables are regarded as user unit.

FALSE means that the system units are used (parameter 20300.1).

TecFunctionCamExpant

Default setting = FALSE. Usually, due to the automatic scaling of the slave positions it is not possible to expand a curve in the numerator/denominator scaling, but to compress it. However, if you intentionally keep the curve smaller via the "maximum shift factor" parameter, you can also expand the curve by setting the "TecFunctionCamExpant" bit.

TecFunctionTransition

Default setting = MX_CAM_LPG1. The transition functions that can follow a mathematical curve are calculated with the linear profile generator 1. That means that you can specify a maximum velocity and a maximum acceleration. This function can be switched accordingly, see "MC_CAM_TRANSTYP_MX".

stInitialParameter_MX

Here you can enter MOVIAXIS® parameters in an array that are transferred to the unit by the "MC_LinkTecCam_MX" module.

3.4.5 ExtConfig with MC_SetCamConfig_MX

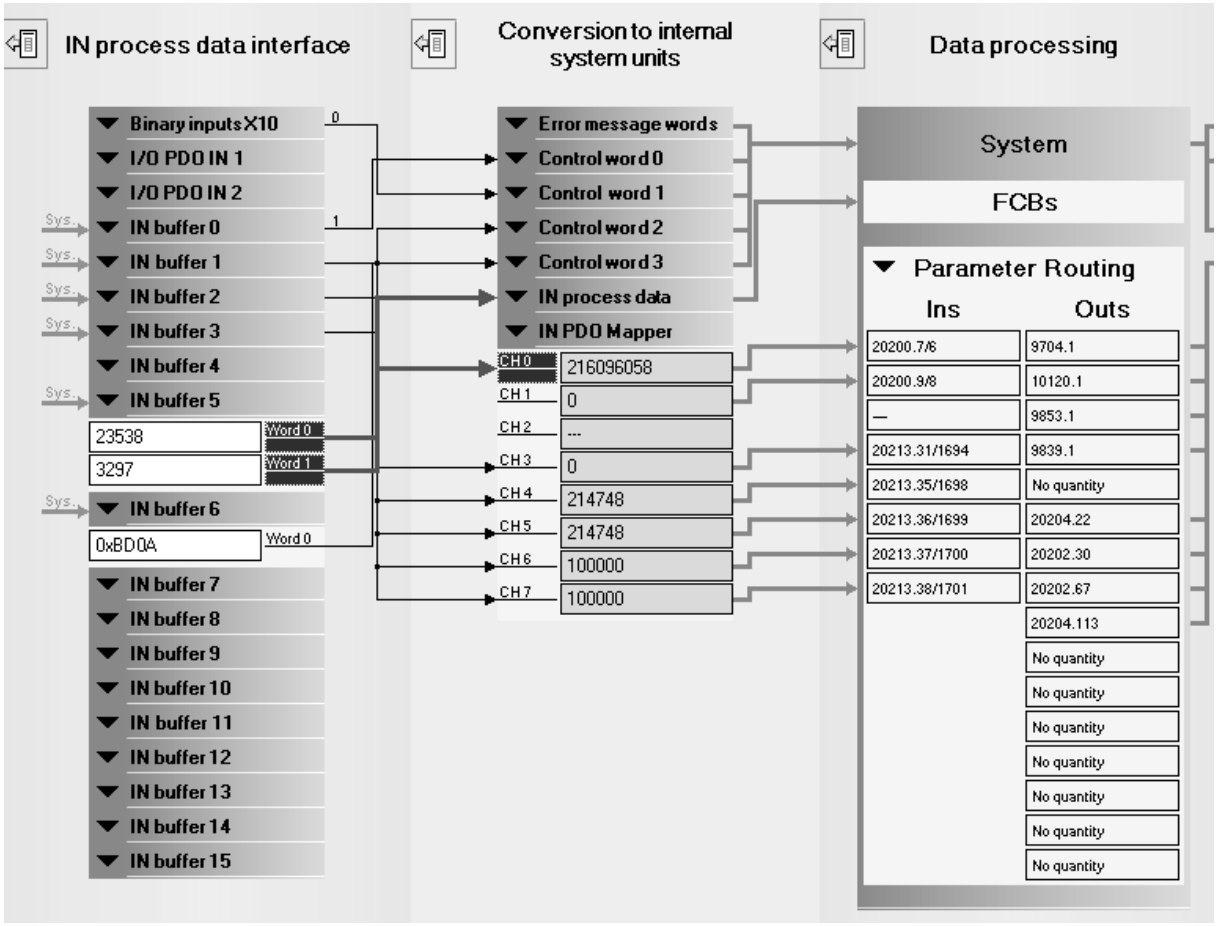
The extended configuration of the "SetCamConfig" includes variables that are used to set additional filters for conditioning the master signal.

For detailed information on the function and effects of these filters, refer to the MOVIAXIS® documentation "MOVIAXIS® MX Multi-Axis Servo Inverter Technology Function Electronic Gear Unit".

The filters are deactivated by default. With the virtual encoder of MOVI-PLC®, the *MSignal_InterpolationTime* value is set to 5 ms.

3.4.6 PDO configuration

In addition to the configuration made by "MC_ConnectAxis_MX" module, the "MC_LinkTecCam_MX" module also performs some settings at the process data interface.



64385axx

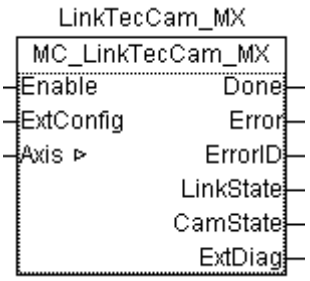
The IN buffer 5 is set to receive a cyclical object with ID 2 that is assigned to channel 0

of the IN-PDO mapper. With this identifier, if not configured otherwise, MOVI-PLC® sends the actual position of the virtual encoder.

When selecting "MX_CAM_RECEIVE_PDO_1" as master source in the "MC_SetCamConfig_MX" module, this object is set as master value for the Cam function.

3.5 The function modules of the MPLCTecCamMotion_MX library

3.5.1 MC_LinktecCam_MX



64386axx

Description:

The "MC_LinkTecCam_MX" function module is the logical interface between the axis and the "Cam" technology function. Once the ConnectAxis has established the cyclical communication to the axis and the activation has been performed, the individual technology functions of MOVIAXIS® are set-up. Only the technology functions required for Cam are activated

The PDO interface (IN-PDO 5 for receiving the master position via SBus) is also configured and the communication parameters for SBus synchronization are set.

In cyclical operation, "MC_LinkTecCam_MX" ensures the changeover from mathematical curves as soon as the axis is in sync (necessary to realize the offset travel option).

Additionally, various status signals are available as outputs.

Important:

The "MC_LinkTecCam_MX" module is absolutely essential for using the cam functionality. It must be called cyclically in the program.

If *LinkState* = NotLinked, none of the function modules of the "MPLCTecCamMotion_MX" can be executed, a corresponding fault message is issued.

Prerequisite:

MOVI-PLC® advanced in technology version T1 or higher.

ConnectAxis for corresponding axes is integrated in the cyclical program.

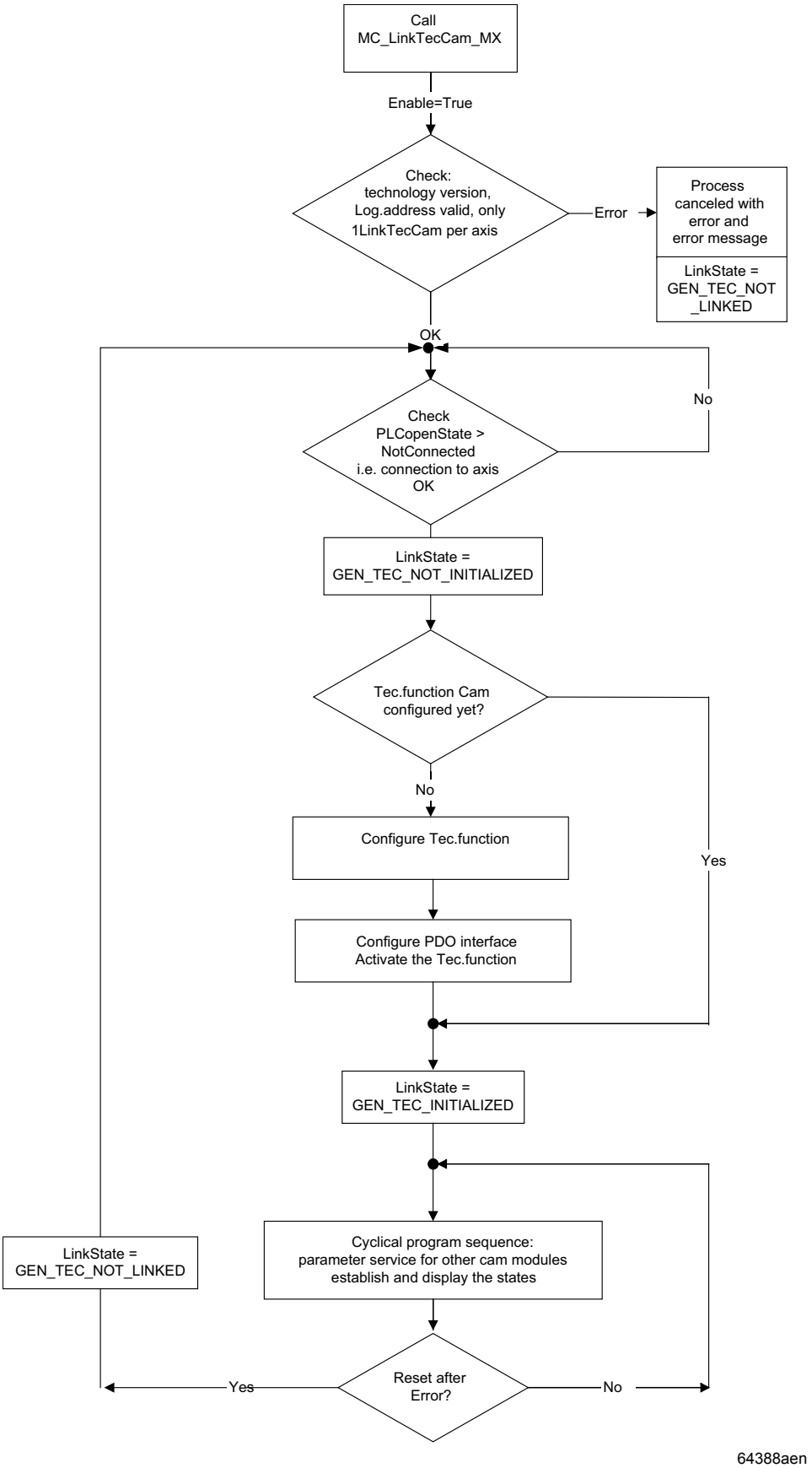
Inputs:

Name	Type	Meaning
Enable	BOOL	The module is processed as long as Enable „true“. Initialization is performed with a rising edge.
ExtConfig	MC_Cam_EXTENDED_CONFIG_MX	Extended configuration, see page 126
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

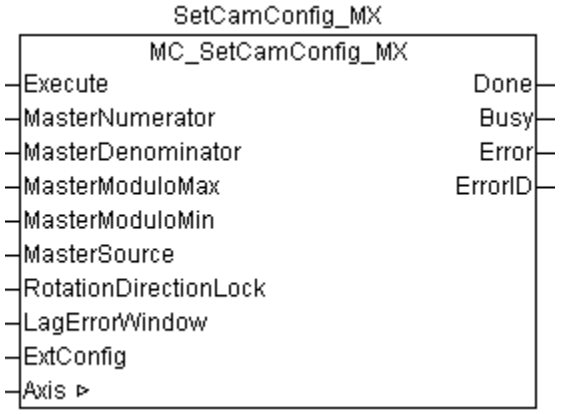
Outputs:

Name	Type	Meaning
Done	BOOL	Initialization successfully completed, connection with "Cam" technology function established.
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126
LinkState	MC_LINKTECSTATE	Current status: GEN_TEC_NOTLINKED GEN_TEC_NOTINITIALISED GEN_TEC_INITIALISED
CamState	MC_Cam_STATE_MX	Current status: MX_CAM_OUTCAM :=0, MX_CAM_INCAM:=1 MX_CAM_INCAM_ADJUST:=2, MX_CAM_INCAM_OFFSET:=3 MX_CAM_INCAM_ADJUST_OFFSET: =4 MX_CAM_MOVE_TO_MASTER: =5 MX_CAM_NOTLINKED:=6
ExtDiag	MC_Cam_EXTENDED_DIAG_MX	Diagnostics information CamFlow1ActCurve:UINT; Active curve number displayed in curve sequence 1 0 = Cam_Wait_MX curve activated 1 = Cam_1_MX curve activated 2 = Cam_2_MX curve activated 3 = Cam_3_MX curve activated 4 = Cam_4_MX curve activated 5 = Cam_5_MX curve activated 6 = Cam_6_MX curve activated 7 = Cam_7_MX curve activated 8 = Cam_8_MX curve activated 9 = Cam_9_MX curve activated 10 = Cam_10_MX curve activated 11 = Cam_1_MX transition activated 12 = Cam_2_MX transition activated 13 = Cam_3_MX transition activated 14 = Cam_4_MX transition activated 15 = Cam_5_MX transition activated 16 = Cam_6_MX transition activated 17 = Cam_7_MX transition activated 18 = Cam_8_MX transition activated 19 = Cam_9_MX transition activated 20 = Cam_10_MX transition activated CamFlow2ActCurve:UINT; Active curve number displayed in curve sequence 2 0 = Cam_Wait_MX activated 1 = Offset travel transition activated 2 = Cam_Wait_MX activated

MC_LinktecCam_MX: initialization



3.5.2 MC_SetCamConfig_MX



64389axx

Description:

This module sets basic parameters of the Cam function and transfers them to the respective address. The set values are adopted and sent with the rising edge on the Execute input.

The *MasterSource* input specifies the source of the master encoder signal. Here, you can choose between "real" encoders that are connected to the axis or "virtual" master signals sent via SBus. See section "Data types of the MPLCTecCamMotion_MX library", page 69 et seq.

The master signal can be scaled with parameters "MasterNumerator" and "MasterDenominator".

Example:

Signal of 65536 Inc per master cycle

- MasterNumerator = 62500
- MasterDenominator = 4096

i.e. one master cycle corresponds to 1000000 user units [mm].

The curves can be expressed using master units in mm.

If the master signal is transferred in modulo format, the *ModuloMax/ModuloMin* inputs should be set to the respective limit values of the master signal. This avoids setpoint jumps when the modulo range overflows. The setting of these values is irrelevant for a linear master signal.

At the *ExtConfig* input, a structure can be transferred for setting various filters. This can be used, for example, to filter a "noisy" master signal. These filters are set to suitable default values, which means that this input is usually not connected.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be GEN_TEC_NOTLINKED.

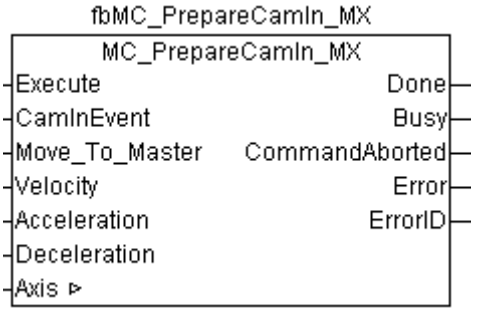
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module with a rising edge
MasterNumerator	DINT	Numerator value for scaling the master signal
MasterDenominator	UDINT	Denominator value for scaling the master signal
ModuloMax	DINT	Maximum modulo value of master signal
ModuloMin	DINT	Minimum modulo value of master signal
MasterSource	MC_CAM_MASTERSOURCE_MX	Source of the master position MX_CAM_SYSTEMPOS_ENCODER1 MX_CAM_SYSTEMPOS_ENCODER2 MX_CAM_SYSTEMPOS_ENCODER3 MX_CAM_MODULOPOS_ENCODER1 MX_CAM_MODULOPOS_ENCODER2 MX_CAM_MODULOPOS_ENCODER3 MX_CAM_MODULO_VIRTUAL_ENCODER_MX MX_CAM_LINEAR_VIRTUAL_ENCODER_MX MX_CAM_RECEIVE_PDO_1MX_CAM_RECEIVE_PDO_2
RotationDirectionLock	BYTE	Direction of rotation lock 0 = both directions enabled 1 = only CCW enabled 2 = only CW enabled
LagErrorWindow	UDINT	Lag error window in [inc]
ExtConfig	MC_CAM_EXTENDED_CONFIG_MX	Extended configuration: MSignal_InterpolationTime in [0.5 ms] MSignal_DelayTime in [µs] AverageFilterTime in [0.5 ms] CompFilterTime in [0.5 ms] DeathTimeCorr_On [on / off]
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	All parameters have been transferred successfully
Busy	BOOL	Parameters are being transferred
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126

3.5.3 MC_PrepareCamIn_MX



64390axx

Description:

You can use this module to activate the interrupt-controlled startup cycle and to switch the *MovetoMaster* function.

The dynamic parameters for the alignment process after a controller inhibit with the *MovetoMaster* function are written as well.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be GEN_TEC_NOTLINKED.

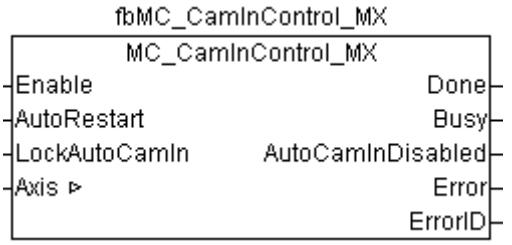
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
CamInEvent	MC_Cam_IN_SOURCE_MX	Possible startup cycle events: MX_CAM_DIRECT MX_CAM_INTERRUPT_DI01_POS_FLAG MX_CAM_INTERRUPT_DI01_NEG_FLAG MX_CAM_INTERRUPT_DI01 ... MX_CAM_INTERRUPT_DI08_POS_FLAG MX_CAM_INTERRUPT_DI08_NEG_FLAG MX_CAM_INTERRUPT_DI08 MX_CAM_INTERRUPT_C_TRACK_ENC1 MX_CAM_INTERRUPT_C_TRACK_ENC2 MX_CAM_INTERRUPT_C_TRACK_ENC3 Explanation: DIRECT = direct startup cycle POS_FLAG= positive edge NEG_FLAG= negative edge only the input = both edges DI01 of MOVIAXISDI01 = input DI02 of MOVIAXISDI02 = input DI03 of MOVIAXISDI03 = input DI04 of MOVIAXISDI04 = input DI05 of MOVIAXISDI05 = input DI06 of MOVIAXISDI06 = input DI07 of MOVIAXISDI07 = input DI08 of MOVIAXISDI08 = input C_TRACK_ENC1 = C track encoder 1 C_TRACK_ENC2 = C track encoder 2 C_TRACK_ENC3 = C track encoder 3
MovetoMaster	BOOL	True: Function switched on for MC_CamInDirekt_MX False: Function switched on for MC_CamInDirekt_MX
Velocity	DINT	Velocity [rpm] or user unit scaling
Acceleration	DINT	Ramp up time in ms relating to 3000 rpm /or user unit scaling
Deceleration	DINT	Ramp down time in ms relating to 3000 rpm /or user unit scaling
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	All parameters have been transferred successfully
Busy	BOOL	Parameters are being transferred
Command Aborted	BOOL	Module was aborted by another instance of a PrepareCamIn function module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126

3.5.4 MC_CamInControl_MX



64395axx

Description:

The "MC_CamInControl_MX" module offers additional control functions for the interrupt- or counter-controlled startup cycle events.

The *AutoRestart* input enables repeated startup cycles without repeated preparation using the "PrepareCamIn" function module. Interrupt-/counter-controlled startup cycles can be inhibited by setting the *LockAutoCamIn* input to 'True'. Startup cycles are also inhibited after an axis fault (Error stop). In this case, interrupt-/counter-controlled startup cycles must be activated again with the "PrepareCamIn" module.

The *AutoCamInDisabled* output displays whether startup cycles are disabled.

Important:

Only interrupt-controlled startup cycles can be inhibited, startup cycles using "MC_CamInDirect_MX" or "MC_CamInAdjust_MX" are still possible.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

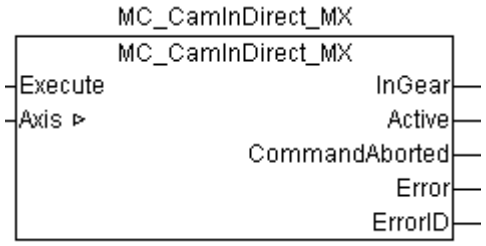
Inputs:

Name	Type	Meaning
Enable	BOOL	The module is processed as long as this input = True.
AutoRestart	BOOL	True: Automatic repeated startup cycles activated False: Startup cycles deactivated
LockAutoCamIn	BOOL	True: Startup cycle disabled False: Startup cycles enabled
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Required function set
Busy	BOOL	Parameters are being transferred
AutoCamInDisabled	BOOL	Startup cycle disabled
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126

3.5.5 MC_CamInDirect_MX



64398axx

Description:
A rising edge at the execute input causes a direct startup cycle.
The module interrupts "Move" modules that are still active.

Prerequisite:
"MC_LinkTecCam_MX" has the "Cam" technology function set-up.
LinkState must not be "GEN_TEC_NOTLINKED".
CamState = "MX_CAM_OUTCAM".

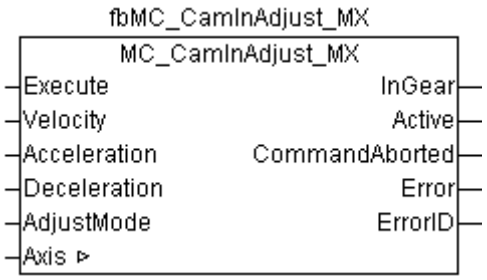
Inputs:

Name	Type	Meaning
Execute	BOOL	This input initiates the direct startup cycle
Axis	AXIS_REF	Logical address of the axis that is to be addressed by the function block.

Outputs:

Name	Type	Meaning
InGear	BOOL	Startup cycle successful, axis is in sync with the master on the cam
Active	BOOL	Axis is in the startup cycle state
Command Aborted	BOOL	Function module was interrupted by calling up another Move module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126

3.5.6 MC_CamInAdjust_MX



64400axx

Description:

If the selected adjust mode is MX_CAMINDIRECT, the cam is initialized and switched to curve operation FCB 16 in the current position in the event of a rising edge at the execute input.

If the selected adjust mode is MX_ADJUST, the cam is adjusted to the current curve position via the selected dynamic parameters and switched to curve operation FCB 16 in the current position in the event of a rising edge at the execute input. After a controller inhibit, an emergency off etc., the axis is aligned accordingly.

After a mains failure, the axis has to be moved back to the startup cycle position and engaged with the MX_CAMINDIRECT adjustmode, as it is no longer synchronized. For the next startup cycle, you can use the MX_Adjust adjust mode.

The module interrupts "Move" modules that are still active.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

CamState = "MX_CAM_OUTCAM".

The axis must be referenced.

