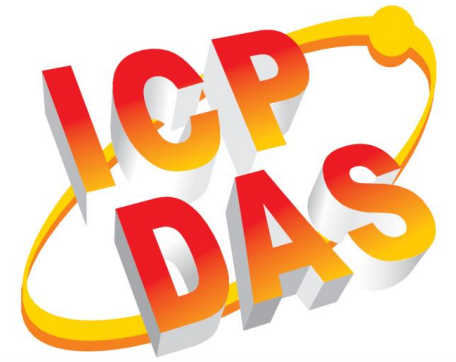
Motion Control Function (PLCopen)

for Win-GRAF

User Manual

(Version 1.1.1)



WARRANTY

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

WARNING

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

COPYRIGHT

TRADEMARK

Names are used for identification only and may be registered trademarks of their respective companies.

CONTACT US

If you have any questions, please feel free to contact us via email at:
service@icpdas.com
service.icpdas@gmail.com

Revision

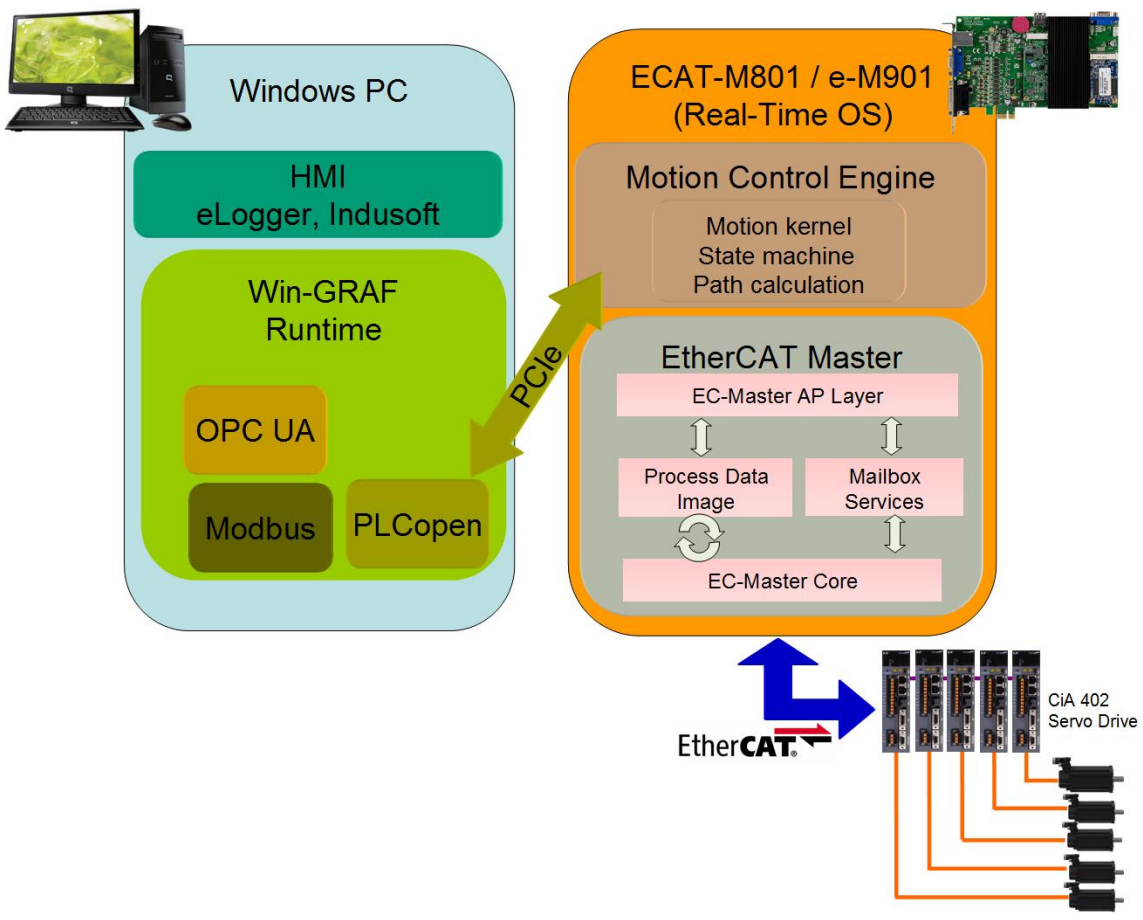
Revision	Date	Description	Author
1.0	25.04.2022	Initial version	M. K.
1.1	09.04.2024	Describe PDO function block Describe modified ECAT Utility UI	M. K.
1.1.1	30.12.2024	Fixed in correct figure and table linkage	M.K.

Contents

1 Product Overview

1.1 Introduction

The PLCopen function blocks described in this manual can only be used in conjunction with the EtherCAT master PCIe cards ECAT-M801 and the programmable automation controller EMP-9xx8-xx. The EMP-9xx8-xx has got a built-in master card which runs on a dedicated processor. The main motion control engine and path generator are implemented in the master cards which guarantees real time and deterministic behaviour. The Win-GRAF runtime for the standard Window OS system is not real time and it is therefore necessary for the runtime to rely on the real time characteristics of the master card for motion control. The PLCopen function of the runtime directly calls the motion function of the ECAT-M801/ EMP-9xx8-xx master card to control the execution path of each axis.



The ECAT-M801/EMP-9xx8-xx master card series support different number of axes (Table 1/Table 2). The axis can either be real (servo or stepper drives) or virtual. The supported number of axis of the Win-GRAF runtime is limited by the master card. To get the newest product release of the ECAT-M801/EMP-9xx8-xx series and the supported axis go to the ICPDAS website and search either for 'ECAT-M801' or ' EMP-9058'.

PCIe Card	Supported Axes Qty
ECAT-M801-8AX	8
ECAT-M801-16AX	16
ECAT-M801-32AX	32

Table 1: Supported axes number of the ECAT-M801 series

EMP-9xx8-xx	Supported Axes Qty
EMP-9xx8-16	16
EMP-9xx8-32	32

Table 2: Supported axes number of the EMP-9xx8-xx series

1.2 PLCopen Library

1.2.1 Workbench PLCopen Library

The PLCopen library installed together with the workbench. The library name is 'EtherCAT_PCI_Vxx' , the postfix 'Vxx' indicates the version number. The workbench library can be found in the following directory:
C:\Users\Public\Documents\Win-GRAF Workbench\Win-GRAF Wb 10.0\HwDef\ EtherCAT_PCI_V1a

In the future ICPDAS will extend the PLCopen motion control product series. Due to hardware restriction the implementation of some PLCopen function blocks may differ between platforms, e.g. the number of supported in and output parameters of a function block may vary. Therefore post-fixes are added the function blocks to allow the use to clearly identify the supported target platform for a set of PLCopen function blocks. For the ECAT-M801/EMP-9xx8-xx series a '_I' (Figure 1) is added to the end of each PLCopen define command.

Attention:
PLCopen names ending with '_I' should only be used in conjunction with the ECAT-M801/EMP-9xx8-xx series.

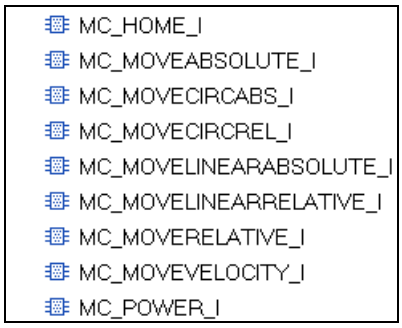


Figure 1: PLCopen names ending with '_I' are reserved for the ECAT-M801/EMP-9xx8-xx series

1.2.2 Runtime PLCopen Library

The runtime requires the following PLCopen libraries:

- ECAT8K.dll
- Fb_EcatPlcopen.dll
- libecatdevice.dll

These libraries files should be in the same directory as the runtime execution file. The runtime installation program will install these libraries by default to the runtime execution file directory.

1.3 Demo Program

Demo programs in several PLC programming language are provided to show how to initialize the EtherCAT master, set up the axis configurations and call the PLCopen function blocks.

The demo programs are listed in the following directory:

- *C:\Users\Public\Documents\Win-GRAF Workbench\Win-GRAF Wb xx.x\ Projects\EMP-9000\PLCopen*
- *C:\Users\Public\Documents\Win-GRAF Workbench\Win-GRAF Wb xx.x\ Projects\Windows PC\PLCopen*

The demo programs are installed together with the Win-GRAF workbench.

2 Motion Function Block (MFB) Definitions and Interfaces

2.1 Function Block Instance

Each motion function block (MFB) used in the application program has its own instance memory. The memory e.g. stores data from the previous cycle or stores the MFB input value after it has been triggered for execution. The data of the internal memory is not visible to programmer. The internal values of the instance memory persist from one execution of the function block to the next so that the status of the function block execution is retained.

2.2 Multiple Instances Per Axis

Multiple instances of an enabled motion function block (MFB) can refer to the same axis in a single application. If two function blocks of the same type attempt to operate on the same axis, they are processed in the order they are encountered during the execution of the application logic.
When a new instance invocation is encountered, the currently active instance for the function block type is terminated and the new instance invocation assumes control.

2.3 Administrative vs. Motion-Generating Functions and Function Blocks

Motion functions and function blocks are divided into two action types: Administrative and Motion.

- Administrative functions and function blocks do not cause axis motion, while
- motion functions and function blocks control axis motion.

Administrative	Single Axis Motion
MC_Power	MC_Home
MC_ReadStatus	MC_Stop
MC_ReadAxisError	MCV_Halt

Administrative	Single Axis Motion
MC_ReadParameter	MC_MoveAbsolute
MC_ReadBoolParameter	MC_MoveRelative
MC_WriteParameter	MC_MoveVelocity
MC_WriteBoolParameter	
MC_ReadDigitalInput	
MC_WriteDigitalOutput	
MC_ReadActualPosition	
MC_ReadActualVelocity	
MC_ReadAxisInfo	
MC_ReadMotionState	
MC_SetPosition	
MC_Reset	

Table 3: Single axis PLCopen Function Blocks

2.4 Function Block Triggering

PLCopen and ICPDAS defined function blocks are activated either by an '***Enable***' input (level-triggered) or by an '***Execute***' input (edge triggered).

- Function blocks that have an '*Enable*' input, the input parameters are applied when *Enable* is true.
- Function blocks that use an '*Execute*' input, the input parameters are applied with the rising edge of the '*Execute*' input parameter.

2.4.1 Executed (Edge-Triggered) Function Blocks

Motion generating function blocks, such as MC_Stop and MC_Home, are edge-triggered and operate only on the rising edge of the '*Execute*' input. The input parameters are only applied when the '*Execute*' input changes from FALSE to TRUE. To start the MFB execution again after it has finished the '*Execute*' input first has to be set to FALSE for one PLC cycle before calling the MFB with a TRUE value for the '*Execute*' input.

Input parameters are stored in the instance memory with the rising edge of the '*Execute*' input. To modify one or more input parameters, it is necessary to first set the '*Execute*' to FALSE for one cycle, change the input parameters and then trigger the function block again. It is important that the Execute input has to stay FALSE for at least one cycle.

The falling edge of '*Execute*' does not stop or influence the execution of the MFB (Figure 2). The action that was initiated by the MFB will continue until it is completed or until

another MFB interrupts or stops (e.g. MC_Stop) the action.