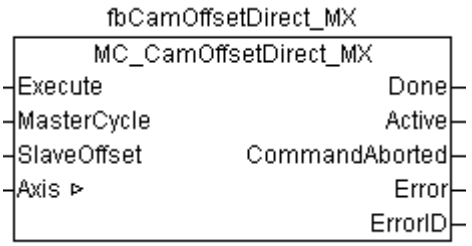
Inputs:

Name	Type	Meaning
Execute	BOOL	This input initiates the direct startup cycle
Velocity	DINT	Velocity [rpm] or user unit scaling
Acceleration	DINT	Ramp up time in ms relating to 3000 rpm or user unit scaling
Deceleration	DINT	Ramp down time in ms relating to 3000 rpm or user unit scaling
Adjust mode	MC_CAM_ADJUSTMODE_MX	MX_CAMINDIRECT (0) Master position of the cam is initialized with 0 (XMASTERPOSZERO) MX_ADJUST(1) Current master position of the cam (XMASTERPOSAKT)
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
InGear	BOOL	Startup cycle successful, axis is in sync with the master on the cam
Active	BOOL	Axis is in the startup cycle state
Command Aborted	BOOL	Function module was interrupted by calling up another Move module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126

3.5.7 MC_CamOffsetDirect_MX



64401axx

Description:

This module adds an offset travel to the active curve depending on the position.

The *CamState* is switched.

MX_Cam_InCam -> MX_Cam_InCam_Offset

MX_Cam_InCamAdjust -> MX_Cam_InCamAdjust_Offset

After successful offset, the axis switches back to the normal curve operation.

The *CamState* is reset to the original state.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

Axis is in curve operation.

CamState = MX_Cam_InCam or *CamState* = MX_Cam_InCamAdjust.

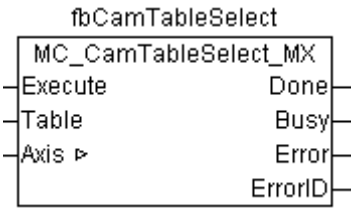
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
MasterCycle	DINT	Master distance for the offset travel (delta X)
SlaveOffset	DINT	Offset distance the slave is to travel in addition to the curve (delta Y)
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Offset was processed, axis is synchronous again
Active	BOOL	Offset processing ongoing.
Command Aborted	BOOL	Function module was interrupted by calling up another Move module
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 126

3.5.8 MC_CamTableSelect_MX



64415axx

Description

Use this module to select a curve. If another curve is activated, the switchover is performed at the end of the active curve.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

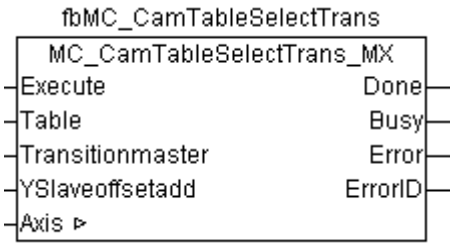
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
Table	MC_CAM_TABLENUMBER_MX	Cam selection Cam_Wait_MX (0), Wait curve selected CAM_1_MX (1), Curve 1 selected CAM_2_MX (2), Curve 2 selected CAM_3_MX (3), Curve 3 selected CAM_4_MX (4), Curve 4 selected CAM_5_MX (5), Curve 5 selected CAM_6_MX (6), Curve 6 selected CAM_7_MX (7), Curve 7 selected CAM_8_MX (8), Curve 8 selected CAM_9_MX (9), Curve 9 selected CAM_10_MX (10), Curve 10 selected
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Curve selected
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.9 MC_CamTableSelectTrans_MX



64416axx

Description:

Use this module to select a curve with a connected transition.

If another curve is activated, the switchover is performed at the end of the active curve. The *Transition master* input can be used to specify the master distance (delta X), and the *Yslaveoffsetadd* input is used to specify the slave distance. The movement is performed with the LPG1. A switchover is possible with the extended configuration, see *ExtendedCamConfig input*.

TecFunctionTransition - data type "MC_CAM_TRANSTYP_MX".

The dynamic parameters can be edited with the "MC_SetCamTransition Dynamics_MX" module.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

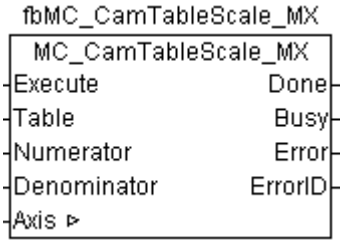
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
Table	MC_CAM_TABLENUMBER_MX	Cam selection Cam_Wait_MX (0), Wait curve selected CAM_1_MX (1), Curve 1 selected CAM_2_MX (2), Curve 2 selected CAM_3_MX (3), Curve 3 selected CAM_4_MX (4), Curve 4 selected CAM_5_MX (5), Curve 5 selected CAM_6_MX (6), Curve 6 selected CAM_7_MX (7), Curve 7 selected CAM_8_MX (8), Curve 8 selected CAM_9_MX (9), Curve 9 selected CAM_10_MX (10), Curve 10 selected
Transition master	DINT	Master distance of the transition function
YSlaveoffsetadd	DINT	Slave distance of the transition function
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Curve selected
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.10 MC_CamTableScale_MX



64417axx

Description:

You can use this module to compress a curve, i.e. denominator > numerator.

If you want to expand a curve, you can set the *ExtendedCamConfig.TecFunction-CamExpant* bit in the extended configuration to TRUE. This allows you to expand the curve.

However, using this function requires the *SlaveShiftFactor* to be limited or the curves to be specified accordingly.

Recommendation: **Do only compress curves**.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

The selected curve must not be activated.

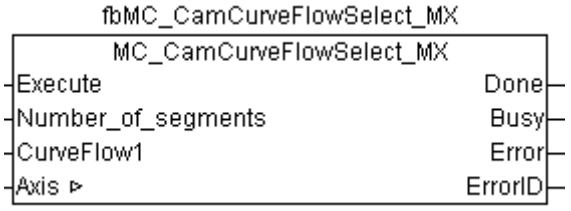
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
Table	MC_CAM_TABLENUMBER_MX	Cam selection Cam_Wait_MX (0), Wait curve selected CAM_1_MX (1), Curve 1 selected CAM_2_MX (2), Curve 2 selected CAM_3_MX (3), Curve 3 selected CAM_4_MX (4), Curve 4 selected CAM_5_MX (5), Curve 5 selected CAM_6_MX (6), Curve 6 selected CAM_7_MX (7), Curve 7 selected CAM_8_MX (8), Curve 8 selected CAM_9_MX (9), Curve 9 selected CAM_10_MX (10), Curve 10 selected
Numerator	DINT	Numerator value for scaling the curve
Denominator	UDINT	Denominator value for scaling the curve
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.11 MC_CamCurveFlowSelect_MX



64423axx

Description:

You can use this module to parameterize a curve sequence with up to 10 curves. Each curve can be followed by a transition.

The *CurveFlow1* input determines the sequence.

The parameters are transferred in an array. This array can be transferred with an initialization module.

Example:

```
FUNCTION_BLOCK InitCamFlow
VAR_INPUT
    Execute:BOOL; (* select the curve flo`*)
    default: BOOL;(*select all curve standalone*)
END_VAR
VAR_OUTPUT
    numberofsegments:UINT;
    CurveFlow1:MC_CAM_CURVE_FLOW_MX;
END_VAR

EXE:=Execute AND NOT bREdgeexecute;
bREdgeexecute:=Execute;
IF EXE  THEN
    numberofsegments:=10;
    (*curve segment  1 first curve*)
    CurveFlow1.Camsegment[1].Curvenumber :=1;
    CurveFlow1.Camsegment[1].Nextcurve :=2;
    CurveFlow1.Camsegment[1].Transitionmaster :=5000;
    CurveFlow1.Camsegment[1].YSlaveoffsetadd :=5000;
    (*curve segment  2 *)
    CurveFlow1.Camsegment[2].Curvenumber :=2;
    CurveFlow1.Camsegment[2].Nextcurve :=3;
    CurveFlow1.Camsegment[2].Transitionmaster :=0;
    CurveFlow1.Camsegment[2].YSlaveoffsetadd :=0;
    (*curve segment  3 *)
    CurveFlow1.Camsegment[3].Curvenumber :=3;
```

```
    CurveFlow1.Camsegment[3].Nextcurve :=4;
    CurveFlow1.Camsegment[3].Transitionmaster :=10000;
    CurveFlow1.Camsegment[3].YSlaveoffsetadd :=5000;
    (*curve segment  4 *)
    CurveFlow1.Camsegment[4].Curvenumber :=4;
    CurveFlow1.Camsegment[4].Nextcurve :=5;
    CurveFlow1.Camsegment[4].Transitionmaster :=10000;
    CurveFlow1.Camsegment[4].YSlaveoffsetadd :=5000;
    (*curve segment  5 *)
    CurveFlow1.Camsegment[5].Curvenumber :=5;
    CurveFlow1.Camsegment[5].Nextcurve :=6;
    CurveFlow1.Camsegment[5].Transitionmaster :=10000;
    CurveFlow1.Camsegment[5].YSlaveoffsetadd :=5000;
    (*curve segment  6 *)
    CurveFlow1.Camsegment[6].Curvenumber :=6;
    CurveFlow1.Camsegment[6].Nextcurve :=7;
    CurveFlow1.Camsegment[6].Transitionmaster :=10000;
    CurveFlow1.Camsegment[6].YSlaveoffsetadd :=5000;
    (*curve segment  7 *)
    CurveFlow1.Camsegment[7].Curvenumber :=7;
    CurveFlow1.Camsegment[7].Nextcurve :=8;
    CurveFlow1.Camsegment[7].Transitionmaster :=10000;
    CurveFlow1.Camsegment[7].YSlaveoffsetadd :=5000;
    (*curve segment  8 *)
    CurveFlow1.Camsegment[8].Curvenumber :=8;
    CurveFlow1.Camsegment[8].Nextcurve :=9;
    CurveFlow1.Camsegment[8].Transitionmaster :=10000;
    CurveFlow1.Camsegment[8].YSlaveoffsetadd :=5000;
    (*curve segment  9 *)
    CurveFlow1.Camsegment[9].Curvenumber :=9;
    CurveFlow1.Camsegment[9].Nextcurve :=10;
    CurveFlow1.Camsegment[9].Transitionmaster :=10000;
    CurveFlow1.Camsegment[9].YSlaveoffsetadd :=5000;
    (*curve segment  10  last curve *)
    CurveFlow1.Camsegment[10].Curvenumber :=10;
    CurveFlow1.Camsegment[10].Nextcurve :=1;
    CurveFlow1.Camsegment[10].Transitionmaster :=10000;
    CurveFlow1.Camsegment[10].YSlaveoffsetadd :=5000;
END_IF
```

See data type "MC_CAM_CURVE_FLOW_MX" and
"MC_CAM_CFLOW_MX"

The following values are specified for each segment:

- the curve number
- the next curve number
- the master distance of the transition
- the slave distance of the transition to be travelled

If 0 is entered for *Transitionmaster* and *Yslaveoffsetadd*, there is no transition added automatically.

The *Transition master* input can be used to specify the master distance (delta X), and the *Yslaveoffsetadd* input is used to specify the slave distance. The movement is performed with the LPG1.

A switchover is possible with the extended configuration, see input *ExtendedCamConfig*.*TecFunktionTransition* Datatype "MC_CAM_TRANSTYP_MX".

The dynamic parameters can be edited with the "MC_SetCamTransition Dynamics_MX" module.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

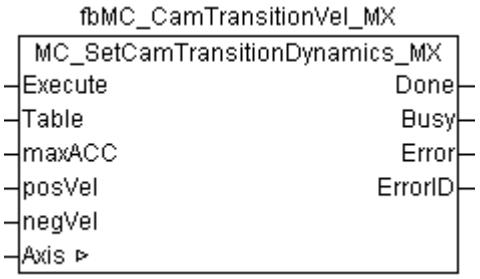
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
Number_of_segments	UINT	Number of curves that are interlinked
CurveFlow1	MC_CAM_CURVE_FLOW_MX (Camsegment: ARRAY[1..10] OF MC_CAM_CFLOW_MX;)	Curve sequence selection Camsegment[1].Curvenumber, curve number- Camsegment[1].Nextcurve, next curve number Camsegment[1].Transitionmaster , master distance of the tr. Camsegment[1].YSlaveoffsetadd , slave distance of the tr .... Camsegment[10].Curvenumber Camsegment[10].Nextcurve Camsegment[10].Transitionmaster Camsegment[10].YSlaveoffsetadd
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.12 MC_SetCamTransitionDynamics_MX



64427axx

Description:

You can use this block to edit the dynamic parameters of a transition. The transitions are assigned curves 1 10, this is why they are selected via this reference.

See *ExtDiag* of "MC_LinkTecCam_MX".

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

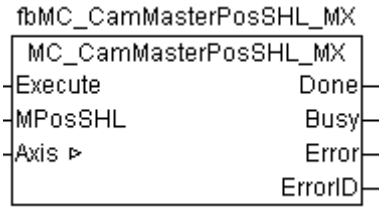
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates offset travel
Table	MC_CAM_TABLENUMBER_MX	Curve selection Cam_Wait_MX (0), wait curves selected CAM_1_MX (1), curve 1 selected CAM_2_MX (2), curve 2 selected CAM_3_MX (3), curve 3 selected CAM_4_MX (4), curve 4 selected CAM_5_MX (5), curve 5 selected CAM_6_MX (6), curve 6 selected CAM_7_MX (7), curve 7 selected CAM_8_MX (8), curve 8 selected CAM_9_MX (9), curve 9 selected CAM_10_MX (10), curve 10 selected
maxACC	REAL	maximum acceleration/deceleration for LPG1/LPG2 Default = 0.000 000 001
posVel	REAL	maximum negative velocity for LPG1Default = 1.0
negVel	REAL	maximum negative velocity for LPG1Default = 1.0
Axis	AXIS_REF	Logical address of the axis that is to be addressed by the function block.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.13 MC_CamMasterPosSHL_MX



64429axx

Description:

Use this block to scale up the master position in order to gain a better resolution. Thus the axis can interpolate better, i.e. the curve operation becomes smoother.

The master cycle lengths of the curves are also scaled up automatically by factor $2^{MPosSHL}$. The factor is automatically limited regarding the largest master cycle of the curves. Display of the internal variables *SEWTecCamMotoion* [Axis number].*ExtDiag.MasterCycle*.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates the parameter changes
MPosSHL	UINT	The master position is shifted to the left by the value i.e. MPosSHL = 10 master position $\times 2^{10}$ = master position $\times$ 1024
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.14 MC_SetMasterCycle_MX

Description:

You can use this block to change the master cycle.

If the selected curve is active, it is completed with the old master cycle, the new master cycle is active with the next run. The master cycle is automatically scaled up by the *MasterShiftFactor*.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

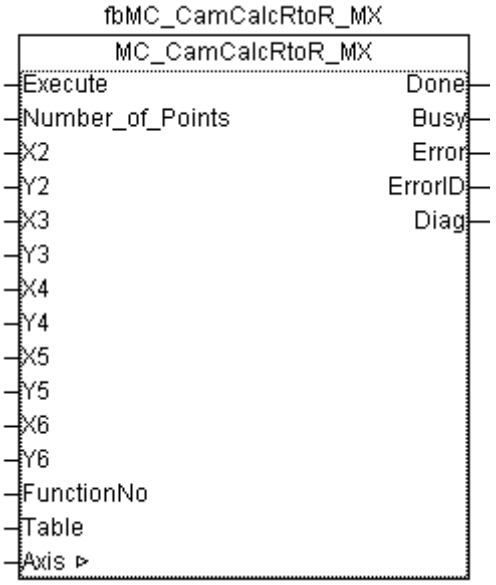
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates the parameter changes
Table	MC_CAM_TABLENUMBER_MX	Cam selectionCam_Wait_MX (0), Wait curve selectedCAM_1_MX (1), Curve 1 selectedCAM_2_MX (2), Curve 2 selectedCAM_3_MX (3), Curve 3 selectedCAM_4_MX (4), Curve 4 selectedCAM_5_MX (5), Curve 5 selectedCAM_6_MX (6), Curve 6 selectedCAM_7_MX (7), Curve 7 selectedCAM_8_MX (8), Curve 8 selectedCAM_9_MX (9), Curve 9 selectedCAM_10_MX (10), Curve 10 selected
Mastercycle	DINT	New master cycle of the selected curve
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.15 MC_CamCalcRtoR_MX



64430axx

Description:

Calculation of curves based on 2 6 coordinate points. The curve always begins at 0 / 0 (X1 / Y1).

- The calculation is always performed section-by-section.
- The master cycle corresponds to the biggest master position e.g. points = 5 → master cycle = X5.
- In order to gain a better resolution, the master cycle is scaled up according to the master shift factor, e.g. master cycle = 10000 → master shift factor = 6. 10000 $\times 2^6$ = 64000 is written in MOVIAXIS®. The master shift factor can be adjusted with the "MC_CamMasterPosSHL_MX" block.
- Position scaling is adjusted according to the highest y-value. The slave coordinates (Y2-Y6) are multiplied by factor $2^{\text{Shift factor}}$. Example: Y4 = 4096, shift factor = 5 → Y4_intern = 4096 $\times 2^5$ = 131072. The shift factor is entered in the CamFlow1 of the respective curve. The slave shift factor can be limited with the *ExtConfig.Calc_SlaveShift_max* variable, (initial value:10).
- The minimum and maximum velocity in the curve is output.
- The minimum and maximum acceleration is output.
- The curve is loaded according to the table selected in MOVIAXIS®. The block has an internal array of 512 control points in accordance with the MOVIDRIVE® configuration.

Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the block.
Number_of_Points	UINT	Number of coordinate control points
X2	DINT	2. control point - master position
Y2	DINT	2. control point - slave position
X3	DINT	3. control point - master position
Y3	DINT	3. control point - slave position
X4	DINT	4. control point - master position
Y4	DINT	4. control point - slave position
X5	DINT	5. control point - master position
Y5	DINT	5. control point - slave position
X6	DINT	6. control point - master position
Y6	DINT	6. control point - slave position
FunctionNo	MC_CAM_CALC_FUNCTIONNO_MX	0 = Straight line1 = 5th order polynomial2 = Sine3 = 3rd order polynomial4 = Sinoids from Bestehorn5 = Modified acceleration trapezoid
Table	MC_CAM_TABLENUMBER_MX	Curve selection 1-10 CAM_1_MX CAM_2_MX CAM_3_MX CAM_4_MX CAM_5_MX CAM_6_MX CAM_7_MX CAM_8_MX CAM_9_MX CAM_10_MX
Axis	AXIS_REF	This input specifies the motor axis on which the actions of the function module are to be executed.

Outputs:

Name		Type	Meaning
Done		BOOL	Calculation, curve and parameter download finished
Busy		BOOL	Processing of the block not completed yet.
Error		BOOL	Fault in module during execution
ErrorID		UDINT	see page 126
DIAG	Vmax	DINT	maximum "velocity" V(i)=Y(i+1)-Y(i) within the cam.
	Vmin	DINT	minimum "velocity" V(i)=Y(i+1)-Y(i) within the cam.
	AbsVmax	DINT	maximum value of the "velocity" V(i)=Y(i+1)-Y(i) Y(i) within the cam.
	Amax	DINT	maximum "acceleration" A(i) = V(i)-V(i-1) within the cam.
	Amin	DINT	minimum "acceleration" A(i) = V(i)-V(i -1) within the cam.
	AbsAmax	DINT	maximum "acceleration" value A(i) = V(i)-V(i-1) within the cam.
	Mastercycle	DINT	Master cycle
	Shift factor	DINT	Slave scaling $\times 2^n$
	State	BYTE	Processing status
	MasterShiftTab	DINT	not used

Functional principles of the block::

The block calculates a curve based on the coordinates (X = Master / Y = Slave).

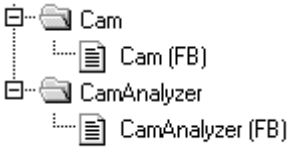
The curve is made up of a number of sequences. The transitions of the sequences are section-by-section. The sequences are calculated via the "CAM" block. For calculation and download purposes the module requires several program cycles (library 11_System\MPLCTechFct_Cam.lib).

The table points are stored in an array. Then the minimum and the maximum velocity and acceleration is determined via the "CamAnalyser" module.

You can use parameter "MC_WriteParameter_MX" to write the 512 control points into the MOVIAXIS®. With a cycle time of 3 ms, the block requires about 3.5 s. If everything was successful, the *Done Bit* is set to TRUE.

Internally used blocks:

11_System\MPLCTechFct_Cam.lib



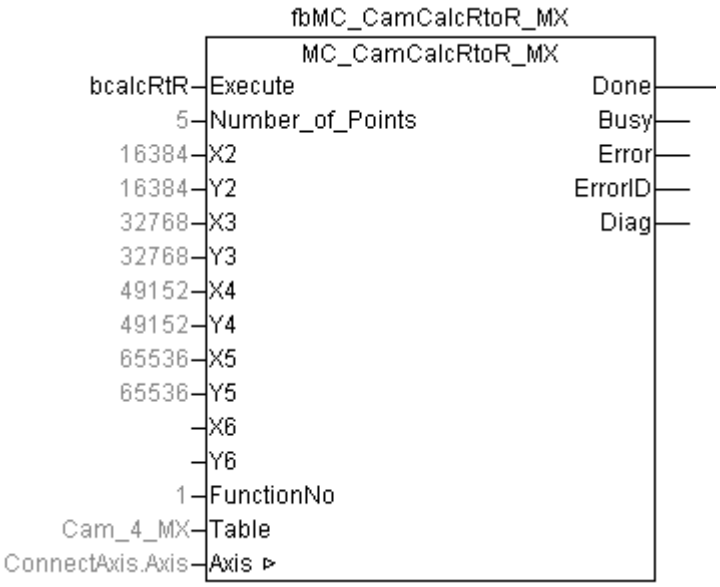
64431axx

Please note:

- The user must make sure that the curve to be calculated is not selected or that the drive is in standstill.
- You can use one of the following blocks to activate the corresponding curve table in the MX: "MC_CamTableSelect_MX", "MC_CamTableSelectTrans_MX", "MC_CamCurveFlowSelect_MX".
- Use the cam editor to download the table from the MOVIAXIS®.

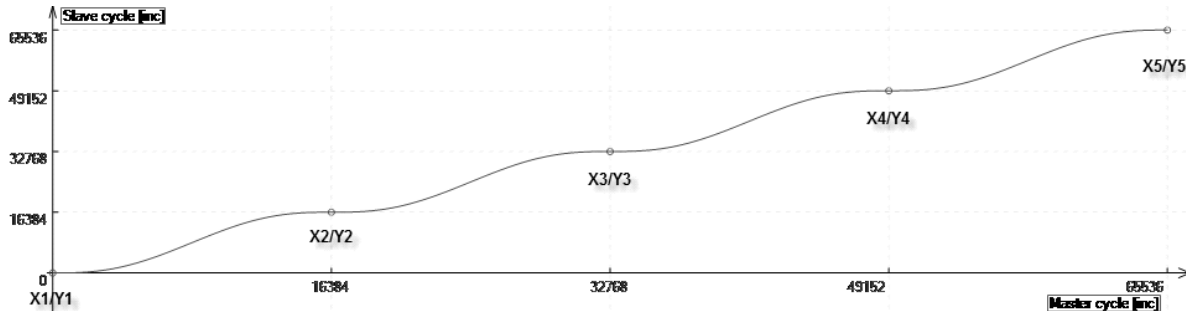
Sample curve *Number_of_Points* = 5.

Module call:



64432axx

Curve that results from control point table:



64433axx

Sequence of motion:

The drive moves one revolution before the stop position at Y2 (Vm moves from X1–X2).

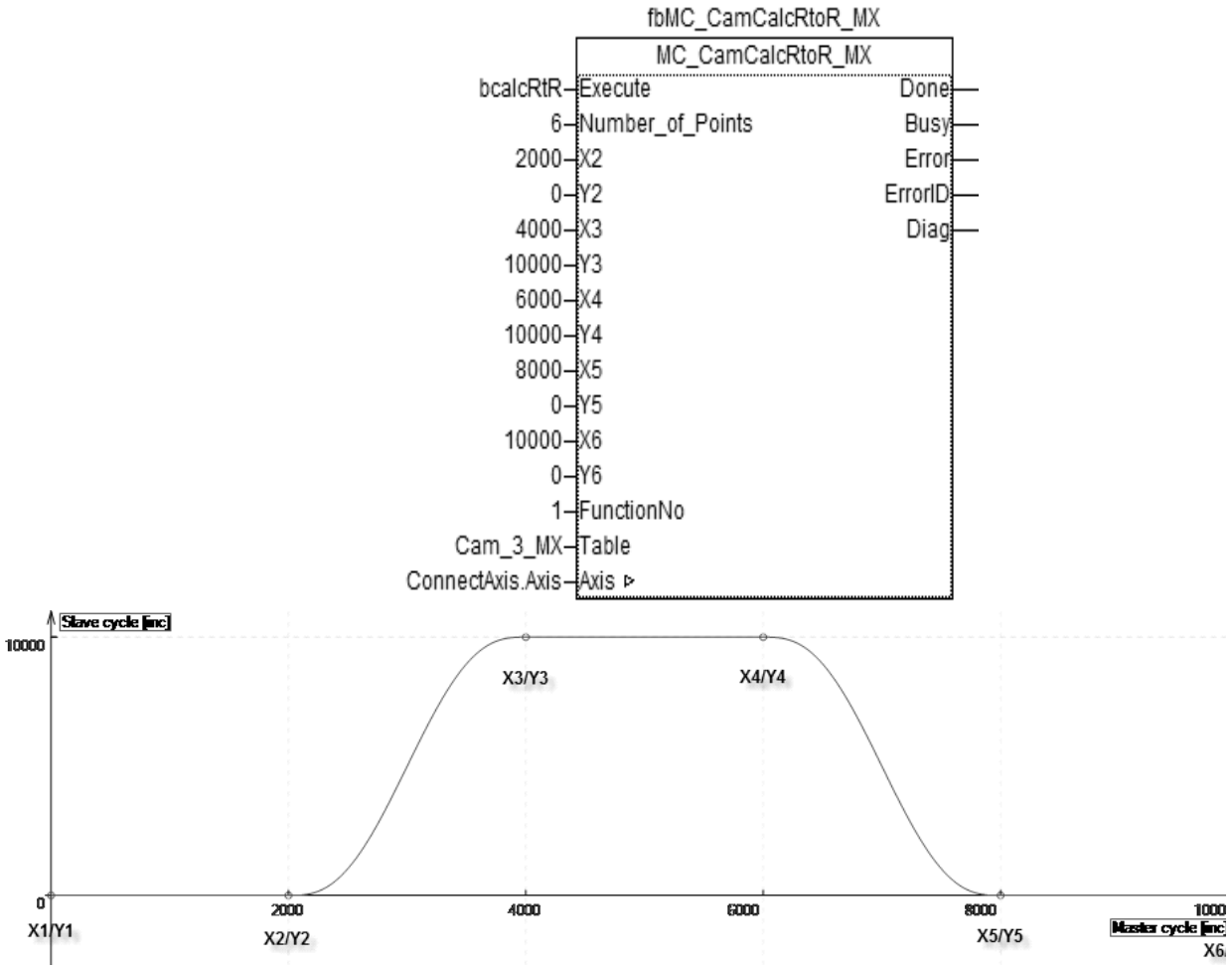
The drive moves one revolution before stop position at Y3 (Vm moves from X2–X3). The drive moves one revolution before stop position at Y4 (Vm moves from X3–X4).

The drive moves one revolution before the stop position at Y5 (Vm moves from X4–X5).

The curve begins again at X1.

Sample curve *Points* = 6.

Module call and curve that results from control point table:



64435axx

Sequence of motion:

The drive stands in section 1 (virtual master moves from X1–X2)

The drive moves forward by one revolution (Vm moves from X2–X3)

The drive stands in section 3 (position Y3 = Y4/Vm moves from X3–X4)

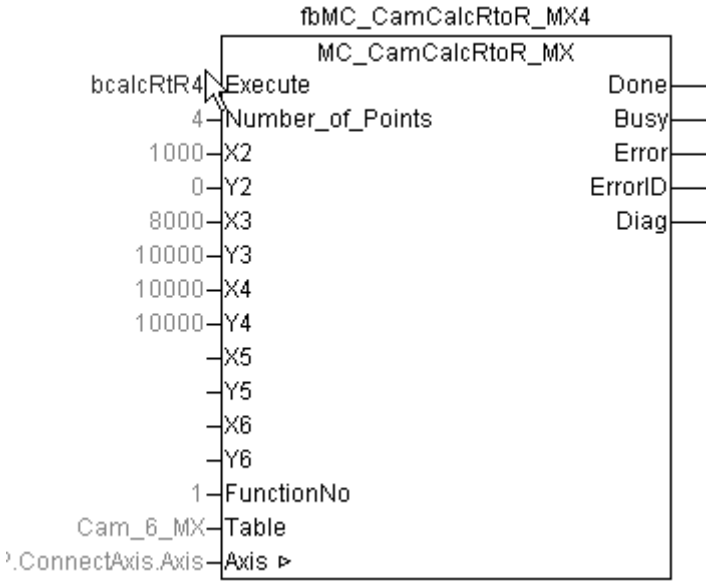
The drive moves backward by one revolution. (Vm moves from X4–X5)

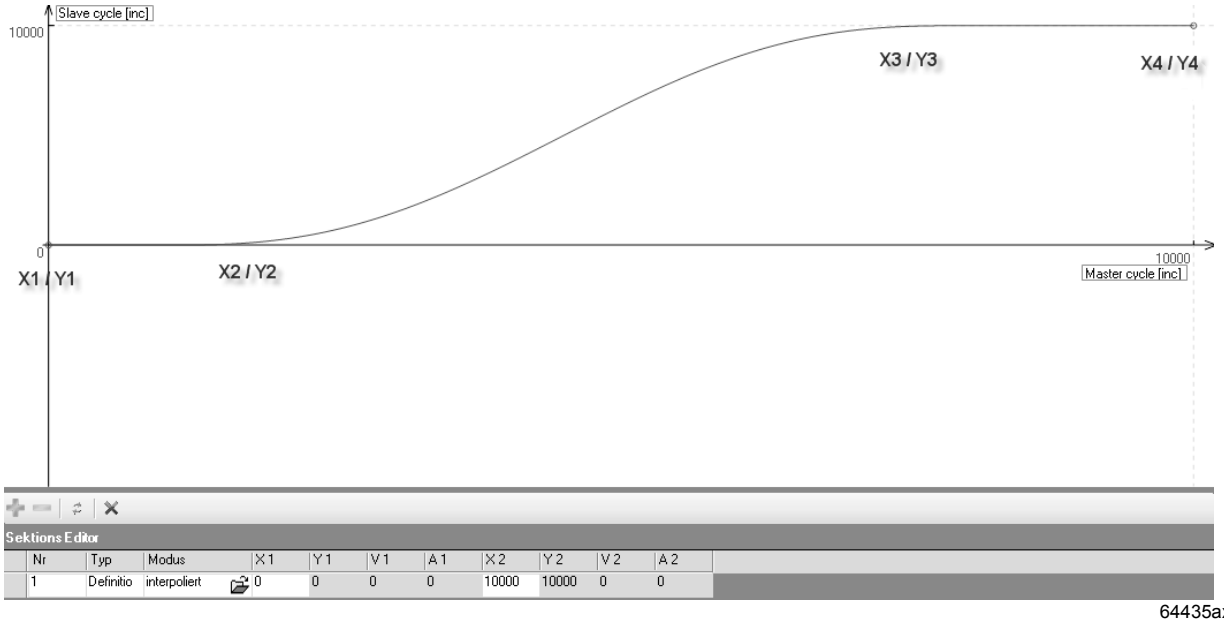
The drive stands in section 5 (Y6 = Y1/Vm moves from X5–X6)

The curve begins again at X1/Y1.

Sample curve *Points* = 4

Module call:





64435axx

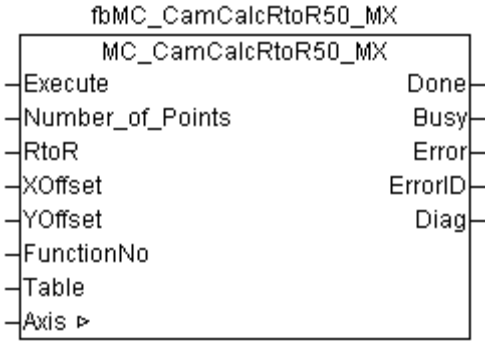
The drive stands in section 1 (virtual master moves from X1–X2)

The drive moves forward by one revolution (Vm moves from X2–X3)

The drive stands in section 3 (position Y3 = Y4/ Vm moves from X3–X4)

The curve begins again at X1.

3.5.16 MC_CamCalcRtoR50_MX



644336axx

Description:

You can use this block to parameterize a curve with up to 50 stop points. Each curve can be followed by a transition.

You can use the *RtoR* input to transfer the coordinates (X/Y) in an array. This array can be transferred with an initialization module.

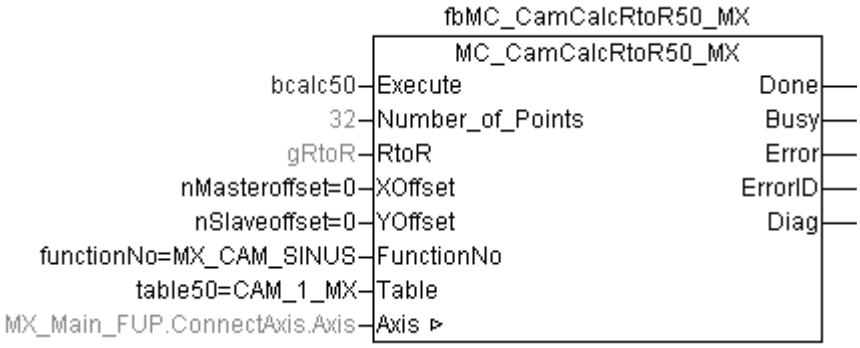
The X values are master values, the Y values are slave values.

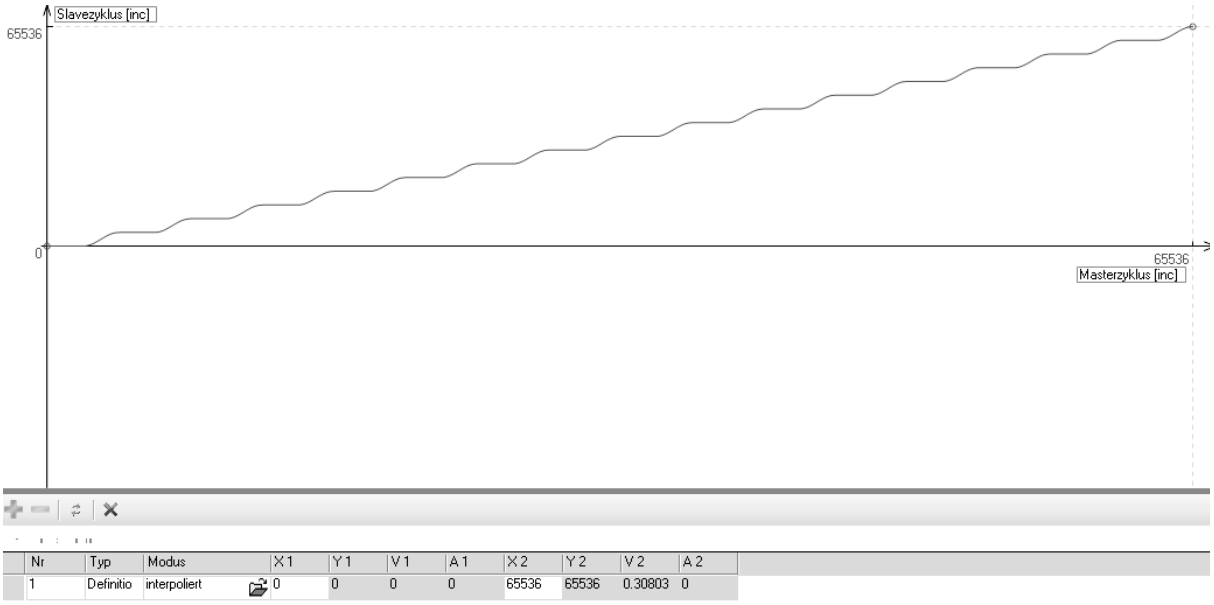
Example:

```
FUNCTION_BLOCK InitRtR50
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    bDone: BOOL;
END_VAR
IF NOT bDone THEN
     gRtoR.Point[1].X:= 2048;
    gRtoR.Point[1].Y := 0 × 4096;
      gRtoR.Point[2].X:=2 × 2048;
    gRtoR.Point[2].Y := 1 × 4096;
    gRtoR.Point[3].X := 3 × 2048;
    gRtoR.Point[3].Y := 1 × 4096;
    gRtoR.Point[4].X := 4 × 2048;
    gRtoR.Point[4].Y := 2 × 4096;
    gRtoR.Point[5].X := 5 × 2048;
    gRtoR.Point[5].Y := 2 × 4096;
    gRtoR.Point[6].X := 6 × 2048;
    gRtoR.Point[6].Y := 3 × 4096;
    gRtoR.Point[7].X := 7 × 2048;
    gRtoR.Point[7].Y := 3 × 4096;
```

```
gRtoR.Point[8].X:= 8 × 2048;
gRtoR.Point[8].Y := 4 × 4096;
gRtoR.Point[9].X := 9 × 2048;
gRtoR.Point[9].Y := 4 × 4096;
gRtoR.Point[10].X := 10 × 2048;
gRtoR.Point[10].Y := 5 × 4096;
gRtoR.Point[11].X := 11 × 2048;
gRtoR.Point[11].Y := 5 × 4096;
gRtoR.Point[12].X := 12 × 2048;
gRtoR.Point[12].Y := 6 × 4096;
gRtoR.Point[13].X := 13 × 2048;
gRtoR.Point[13].Y := 6 × 4096;
gRtoR.Point[14].X := 14 × 2048;
gRtoR.Point[14].Y := 7 × 4096;
gRtoR.Point[15].X := 15 × 2048;
gRtoR.Point[15].Y := 7 × 4096;
gRtoR.Point[16].X := 16 × 2048;
gRtoR.Point[16].Y := 8 × 4096;
gRtoR.Point[17].X := 17 × 2048;
gRtoR.Point[17].Y := 8 × 4096;
gRtoR.Point[18].X := 18 × 2048;
gRtoR.Point[18].Y := 9 × 4096;
gRtoR.Point[19].X := 19 × 2048;
gRtoR.Point[19].Y :=9 × 4096;
gRtoR.Point[20].X := 20 × 2048;
gRtoR.Point[20].Y := 10 × 4096;
gRtoR.Point[21].X := 21 × 2048;
gRtoR.Point[21].Y := 10 × 4096;
gRtoR.Point[22].X := 22 × 2048;
gRtoR.Point[22].Y := 11 × 4096;
gRtoR.Point[23].X := 23 × 2048;
gRtoR.Point[23].Y := 11 × 4096;
gRtoR.Point[24].X:= 24 × 2048;
gRtoR.Point[24].Y := 12 × 4096;
gRtoR.Point[25].X := 25 × 2048;
gRtoR.Point[25].Y := 12 × 4096;
gRtoR.Point[26].X := 26 × 2048;
gRtoR.Point[26].Y := 13 × 4096;
gRtoR.Point[27].X := 27 × 2048;
gRtoR.Point[27].Y := 13 × 4096;
gRtoR.Point[28].X := 28 × 2048;
```

```
    gRtoR.Point[28].Y := 14 × 4096;
    gRtoR.Point[29].X := 29 × 2048;
    gRtoR.Point[29].Y := 14 × 4096;
    gRtoR.Point[30].X := 30 × 2048;
    gRtoR.Point[30].Y := 15 × 4096;
    gRtoR.Point[31].X := 31 × 2048;
    gRtoR.Point[31].Y := 15 × 4096;
    gRtoR.Point[32].X := 32 × 2048;
    gRtoR.Point[32].Y := 16 × 4096;
    bDone := TRUE;
END_IF
```





64437axx

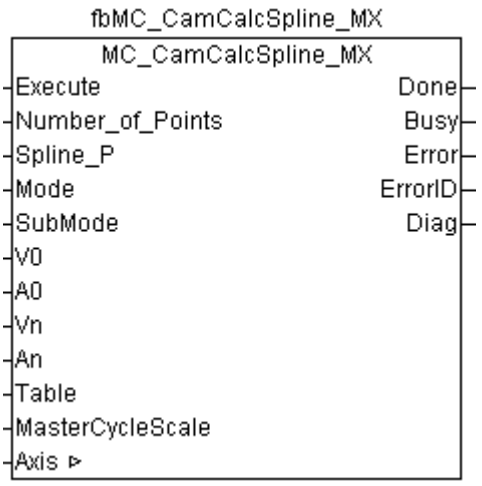
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
Number_of_Points	UINT	Number of coordinate control points
RtoR	MC_Cam_RtoR_MX(Point: ARRAY[1..50] OF MC_CAM_Point_MX) (X: DINT masterpoints Y: DINT slavepoints)	The coordinates (X/Y) are transferred in an array. The X value is for the master and the Y value for the slave.
XOffset	DINT	Cam shifted towards master
YOffset	DINT	Cam shifted towards slave
FunctionNo	MC_CAM_CALC_FUNCTIONNO_MX	0 = Straight line 1 = 5th order polynomial 2 = Sine 3 = 3rd order polynomial 4 = Sinoids from Bestehorn 5 = Modifed acceleration trapezoid
Table	MC_CAM_TABLENUMBER_MX	Curve selection 1-10 CAM_1_MX CAM_2_MX CAM_3_MX CAM_4_MX CAM_5_MX CAM_6_MX CAM_7_MX CAM_8_MX CAM_9_MX CAM_10_MX
Axis	AXIS_REF	This input specifies the motor axis on which the actions of the function module are to be executed.

Outputs:

Name		Type	Meaning
Done		BOOL	Calculation, curve and parameter download finished
Busy		BOOL	Processing of the block not completed yet.
Error		BOOL	Fault in module during execution
ErrorID		UDINT	see page 126
DIAG	Vmax	DINT	maximum "velocity" V(i)=Y(i+1)-Y(i) within the cam.
	Vmin	DINT	minimum "velocity" V(i)=Y(i+1)-Y(i) within the cam.
	AbsVmax	DINT	maximum value of the "velocity" V(i)=Y(i+1)-Y(i) Y(i) within the cam.
	Amax	DINT	maximum "acceleration" A(i) = V(i)-V(i-1) within the cam.
	Amin	DINT	minimum "acceleration" A(i) = V(i)-V(i -1) within the cam.
	AbsAmax	DINT	maximum "acceleration" value A(i) = V(i)-V(i-1) within the cam.
	Mastercycle	DINT	Master cycle
	Shift factor	DINT	Slave scaling $\times 2^n$
	State	BYTE	Processing status
	MasterShiftTab	DINT	Table control point was shifted by

3.5.17 MC_CamCalcSpline_MX



64445axx

Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
Number_of_Points	UINT	Number of coordinate control points
Spline_P	ARRAY[0..29] OF Spline_Point(X: INT masterpoints Y: DINT slavepoints)	The coordinates (X/Y) are transferred in an array. The X value is for the master, the table control point number (1-512) is always specified. The Y value corresponds to the slave position.
Mode	MC_CAM_CALC_SPLINEMODE_MX	MX_CAM_Spline_0 = Spline-Interpolation MX_CAM_Spline_B = Approximation MX_CAM_Spline_1 = fast, piecewise Interpolation
SubMode	UINT	Mode Control only used if Mode MX_CAM_Spline_B Submode 0 - Polynom 3 Submode 1- not used Submode 2 - Polynom 2 Submode 3 - Polynom 3 Submode 4 - Polynom 4 Submode 5 - Polynom 5 Submode 6 - Polynom 6 Submode 7 - Polynom 7
V0	REAL	Velocity in the start point
A0	REAL	Acceleration in the start point
Vn	REAL	Velocity in the end point
To	REAL	Acceleration in the end point
Table	MC_CAM_TABLENUMBER_MX	Curve selection 1-10 CAM_1_MX CAM_2_MX CAM_3_MX CAM_4_MX CAM_5_MX CAM_6_MX CAM_7_MX CAM_8_MX CAM_9_MX CAM_10_MX
MasterCycleScale	DINT	Master cycle of the curve
Axis	AXIS_REF	This input specifies the motor axis on which the actions of the function module are to be executed.