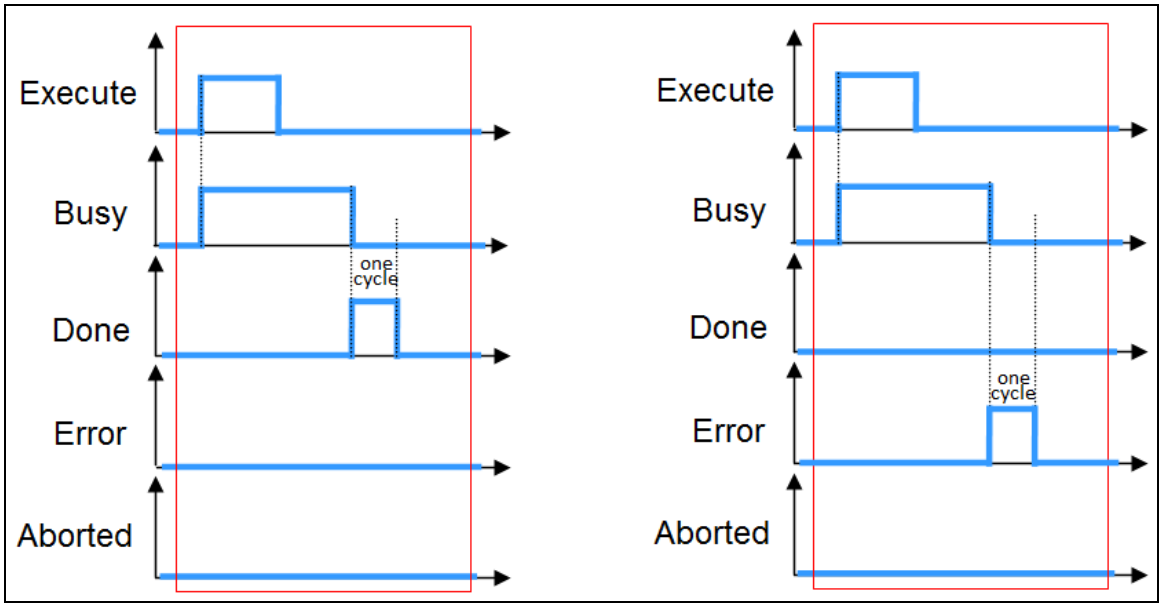


Figure 2: Falling edge of Execute does not stop the FB execution

2.4.1.1 Outputs of Executed (Edge-Triggered) Function Blocks

The outputs '*Busy*', '*Done*', '*Error*', and '*CommandAborted*' are mutually exclusive which means only one of them can be TRUE. If the input '*Execute*' is TRUE, one of these outputs has to be TRUE.

The '*Busy*' output indicates that the execution of the MFB is not complete, and new output states (values) are expected to be generated. The '*Busy*' output is reset to FALSE once the MFB execution has finished and one of the other outputs (Done, CommandAborted, Error) changes from FALSE to TRUE.

The falling edge of '*Execute*' resets the following outputs:
- Done
- Error
- ErrorID, if associated with an error
- CommandAborted

The values of the output arguments are set on (TRUE) for at least one cycle, even if the '*Execute*' transitions from TRUE to FALSE before the function block completed its action.

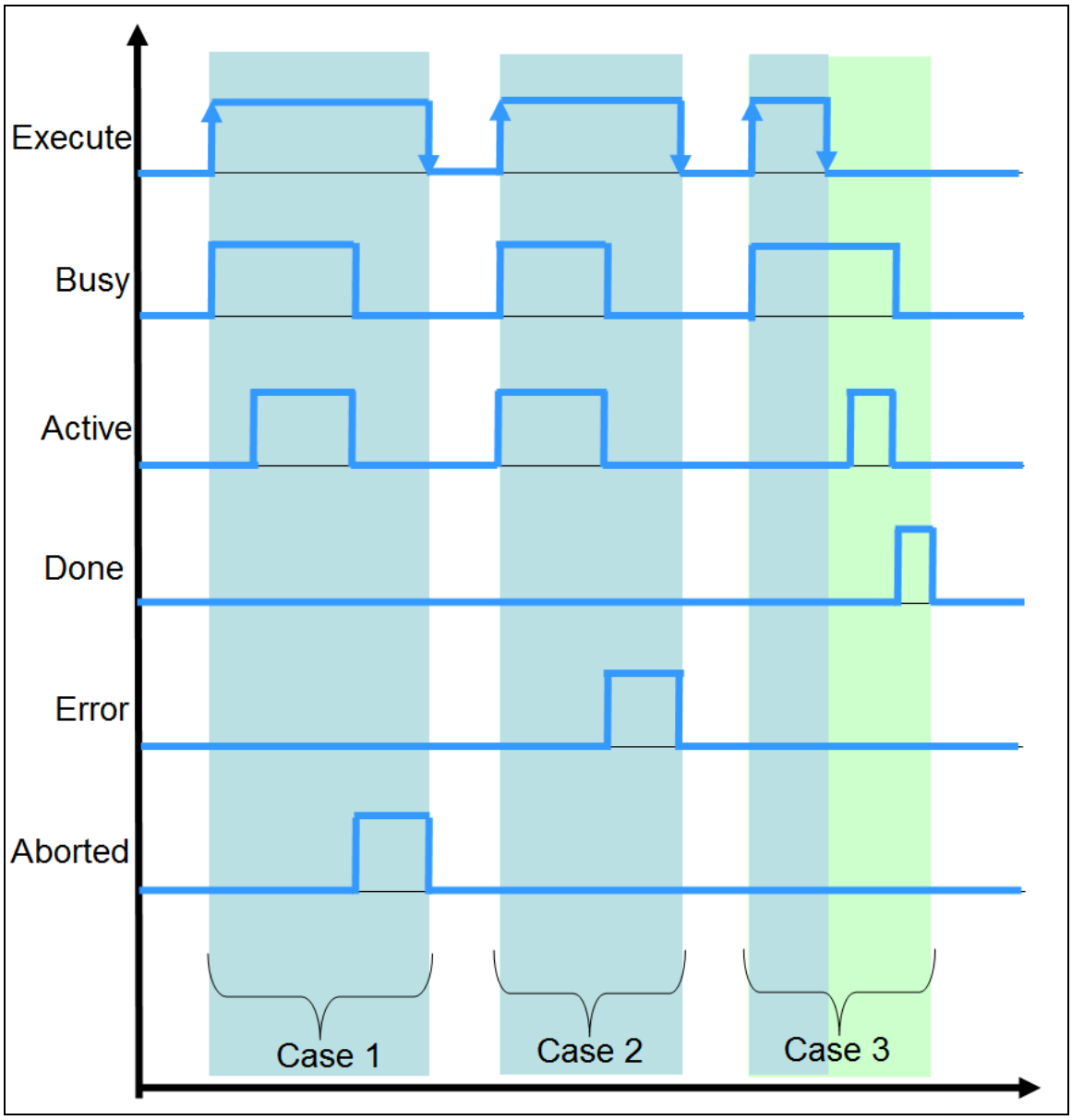


Figure 3: Behavior of Execute/Done as defined by PLCopen

2.4.1.2 Re-execution of a Function Block

If an instance of an executing MFB receives a new execute before it has finished its current command ('*Busy*'=TRUE), the MFB will simple ignore the execute trigger and continue running its current command until it has finished or being aborted (e.g. by a '*MC_Stop*' MFB). If the input values are changed and '*Execute*' is triggered again while the MFB is executing ('*Busy*'=TRUE), the '*Execute*' and all the other inputs will be ignored.

2.4.2 Enabled Function Blocks

The enabled MFBs are active only while the input parameter '*Enable*' is TRUE. The '*Enable*' parameter is used to perform cyclical actions, for example reading the current encoder position in every cycle ('*MC_ReadActualPosition*'). In contrast to '*Execute*' the '*Enable*' input results in an action being executed continuously, as long as Enable is TRUE.

- '*Enable*' input:
  - The '*Enable*' is level sensitive which means once the '*Enable*' is set to TRUE the MFB will be executed in every task cycle until the it is set to FALSE. The '*Valid*' output indicates whether the FB outputs are valid.
  - The input parameters are used with the rising edge of the '*Enable*' input and can be modified continuously.
  - The outputs '*Valid*' and '*Error*' are mutually exclusive: only one of them can be TRUE at a time.
  - The '*Valid*', '*Busy*', '*Error*', and '*ErrorID*' outputs are reset (to FALSE) with the falling edge of Enable as soon as possible.
- The '*Valid*' output is TRUE as long as a valid output value is available and the '*Enable*' input is TRUE. The relevant output value are updated in each cycle time as long as the input '*Enable*' is TRUE.
- MFB with an output '*Busy*' parameter, indicates that it needs more than one cycle time for a complete execution. The '*Busy*' output indicates that the execution of the MFB is not complete, and new outputs are expected to be generated in the next cycle. '*Busy*' is set at the rising edge of '*Enable*' and remains set as long as the MFB is performing any action.
- If there is a MFB error, the output is not valid ('*Valid*' set to FALSE). When the error condition disappears, the values will reappear and '*Valid*' output will be set ('*Valid*'=TRUE) again.

2.4.3 Timing example for the "Enable" input

Example 1 (Figure 2):

- Case 1: This shows the normal operation. The '*Valid*' output turns TRUE once the output parameter (e.g. status in '*MC_Power*') is valid.
- Case 2: An error occurred during the operation of the MFB while '*Enable*' is TRUE. During the time span when the error occurred the '*Error*' output is set to TRUE and '*Valid*' output to FALSE. The output '*Busy*' stays high. After the error has been reset, the normal operation procedure is restored. It may take some time for the '*Valid*' output to show TRUE after the error has been cleared.