Outputs:

Name		Type	Meaning
Done		BOOL	Calculation, curve and parameter download finished
Busy		BOOL	Processing of the block not completed yet.
Error		BOOL	Fault in module during execution
ErrorID		UDINT	see page 126
DIAG	Vmax	DINT	maximum "velocity" V(i) = Y(i+1)-Y(i) within the cam.
	Vmin	DINT	minimum "velocity" V(i) = Y(i+1)-Y(i) within the cam.
	AbsVmax	DINT	maximum value of the "velocity" V(i) = Y(i+1)-Y(i) Y(i) within the cam.
	Amax	DINT	maximum "acceleration" A(i) = V(i)-V(i-1) within the cam.
	Amin	DINT	minimum "acceleration" A(i) = V(i)-V(i -1) within the cam.
	AbsAmax	DINT	maximum "acceleration" value A(i) = V(i)-V(i-1) within the cam.
	Mastercycle	DINT	Master cycle
	Shift factor	DINT	Slave scaling $\times 2^n$
	State	BYTE	Processing status
	MasterShiftTab	DINT	Not used

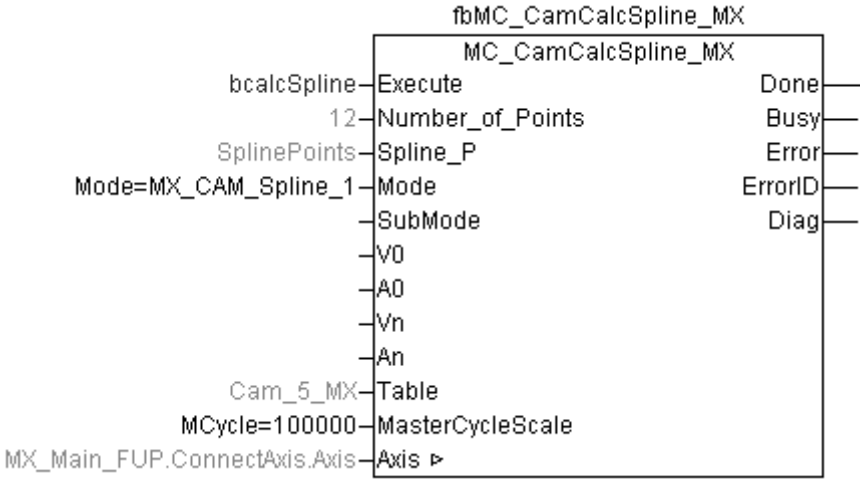
Example:

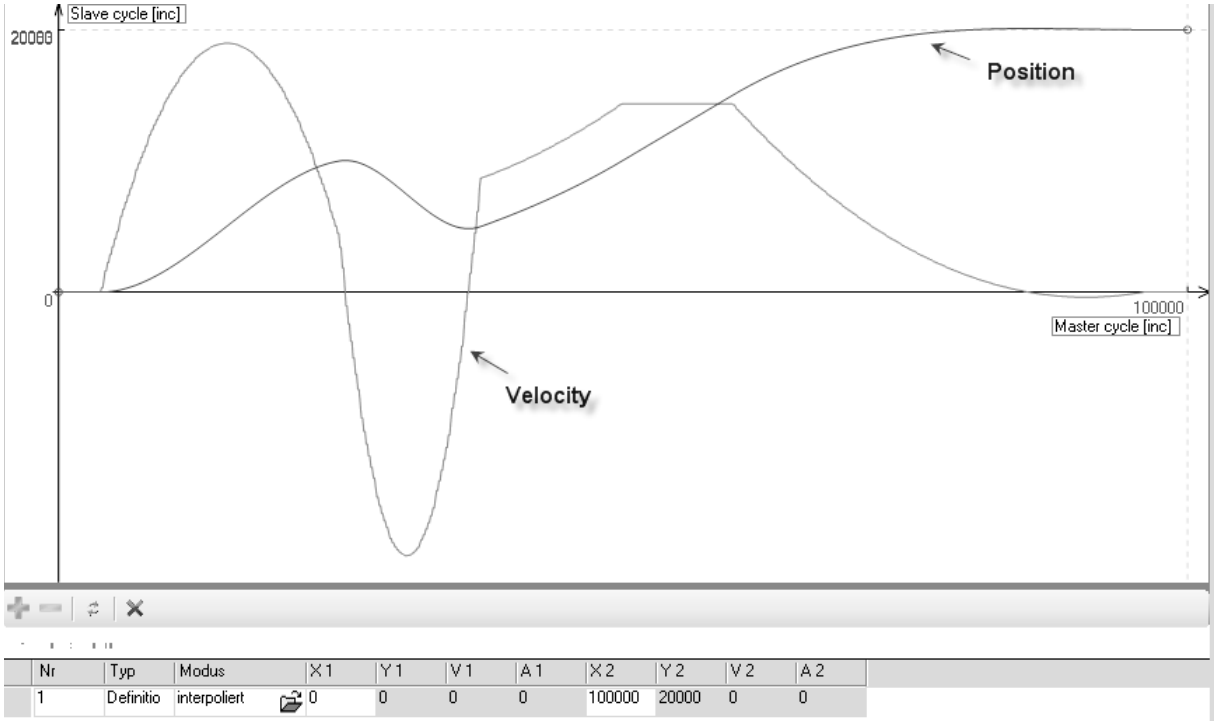
SplinePoints: ARRAY [0..29] OF SPLINE_POINT :=

(x:=1, y:=0), (x:=10, y:=0),(x:=20, y:=0), (x:=128, y:=10000), (x:=192, y:=5000), (x:=256, y:=10000), (x:=266, y:=11000), (x:=286, y:=13000), (x:=306, y:=15000), (x:=492, y:=20000), (x:=502, y:=20000), (x:=512, y:=20000);

X corresponds to the master table control point number (1 – 512)Y corresponds to the slave position

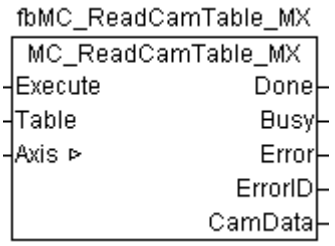
Mode: MC_CAM_CALC_SPLINEMODE_MX:=2;MCycle: DINT:=100000;


fbMC_CamCalcSpline_MX
MC_CamCalcSpline_MX
bcalcSpline
Execute
Done
12
Number_of_Points
Busy
SplinePoints
Spline_P
Error
Mode=MX_CAM_Spline_1
Mode
ErrorID
SubMode
Diag
V0
A0
Vn
An
Cam_5_MX
Table
MCycle=100000
MasterCycleScale
MX_Main_FUP.ConnectAxis.Axis
Axis


Slave cycle [inc]
20000
0
Position
Velocity
100000
Master cycle [inc]
Nr
Typ
Modus
X1
Y1
V1
A1
X2
Y2
V2
A2
1
Definitio
interpoliert
0
0
0
0
100000
20000
0
0

64446axx

3.5.18 MC_ReadCamTable_MX



64447axx

Description:

You can use this module to read a control point table from MOVIAXIS®.

The control points are stored in an array.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

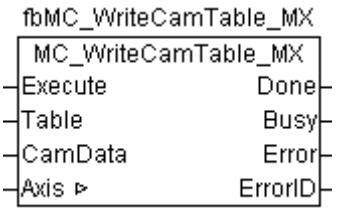
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates the reading of the control points
Table	MC_CAM_TABLENUMBER_MX	Curve selection limited to the values 1-10 CAM_1_MX (1), curve 1 selected CAM_2_MX (2), curve 2 selected CAM_3_MX (3), curve 3 selected CAM_4_MX (4), curve 4 selected CAM_5_MX (5), curve 5 selected CAM_6_MX (6), curve 6 selected CAM_7_MX (7), curve 7 selected CAM_8_MX (8), curve 8 selected CAM_9_MX (9), curve 9 selected CAM_10_MX(10), curve 10 selected
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters read
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126
CamData	ARRAY[0..511]	Storage location for the 512 control points

3.5.19 MC_WriteCamTable_MX



64448axx

Description:

You can use this module to write a control point table to MOVIAXIS®.

The control points are transferred with an array.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

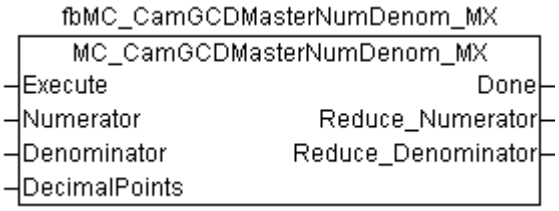
Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates the writing of the control points
Table	MC_CAM_TABLENUMBER_MX	Curve selection limited to the values 1-10 CAM_1_MX (1), curve 1 selected CAM_2_MX (2), curve 2 selected CAM_3_MX (3), curve 3 selected CAM_4_MX (4), curve 4 selected CAM_5_MX (5), curve 5 selected CAM_6_MX (6), curve 6 selected CAM_7_MX (7), curve 7 selected CAM_8_MX (8), curve 8 selected CAM_9_MX (9), curve 9 selected CAM_10_MX(10), curve 10 selected
CamData	ARRAY[0..511]	Storage location for the 512 control points
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.5.20 MC_CamGCDMasterNumDenom_MX



64449axx

Description:

You can use this module to calculate the reduced numerator and denominator for the master position. No parameters are written.

You can use the numerator and denominator values with the "MC_SetCamConfig_MX" module.

GCD -> **g**reatest **c**ommon **d**ivisor

	Example 1	Example 2
Numerator	65536	65536
Denominator	10000	20
DecimalPoints	0	3
Result: Reduce_Numerator	4096	2048
Result: Reduce_Denominator	625	625

Prerequisite: None

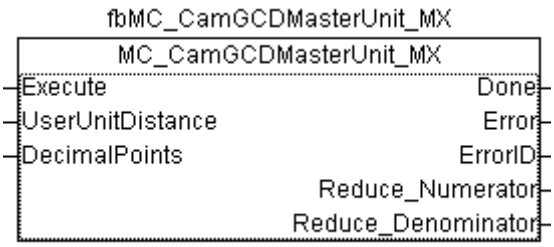
Inputs:

Name	Type	Meaning
Execute	BOOL	This input triggers the calculation
Numerator	UDINT	Numerator value of the master
Denominator	UDINT	Denominator value of the master
DecimalPoints	BYTE	Number of decimal places

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Reduce_Numerator	UDINT	reduced numerator value
Reduce_Denominator	UDINT	reduced denominator value

3.5.21 MC_CamGCDMasterUnit_MX



64457axx

Description:

You can use this module to calculate the numerator and denominator for the master position. No parameters are written.

You can use the numerator and denominator values with the "MC_SetCamConfig_MX" module.

GCD: **g**reatest **c**ommon **d**ivisor.

Calculation:

MasterPosition [inc](65536inc) = MasterPosition [user unit] $\times$ $10^{DecimalPoints}$ $\times$ Numerator / Denominator

	Example 1	Example 2
UserUnitDistance	12.345	1.000.000 [Unit]
DecimalPoints	3	0
Result: Reduce_Numerator	65536	1024
Result: Reduce_Denominator	12345	15625

Prerequisite:

UserDistance > 0.

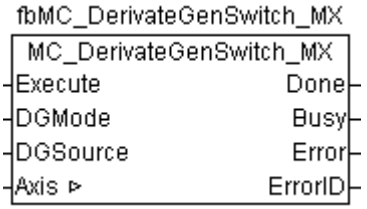
Inputs:

Name	Type	Meaning
Execute	BOOL	This input triggers the calculation
UserUnitDistance	REAL	Master cycle length in user units
DecimalPoints	BYTE	Number of decimal places

Outputs:

Name	Type	Meaning
Done	BOOL	Parameters written
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126
Reduce_Numerator	UDINT	reduced numerator value
Reduce_Denominator	UDINT	reduced denominator value

3.5.22 MC_DerivateGenSwitch_MX



64458axx

Description:

You can use this module to change the "Source address" and "Operating mode" parameters of the derivate generator.

The "MC_LinkTeCam_MX" module sets the source to the output of cam 1 and the operating mode to Position.

The library uses CamFlow 1 for the 10 curves and CamFlow 2 for processing the offset. CamFlow 3 is reserved for special applications.

Prerequisite:

"MC_LinkTecCam_MX" has the "Cam" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

Inputs:

Name	Type	Meaning
Execute	BOOL	This input activates the writing of the parameters
DGMode	DINT	Change of operating mode 0 = none 1 = Position 2 = Velocity 3 = Torque
DGSource	UINT	Source for the derivate generator 1 = Curve sequence 1 (CamFlow 1)-> default 2 = Curve sequence 2 (CamFlow 2) 3 = Curve sequence 3 (CamFlow 3)
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

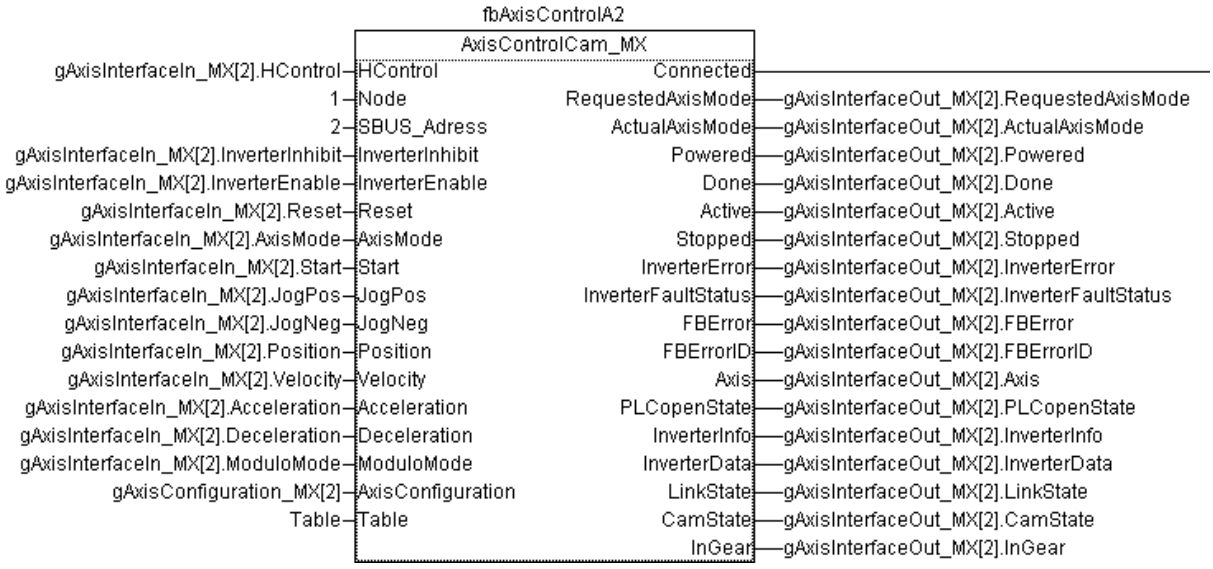
Name	Type	Meaning
Done	BOOL	Parameters written
Busy	BOOL	Block is being processed.
Error	BOOL	Fault in module during execution
ErrorID	DWORD	see page 126

3.6 Example

3.6.1 Cam with virtual encoder as master template

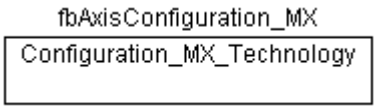
Creating a new PLC project using the example of "AxisControl_MX_Technology".

Delete the "AxisControl_MX" and "AxisControlGear_MX" modules from the "PRG_Technology_MX" module.



64459axx

Set the initialization parameters in the "Configuration_MX_Technology" module.



64460axx

Configuration settings in the Configuration_MX_Technology module

Virtual encoder

```
IF NOT bInit THEN
    (*********************************************************************
        Initialization of AxisControlVirtual
    *********************************************************************)
    (* Configuration virtual encoder *)
    gVirtualEncoderConfiguration.StartRisingEdge := FALSE;
    gVirtualEncoderConfiguration.LinkTecAxisVirtual.CanNode := SBUS_NODE_1;
(* SBUS_NODE_1 SBUS_NODE_2 *)
    gVirtualEncoderConfiguration.LinkTecAxisVirtual.SendObjectID:=  MX_VIRTUAL_
ENCODER_ID1;
    gVirtualEncoderConfiguration.EncoderModeVirtual := VIRTUAL_MX_MODE;
    (*********************************************************************
        Initialization of AxisControl axis 2 (single axis)
    *********************************************************************)
```

```
    (* general axis control parameters *)
    gAxisConfiguration_MX[2].StartRisingEdge := FALSE;
    (* configuration modulo parameter --> only necessary if modulo mode is used *)
    gAxisConfiguration_MX[2].SetModuloParameters.Overflow: = 1;
    gAxisConfiguration_MX[2].SetModuloParameters.Underflow: = 1;
    gAxisConfiguration_MX[2]. Adjustmode:= MX_MOD_ABS_SHORT_ XMASTERPO-
SZERO;


    (* Configuration cam parameters   *)
    (* the master signal is 65536 Inc/one revolution
     if you want to have 1,000,000 Unit/revolution use the follow parameters
       1.000.000/65536 = 62500/4096
     1,000,000 UserUnit  =  65536 Inc * 62500/4096*)
    gAxisConfiguration_MX[2].SetCamConfig.MasterNumerator:= 62500;
    (* Numerator for scaling of the master increments *)
    gAxisConfiguration_MX[2].SetCamConfig.MasterDenominator:= 4096;
(* Denominator for scaling of the master increments *)
    gAxisConfiguration_MX[2].SetCamConfig.MasterModuloMax: = 1.000.000;
(* Modulo maximum Value of the master position*)
    gAxisConfiguration_MX[2].SetCamConfig.MasterModuloMin: = 0;
(* Modulo minimum Value of the master position*)
    gAxisConfiguration_MX[2].SetCamConfig.MasterSource:=     MX_CAM_RECEIVE_
PDO_1;
            (* select physical master or receive message box master is the SBus receive
object PDO1 *)
    gAxisConfiguration_MX[2].SetCamConfig.LagErrorWindow := 1000;
    (* lag error window in [inc] *)
    gAxisConfiguration_MX[2].SetCamConfig.RotationDirectionLock :=0;
(* 0 = both directions, 1 = only CW enabled, 2 = only CCW enabled *)
    gAxisConfigura-tion_MX[2].SetCamConfig.ExtConfig.ReactionLagEr-
ror:=MX_CAM_INHIBIT_WAITING;
    gAxisConfiguration_MX[2].SetCamConfig.ExtConfig.MSignal_InterpolationTime   :=
10;
    (* Interpolation time: range 1...60 means 0.5....30 ms filter time*)
    gAxisConfiguration_MX[2].SetCamConfig.ExtConfig.AverageFilterTime := 10;
(* AverageFilter: range 1...60 means 0.5....30 ms filter time*)
    gAxisConfiguration_MX[2].SelectUserFcb.FCB_Number: = 26;
(* Designated FCB number *)
    gAxisConfiguration_MX[2].SelectUserFcb.FCB_Instance: = 0;
(* Designated FCB instance number *)
    bInit:=TRUE;
END_IF;
```

Description:

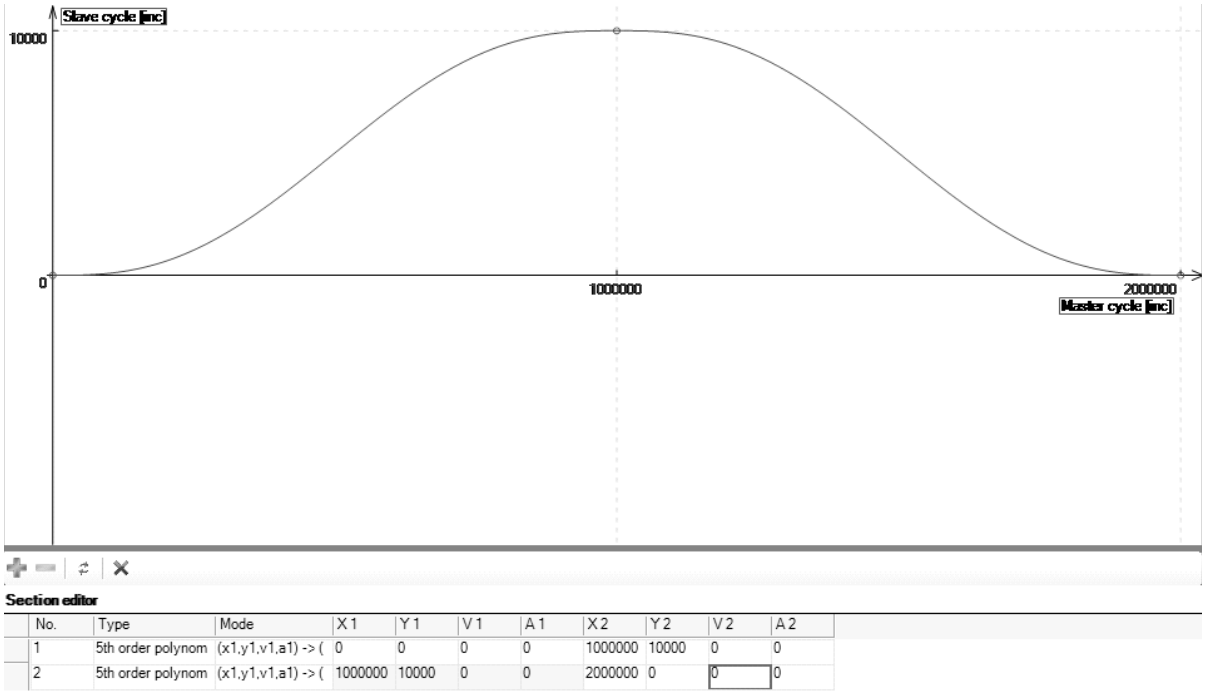
Set the SBus address 2 at the supply module.

Perform a drive startup for MOVI-PLC®. If required, restore the factory settings.

One revolution should correspond to 10000 mm.

64471axx

Create curve 1 with the cam editor and download it.



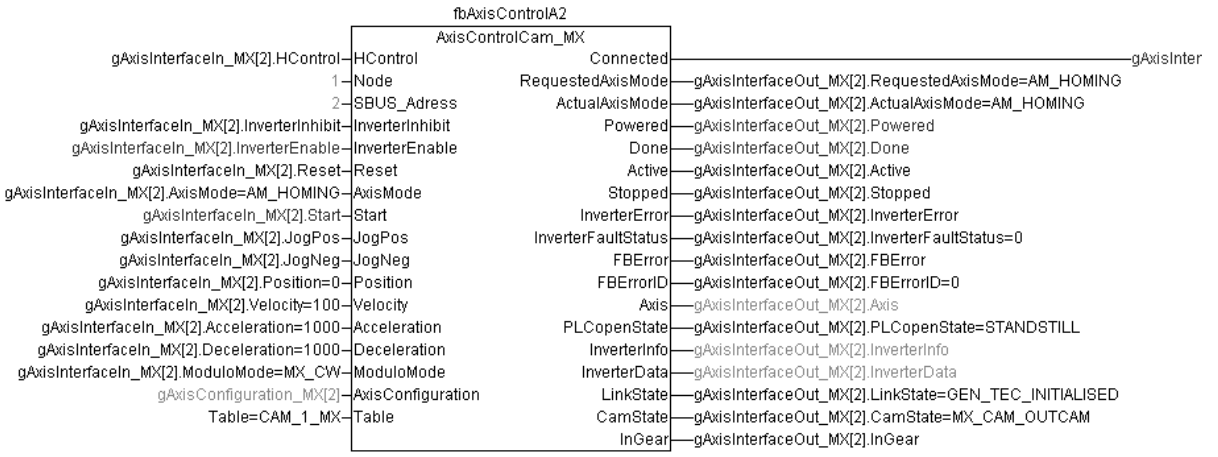
No.	Type	Mode	X 1	Y 1	V 1	A 1	X 2	Y 2	V 2	A 2
1	5th order polynom	(x1,y1,v1,a1) -> (	0	0	0	0	1000000	10000	0	0
2	5th order polynom	(x1,y1,v1,a1) -> (	1000000	10000	0	0	2000000	0	0	0

64472axx

Start the PLC program.

Enable the axis at input DI 0 and set the *InverterEnable* bit.

Select "AxisMode AM_Homing", set the start bit and reference the axis.

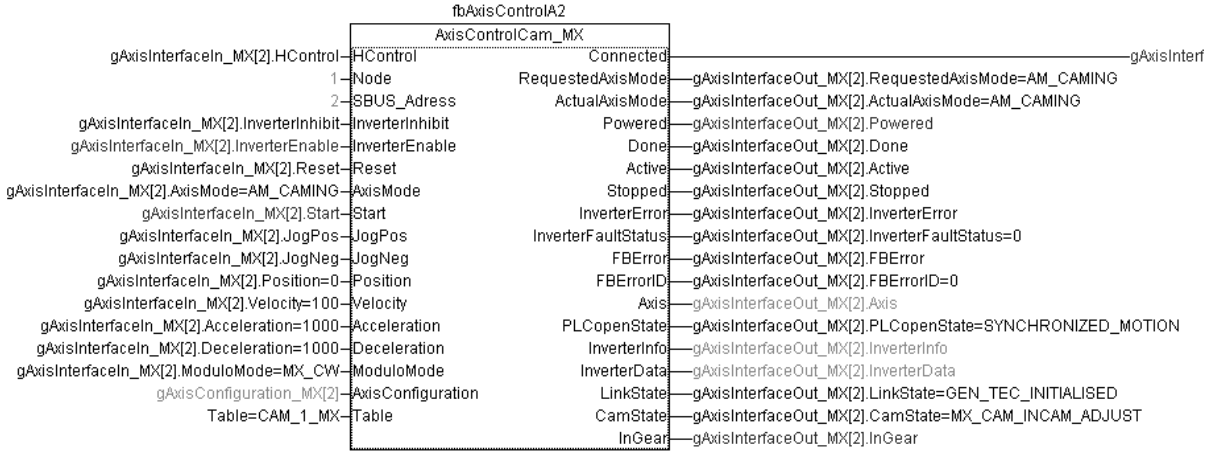


64473axx

Specify the velocity and the acceleration.

Select "AxisMode AM_Camimg" and switch the axis to curve operation.

The axis aligns with the "MC_CamInAdjust_MX" module and engages.

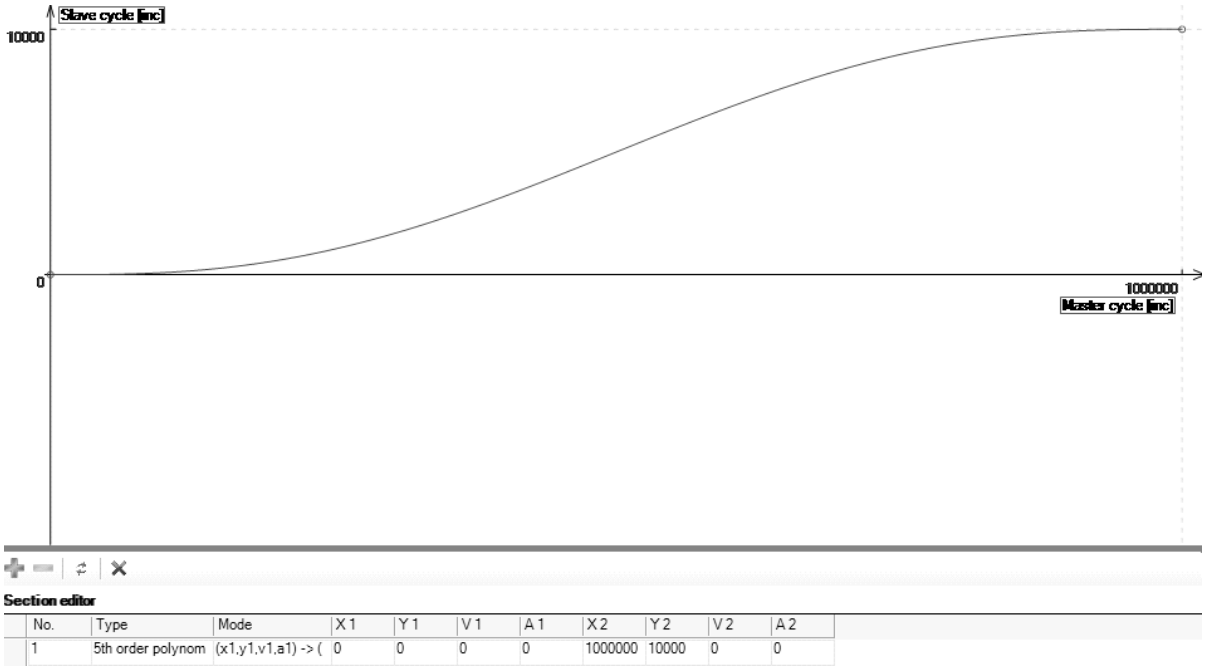


64474axx

Start the virtual master via axis mode "Velocity" in the "PRG_VirtualEncoder"

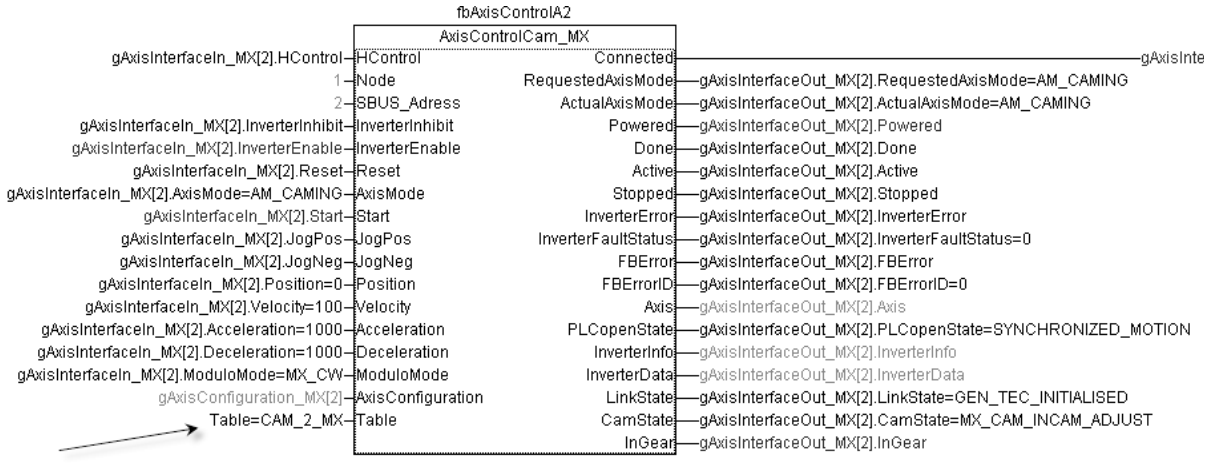
The axis moves one revolution forward and one revolution backward.

Create "curve 2" with the cam editor and download it.



64475axx

Set the *Table* input of the "AxisControlCam_MX module to Cam _2_MX".



64476axx

Now switch from curve 1 to curve 2. The axis turns by one revolution per master revolution.

Deactivate the *DI0* or *InverterEnable*.

- Axis stops,
- Virtual master continues.

Set the *Start input* of the virtual master to FALSE.

- Virtual master stops.

Activate the *DI0* or *InverterEnable*.

- The axis returns to the slave position that corresponds to the master position on the shortest way.

Set the *Start input* of the virtual master to TRUE.

- The axis follows the master with the selected curve.

3.7 Appendix

3.7.1 Error detection (ErrorID)

Error code	Error designation	Problem
FA0010	E_IEC_GENERAL_INVALID_TECHNOLOGIE_OPTION	Requested technology function not supported
FA020...	**General link tec error**	
FA0200	E_TEC_GENERAL_MULTIPLE_TECLINKS	The LinkTec FB may not be called repeatedly.
FA0201	E_TEC_GENERAL_INVALID_LINKSTATE	Invalid LinkState for function call
FA0202	E_TEC_GENERAL_NOT_LINKED	LinkTec FB has no logical connection
FA0203	E_TEC_GENERAL_NOT_INITIALIZED	Technology function not yet active
FA0204[1)]	E_TEC_GENERAL_SERVICE_NOT_IMPLEMENTED	Requested service not supported
FB02002	E_TEC_CAM_MX_INVALID_CAM_TEC_EDITOR_CONFIGURATION	No valid technology editor settings
FA023…	**Cam Technology function Error**	
FA0233	E_TEC_CAM_MX_CURVE_AKTIV	Not permitted while a curve is active.
FA0234	E_TEC_CAM_MX_SLAVESHIFT_TO_LOW	The calculated shift factor is smaller than the maximum
	IEC general error messages	
FA0070	E_IEC_PARAMETER_VALUE_OUT_OF_RANGE	Invalid entry value for parameter
FB0071	E_MDX_MOTIONBLOCK_LOG_ADR_NOT_INITIALIZED	MC_ConnectAxis_MDX has not yet assigned a log address
FB0072	E_MDX_MOTIONBLOCK_INVALID_LOG_ADR	Invalid log address
FB0073	E_MDX_MOTIONBLOCK_INVALID_STATE	Function call in current PLCopen state not permitted
FB0074[1)]	E_MDX_MOTIONBLOCK_INVALID_OPERATING_MODE	The requested technology function is not supported by the set MOVIDRIVE® operating mode

1) This error message is not relevant for MOVIAXIS® (only for MOVIDRIVE®)

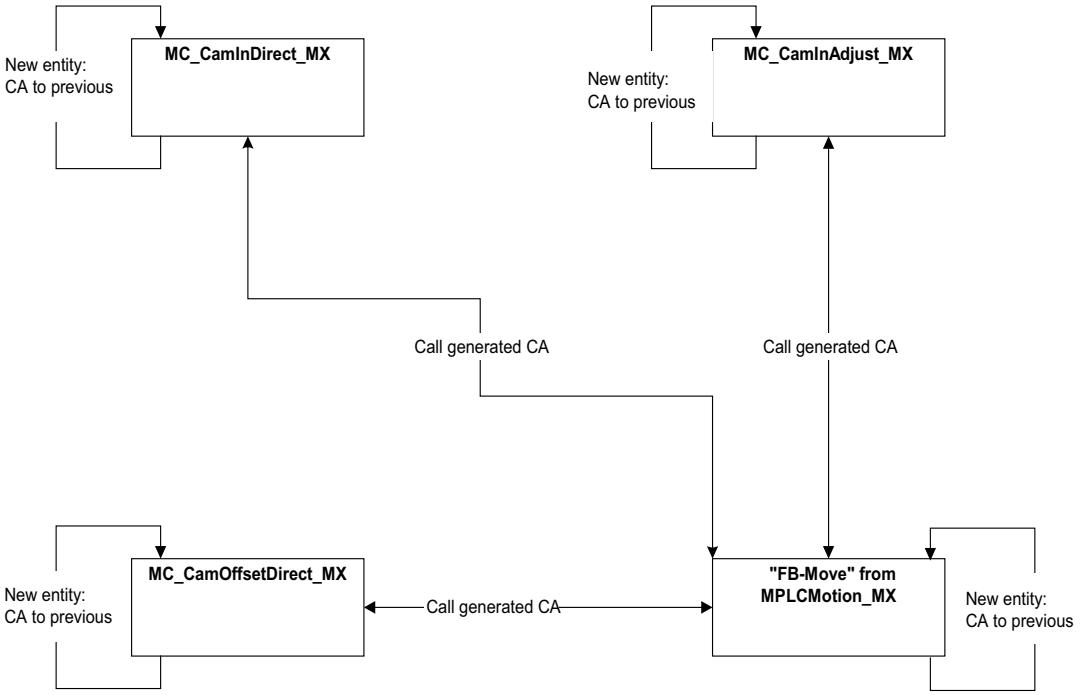
3.7.2 Interrupting the modules (CommandAborted)

The "CommandAborted" output signal is used to signal repeated calls of a motion module or the call of a different motion module. The aborted command of the function module is no longer executed (see detailed information in the "MPLCMotion_MDX/MX library for MOVI-PLC®" manual).

The following table shows the mutual interruption of the "Motion function blocks".

Activated module	Generates CommandAborted at:
MC_CamInDirect_MX	- Instances of MC_CamInDirect_MX - “Move-FB”
MC_CamInAdjust_MX	- Instances of MC_CamInAdjust_MX - "Move FB"
MC_CamOffsetDirect_MX	- Instances of MC_CamOffsetDirect_MX - "Move FB"
“Move FB”	- MC_CamInDirect_MX - MC_CamInAdjust_MX - MC_CamOffsetDirect_MX

Schematic representation of "CommandAborted":



Comment:

"Move FBs" of the MPLCMotion_MX library are:

- MC_MoveAbsolute_MX
- MC_MoveRelative_MX
- MC_MoveTargetPosition_MX
- MC_MoveTargetSpeed_MX
- etc.

4 MPLCTecCAMSwitch_MX Library

4.1 Introduction

The MOVIAXIS® multi-axis servo inverter offers comprehensive technology functions, for example:

- Electronic gear unit,
- Cams,
- Touch probe,
- Event control,
- Cam control.

The "Motion Technology Editor" startup assistant of MotionStudio enables configuration of all technology functions. Basic settings are configured this way. However, certain values or settings often have to be changed dynamically during a process, or various functions must be combined and run in sequence.

The function blocks of the "MPLCTecCamSwitch_MX" library set the general cam control parameters.

This means that in-depth familiarization with the individual parameters of the respective technology functions is not necessary. Configuration with the Technology Editor is not necessary, because the "MPLCTecCamSwitch_MX" library performs all relevant settings.

Advantages of the "MPLCTecCamSwitch_MX" library:

- Programming in accordance with IEC 61131, which means customer-specific applications can be implemented easily.
- Library for controlling the technology function Cam controller. This allows for simple configuration and control of the cam controller functionality without detailed knowledge of the individual technology function.
- Central control of several cam controllers of various axes.
- Additional functions, such as specifying master values via a virtual master encoder, are controlled centrally using the MOVI-PLC®.
- Different data sources of the individual tracks can be parameterized easily.
- Cam windows can easily be changed during operation or format changes.

Additional notes are can be found in the following documentation:

- For general notes on the operating principle of the libraries, refer to the publication "MPLCMotion_MDX and MPLCMotion_MX libraries for MOVI-PLC®".
- "MOVIAXIS® MX Multi-Axis Servo Inverter, Electronic Gear Unit Technology Function" manual.

4.2 Areas of application

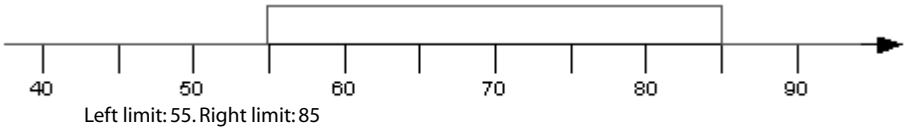
4.2.1 Cam windows and tracks

The MOVIAXIS® cam controller is equipped with 8 cam tracks, each of which can be assigned 16 of 32 cam windows. The cam windows are described with a left and a right limit value.

Each cam track has n operating mode. It can be used to deactivate a track and to decide as to whether the assigned cams shall be regarded to be in standard or modulo operation.

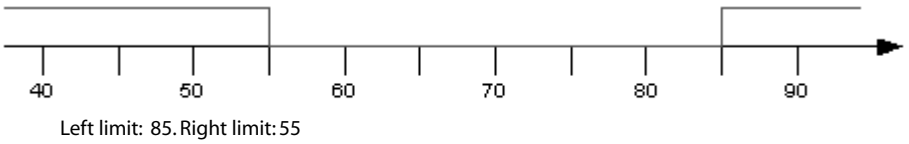
In "Standard", the left limit value of each assigned cam must be smaller than or equal to the right limit value. In "Modulo", the limit values of a cam can be any value. The cam limits are parts of the cam.

The following figure shows a modulo/standard cam track : left limit < right limit.



64568aen

The following figure shows a modulo/standard cam track: left limit > right limit



64569aen

A cam window is also equipped with a processing direction. With the direction you can determine as to whether it is active coming the left, the right, both or no side at all.

Observe the following when switching off a cam:

- The value sequence must correspond to the effective direction of the cam
- The current value must be part of the cam:
  - For standard cams with the left limit value smaller than or equal to the right limit value: Value ≥ left limit value **and** value ≤ right.
  - For Modulo cams with the limit value exceeding the right limit value: ≥ left limit value **and** right limit value ≤.

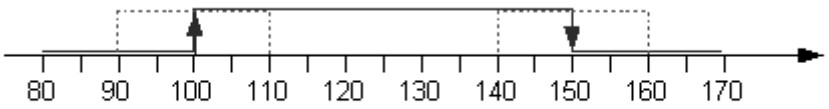
This definition allows for a directional effect when passing the cam, as well as correct switching result when waking up on a cam and moving within the cam area.

Observe the following notes :

- The current value must be outside the cam area.

The direction is no longer considered during the switch off. It is only important for activation This will result in the desired behavior that, in the event of a reversal on a directional cam, the result of the track remains '1' until the cam has been left instead of changing to '0' right after the reversal.

Example:



Left limit: 100. Right limit: 150. Hysteresis 10, from left

64570aen

Value	Result of the track	Comment
80	0	Value outside the cam
90	0	Value outside the cam
100	1	Cam limit belongs to the cam hysteresis window: 90/110
120	1	Value on cam
119	1	Reversal, but still on cam
100	1	Value on cam
99	0	Cam being left, hysteresis window: 90/110

However, these rules might also result in the following possibly undesired behavior:

Value	Result of the track	Comment
99	0	Value outside the cam
100	1	Cam limit belongs to the cam, direction correct hysteresis window: 90/110
130	1	Value in cam area
150	1	Cam limit belongs to the cam
151	0	Cam being left, hysteresis window: 140/160
130	0	Hysteresis window being left, hysteresis deactivated
135	1	Value in cam area, direction correct, no hysteresis
100	1	Value in cam area
99		Cam being left, hysteresis window: 90/110

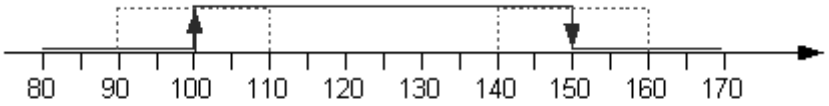
For each track there are several setpoint and actual values. Any value existing in the DDB is possible. For this variable, you can determine a dead-time compensation of -500 µs to +500 µs in steps of 0.1 µs. Thus, the value calculated with the dead-time compensation is used as a basis of comparison.

This procedure might result in incorrect values due to mixed-up actual values. In order to achieve a smoothing, you can specify a time basis for a pre-calculation. This time determines the age of the previous value for the linear extrapolation. The direction of this line corresponds to the direction the cam direction is compared with, i.e. the time basis also smoothens the direction determined by the comparison values.

If an actual value jitters about the startup transition of a cam, the switching result jitters as well. In order to eliminate this usually undesired behavior, each cam track has a hysteresis value. This value determines a window around the active switching transition, e.g. left cam limit value – hysteresis to cam limit value + hysteresis.

This window is determined and activated once the switching results of a track have been changed. As long as the value of the data source is in this window, the track is not re-evaluated. The limit values are within the hysteresis window.

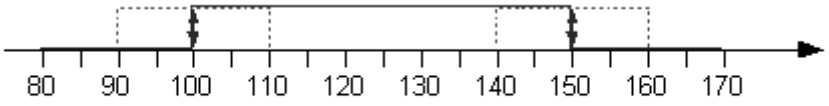
Examples for switching results:



Left limit: 100. Right limit: 150. Hysteresis 10, from left

64571ade

Value	Result of the track	Comment
80	0	Value outside the cam
90	0	Value outside the cam
100	1	Cam limit belongs to the cam hysteresis window: 90/110
90	1	Hysteresis limit is within hysteresis → no re-evaluation.
150	1	Cam limit belongs to the cam
151	0	Value outside the cam, hysteresis window: 140/160
140	0	Hysteresis limit is within hysteresis → no re-evaluation.
139	0	Value now on cam, outside hysteresis, hysteresis window deactivated, wrong direction.
151	0	Value outside the cam
170	0	Value outside the cam
150	0	Value now on cam, wrong direction.
160	0	Value outside the cam
161	0	Value outside the cam



Left limit: 100. Right limit: 150. Hysteresis 10, from both sides

64572aen

Value	Result of the track	Comment
80	0	Value outside the cam
90	0	Value outside the cam
100	1	Cam limit belongs to the cam hysteresis window: 90/110
90	1	Hysteresis limit is within hysteresis → no re-evaluation.
150	1	Cam limit belongs to the cam
151	0	Value outside the cam, hysteresis window: 140/160
140	0	Hysteresis limit is within hysteresis → no re-evaluation.
139	1	Value now on cam, outside hysteresis, hysteresis window deactivated
151	0	Value outside the cam
170	0	Value outside the cam
150	1	Cam limit belongs to the cam hysteresis window: 140/160
160	1	Hysteresis limit is within hysteresis → no re-evaluation.
161	0	Value outside hysteresis

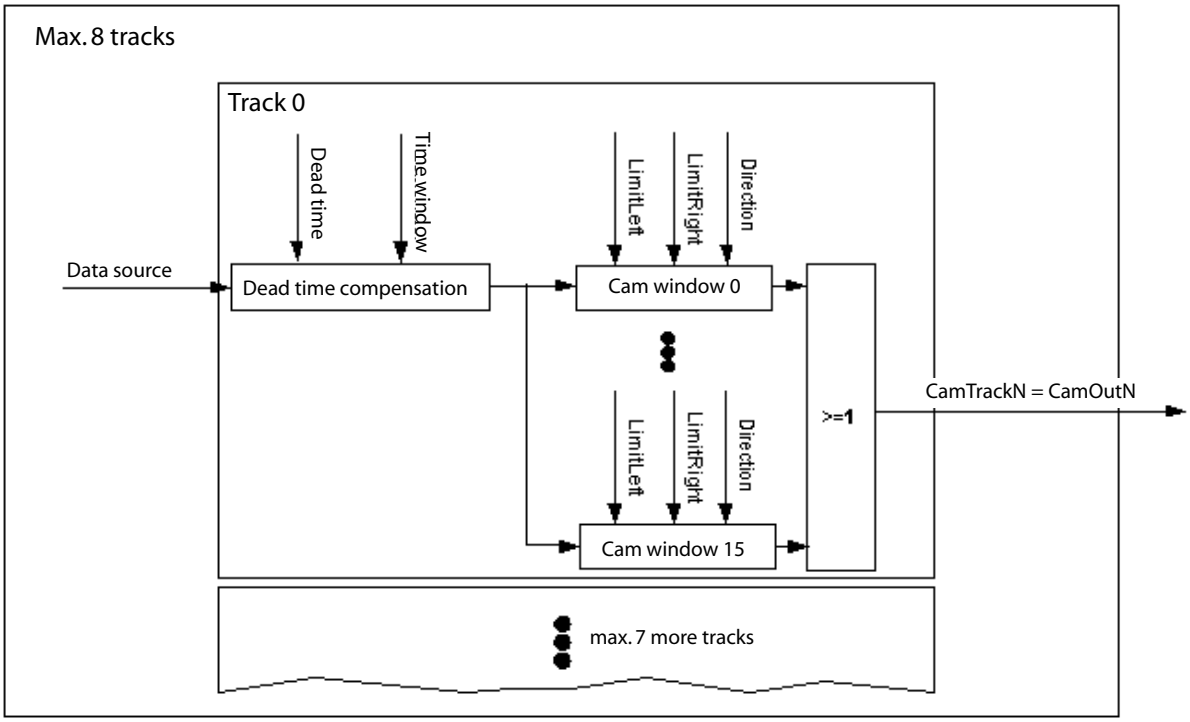
4.2.2 Processing speed

A maximum of 8 tracks can be processed in the cam controller in 1 ms.

The tracks use the dynamic linear procedure. With this procedure, all the cam windows of all the cam tracks are searched during the evaluation until the end is reached or one window determines the output.

In order to achieve a precise output switching process, you can use the results of these 4 tracks as setpoint values for *Fast IOs*. The results of the tracks are mirrored to index 10488.2. The index is assigned to process output word 9 in the PDO editor. From there, you can use the PDO editor to assign the respective result to a bit in a control word, and this control word to the digital outputs of the unit. With this setting, the cam controller automatically addresses the IO as *Fast IOs*. You can also select a connection to the option cards.

The following display shows the block diagram of the cam controller.