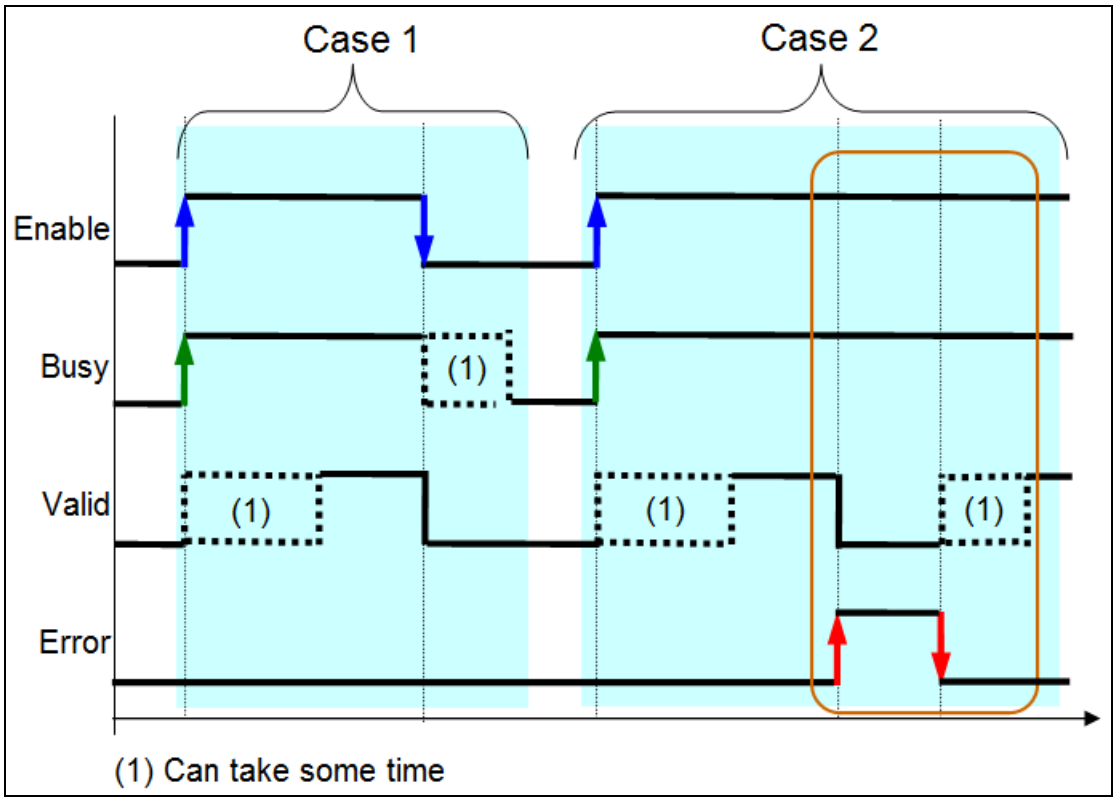


Figure 4: Example 1 - Error handling with "Enable" input

Example 2 (Figure 5):

- Case 2: This example shows an error that cannot be automatically cleared. If the outputs '*Busy*' and '*Valid*' are FALSE while the '*Enable*' is TRUE then the '*Enable*' has to be set to FALSE and to TRUE again before function block can continue.

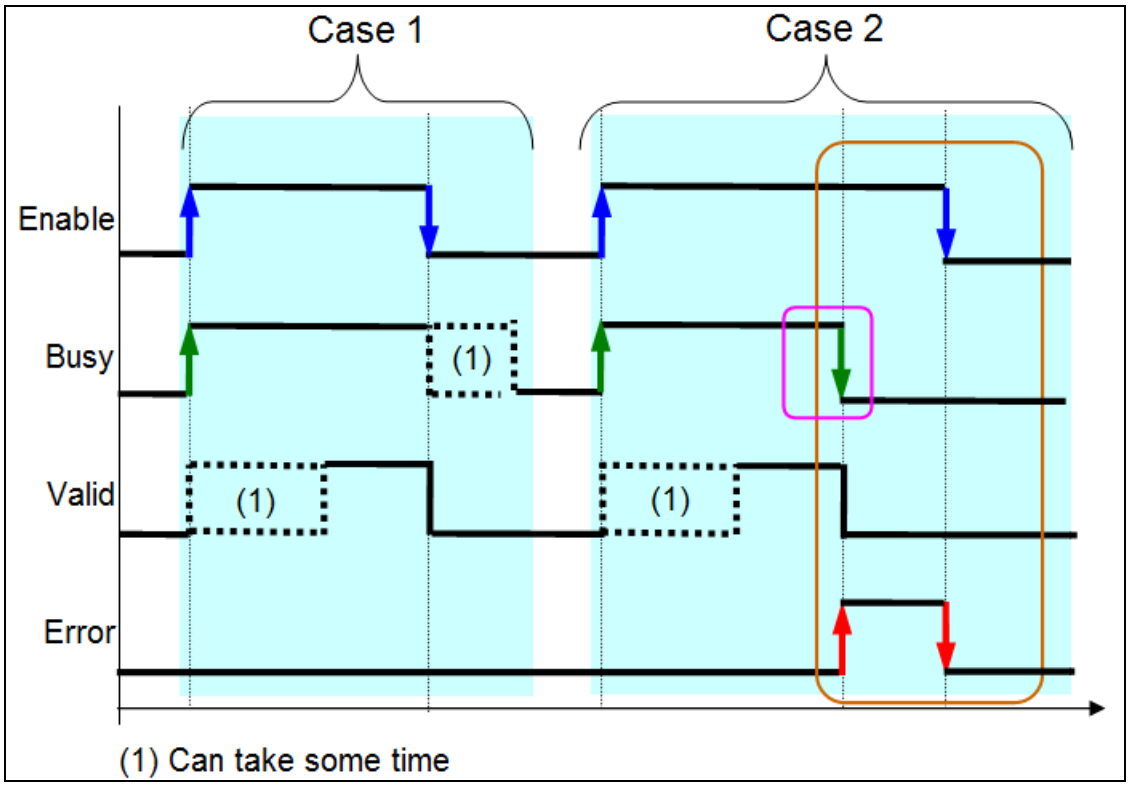


Figure 5: Example 2 - Error handling with "Enable" input

2.5 Buffer Mode

The '*BufferMode*' input controls the command flow with several function blocks and specifies whether an incoming command interrupts another command (non-queued mode, non-buffered mode) or whether the following command is only executed after the previous command (queued mode). The modes determine when the motion action of a MFB is started.

Non-queued mode (non-buffered mode, aborting mode):

In non-queued mode a triggered MFB command leads to termination of a running MFB. In this case the terminated MFB sets the '*CommandAborted*' output. A '*mcAborting*' command is a non-buffered command and acts immediately, even if this interrupts another motion and clears the command buffer. The command buffer stores all the previous commands which have not been executed yet.

Queued mode (buffered mode):

A command in a buffered mode waits until the current MFB execution ends and sets its '*Done*' output to TRUE. In other words the following MFB waits until all the

previous MFB in the queue are completed. Buffered commands are not possible if an endless motion command ('*MC_MoveVelocity*') is active. In this case the endless motion command has first to be terminated. Limitation of ECAT-M801: it is not possible to switch between discrete and continuous motion without stopping.

While the MFB is busy ('*Busy*' = TRUE) it can not be retriggered and a change of its input parameters will not take affect. It is therefore not possible to trigger the same instance of a MFB to add commands to the queue, as is not supported to trigger the same function block with different parameters as long as it is busy. Another instance of a MFB is required for each new queue entry.

By default up to 100 commands can be queued in the buffer while another command is executed. The max number of commands which can be stored in the queue can be set by the FB '*MC_WRITEPARAMETER*'. It is not possible to set the number higher than 100 commands. If the subsequent MFB encounters a full buffer the queuing is rejected with an error (Error -1138 Queue is full).

If the last command is started in non-queued mode ('*mcAborting*'), it becomes active and interrupts the running command and clears all the queued commands.

Every MFB which supports buffering has an '*Active*' output that is set when the function block takes control of the axis. The input parameters of a MFB can not be changed after the '*Execute*' input has been triggered.

Supported '*BufferModes*':

- Aborting: No command buffering. The command is executed immediately and interrupts any other command that may be running.
- Buffered: The command is executed once no other command is running on the axis. The previous movement continues until it has stopped. The following command is started from standstill.
- BlendingNext : The command is executed once no other command is running on the axis. In contrast to '*Buffered*' the axis does not stop at the previous target, but passes through this position with the velocity of the last command.

No	MC_BUFFER_MODE	Description
0	mcAborting	Start FB immediately (default mode)
1	mcBuffered	Start FB after current motion has finished
4	mcBlendingNext	The velocity is blended with velocity of the second FB

Table 4: Supported MC_BUFFER_MODE

Examples:
The first MFB moves an axis from position P_1 to P_2 and the second from P_2 then to P_3.

The second MFB is triggered during constant velocity movement towards P_2. The reference point for the different velocity profiles is always P_2. The '*BufferMode*' specifies the velocity v_1 or v_2 at the point P_2.

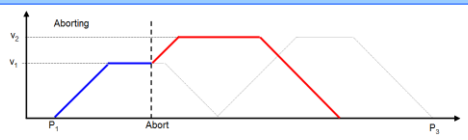
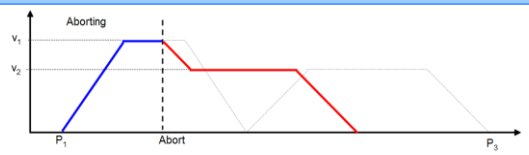
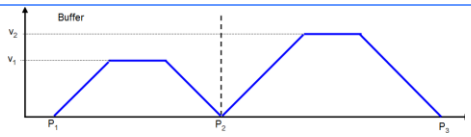
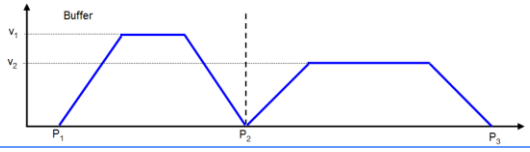
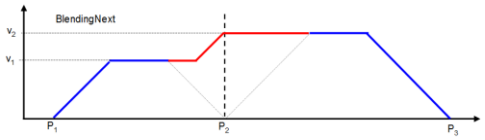
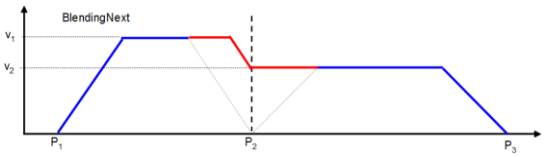
BufferMode	$V_2>V_1$	$V_2<V_1$
mcAborting		
mcBuffered		
mcBlendingNext		

Table 5: Velocity profiles for the different '*BufferModes*'

3 Single Axis State Diagram

The state diagram defines the behavior of a single axis. The axis is always in one of the defines states as shown in Figure 6. In the diagram the motion commands listed above the states transits the axis to the corresponding motion state. The motion command may also be executed if the axis is already in the corresponding motion state. Arrows with a solid line show the state transitions when a motion command is being issued. Arrows with dashed lines indicate the state transitions that occur when a command of an axis has ended.