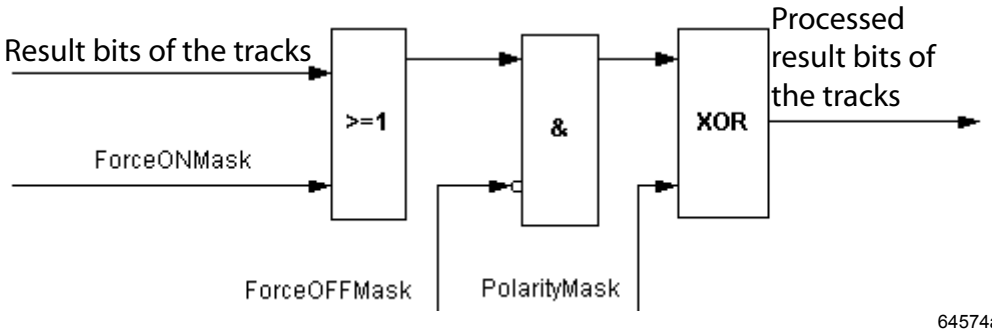


64573aen

4.2.3 Results

The outputs of the individual cam tracks depend on various parameters. Use the "ForceOnMask" to permanently switch on certain outputs. The "ForceOffMask" permanently deactivates the outputs. It has higher priority than the "ForceOnMask". With the "PolarityMask" you can determine as to whether an output is *High active* or *Low active*.

The following figure shows the processing of the track outputs.



64574aen

The cam controller always writes the outputs in the value range in the Data Distribution Buffer. From there you can handle the result as a process date. Thus the outputs of the tracks can be provided in various forms, e.g. via the physical outputs or a fieldbus. In addition, the result is on the fixed parameter 10488.2.

	Operation	Example values	Comment
Results of the tracks		0b10101010	
ForceOn mask	OR	0b00010101	Tracks 0 and 2 are always set to 1
ForceOff mask	AND, value inv.	0b00010010	Tracks 1 and 4 are always set to 0 Ingnores previous lines.
Polarity mask	XOR	0b00000100	Track 2 -> reversed polarity
Final result in 10488.2		0b10101001	

4.3 Project planning

To use the "MPLCTecCamSwitch_MX" library, you need a MOVI-PLC® controller in technology version T1 or higher.

Further, the "MPLCMotion_MX" library must be integrated. Otherwise, the requirements and limitations apply according to the "MPLCMotion_MX"

4.3.1 Basic principle and notes

- Configuring the relevant technology functions of the axis using IEC function modules.
- The "MC_LinkTecCamSwitch_MX" function module is absolutely essential as a central data interface to the "Cam controller" technology function.
- The Technology Editor of MotionStudio is not necessary, but for diagnostic purposes, an online connection of the monitor can be established with the sample project "MOVI-PLC® Default project.....Tec-Function. The sample project may not be downloaded, otherwise settings made by the library would be overwritten.

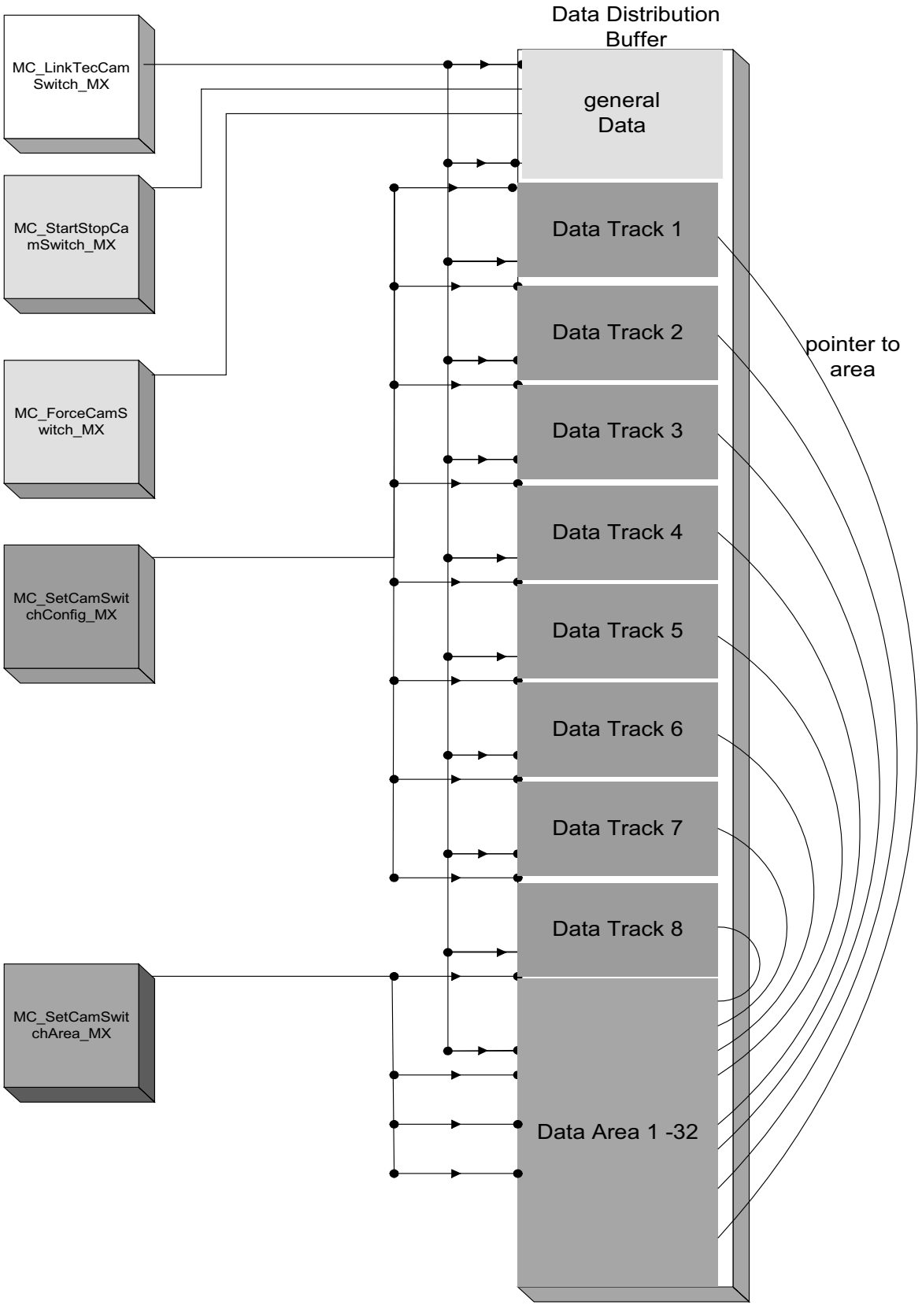
4.3.2 Project planning procedure

Procedure

- Integrate the "MPLCTecCamSwitch_MX" library with the library manager into the project in addition to the "MPLCMotion_MX" library.
- The "MC_LinkTecCamSwitch_MX" function module constitutes the interface between technology function and axis and must therefore be called cyclically in the program. Once the "MC_ConnectAxis_MX" module has established the connection to the axis (*Done*), "LinkTecCamSwitch" activates the technology functions of the MOVIAXIS®" necessary for the cam controller. In addition to that, the necessary settings are made in the PDO Editor.
- The cam controller is activated with the "MC_StartStopCamConfig_MX" function module.
- You can use the "MC_SetCamSwitchAreas_MX" module to write the cam limits into the MOVIAXIS®.
- The "MC_SetCamSwitchConfig_MX" module is used to set the track parameters and the cam references.
- With the "M_ForceCamSwitch_MX" module, you can set or reset the outputs (tracks) of the cam controller permanently bit by bit.

4.3.3 Location of the data in MOVIAXIS®

Overview of the data location



All values and settings of the cam controller are stored in the Data Distribution Buffer. The start of the DDB structure with the cam controller configuration is stored in parameter 10488.1.

The following section provides detailed information on the DDB data blocks. Offset and length of the data blocks are calculated in DDB variables i.e. as 32 bit value.

4.3.4 Total DDB structure

Offset	Name	GUI name	Value range
0	Length	-	582
1	Type and version	-	Type: 3 (High word) Main version: 1 (High byte of the Low word) Subversion: 1 (Low byte of the Low word)
2	General data block		
22	8 cam track data blocks		
262	32 cam window data blocks		

General settings

The general data is written into the MOVIAXIS® using the "MC_LinkTecCamSwitch_MX" module. The tracks of the cam controller can be routed to the standard outputs and the option card outputs.

The outputs of the cam controller can be set to '1' or '0' individually. Further, you can reverse the logic. You can use the "MC_ForceCamSwitch_MX" module to manually force process output data. The cam controller can be activated and deactivated with the "MC_StartStopCamSwitch_MX" module.

If the start address (10488.1) of the cam controller is set to 815, the following addresses apply:

Offset	Index/subindex	Description
819	20206.52	Operating mode (0 = deactivated / 1 = activated)
820	20206.53	Sub operating mode
821	20206.54	Status display
822	20206.55	« Force-On « Mask
823	20206.56	« Force OFF « Mask
824	20206.57	« Polarity « Mask
825	20206.58	Results of the cam controller

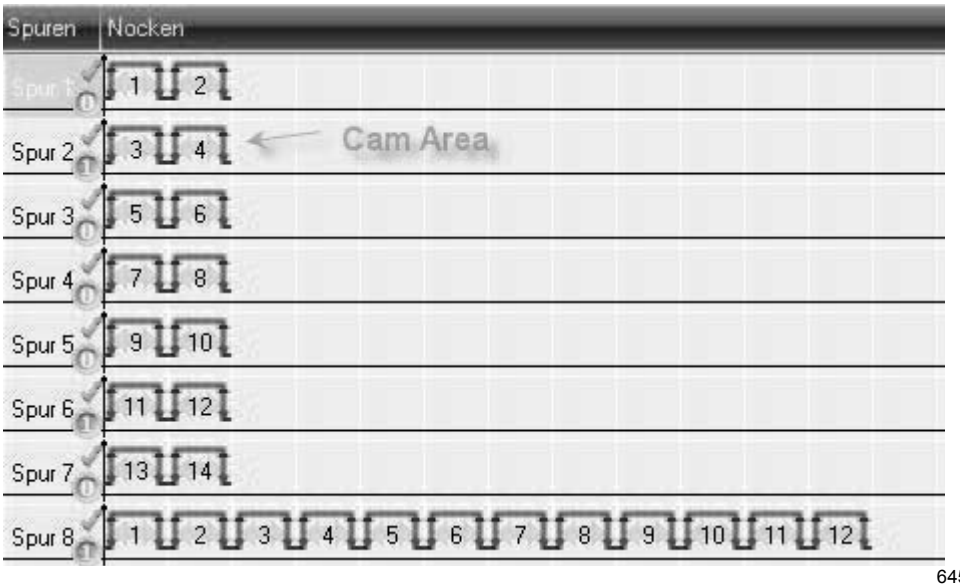
Location of the CAM tracks 1 – 8

You can set 1–8 cam tracks. Each track has separate seting options, see data type: "MC_CAMSWITCH_CONFIG_MX". The data is written into the MOVIAXIS® with the "MC_SetCamSwitchConfig_MX". If the start address (10488.1) of the cam controller is set to 815, the following addresses apply:

Offset	Index/subindex	Description	No.
839	20206.72	Operating mode (0 = deactivated / 1 = activated)	1
840	20206.73	Sub operating mode (0=Standard / 1=Modulo)	1
841	20206.74	Status	1
842	20206.75	Data source	1
843	20206.76	Dead time (-500000 ms -500000 ms)	1
844	20206.77	Time window (0 16 ms)	1
845	20206.78	Min modulo value	1
846	20206.79	Max. modulo value	1
847	20206.80	Hysteresis	1
851	20206.84	Cam reference 1	1
852	20206.85	Cam reference 2	1
853	20206.86	Cam reference 3	1
854	20206.87	Cam reference 4	1
855	20206.88	Cam reference 5	1
856	20206.89	Cam reference 6	1
857	20206.90	Cam reference 7	1
858	20206.91	Cam reference 8	1
859	20206.92	Cam reference 9	1
860	20206.93	Cam reference 10	1
861	20206.94	Cam reference 11	1
862	20206.95	Cam reference 12	1
863	20206.96	Cam reference 13	1
864	20206.97	Cam reference 14	1
865	20206.98	Cam reference 15	1
866	20206.99	Cam reference 16	1
...			
869	20206.72	Operating mode	2
...			
896	20207.01	Cam reference 16	2
...			
899	20207.04	Operating mode	3
...			
926	20207.31	Cam reference 16	3
...			
929	20207.34	Operating mode	4
...			
956	20207.61	Cam reference 16	4
...			

Offset	Index/subindex	Description	No.
959	20207.64	Operating mode	5
...			
986	20207.91	Cam reference 16	5
...			
989	20207.94	Operating mode	6
...			
1016	20207.121	Cam reference 16	6
...			
1019	20207.124	Operating mode	7
...			
1046	20208.23	Cam reference 16	7
...			
1059	20208.24	Operating mode	8
...			
1076	20208.53	Cam reference 16	8

You can open the motion technology editor with a default project to open the cam references.



64592axx

Note:

The maximum modulo value cannot be achieved, thus it cannot serve as cam limit.

	TIPS
	The correctness of the values of the referenced cams in conjunction with the track is only ensured when the track is activated. However, the cam values can be changed afterwards too. However, due to run time issues the values of the referenced cams of an activated track are not checked.

Location of the cam areas 1 – 32 (cam windows)

The maximum number of cams is 32. Each cam has a left limit and a right limit, as well as a direction of action. See data type: "MC_CAMSWITCH_AREAS_MX".

The general data is written into the MOVIAXIS® using the "MC_SetCamSwitchAreas_MX" module. If the start address (10488.1) of the cam controller is set to 815, the following addresses apply:

Offset	Index/subindex	Description	No.
1079	20208.56	Cam left limit value	1
1080	20208.57	Cam right limit value	1
1081	20208.58	Cam direction	1
..			
1089	20208.66	Cam left limit value	2
1090	20208.67	Cam right limit value	2
1091	20208.68	Cam direction	2
...			
1099	20208.76	Cam left limit value	3
1100	20208.77	Cam right limit value	3
1101	20208.78	Cam direction	3
...			...
...			...
1179	20209.28	Cam left limit value	10
1180	20209.29	Cam right limit value	10
1181	20209.30	Cam direction	10
...			
1189	20209.38	Cam left limit value	11
1190	20209.39	Cam right limit value	11
1191	20209.40	Cam direction	11
...			...
...			...
1279	20209.128	Cam left limit value	20
1280	20210.1	Cam right limit value	20
1281	20210.2	Cam direction	20
...			
1289	20210.10	Cam left limit value	21
1290	20210.11	Cam right limit value	21
1291	20210.12	Cam direction	21
...			...
...			...
1379	20210.100	Cam left limit value	31
1380	20210.101	Cam right limit value	31
1381	20210.102	Cam direction	31
...			
1389	20210.100	Cam left limit value	32
1390	20210.101	Cam right limit value	32
1391	20210.102	Cam direction	32

4.4 The MPLCTecCamSwitch_MX library

4.4.1 Brief overview of the MPLCTecCamSwitch_MX libraries

64593axx

MC_LinkTecCamSwitch_MX

Description:

The "MC_LinkTecCamSwitch_MX" function module has various functions:

- Setting up the technology functions of MOVIAXIS® and activating the functions and parameters required for Cam Switch.
- Configuring the MOVIAXIS® process data interface.
- Routing the tracks of the cam controller to digital outputs of the MOVIAXIS®
- Displaying the initialization state.

MC_SetCamSwitchAreas_MX

Description:

This function module writes the cam windows into the MOVIAXIS®. You can write individual windows or all 32 windows.

Prerequisites:

"MC_LinkTecCamSwitch_MX" signals done.

MC_SetCamSwitchConfig_MX

Description:

This function module activates the selected track (1–8) and writes the general parameters and the cam references. You can assign up to 16 cms (window or area) to one track.

Prerequisites:

"MC_LinkTecCamSwitch_MX" signals done.

MC_StartStopCamSwitch_MX

Description:

This function module activates or deactivates the cam controller.

Prerequisites:

"MC_LinkTecCamSwitch_MX" signals done.

MC_ForceCamSwitch_MX

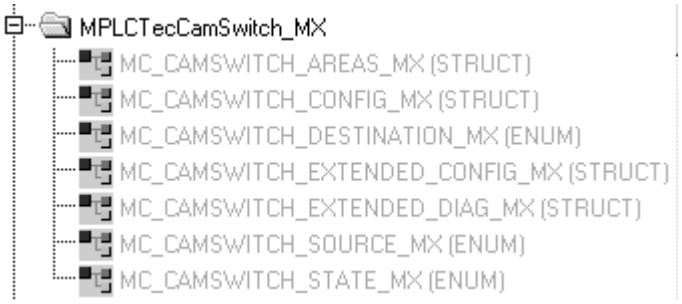
Description:

With this function module you can set the tracks to '1' or '0' individually. You can reverse the logic.

Prerequisites:

"MC_LinkTecCamSwitch_MX" signals done.

4.4.2 The data types of the MPLCTecCamSwitch_MX library



64594axx

MC_CAMSWITCH_AREAS_MX

```
TYPE MC_CAMSWITCH_AREAS_MX :
STRUCT
    LeftLimit:      DINT;  (*Left limit cam window*)
    RightLimit:     DINT;  (*Right limit cam window*)
    Direction:      DINT;  (* 0 = off , 1 = from left , 2 = from right , 3 = both direction*)
END_STRUCT
END_TYPE
```

64595axx

MC_CAMSWITCH_CONFIG_MX

```
TYPE MC_CAMSWITCH_CONFIG_MX :
STRUCT
    Mode             :BOOL;                    (* 0 = off , 1 = on*)
    ModeControl      :BOOL;                    (* 0 = Standard cam track , 1 = Modulo cam track*)
    DataSource:  MC_CAMSWITCH_SOURCE_MX;       (*datasource of the cam track*)
    DeadTime         :DINT;                    (*-500.000  -  + 500.000  microsecond steps 0.1ms*)
    TimeFrame        :DINT;                    (* 1-16  in millisecond steps*)
    MinModuloValue   :DINT;                    (* min modulo value*)
    MaxModuloValue   :DINT;                    (* max modulo value*)
    Hysteresis       :DINT;                    (*Hysteresis of the cam window*)
    SelectArea:   ARRAY[1..16] OF DINT;        (* number of the Cam areas 1 - 32 possible*)
    State            : DINT;                   (*   0 =      off
                                                   1 =      false lenght error 38 suberror 184
                                                   2 =      false typ  error 38 suberror 185
                                                   3 =      false version  error 38 suberror 186
                                                   4 =      false datasource  error 38 suberror 187
                                                   5 =      false cam number  error 38 suberror 188
                                                   6 =      false cam data  error 38 suberror 189
                                                   7 =      false cam track  error 38 suberror 190
                                                   8 =      not enough comparison value
                                                   9 =      initialisation
                                                   10=      activ*)
END_STRUCT
END_TYPE
```

64596axx

MC_CAMSWITCH_DESTINATION_MX

```
TYPE MC_CAMSWITCH_DESTINATION_MX :
(
    MX_CAMSWITCH_DEST_NONE              := 0,
    MX_CAMSWITCH_DEST_X11               := 1,
    MX_CAMSWITCH_DEST_OPT1              := 2,
    MX_CAMSWITCH_DEST_OPT2              := 4,
    MX_CAMSWITCH_DEST_X11_OPT1          := 3,
    MX_CAMSWITCH_DEST_X11_OPT2          := 5,
    MX_CAMSWITCH_DEST_OPT1_OPT2         := 6,
    MX_CAMSWITCH_DEST_X11_OPT1_OPT2:= 7
);
END_TYPE
(*  MX_CAMSWITCH_DEST_NONE              := 0,       none
    MX_CAMSWITCH_DEST_X11               := 1,       Standard output  / X11 DO 0 - 3
    MX_CAMSWITCH_DEST_OPT1              := 2,       PDO Output option 1  / DO 0-7
    MX_CAMSWITCH_DEST_OPT2              := 4        PDO Output option 2  / DO 0-7
    MX_CAMSWITCH_DEST_X11_OPT1          := 3,       Standard output  / X11 DO 0 - 3 and PDO Output option 1    / DO 0-7
    MX_CAMSWITCH_DEST_X11_OPT2          := 5,       Standard output  / X11 DO 0 - 3 and PDO Output option 2    / DO 0-7
    MX_CAMSWITCH_DEST_OPT1_OPT2         := 6,       PDO Output option 1  / DO 0-7 and PDO Output option 2    / DO 0-7
    MX_CAMSWITCH_DEST_X11_OPT1_OPT2:= 7         Standard output   and PDO Output option 1 and PDO Output option 2
*)
```

64597axx

MC_CAMSWITCH_SOURCE_MX

```
TYPE MC_CAMSWITCH_SOURCE_MX :
    (
        MX_CAMSWITCH_SYSTEMPOS_ENCODER1:=109,              (* Master position is the linear position of encoder 1*)
        MX_CAMSWITCH_SYSTEMPOS_ENCODER2:=110,              (* Master position is the linear position of encoder 2*)
        MX_CAMSWITCH_SYSTEMPOS_ENCODER3:=111,              (* Master position is the linear position of encoder 3*)
        MX_CAMSWITCH_MODULOPOS_ENCODER1:=112,              (* Master position is the modulo position of encoder 1*)
        MX_CAMSWITCH_MODULOPOS_ENCODER2:=113,              (* Master position is the modulo position of encoder 2*)
        MX_CAMSWITCH_MODULOPOS_ENCODER3:=114,              (* Master position is the modulo position of encoder 3*)
        MX_CAMSWITCH_USERPOS_ENCODER1:=115,                (* Master position is the modulo position of encoder 1*)
        MX_CAMSWITCH_USERPOS_ENCODER2:=116,                (* Master position is the modulo position of encoder 2*)
        MX_CAMSWITCH_USERPOS_ENCODER3:=117,                (* Master position is the modulo position of encoder 3*)
        MX_CAMSWITCH_MODULO_VIRTUAL_ENCODER_MX:=156,       (* Master position is the modulo position of the MX virtual encoder *)
        MX_CAMSWITCH_LINEAR_VIRTUAL_ENCODER_MX:=157,       (* Master position is the linear position of the MX virtual encoder *)
        MX_CAMSWITCH_RECEIVE_PDO_1:=6,                     (* Master position is the SBus receive objekt PDO1 *)
        MX_CAMSWITCH_RECEIVE_PDO_2:=8,                     (* Master position is the SBus receive objekt PDO2 *)
        MX_CAMSWITCH_VAR_INPUT_2:=10,                      (* Master position is the Variablen Input 2 *)
        MX_CAMSWITCH_VAR_INPUT_3:=12,                      (* Master position is the Variablen Input 3 *)
        MX_CAMSWITCH_VAR_INPUT_4:=14,                      (* Master position is the Variablen Input 4 *)
        MX_CAMSWITCH_VAR_INPUT_5:=16,                      (* Master position is the Variablen Input 5 *)
        MX_CAMSWITCH_VAR_INPUT_6:=18,                      (* Master position is the Variablen Input 6 *)
        MX_CAMSWITCH_VAR_INPUT_7:=20,                      (* Master position is the Variablen Input 7 *)
        MX_CAMSWITCH_PSG_GEAR:=49,                         (* Master position is the Output Gear off the Position setpoint generator*)
        MX_CAMSWITCH_PSG_CAM:=50                           (* Master position is the Output Cam off the Position setpoint generator*)
    );
END_TYPE
```

64598axx

MC_CAMSWITCH_STATE_MX

```
TYPE MC_CAMSWITCH_STATE_MX :
    (
        MX_CAMSWITCH_INACTIVE,
        MX_CAMSWITCH_ERROR_LENGHT,
        MX_CAMSWITCH_ERROR_TYP,
        MX_CAMSWITCH_ERROR_VERSION,
        MX_CAMSWITCH_ERROR_DATASOURCE,
        MX_CAMSWITCH_ERROR_CAMNUMBER,
        MX_CAMSWITCH_ERROR_CAMAREA,
        MX_CAMSWITCH_ERROR_CAMTRACK,
        MX_CAMSWITCH_NOCOMPAREVALUE,
        MX_CAMSWITCH_INIT,
        MX_CAMSWITCH_ACTIVE
    );
END_TYPE
(*  MX_CAMSWITCH_INACTIVE              0 =    off
    MX_CAMSWITCH_ERROR_LENGHT          1 =    false lenght error 38 suberror 184
    MX_CAMSWITCH_ERROR_TYP             2 =    false typ  error 38 suberror 185
    MX_CAMSWITCH_ERROR_VERSION         3 =    false version  error 38 suberror 186
    MX_CAMSWITCH_ERROR_DATASOURCE      4 =    false datasource  error 38 suberror 187
    MX_CAMSWITCH_ERROR_CAMNUMBER       5 =    false cam number  error 38 suberror 188
    MX_CAMSWITCH_ERROR_CAMAREA         6 =    false cam area  error 38 suberror 189
    MX_CAMSWITCH_ERROR_CAMTRACK        7 =    false cam track  error 38 suberror 190
    MX_CAMSWITCH_NOCOMPAREVALUE        8 =    not enough comparison value
    MX_CAMSWITCH_INIT                  9 =    initialisation
    MX_CAMSWITCH_ACTIVE                10=    activ*)
```

64599axx

MC_CAMSWITCH_EXTENDED_CONFIG_MX

```
TYPE MC_CAMSWITCH_EXTENDED_CONFIG_MX :
STRUCT
    Dummy : BOOL;   (*prepared for further use*)
END_STRUCT
END_TYPE
```

64600axx

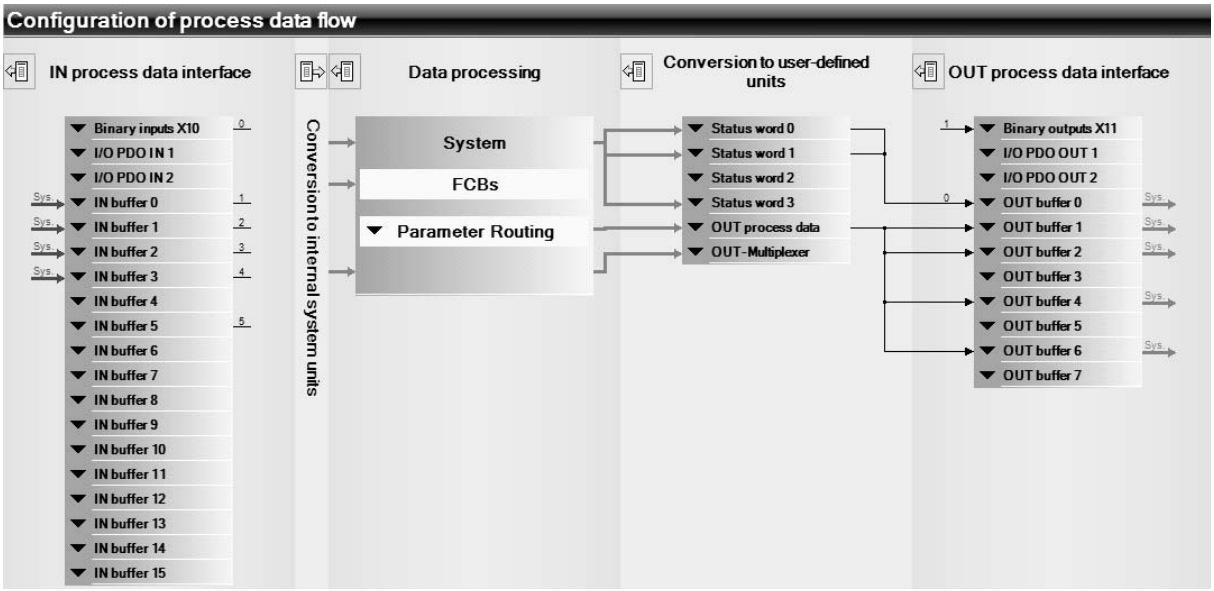
MC_CAMSWITCH_EXTENDED_DIAG_MX

```
TYPE MC_CAMSWITCH_EXTENDED_DIAG_MX :
STRUCT
    Dummy : BOOL;       (*prepared for further use*)
END_STRUCT
END_TYPE
```

64601axx

4.4.3 PDO configuration

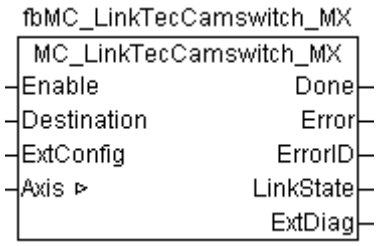
In addition to the configuration made by "MC_ConnectAxis_MX", the "MC_LinkTecCamSwitch_MX" module also performs some settings at the process data interface. The result of the tracks is written to *Out Process Data Low Word 9*. Depending on the destination, it is routed to the hardware outputs, see data type "MC_CAMSWITCH_DESTINATION_MX".



64602axx

4.5 The function modules of the MPLCTecCamSwitch_MX library

4.5.1 MC_LinkTecCamSwitch_MX



64603axx

Description:

The "MC_LinkTecCamSwitch_MX" function module is the logical interface between the axis and the "Cam controller" technology function. Once "ConnectAxis" has established the cyclical communication with the axis and after the unit has been enabled, the individual MOVIAXIS® technology functions are first configured and the ones necessary for the cam controller are activated.

Further, the process data interface is set-up (OUT-Low Word 9), and the routing to the hardware outputs is established.

Important:

The "MC_LinkTecCamSwitch_MX" module is absolutely essential for using the cam controller. It must be called cyclically in the program.

If *LinkState* = NotLinked, none of the function modules of the "MPLCTecCamSwitch_MX" can be executed, a corresponding error message is issued.

Prerequisite:

MOVI-PLC® *advanced* in technology version T1 or higher.

ConnectAxis for corresponding axes is integrated in the cyclical program.

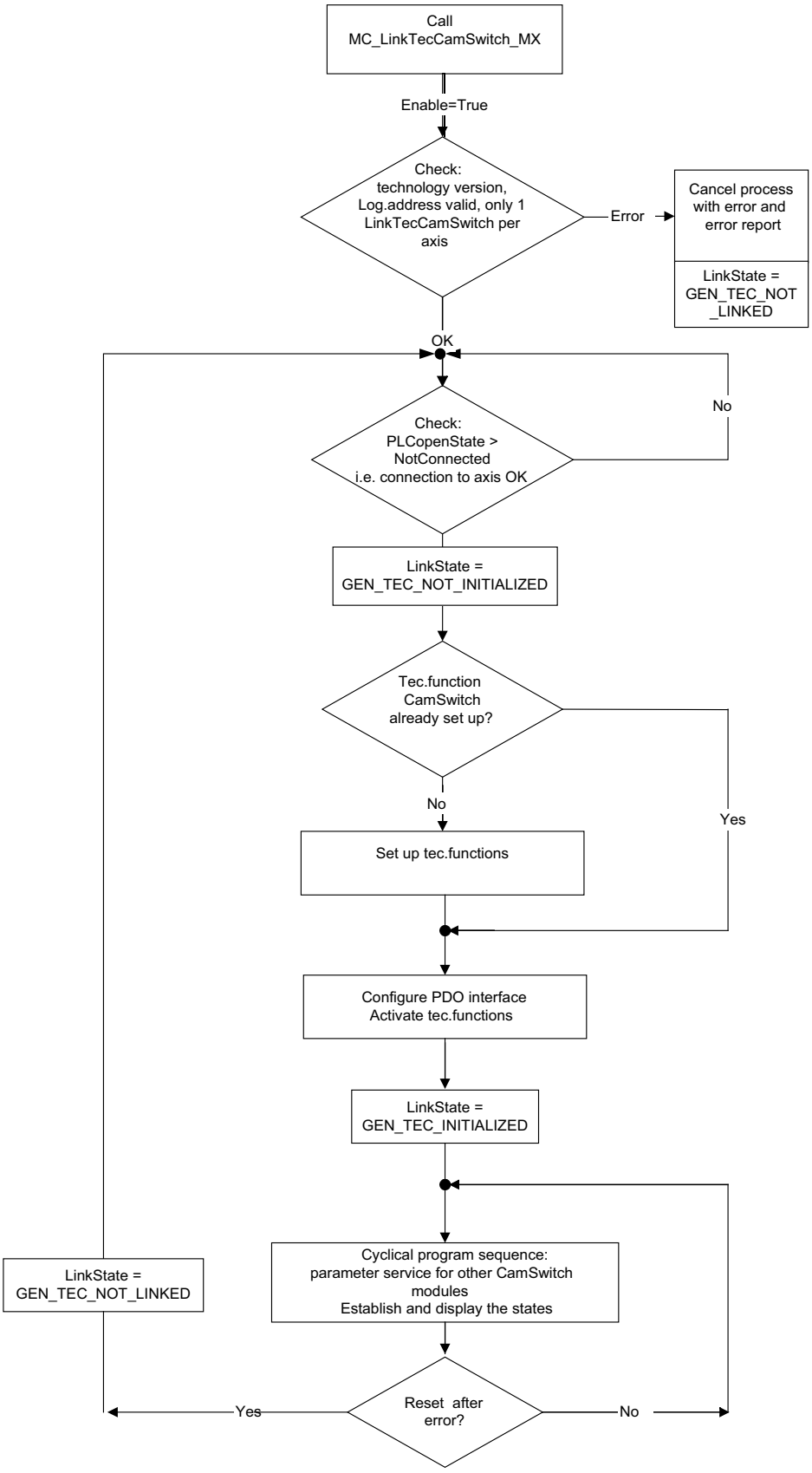
Inputs:

Name	Type	Meaning
Enable	BOOL	The module is processed as long as Enable „true“. Initialization is performed with a rising edge.
Destination	MC_CAMSWITCH_DESTINATION_MX	MX_CAMSWITCH_DEST_NONE = 0MX_CAMSWITCH_DEST_X11 = 1MX_CAMSWITCH_DEST_OPT1 = 2MX_CAMSWITCH_DEST_OPT2 = 4MX_CAMSWITCH_DEST_X11_OPT1 = 3MX_CAMSWITCH_DEST_X11_OPT2 = 5MX_CAMSWITCH_DEST_OPT1_OPT2 = 6MX_CAMSWITCH_DEST_X11_OPT1_OPT2= 7 Explanation of the aabreviations: X11 – Standard outputs X11 D0–D3 OPT1 – Option card 1 XIO11A / XIA11A OPT2 – Option card 2 XIO11A / XIA11A
ExtConfig	MC_CAMSWITCH_EXTENDED_CONFIG_MX	Extended configuration
Axis	AXIS_REF	Logical address of the axis that is to be addressed by the function block.

Outputs:

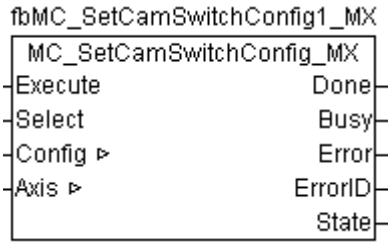
Name	Type	Meaning
Done	BOOL	Initialization successfully completed, connection with Cam technology function established
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 163
LinkState	MC_LINKTECSTATE	Current status: GEN_TEC_NOTLINKED (no connection) GEN_TEC_NOTINITIALISED (ho initialization) GEN_TEC_INITIALISED (Cam switch technology function initialized)
ExtDiag	MC_CAMSWITCH_EXTENDED_DIAG_MX	Extended diagnostics

MC_LinkTecCam_MX: Initialization



64604axx

4.5.2 MC_SetCamSwitchConfig_MX



64605axx

Description:

This module sets basic parameters for the respective cam track.

The track number is transferred with the *Select* input, and the track parameters are transferred with the *Config* ARRAY. Each track requires an own instance of the module. The parameters are taken over and written to the respective locations in the MOVIAXIS® with the rising edge at the execute.

Recommendation:

Create an array for the initialization of the config. parameters.

Example: CamTrack: ARRAY[1..8] OF MC_CAMSWITCH_CONFIG_MX.

The parameters of the array are descibed with an initialization module.

Each track can be assigned a maximum of 16 cam references. There are 32 cams (areas) available. They can also be assigned several tracks.

Prerequisite:

MC_LinkTecCamSwitch_MX has the "CamSwitch" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

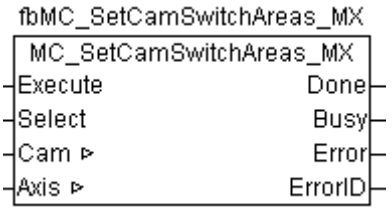
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module with a rising edge
Select	UINT	1 = Download parameter Cam track 1 2 = Download parameter cam track 2 3 = Download parameter cam track 3 4 = Download parameter cam track 4 5 = Download parameter cam track 5 6 = Download parameter cam track 6 7 = Download parameter cam track 7 8 = Download parameter cam track 8
Config	MC_CAMSWITCH_CONFIG_MX	Mode: 0= deactivated / 1= activated ModeControl : 0 - Standard cam track 1 - Modulo cam track source : Data source for the track, see MC_CAMSWITCH_SOURCE_MX DeadTime: -500.000 [µs] - +500.000 [µs] in steps of 100 µs TimeFrame: 0-16 ms MinModulo: minimum modulo value MaxModulo: maximum modulo value Hysteresis: Hysteresis of the cams SelectArea[1] – number of cam * SelectArea[2] – number of cam * .... SelectArea[16] – number of cam * * The number of the cam (1 – 32) is displayed.
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	All parameters have been transferred successfully
Busy	BOOL	Parameters are being transferred
Error	BOOL	Fault in module during execution
ErrorID	UINT	See the section Error identifier
State	MC_CAMSWITCH_STATE_MX	MX_CAMSWITCH_INACTIVE = 0 MX_CAMSWITCH_ERROR_LENGTH = 1 MX_CAMSWITCH_ERROR_TYP = 2 MX_CAMSWITCH_ERROR_VERSION = 3 MX_CAMSWITCH_ERROR_DATASOURCE =4 MX_CAMSWITCH_ERROR_CAMNUMBER = 5 MX_CAMSWITCH_ERROR_CAMAREA = 6 MX_CAMSWITCH_ERROR_CAMTRACK = 7 MX_CAMSWITCH_NOCOMPAREVALUE = 8 MX_CAMSWITCH_INIT = 9 MX_CAMSWITCH_ACTIVE = 10

4.5.3 MC_SetCamSwitchAreas_MX



64607axx

Description:

This module is for transferring the cam data for max. 32 cams (areas).

The *Select* input is used for transferring the cam number and the ARRAY is used for transferring the cam data. If the select input is not assigned or the value 0 is transferred, all parameters of the 32 cams are transferred.

The parameters are taken over and written to the respective locations in the MOVIAXIS® with the rising edge at the execute.

Prerequisite:

MC_LinkTecCamSwitch_MX has the "CamSwitch" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

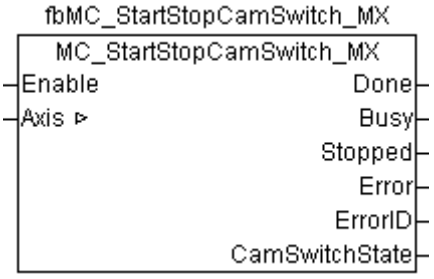
Inputs:

Name	Type	Meaning
Execute	BOOL	This input starts the task of the module.
Select	UINT	0 – Download all areas (cams) 1 – Download parameters area 1 2 – Download parameters area 2 … 32– Download parameter area 32
Cam	Array [1..32] OFMC_CAMSWITCH_AREAS_MX	Cam[1].LeftLimit – left cam limit Cam[1].RightLimit – right cam limit Cam[1].Direction – efficiency limit 0 – Off 1 – from left 2 – from right 3 – both directions ... Cam[...].LeftLimit – left cam limit Cam[...].RightLimit – right cam limit Cam[...].Direction – efficiency limit ... Cam[32].LeftLimit – left cam limit Cam[32].RightLimit – right cam limit Cam[32].Direction – efficiency limit
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	All parameters have been transferred successfully
Busy	BOOL	Parameters are being transferred
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 163

4.5.4 MC_StarStopCamSwitch_MX



64609axx

Description:

The "MC_StartStopCamSwitch" module activates the cam controller with *Enable* = True and deactivates the cam controller with *Enable* = False.

Important:

The *CamSwitchState* is only updated with the positive or negative edge of enable.

Prerequisite:

MC_LinkTecCamSwitch_MX has the CamSwitch technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

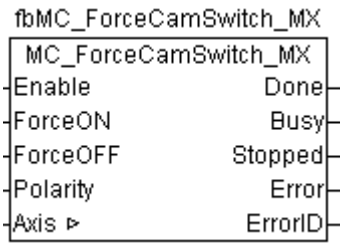
Inputs:

Name	Type	Meaning
Enable	BOOL	True = cam controller activation False = cam controller deactivation
Axis	AXIS_REF	Logical address of the axis that is to addressed by the function blocks.

Outputs:

Name	Type	Meaning
Done	BOOL	Cam controller activated
Busy	BOOL	Parameters are being transferred
Stopped	BOOL	Cam controller deactivated
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 163
CamSwitchState	MC_CAMSWITCH_STATE_MX	MX_CAMSWITCH_INACTIVE = 0 MX_CAMSWITCH_ERROR_LENGTH = 1 MX_CAMSWITCH_ERROR_TYP = 2 MX_CAMSWITCH_ERROR_VERSION = 3 MX_CAMSWITCH_ERROR_DATASOURCE =4 MX_CAMSWITCH_ERROR_CAMNUMBER = 5 MX_CAMSWITCH_ERROR_CAMAREA = 6 MX_CAMSWITCH_ERROR_CAMTRACK = 7 MX_CAMSWITCH_NOCOMPAREVALUE = 8 MX_CAMSWITCH_INIT = 9 MX_CAMSWITCH_ACTIVE = 10

4.5.5 MC_ForceCamSwitch_MX



64611axx

Description:

With the "MC_ForceCamSwitch" module, you can set or reset the outputs of the cam controller individually. Further, you can reverse the polarity.

If the *Enable* input is logically one, the parameters are written. If the input is logically zero, the setting/resetting, and the reversal of the logic are canceled.

Important:

If an output is set and the logic is reversed at the same time, the output is reset permanently. That means that the logic is reversed according to the manual specification of the process output data (forcing).

Prerequisite:

"MC_LinkTecCamSwitch_MX" has the "CamSwitch" technology function set-up.

LinkState must not be "GEN_TEC_NOTLINKED".

Inputs:

Name	Type	Meaning
Enable	BOOL	True = Forcing False = no forcing
ForceON	Byte	Track 1-8 <---> bit 0-7 bit = 0 output not set bit = 1 track is permanently true
ForceOFF	Byte	Track 1-8 <---> bit 0-7 bit = 0 output not reset bit = 1 track is permanently false
Polarity	Byte	Track 1-8 <---> bits 0-7 bit = 0 no reversal bit = output is reversed (XOR)
Axis	AXIS_REF	Logical address of the axis that is to be addressed by the function block.

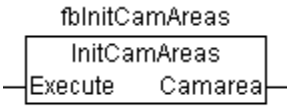
Outputs:

Name	Type	Meaning
Done	BOOL	Forcing
Busy	BOOL	Parameters are being transferred
Stopped	BOOL	No forcing
Error	BOOL	Fault in module during execution
ErrorID	UINT	see page 163

4.6 Example

4.6.1 8 tracks and 32 cams

In the "InitCamAreas" module, you have to set the initialization parameters for the cams.



64612axx

Declaration

```
FUNCTION_BLOCK InitCamAreas
VAR_INPUT
    Execute:BOOL;
END_VARVAR_OUTPUT
    Camarea: ARRAY[1..32] OF MC_CAMSWITCH_AREAS_MX;
END_VAR
VAR
    EXE: BOOL;
    bREdgeexecute: BOOL;
END_VAR
```

Program

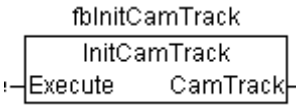
```
EXE:=Execute AND NOT bREdgeexecute;
bREdgeexecute:=Execute;
IF EXE  THEN
    (*area 1*)
    Camarea[1].LeftLimit :=1000;
    Camarea[1].RightLimit :=2000;
    Camarea[1].Direction :=3;
    (*area 2*)
    Camarea[2].LeftLimit :=3000;
    Camarea[2].RightLimit :=4000;
    Camarea[2].Direction :=3;
    (*area 3*)
    Camarea[3].LeftLimit :=5000;
    Camarea[3].RightLimit :=6000;
    Camarea[3].Direction :=3;
    (*area 4*)
    Camarea[4].LeftLimit :=7000;
    Camarea[4].RightLimit :=8000;
    Camarea[4].Direction :=3;
    (*area 5*)
    Camarea[5].LeftLimit :=9000;
    Camarea[5].RightLimit :=10000;
    Camarea[5].Direction :=3;
    (*area 6*)
```

```
Camarea[6].LeftLimit :=11000;
Camarea[6].RightLimit :=12000;
Camarea[6].Direction :=3;
(*area 7*)
Camarea[7].LeftLimit :=13000;
Camarea[7].RightLimit :=14000;
Camarea[7].Direction :=3;
(*area 8*)
Camarea[8].LeftLimit :=15000;
Camarea[8].RightLimit :=16000;
Camarea[8].Direction :=3;
(*area 9*)
Camarea[9].LeftLimit :=17000;
Camarea[9].RightLimit :=18000;
Camarea[9].Direction :=3;
(*area 10*)
Camarea[10].LeftLimit :=19000;
Camarea[10].RightLimit :=20000;
Camarea[10].Direction :=3;
(*area 11*)
Camarea[11].LeftLimit :=21000;
Camarea[11].RightLimit :=22000;
Camarea[11].Direction :=3;
(*area 12*)
Camarea[12].LeftLimit :=23000;
Camarea[12].RightLimit :=24000;
Camarea[12].Direction :=3;
(*area 13*)
Camarea[13].LeftLimit :=25000;
Camarea[13].RightLimit :=26000;
Camarea[13].Direction :=3;
(*area 14*)
Camarea[14].LeftLimit :=27000;
Camarea[14].RightLimit :=28000;
Camarea[14].Direction :=3;
(*area 15*)
Camarea[15].LeftLimit :=29000;
Camarea[15].RightLimit :=30000;
Camarea[15].Direction :=3;
(*area 16*)
Camarea[16].LeftLimit :=31000;
Camarea[16].RightLimit :=32000;
Camarea[16].Direction :=3;
```

```
(*area 17*)
Camarea[17].LeftLimit :=33000;
Camarea[17].RightLimit :=34000;
Camarea[17].Direction :=3;
(*area 18*)
Camarea[18].LeftLimit :=35000;
Camarea[18].RightLimit :=36000;
Camarea[18].Direction :=3;
(*area 19*)
Camarea[19].LeftLimit :=37000;
Camarea[19].RightLimit :=38000;
Camarea[19].Direction :=3;
(*area 20*)
Camarea[20].LeftLimit :=39000;
Camarea[20].RightLimit :=40000;
Camarea[20].Direction :=3;
(*area 21*)
Camarea[21].LeftLimit :=41000;
Camarea[21].RightLimit :=42000;
Camarea[21].Direction :=3;
(*area 22*)
Camarea[22].LeftLimit :=43000;
Camarea[22].RightLimit :=44000;
Camarea[22].Direction :=3;
(*area 23*)
Camarea[23].LeftLimit :=45000;
Camarea[23].RightLimit :=46000;
Camarea[23].Direction :=3;
(*area 24*)
Camarea[24].LeftLimit :=47000;
Camarea[24].RightLimit :=48000;
Camarea[24].Direction :=3;
(*area 25*)
Camarea[25].LeftLimit :=49000;
Camarea[25].RightLimit :=50000;
Camarea[25].Direction :=3;
(*area 26*)
Camarea[26].LeftLimit :=51000;
Camarea[26].RightLimit :=52000;
Camarea[26].Direction :=3;
(*area 27*)
Camarea[27].LeftLimit :=53000;
Camarea[27].RightLimit :=54000;
```

```
    Camarea[27].Direction :=3;
    (*area 28*)
    Camarea[28].LeftLimit :=55000;
    Camarea[28].RightLimit :=56000;
    Camarea[28].Direction :=3;
    (*area 29*)
    Camarea[29].LeftLimit :=57000;
    Camarea[29].RightLimit :=58000;
    Camarea[29].Direction :=3;
    (*area 30*)
    Camarea[30].LeftLimit :=59000;
    Camarea[30].RightLimit :=60000;
    Camarea[30].Direction :=3;
    (*area 31*)
    Camarea[31].LeftLimit :=61000;
    Camarea[31].RightLimit :=62000;
    Camarea[31].Direction :=3;
    (*area 32*)
    Camarea[32].LeftLimit :=63000;
    Camarea[32].RightLimit :=64000;
    Camarea[32].Direction :=3;END_IF
```

In the "InitCamTrack" module, you have to set the initialization parameters for the tracks.