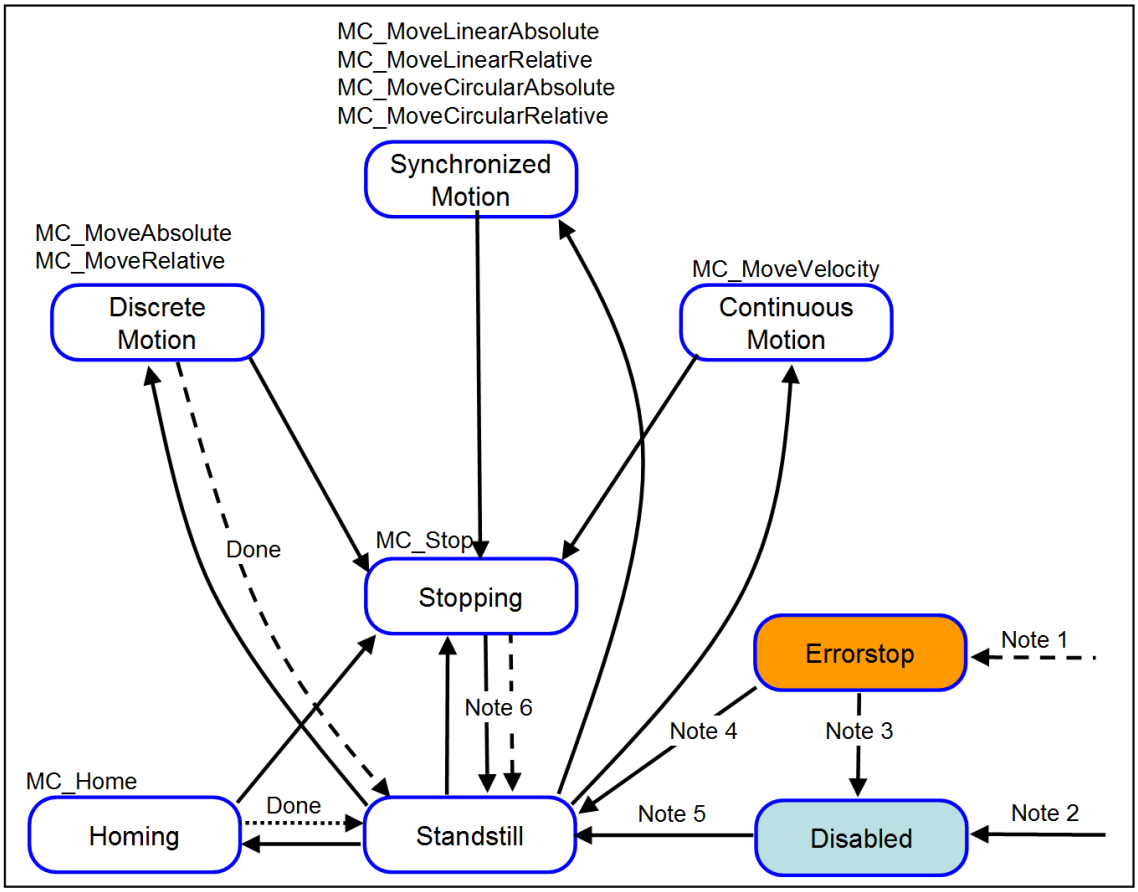


Figure 6: State diagram of the ECAT-M801/M901 EtherCAT master card

Remarks:

- Note 1: The axis can enter the '*Errorstop*' state from any other state except from the '*Disabled*' state
- Note 2: By disabling the '*MC_Power*' the axis can directly enter the '*Disabled*' state:
    i. After the successful initialization of the EtherCAT master ('*EM_InitMaster*') and axis assignment ('*EM_AxisAssign*') when '*MC_Power.Enable*' = FALSE and

'*MC_Power.Status*' = FALSE

ii. From any state except '*ErrorStop*' if '*MC_Power.Enable*' is set to FALSE.

- Note 3: If the axis is in '*Errorstop*' state it is first necessary to switch into '*Standstill*' state by executing '*MC_Reset*' before disabling the power ('*MC_Power.Enable*' = TRUE)
- Note 4: Execute '*MC_Reset*' to change the state from '*Errorstop*' to '*Standstill*'
- Note 5: To switch from the '*Disabled*' to '*Standstill*' state the '*MC_Power.Enable*' has to be set to TRUE. The '*Standstill*' state is entered once the '*MC_Power.Status*' turns TRUE.
- Note 6: The axis changes from any of the three motion states (Discrete motion, Continuous motion , Homing) to the '*Standstill*' state once the '*MC_Stop.Done*'=TRUE and '*MC_Stop.Execute*'=FALSE.

3.1 Standstill

In the '*Standstill*' state the axis is not moving and ready for motion execution. The axis '*MC_Power*' has been enable and the its '*Status*' output set to TRUE. Any of the following motion commands can be performed from this state:

- MC_Home, MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity

The axis enters the '*Standstill*' state every time after it reaches its commanded target position.

3.2 Errorstop

The axis transits into '*Errorstop*' state if an axis error has occurred. The error type can be determines by calling the '*MC_ReadAxisError*' MFB.

Possible axis errors:

- An active hardware limit switch triggered by the axis
- Axis reached the software limit
- Axis drive error (e.g. over torque)
- EtherCAT master encountered an communication error etc.

An error can occurs in any state. Once the axis is in '*Errorstop*' state axis motion stops and no new motion command can be issued. The only way to exit the '*Errorstop*' state is to rectify the cause of the error and issue the '*MC_Reset*' command to transit the axis into '*Standstill*'.

For example an axis has been stopped by an active limit switch:

1. First use '*MC_Reset*' to reset the axis error and switch into standby state.

2. Then move the axis away from the limit switch in oposit direction (e.g. by calling MC_MoveRelative).

3.3 Homing

This state indicates that the axis is performing a home search. The only way to enter this state is using the '*MC_Home*' function block while at '*Standstill*'. Only the '*MC_Stop*' command can interrupt the '*Homing*' state.

3.4 Discrete Motion

In this state the axis is performing a point to point movement. The motion is complete when the axis reaches it commanded target position. Once the position has been reached and no further command waits in the buffer the axis returns back to '*Standstill*'. The following discrete MFBs are supported:

- MC_MoveAbsolute,
- MC_MoveRelative

3.5 Continuous Motion

This state indicates that the move has not a definitive end point. The axis keeps on moving until a '*MC_Stop*' or '*MCV_Halt*' FB brings the axis to a halt or an axis error occurs (e.g. limit switch has been triggered). Currently only the '*MC_MoveVelocity*' is being supported.

3.6 Stopping

'*MC_Stop*' stops a running discrete or continuous motion command. Command stored in the command buffer are aborted and deleted. Once the '*MC_Stop.Execute*'=TRUE has been set the axis immediately enters the '*Stopping*' state. The axis remains in this state until '*MC_Stop.Execute*' is set to FALSE. While the axis is in this state, other MFBs can be called but they will not be executed. When the axis has stopped, '*MC_Stop.Done*' will output TRUE. Only after '*MC_Stop.Done*' is TRUE and 'MC_Stop.Execute' is set to FALSE

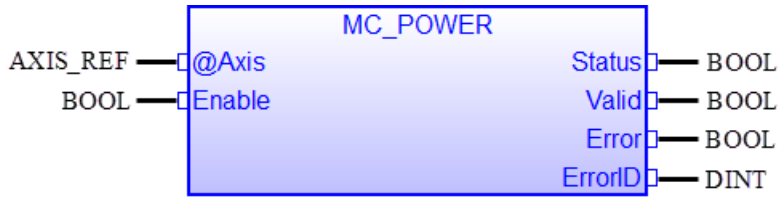
does the state change to '*Standstill*'.

4 Single Axis Motion Function Blocks

This chapter describes the Win-GRAF PLCopen library for single axis motion control.

4.1 MC_Power

To perform motion, an axis must be enabled by a '*MC_Power*' function block. As long as the '*MC_Power*' is active ('*Status*' = TRUE), motion command can be executed.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Enable	BOOL	As long as the "Enable" is TRUE, the servo is being enabled
VAR_OUTPUT	Status	BOOL	Indicates whether the servo drive has been set to ON or OFF
	Valid	BOOL	If TRUE the status output is valid
	Error	BOOL	An error has occurred within the Function Block
	ErrorID	DINT	Error identification

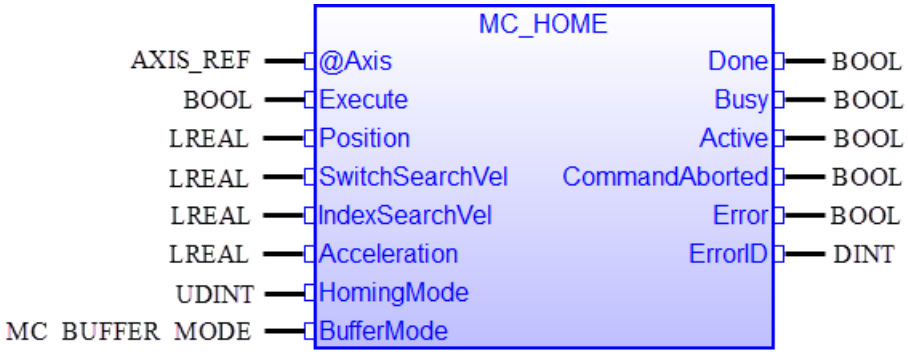
Remarks:

- The '*Enable*' input enables the power stage in the drive and not the FB itself
- If the '*MC_Power*' is called with the '*Enable*' = TRUE when being in '*Disabled*' state, the axis state changes to '*Standstill*'.
- If power fails (also during operation) the axis state will transit into "ErrorStop" state. Example of power failure: servo driver is switched off during motion execution or servo drive enters error state due to over torque.

  The power can only be switch of from any axis state except '*ErrorStop*' by setting '*MC_Power.Enable*' to TRUE. In case the axis is in '*ErrorStop*' state the error has to be cleared before state changes to '*Dsiabled*' by using '*MC_Reset*'.
- Only <u>one</u> FB '*MC_Power*' should be issued per axis. An error will be generated if more than one '*MC_Power*' instance take control over the same axis.

4.2 MC_Home

This MFB commands the axis to perform the 'search home' sequence.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the action at rising edge
	Position	LREAL	The encoder will be set to this absolute value after the home search process has been completed (Unit: user unit)
	SwitchSearchVel	LREAL	Speed during the search for the home switch (Unit: user unit)
	IndexSearchVel	LREAL	Speed for searching the z-phase signal (Unit: user unit)
	Acceleration	LREAL	Homing Acceleration (Unit: user unit/s^2)
	HomingMode	UDINT	Homing modes as defined by CiA402. Homing modes are manufacturer dependant.
	BufferMode	MC_BUFFER_MODE	Reserved (not supported)
VAR_OUTPUT	Done	BOOL	Zero velocity reached. The state machine has transitioned to standstill
	Busy	BOOL	Function block is busy executing
	Active	BOOL	Command is busy executing (same as 'Busy')
	CommandAborted	BOOL	'Command' is aborted by another command
	Error	BOOL	An error has occurred within the Function Block
	ErrorID	DINT	Error identification

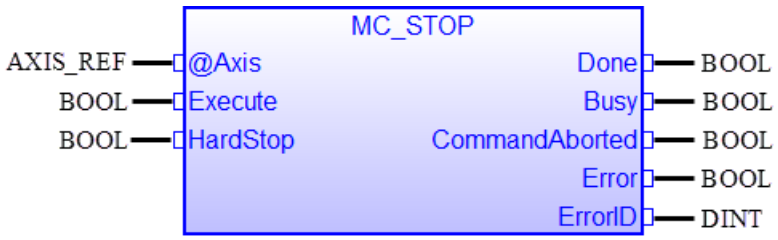
Table 6: Interface of MC_Home

Remarks:

- '*MC_Home*' can only be executed when the axis is in "Standstill" state, otherwise the FB will output an Error ID which indicates an invalid axis state
- '*MC_Home*' does not support aborting and buffer modes. While a '*MC_Home*' command is executing it can not be aborted by any other motion command (e.g. by another '*MC_Home*' or any other '*MC_Move*' function block)
- Homing Modes: CAN in Automation (CiA) 402 defines up to 37 homing modes. Refer to the user manual of the EtherCAT servo drives manufacturer which homing modes are supported.

4.3 MC_Stop

This MFB commands a controlled motion stop and transfers the axis to the state '*Stopping*'. It aborts any ongoing MFB execution. While the axis is in state '*Stopping*', no other MFB can perform any motion on the same axis. After the axis has reached zero velocity, the '*Done*' output is set to TRUE. The axis remains in the state '*Stopping*' as long as '*Execute*' is TRUE. As soon as '*Done*' is set and '*Execute*' is FALSE the axis goes to state '*Standstill*'.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the action at rising edge
	HardStop	BOOL	• FALSE: The axis will decelerate to standstill within the time set to accelerate the axis to the current velocity. • TRUE: Stop without deceleration
VAR_OUTPUT	Done	BOOL	Zero velocity reached. The state machine has transitioned to standstill
	Busy	BOOL	The axis is still decelerating and has not reach standstill yet
	CommandAborted	BOOL	'Command' is aborted by switching the servo power of.
	Error	BOOL	An error has occurred within the Function Block
	ErrorID	DINT	Error identification

Remarks:

- This MFB primarily is intended for emergency stop purposes or motion exception situation
- As long as the '*Execute*' is TRUE, the axis remains in the state of '*Stopping*' and will not execute any other motion commands.

- The deceleration time for '*MC_Stop.HardStop*'= FALSE is determined by the current active moving MFB (e.g. '*MC_MoveAbsolute*', '*MC_MoveRelative*'). The '*AccDecTime*' input of the first MFB in the buffer sets the acceleration and

deceleration time for all following motion commands.

- A timing diagram with the state transitions for '*MC_Stop*' is shown below.
  - In case 1 the '*Done*' output is set (TRUE) before the '*Execute*' is reset (FALSE).
  - In case 2 the *'Execute*' is reset before '*Done*' turns true.

  The axis remains in '*Stopping*'
  - as long as '*Execute*' is set (TRUE)
  - until the condition '*Done*' is TRUE after '*Execute*' has been set to FALSE.

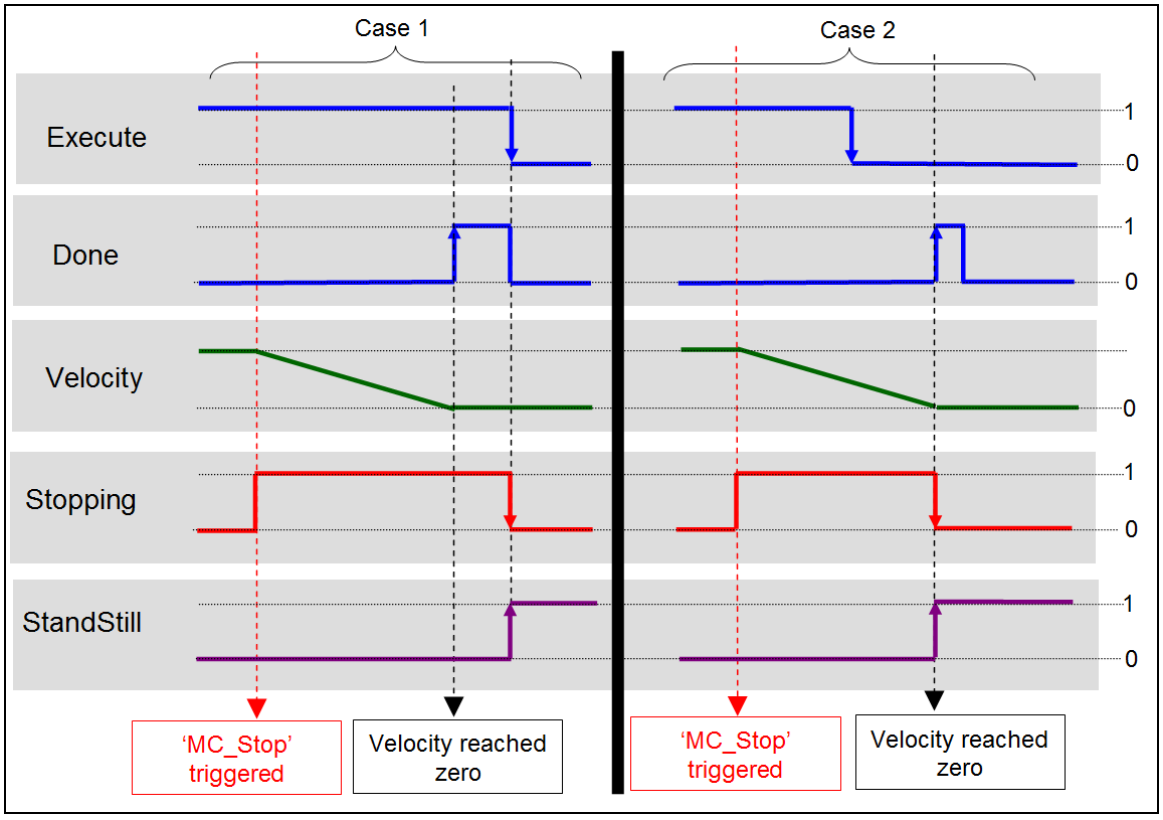


Figure 7: '*MC_Stop*' timing diagram

- The example below (Figure 8) shows the behavior in combination with a '*MC_MoveVelocity*'.

  a) The axis movement is ramped down with '*MC_Stop*'.

  b)The axis rejects motion commands as long as '*MC_Stop. Execute*' is TRUE.

  '*MC_MoveVelocity*' reports an error indicating the enabled '*MC_Stop*' command. The returned error is a function block and not an axis error, which means the state remains in '*Stopping*' and will not change to '*ErrorStop*'. At the third 'Exe_1' rising edge, the axis starts the next movement.

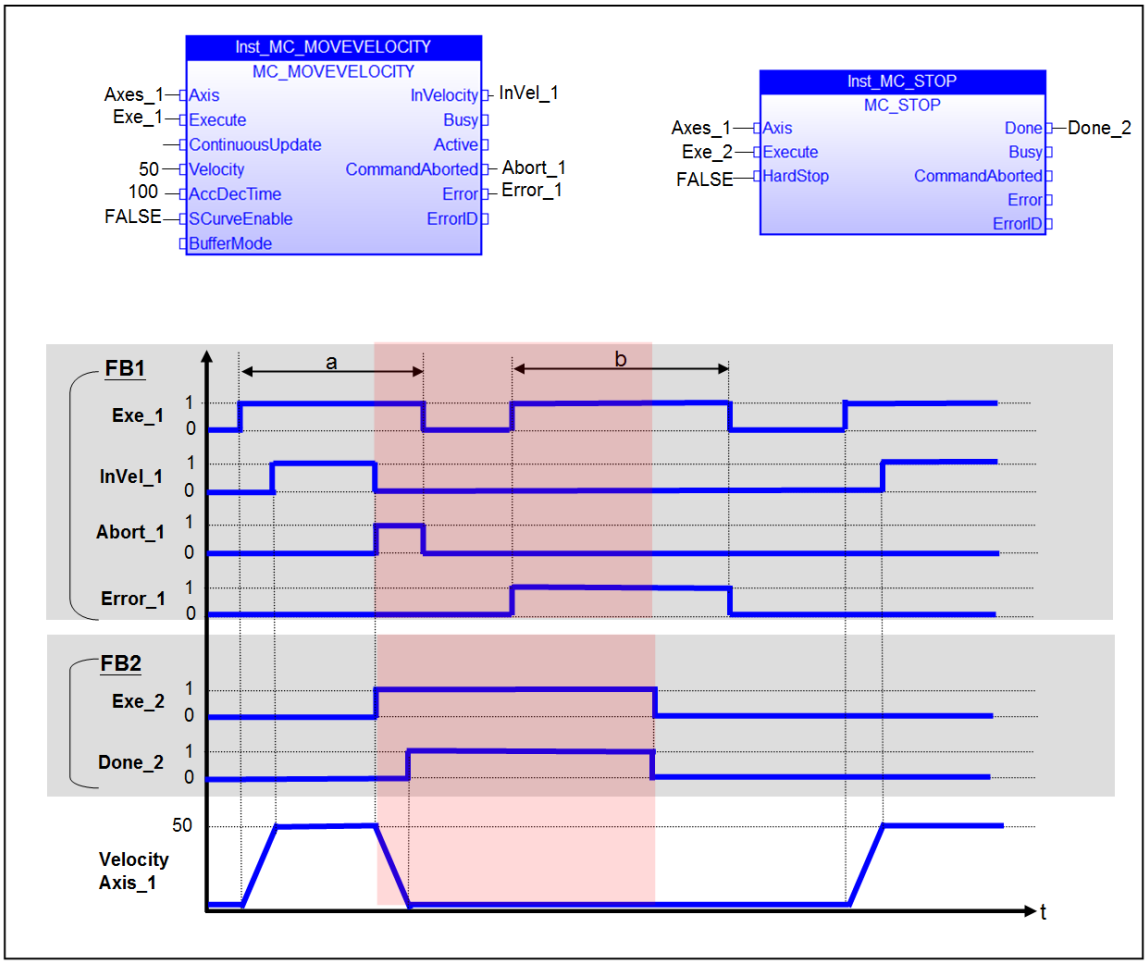
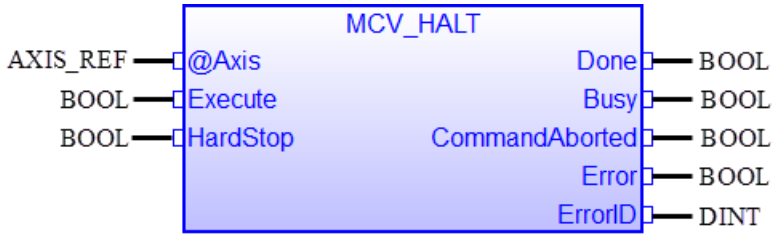


Figure 8: Behavior of '*MC_Stop*' in combination with '*MC_MoveVelocity*'

4.4 MCV_Halt

The MFB transits an axis to the '*Standstill*' state. It commands a controlled motion stop and transition to the '*Standstil*l' state once it has stopped. This MFB first set the moving axis into '*Stopping*' state; after the velocity reaches zero the axis state is automatically transition to '*Standstill*'. With the '*Done*' output set, the state is transferred to '*Standstill*'.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the action at rising edge
	HardStop	BOOL	• FALSE: Use the acceleration time for deceleration • TRUE: Stop without deceleration
VAR_OUTPUT	Done	BOOL	Zero velocity reached. The state machine has transitioned to standstill
	Busy	BOOL	The axis is still decelerating and has not reach standstill yet
	CommandAborted	BOOL	Command is aborted by another command
	Error	BOOL	An error has occurred within the Function Block
	ErrorID	DINT	Error identification

Remarks:

- The '*MCV_Halt*' will not be buffered and take effect immediately.
- Motion commands send in buffer or blending mode will be removed from FIFO memory.
- While the '*MCV_Halt*' command is busy no other motion command for the axis can be executed (see b in Figure 9).
- Once the '*MCV_Halt.Done*' is TRUE the axis is ready for executing the next motion command. Once '*MCV_Halt.Done*' = TRUE for at least one cycle a new movement command can be executed; even if the '*MCV_Halt.Execute*' is still TRUE (see c in Figure 9). In contrast the '*MC_Stop*' only allows the execution of the next movement if the '*MC_Stop.Done*' = TRUE and '*MC_Stop.Execute*' = FALSE.
- It is not possible to execute another motion instruction during deceleration of the axis (see b in Figure 9). This MFB can not be aborted by another MFB while it is busy.
- '*MCV_Halt*' does not support buffer and blending mode. Calling this MFB multiple times while another '*MCV_Halt*' took control of the same axis will not buffer the

MFBs.