64613axx

Declaration

```
FUNCTION_BLOCK InitCamTrack
VAR_INPUT
    Execute:BOOL;
END_VAR
VAR_OUTPUT
    CamTrack: ARRAY[1..8] OF MC_CAMSWITCH_CONFIG_MX;
END_VAR
VAR
    EXE: BOOL;
    bREdgeexecute: BOOL;
END_VAR
```

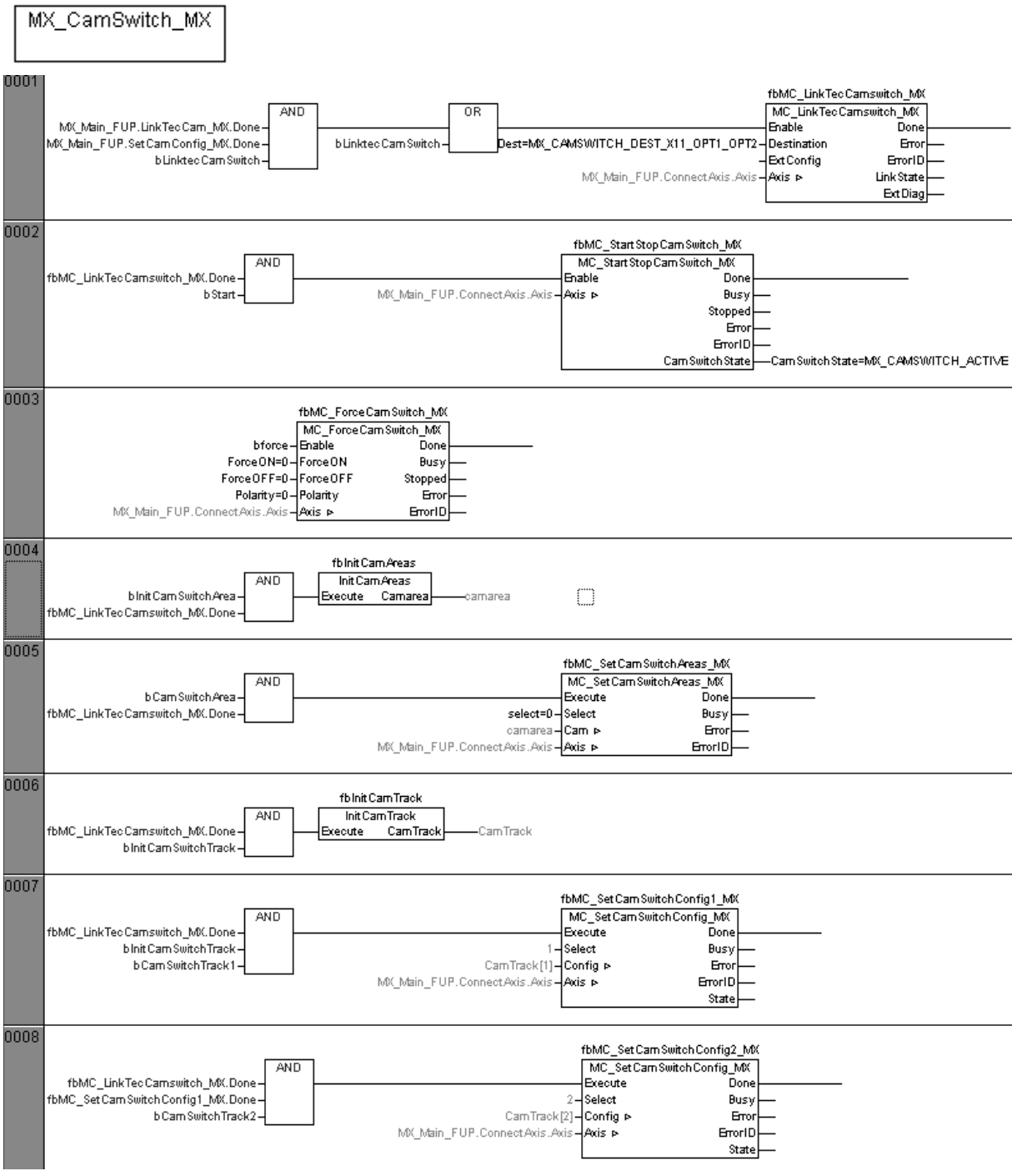
Program

```
EXE:=Execute AND NOT bREdgeexecute;
bREdgeexecute:=Execute;
IF EXE  THEN
    (*track 1*)
    CamTrack[1].Mode :=TRUE;
    CamTrack[1].ModeControl :=FALSE;
    CamTrack[1].DataSource :=MX_CAMSWITCH_SYSTEMPOS_ENCODER1;
    CamTrack[1].DeadTime :=0;
    CamTrack[1].TimeFrame :=1;
    CamTrack[1].MinModuloValue :=0;
    CamTrack[1].MaxModuloValue := 131072;
    CamTrack[1].Hysteresis :=0;
    CamTrack[1].SelectArea[1]:=1;
    CamTrack[1].SelectArea[2]:=2;
    CamTrack[1].SelectArea[3]:=3;
    CamTrack[1].SelectArea[4]:=4;
    CamTrack[1].SelectArea[5]:=5;
    CamTrack[1].SelectArea[6]:=6;
    CamTrack[1].SelectArea[7]:=23;
    CamTrack[1].SelectArea[8]:=24;
    CamTrack[1].SelectArea[9]:=25;
    CamTrack[1].SelectArea[10]:=26;
    CamTrack[1].SelectArea[11]:=27;
    CamTrack[1].SelectArea[12]:=28;
    CamTrack[1].SelectArea[13]:=29;
    CamTrack[1].SelectArea[14]:=30;
    CamTrack[1].SelectArea[15]:=31;
    CamTrack[1].SelectArea[16]:=32;
        (*track 2*)
    CamTrack[2].Mode :=TRUE;
    CamTrack[2].ModeControl :=FALSE;
    CamTrack[2].DataSource :=MX_CAMSWITCH_SYSTEMPOS_ENCODER1;
    CamTrack[2].DeadTime :=0;
    CamTrack[2].TimeFrame :=1;
    CamTrack[2].MinModuloValue :=0;
    CamTrack[2].MaxModuloValue := 131072;
    CamTrack[2].Hysteresis :=0;
    CamTrack[2].SelectArea[1]:=3;
    CamTrack[2].SelectArea[2]:=4;
    (*track 3*)
    CamTrack[3].Mode :=TRUE;
    CamTrack[3].ModeControl :=FALSE;
    CamTrack[3].DataSource :=MX_CAMSWITCH_SYSTEMPOS_ENCODER1;
```

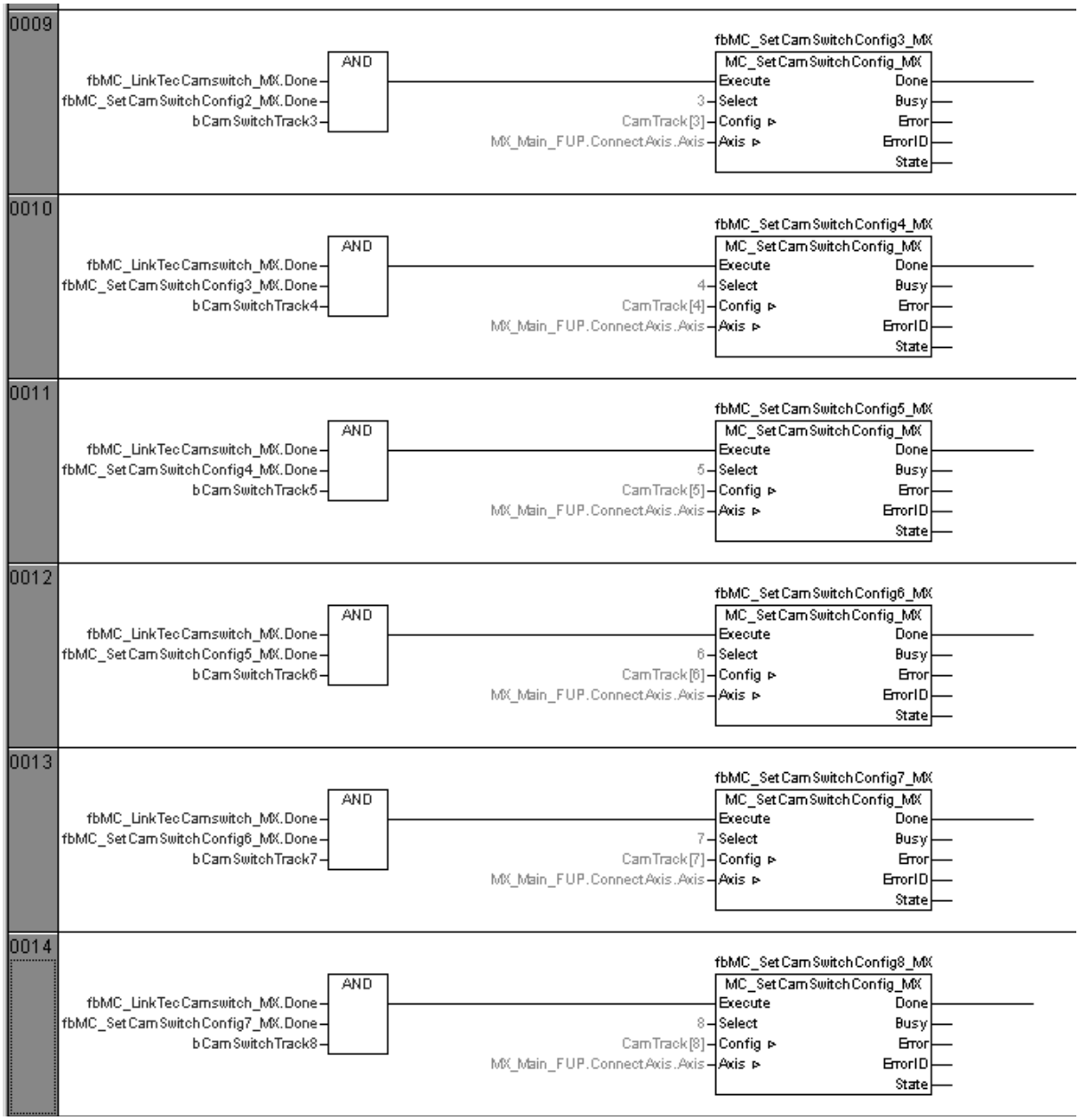
```
CamTrack[3].DeadTime :=0;
CamTrack[3].TimeFrame :=1;
CamTrack[3].MinModuloValue :=0;
CamTrack[3].MaxModuloValue := 131072;
CamTrack[3].Hysteresis :=0;
CamTrack[3].SelectArea[1]:=5;
CamTrack[3].SelectArea[2]:=6;
(*track 4*)
CamTrack[4].Mode :=TRUE;
CamTrack[4].ModeControl :=FALSE;
CamTrack[4].DataSource :=MX_CAMSWITCH_SYSTEMPOS_ENCODER1;
CamTrack[4].DeadTime :=0;
CamTrack[4].TimeFrame :=1;
CamTrack[4].MinModuloValue :=0;
CamTrack[4].MaxModuloValue := 131072;
CamTrack[4].Hysteresis :=0;
CamTrack[4].SelectArea[1]:=7;
CamTrack[4].SelectArea[2]:=8;
    (*track 5*)
CamTrack[5].Mode :=TRUE;
CamTrack[5].ModeControl :=FALSE;
CamTrack[5].DataSource :=MX_CAMSWITCH_SYSTEMPOS_ENCODER1;
CamTrack[5].DeadTime :=0;
CamTrack[5].TimeFrame :=1;
CamTrack[5].MinModuloValue :=0;
CamTrack[5].MaxModuloValue := 131072;
CamTrack[5].Hysteresis :=0;
CamTrack[5].SelectArea[1]:=1;
CamTrack[5].SelectArea[2]:=2;
    (*track 6*)
CamTrack[6].Mode :=TRUE;
CamTrack[6].ModeControl :=FALSE;
CamTrack[6].DataSource := MX_CAMSWITCH_MODULOPOS_ENCODER1;
CamTrack[6].DeadTime :=0;
CamTrack[6].TimeFrame :=1;
CamTrack[6].MinModuloValue :=0;
CamTrack[6].MaxModuloValue := 10000;
CamTrack[6].Hysteresis :=0;
CamTrack[6].SelectArea[1]:=3;
CamTrack[6].SelectArea[2]:=4;
(*track 7*)
CamTrack[7].Mode :=TRUE;
CamTrack[7].ModeControl :=FALSE;
```

```
    CamTrack[7].DataSource := MX_CAMSWITCH_MODULOPOS_ENCODER1;
    CamTrack[7].DeadTime :=0;
    CamTrack[7].TimeFrame :=1;
    CamTrack[7].MinModuloValue :=0;
    CamTrack[7].MaxModuloValue := 10000;
    CamTrack[7].Hysteresis :=0;
    CamTrack[7].SelectArea[1]:=5;
    CamTrack[7].SelectArea[2]:=6;
    (*track 8*)
    CamTrack[8].Mode :=TRUE;
    CamTrack[8].ModeControl :=FALSE;
    CamTrack[8].DataSource :=MX_CAMSWITCH_USERPOS_ENCODER1;
    CamTrack[8].DeadTime :=0;
    CamTrack[8].TimeFrame :=1;
    CamTrack[8].MinModuloValue :=0;
    CamTrack[8].MaxModuloValue := 20000;
    CamTrack[8].Hysteresis :=0;
    CamTrack[8].SelectArea[1]:=1;
    CamTrack[8].SelectArea[2]:=2;
    CamTrack[8].SelectArea[3]:=3;
    CamTrack[8].SelectArea[4]:=4;
    CamTrack[8].SelectArea[5]:=5;
    CamTrack[8].SelectArea[6]:=6;
    CamTrack[8].SelectArea[7]:=7;
    CamTrack[8].SelectArea[8]:=8;
    CamTrack[8].SelectArea[9]:=9;
    CamTrack[8].SelectArea[10]:=10;
    CamTrack[8].SelectArea[11]:=11;
    CamTrack[8].SelectArea[12]:=12;
END_IF
```

Creating and cyclically calling the "MX_CamSwitch_MX" module.



64614axx


0009
fbMC_LinkTecCamswitch_MX.Done
fbMC_SetCamSwitchConfig2_MX.Done
bCamSwitchTrack3
AND
fbMC_SetCamSwitchConfig3_MX
MC_SetCamSwitchConfig_MX
Execute
Select
Config
Axis
Done
Busy
Error
ErrorID
State
3
CamTrack[3]
MX_Main_FUP.ConnectAxis.Axis
0010
fbMC_LinkTecCamswitch_MX.Done
fbMC_SetCamSwitchConfig3_MX.Done
bCamSwitchTrack4
AND
fbMC_SetCamSwitchConfig4_MX
4
CamTrack[4]
0011
fbMC_LinkTecCamswitch_MX.Done
fbMC_SetCamSwitchConfig4_MX.Done
bCamSwitchTrack5
AND
fbMC_SetCamSwitchConfig5_MX
5
CamTrack[5]
0012
fbMC_LinkTecCamswitch_MX.Done
fbMC_SetCamSwitchConfig5_MX.Done
bCamSwitchTrack6
AND
fbMC_SetCamSwitchConfig6_MX
6
CamTrack[6]
0013
fbMC_LinkTecCamswitch_MX.Done
fbMC_SetCamSwitchConfig6_MX.Done
bCamSwitchTrack7
AND
fbMC_SetCamSwitchConfig7_MX
7
CamTrack[7]
0014
fbMC_LinkTecCamswitch_MX.Done
fbMC_SetCamSwitchConfig7_MX.Done
bCamSwitchTrack8
AND
fbMC_SetCamSwitchConfig8_MX
8
CamTrack[8]

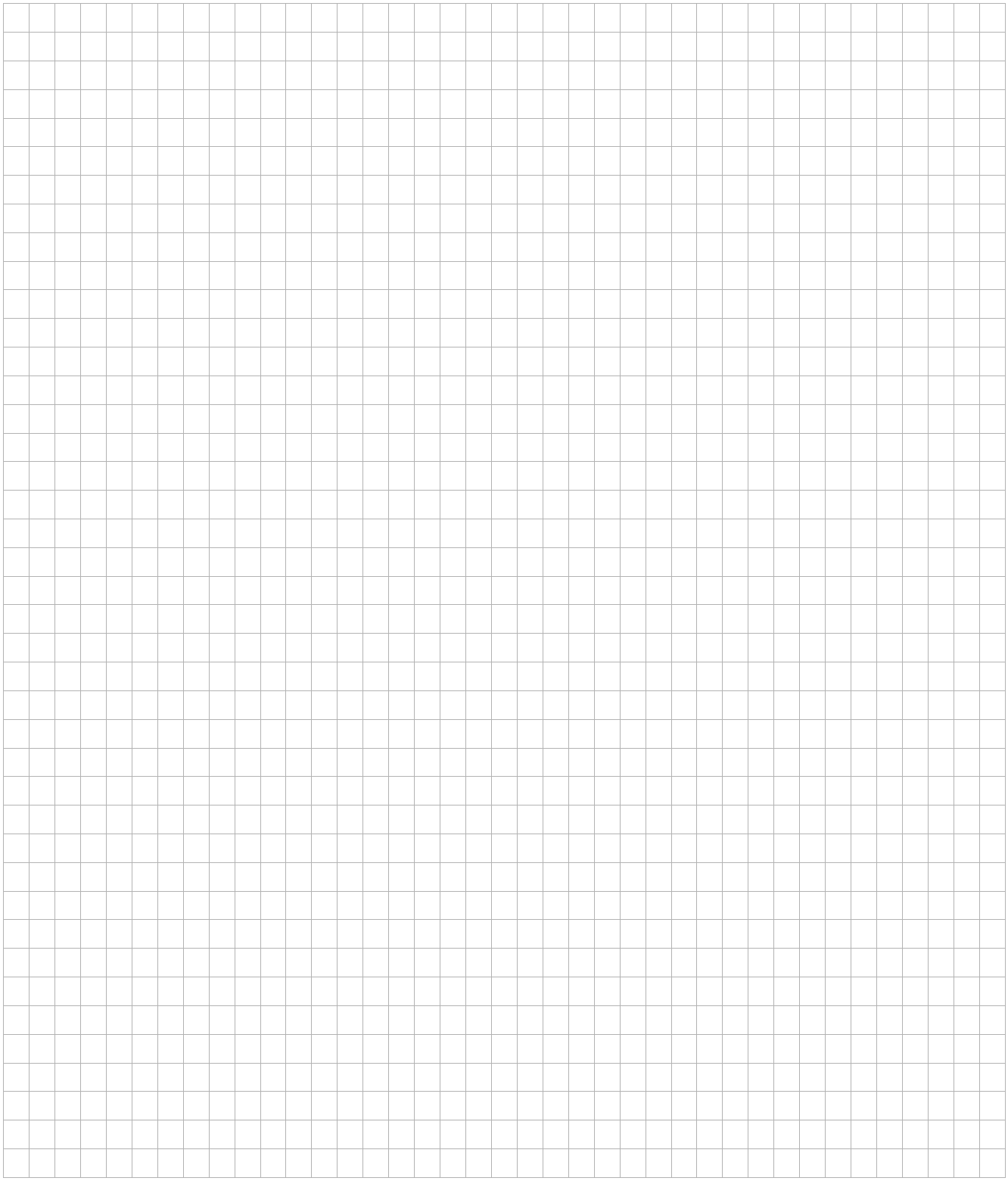
64615axx

4.7 Appendix

4.7.1 Fault detection (ErrorID)

Fault code	Error designation	Problem
FA0010	E_IEC_GENERAL_INVALID_TECHNOLOGIE_OPTION	Requested technology function not supported
FA020...	General link tec error	
FA0200	E_TEC_GENERAL_MULTIPLE_TECLINKS	The LinkTec FB may not be called repeatedly.
FA0201	E_TEC_GENERAL_INVALID_LINKSTATE	Invalid LinkState for function call
FA0202	E_TEC_GENERAL_NOT_LINKED	LinkTec FB has no logical connection
FA0203	E_TEC_GENERAL_NOT_INITIALIZED	Technology function not yet active
	IEC general error messages	
FA0070	E_IEC_PARAMETER_VALUE_OUT_OF_RANGE	Invalid entry value for parameter
FB0071	E_MDX_MOTIONBLOCK_LOG_ADR_NOT_INITIALIZED	MC_ConnectAxis_MDX has not yet assigned a log. address
FB0072	E_MDX_MOTIONBLOCK_INVALID_LOG_ADR	Invalid log. address
FB0073	E_MDX_MOTIONBLOCK_INVALID_STATE	Function call in current PLCopen state not permitted

5 Index



How we're driving the world

With people who think fast and develop the future with you.

With a worldwide service network that is always close at hand.

With drives and controls that automatically improve your productivity.

With comprehensive knowledge in virtually every branch of industry today.

With uncompromising quality that reduces the cost and complexity of daily operations.

SEW-EURODRIVE
Driving the world

With a global presence that offers responsive and reliable solutions. Anywhere.

With innovative technology that solves tomorrow's problems today.

With online information and software updates, via the Internet, available around the clock.

SEW-EURODRIVE GmbH & Co KG
P.O. Box 3023 · D-76642 Bruchsal / Germany
Phone +49 7251 75-0 · Fax +49 7251 75-1970
sew@sew-eurodrive.com

→ **www.sew-eurodrive.com**