- The deceleration of the axis is either the acceleration value (HardStop = FALSE) set by the previous movement command or zero (HardStop = TRUE).

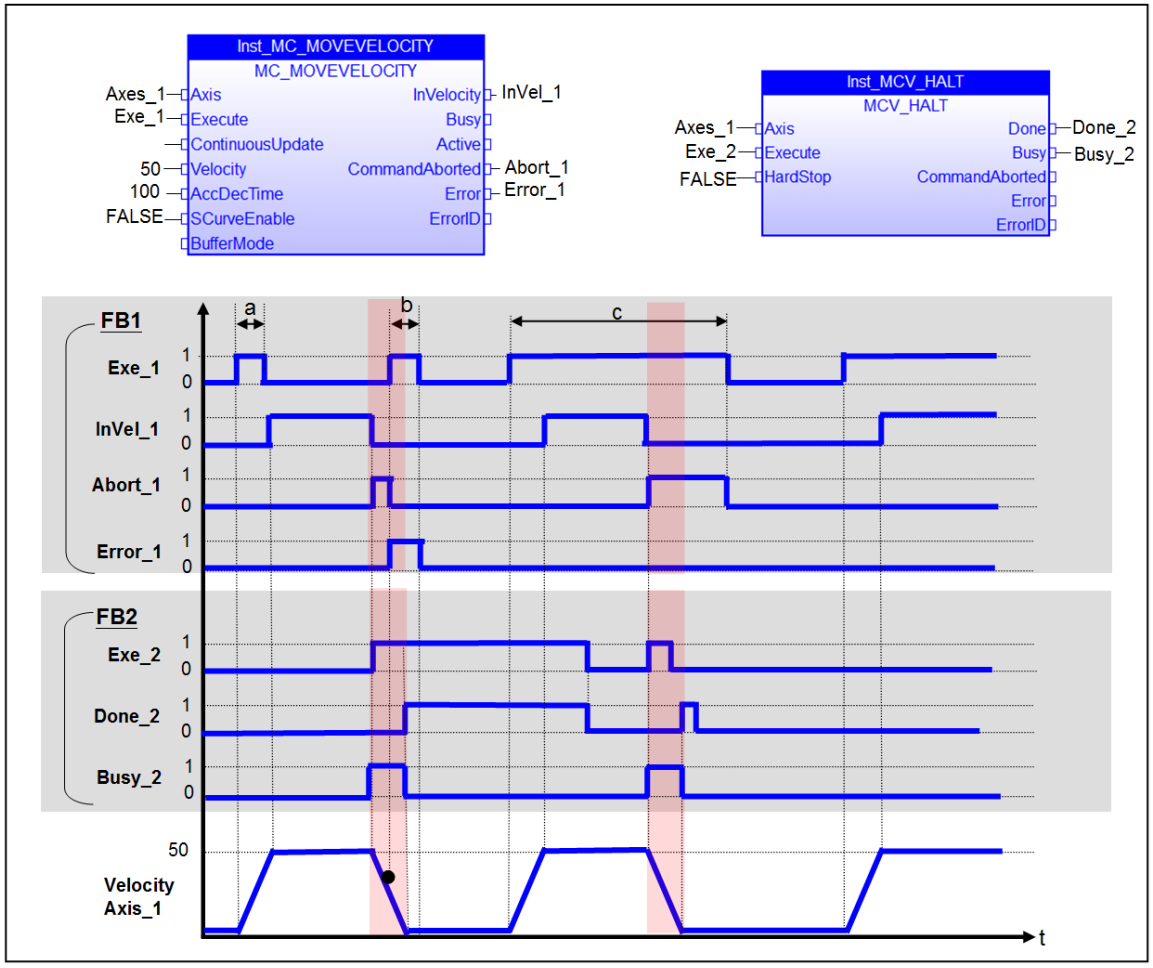
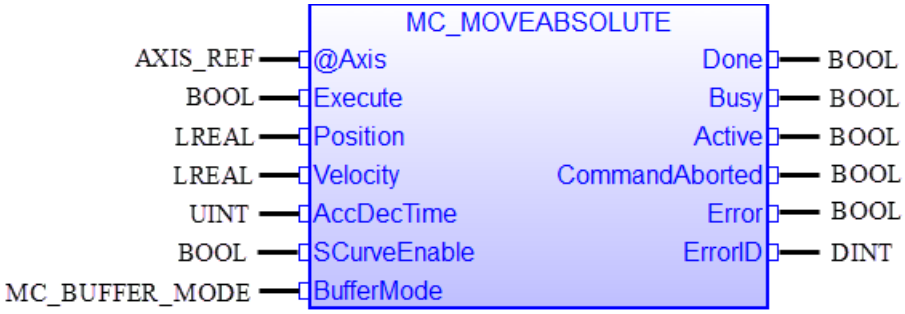


Figure 9: Behavior of '*MCV_Halt*' in combination with '*MC_MoveVelocity*'

4.5 MC_MoveAbsolute

Move to a specified absolute position.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the action at rising edge
	Position	LREAL	Commanded position for the motion (Unit: user unit)
	Velocity	LREAL	Value of the maximum velocity (Unit: user unit). The velocity will not necessary be reached if the distance to move is smaller than the sum of the acceleration/deceleration distances.
	AccDecTime	UINT	Acceleration and deceleration time [milliseconds] to reach the 'Velocity'.
	SCurveEnable	BOOL	Velocity profile: • FALSE: trapezoidal curve (T-curve) • TRUE: sinusoidal curve (S-curve)
	BufferMode	MC_BUFFER_MODE	Defines the MFB buffering behavior.
VAR_OUTPUT	Done	BOOL	The commanded position has been reached
	Busy	BOOL	The MFB has not finished and is still processing the command.
	Active	BOOL	The MFB has control over the axis
	CommandAborted	BOOL	Command is aborted by another command
	Error	BOOL	An error has occurred within the MFB.
	ErrorID	DINT	Error identification

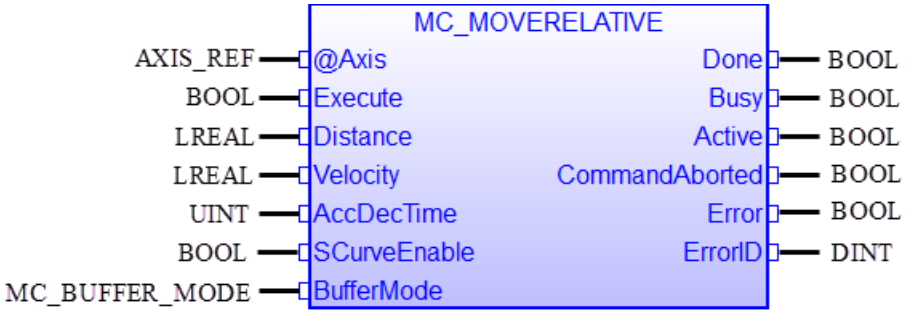
Remarks:

- While this MFB is active the axis is in the '*Discrete Motion*' state.
- When the axis absolute position is reached, the '*Done*' output will be set TRUE and the axis state changes to the '*Standstill*' or starts to execute another move command.
  - This MFB completes with velocity zero if no further actions are pending.
  - In '*mcBuffered*' mode: When the motion is complete, the axis returns to the '*Standstill*' state.
- This MFB can not be used together with the 'MC_MoveVelocity' command.

- Graph: see PLCopen technical specification: "Function blocks for motion control (Version 2.0)" Part 1; figure 18 page 36/141.

4.6 MC_MoveRelative

Move a specified distance relative to the actual position at the time of the execution.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the action at rising edge
	Distance	LREAL	Relative distance for the motion (Unit: user unit)
	Velocity	LREAL	Value of the maximum velocity (Unit: user unit). The velocity will not necessary be reached if the distance to move is smaller than the sum of acceleration/deceleration distance.
	AccDecTime	UINT	Acceleration and deceleration time [milliseconds] to reach the 'Velocity'.
	SCurveEnable	BOOL	Velocity profile: • FALSE: trapezoidal curve (T-curve) • TRUE: sinusoidal curve (S-curve)
	BufferMode	MC_BUFFER_MODE	Defines the MFB buffering behavior.
VAR_OUTPUT	Done	BOOL	The commanded distance has been reached
	Busy	BOOL	The MFB has not finished and is still processing the command.
	Active	BOOL	The MFB has control over the axis
	CommandAborted	BOOL	Command is aborted by another command
	Error	BOOL	An error has occurred within the FB.
	ErrorID	DINT	Error identification

Remarks:

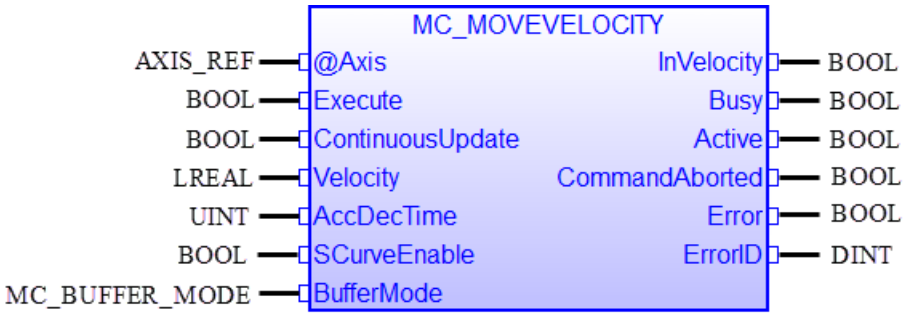
- This MFB commands an axis to move a specified relative distance from the current position.
- While this MFB is active the axis is in the '*Discrete Motion*' state.
- When the axis absolute position is reached, the '*Done*' output will be set TRUE and the axis state changes to the '*Standstill*' or starts to execute another move command.
  - This MFB completes with velocity zero if no further actions are pending.

- In '*mcBuffered*' mode: When the motion is complete, the axis returns to the '*Standstill*' state.

- This FB can not be used together with the '*MC_MoveVelocity*' command.
- Graph: see PLCopen: "Function blocks for motion control (Version 2.0)" Part 1; figure 19 page 38/141.

4.7 MC_MoveVelocity

Move axis at a specified velocity.



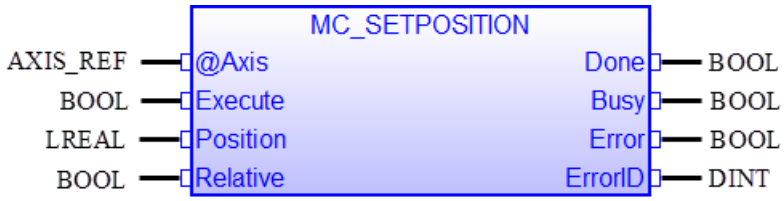
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the motion at rising edge
	ContinuousUpdate	BOOL	Determines whether the 'Velocity' input can be update after the 'Execute' has been triggered: • TRUE: the new target velocity value will be read in each cycle from the 'Velocity' input. • FALSE: Velocity can not be changed after 'Execute' edge has resin.
	Velocity	LREAL	Value of the commanded velocity (Unit: user unit). • Positive value: move in positive direction • Negative value: move in negative direction.
	AccDecTime	UINT	Acceleration and deceleration time [milliseconds] to reach the 'Velocity'.
	SCurveEnable	BOOL	Velocity profile: • FALSE: trapezoidal curve (T-curve) • TRUE: sinusoidal curve (S-curve)
	BufferMode	MC_BUFFER_MODE	Only 'mcAborting' supported
VAR_OUTPUT	InVelocity	BOOL	Indicates that the command 'Velocity' has been reached.
	Busy	BOOL	The FB has not finished and is still processing the command.
	Active	BOOL	The FB has control over the axis
	CommandAborted	BOOL	Command is aborted by another command
	Error	BOOL	An error has occurred within the MFB.
	ErrorID	DINT	Error identification

Remarks:

- This MFB commands a move to the commanded velocity.
- To stop the command, this MFB has to be interrupted by a '*MC_STOP*' or '*MCV_HALT*'.
- The value of the '*Velocity*' determines the velocity and direction. Use positive values for positive direction and negative values for negative direction.
- While this MFB is active the axis is in the '*Continuous Motion*' state.
- When the axis reaches the commanded '*Velocity*' the '*InVelocity*' will be set to TRUE.
- This MFB can not be aborted by the '*MC_MoveAbsolute*' or '*MC_MoveRelative*' commands.

4.8 MC_SetPosition

Assigned a position value to the current encoder position and the actual commanded position. Both encoder and commanded position will be set to the same value. No movement occurs during the change of the position values.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Start the action at rising edge
	Position	LREAL	Specifies the set position. (Unit: user unit)
	Relative	BOOL	Specifies a relative distance or an absolute position. • TRUE: Relative Position • FALSE: Absolute Position
VAR_OUTPUT	Done	BOOL	True when the position change is completed.
	Busy	BOOL	MFB is busy executing
	Error	BOOL	An error has occurred within the MFB
	ErrorID	DINT	Error identification • ECAT_ERR_MC_INVALID_AXIS_STATE: - When a motion command for this axis is busy executing then the position can not be set - If another 'MC_SetPosition' instance for the same axis is busy then the new 'MC_SetPosition' instance call fails.

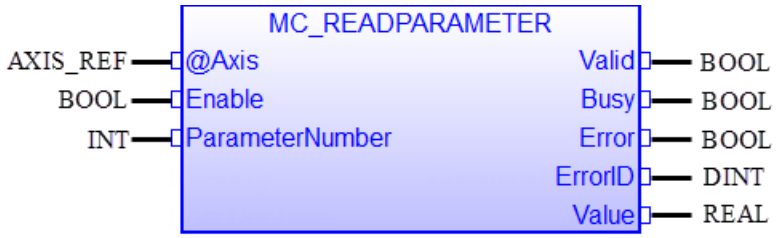
Remarks:

- '*Relative*' means that '*Position*' is added to the actual position value of the axis at the time of execution. This results in a recalibration by a specified distance.
- '*Absolute*' means that the actual position value of the axis is set to the value specified by the '*Position*' input parameter.
- '*MC_SetPosition*' changes the current and commanded position of the servo axis to the specified target position.

- '*MC_SetPosition*' can not be called while a motion command is executing.
- Internally the '*MC_SetPosition*' is executing the CiA402 defined homing method 37. If the servo driver does not support homing method 37 then the MFB fails. After '*MC_SetPosition*' has finished successfully the '*MC_AxisReadInfo*' will show that '*IsHome*' is true. If the servo driver does not support homing method 37 then call '*MC_Home*' with the homing method 35 instead.

4.9 MC_ReadParameter

This MFB returns the value of a configuration parameter of the axis or the EtherCAT master card ECAT-M801/e-M901.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Enable	BOOL	Read the parameter value in each cycle while the input is TRUE.
	ParameterNumber	INT	The parameter number (PN) shown in Table 7.
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'Value' is valid.
	Busy	BOOL	Function block is busy processing the request
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	Value	REAL	Value of the specified parameter

Remarks:

- The configuration parameter is identified by the '*Axis*' and '*ParameterNumber*' (PN) input parameters. The PN are define in the table (Table 7).
- Only data types indicated as REAL in the PN table can be accessed with the '*MC_ReadParameter*' FB.
- PLCopen defined for setting or reading the most common axis parameters separate function blocks. Separate function blocks are provided for axis parameters which are not listed in the PN table.

4.9.1 Parameter Table

The parameter table is a list of parameters for configuring the individual axis. Some parameters are associated with the configuration of the EtherCAT master card (ECAT-M801/e-M901). During the initialization phase the axes and hardware will be automatically set to the default parameters if they are not set by the PLC application

program. Parameter indicated by 'R' in the table can be read by the application to confirm the setting. 'W' indicates that the parameter value can be set.

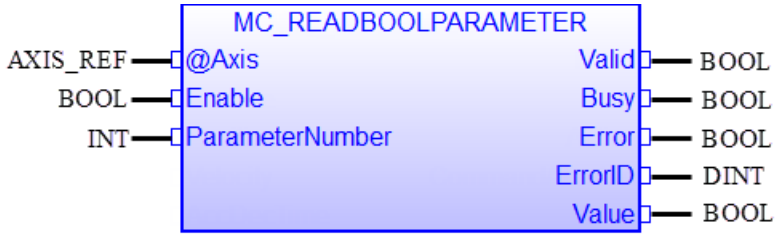
PN	Name	Datatype	R/W	Comments
2	SWLimitPos	REAL	R/W	Positive software limit switch position
3	SWLimitNeg	REAL	R/W	Negative software limit switch position
4	EnableLimitPos	BOOL	R/W	Enable positive software limit
5	EnableLimitNeg	BOOL	R/W	Enable negative Software limit
1000	Pulse/unit	REAL	R/W	Pulse per unit for specified axis
1001	Timeout	REAL	W	Response timeout for the ECAT-M801/ECAT-M901
1002	Heartbeat time	REAL	W	The PLC application has to send the ECAT-M801/ECAT-M901 at the set time interval a command to indicate that PLC is still running. The motion will be automatically stopped if no command has been received within the time interval.
1003	Command buffer size	REAL	W	The number of MFB command to be stored in the buffer
1004	SWLimitDec	REAL	R/W	Not supported Software limit deceleration value. This deceleration will take effect once the software limit has been reached.
1005	LimitSwitchDec	REAL	R/W	Not supported Limit switch deceleration value. This value determines how fast the axis decelerates to stop when the hardware limit switch has been triggered. Has to be set via the SDO function block. Not all CiA402 driver support this setting.

Table 7: 'ParameterNumber' (PN) definitions

Some parameters setting will take immediate effect and some will be rejected while the axis is executing. In general it is a good practice to set the commands only when the axis is in standstill. The axis execution should only start once the setting action as been completed.

4.10 MC_ReadBoolParameter

This MFB returns the value of a specific parameter with data type BOOL.



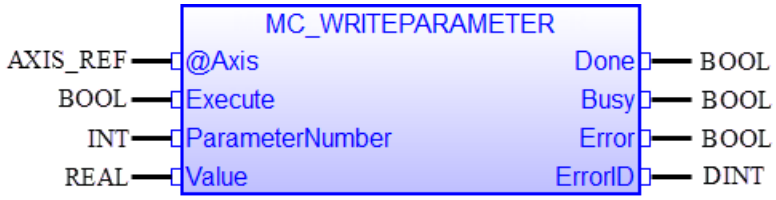
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Enable	BOOL	Read the parameter value in each cycle while the input is TRUE.
	ParameterNumber	INT	The parameter number (PN) shown in Table 7.
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'Value' is valid.
	Busy	BOOL	Function block is busy processing the request
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	Value	BOOL	Value of the specified parameter

Remarks:

- The configuration parameter is identified by the '*Axis*' and '*ParameterNumber*' (PN) input parameters. The PN are define in the table (Table 7).
- Only data types indicated by BOOL in the PN table can be accessed with the '*MC_ReadBoolParameter*' MFB.

4.11 MC_WriteParameter

This MFB set the value of an axis parameter.



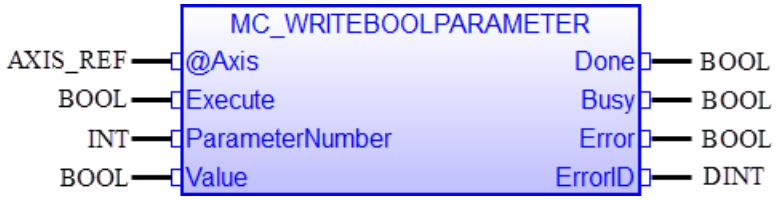
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	A rising edge triggers the execution of the write command.
	ParameterNumber	INT	The parameter number (PN) shown in Table 7.
	Value	REAL	New value to be assigned to the parameter (PN)
VAR_OUTPUT	Done	BOOL	Indicates that the input 'Value' has been successfully written to the parameter
	Busy	BOOL	Function block is busy executing the write command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification

Remarks:

- The configuration parameter is identified by the '*Axis*' and '*ParameterNumber*' (PN) input parameters. The PN are define in the table (Table 7).
- Only data types indicated by REAL in the PN table can be accessed with the '*MC_WriteParameter*' FB.
- PLCopen defined for setting or reading the most common axis parameters separate function blocks. Separate function blocks are provided for axis parameters which are not listed in the PN table.

4.12 MC_WriteBoolParameter

This MFB modifies the value of a selected parameter of type BOOL.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	A rising edge triggers the execution of the write command.
	ParameterNumber	INT	The parameter number (PN) shown in Table 7.
	Value	BOOL	New value to be assigned to the parameter (PN)
VAR_OUTPUT	Done	BOOL	Indicates that the input 'Value' has been successfully written to the parameter
	Busy	BOOL	Function block is busy executing the write command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification

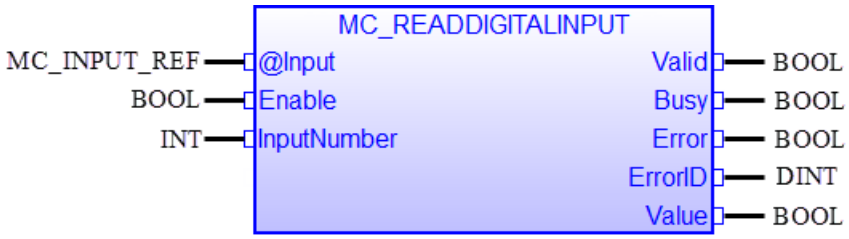
Remarks:

- The configuration parameter is identified by the '*Axis*' and '*ParameterNumber*' (PN) input parameters. The PN are define in the table (Table 7).
- Only data types indicated by BOOL in the PN table can be modified with the '*MC_WriteBoolParameter*' FB.
- PLCopen defined for setting or reading the most common axis parameters separate function blocks. Separate function blocks are provided for axis parameters which are not listed in the PN table.

4.13 MC_ReadDigitalInput

This FB returns the value of a digital input channel. Three different types of DI module can be accessed:

- Local DI of ECAT-M801 board,
- EtherCAT DI slave and
- remote EtherCAT servo drive DI.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Input	MC_INPUT_REF	Reference to the digital input (DI)
VAR_INPUT	Enable	BOOL	Read the digital input channel continuously while enabled
	InputNumber	INT	Channel number
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'Value' is valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	Value	BOOL	The value of the selected input signal

Remarks:

- If the DI signal width is shorter than the PLC cycle time then it may occur that the digital signal will not be detected by the FB. A short digital input could be over before the FB reads the channel in the next cycle.
- The digital input data structure '*MC_INPUT_REF*' is defined as follows (Figure 10):
  - Digital input (DI) structure '*DiType*':
    - 0: Read remote EtherCAT slave DI
    - 1: Read axis DI which is part of the servo drive
    - 2: Read local DI of ECAT-M801 card
  - '*CardNo*': Card number of the ECAT-M801. This variable will be ignored by the FB if 'DiType' is 0 or 1
  - '*SlaveNo*': Number of the EtherCAT DI slave. Only valid if '*DiType*' is 0.
  - 'AxisNo': Number of the axis assigned to the servo drive. Only valid for '*DiType*' 1.
-

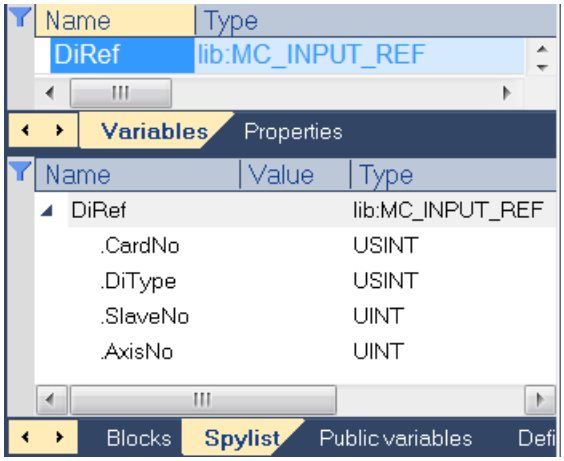
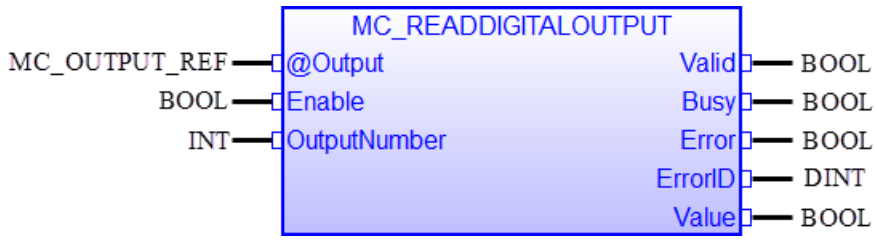


Figure 10: 'MC_INPUT_REF' structure member variables

4.14 MC_ReadDigitalOutput

This FB returns the value of a digital output channel. Two types of DO module can be accessed:

- Local DO of ECAT-M801 board and
- EtherCAT DO slave.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Output	MC_OUTPUT_REF	Reference to the digital output (DO) structure
VAR_INPUT	Enable	BOOL	Read the digital output channel continuously while enabled
	OutputNumber	INT	Channel number
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'Value' is valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	Value	BOOL	The value of the selected output signal

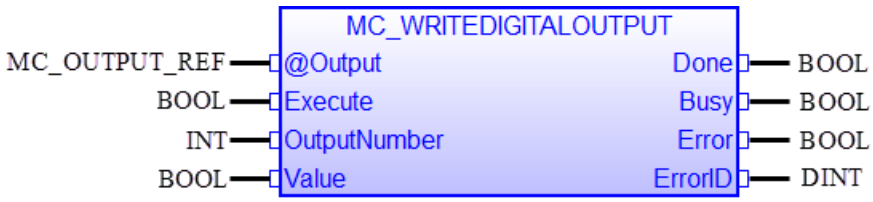
Remarks:

- The operation of the '*MC_ReadDigitialOutput*' function is similar to that of the '*MC_ReadDigitalInput*' function.
- The digital output data structure '*MC_OUTPUT_REF*' is defined as follows:
  - digital output (DO) structure '*DoType*':
    - 0: Read remote EtherCAT slave DO
    - 1: Read axis DO which is part of the servo drive (Not supported).
    - 2: Read local DO of ECAT-M801 card
  - '*CardNo*': Card number of the ECAT-M801. This variable will be ignored by the FB if '*DoType*' is 0 or 1
  - '*SlaveNo*': Number of the EtherCAT DO slave. Only valid if '*DoType*' is 0.
-

4.15 MC_WriteDigitalOutput

This FB write a value to a selected digital output channel. Two types of DO module can be accessed:

- Local DO of ECAT-M801 board and
- EtherCAT DO slave.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Output	MC_OUTPUT_REF	Reference to the digital output (DO) structure
VAR_INPUT	Execute	BOOL	A rising edge triggers the execution of the DO write command.
	OutputNumber	INT	Channel number
	Value	BOOL	The value of the selected digital output channel
VAR_OUTPUT	Done	BOOL	True when the digital output channel has been set.
	Busy	BOOL	Function block is busy executing the write command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification

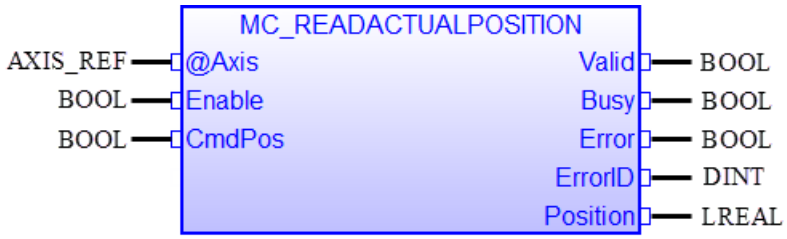
Remarks:

• The digital output data structure '*MC_OUTPUT_REF*' is defined as follows:

- digital output (DO) structure '*DoType*':
  - 0: Read remote EtherCAT slave DO
  - 1: Read axis DO which is part of the servo drive (Not supported).
  - 2: Read local DO of ECAT-M801 card
- '*CardNo*': Card number of the ECAT-M801. This variable will be ignored by the FB if '*DoType*' is 0 or 1
- '*SlaveNo*': Number of the EtherCAT DO slave. Only valid if '*DoType*' is 0.

4.16 MC_ReadActualPosition

This MFB returns either the actual encoder position of the stepper/servo drive or the current commanded position of the motion control engine inside the ECAT-M801/e-M901 card.



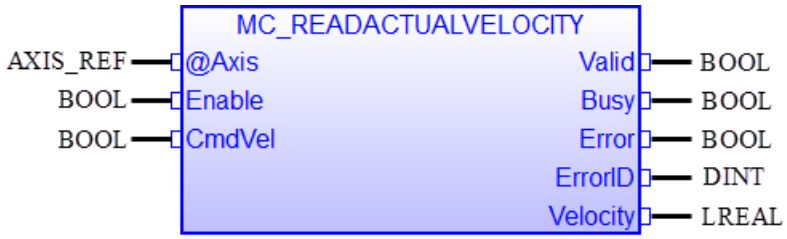
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Enable	BOOL	Continuously read the current position while enabled.
	CmdPos	BOOL	FALSE: Read encoder position (CiA402) TRUE: Read commanded position
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'Position' is valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	Position	LREAL	Either current encoder or commanded position

Remarks:

- To increase the reaction speed of the runtime it is suggested to just use one '*MC_ReadActualPosition*' instance per axis.

4.17 MC_ReadActualVelocity

This MFB either returns the actual velocity of the stepper/servo drive or the commanded velocity of the motion control engine inside the ECAT-M801/e-M901 card.



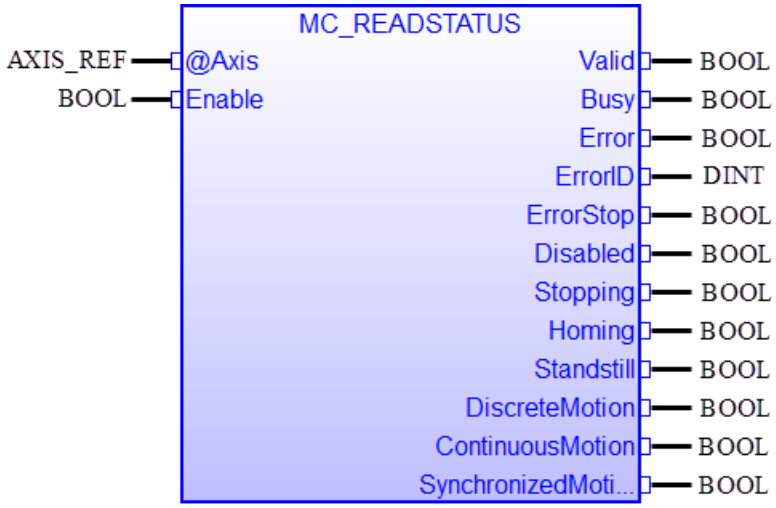
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Enable	BOOL	Continuously read the current velocity while enabled.
	CmdVel	BOOL	FALSE: Read actual velocity (CiA402) TRUE: Read commanded velocity
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'Velocity' is valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	Velocity	LREAL	Either real or commanded velocity

Remarks:

- To increase the reaction speed of the runtime it is suggested to just use one '*MC_ReadActualVelocity*' instance per axis.

4.18 MC_ReadStatus

This MFB returns the status of the state diagram of the selected axis.



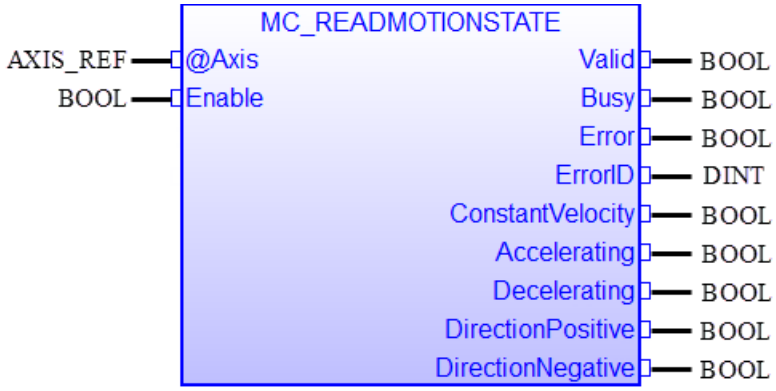
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Enable	BOOL	Continuously read the status of the state diagram while enabled.
VAR_OUTPUT	Valid	BOOL	True when all the output parameters are valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	ErrorStop	BOOL	See PLCopen state diagram
	Disabled	BOOL	See PLCopen state diagram
	Stopping	BOOL	See PLCopen state diagram
	Homing	BOOL	See PLCopen state diagram
	Standstill	BOOL	See PLCopen state diagram
	DiscreteMotion	BOOL	See PLCopen state diagram
	ContinuousMotion	BOOL	See PLCopen state diagram
	SynchronizedMotion	BOOL	See PLCopen state diagram

Remarks:

-

4.19 MC_ReadMotionState

This MFB returns the status of an axis movement.



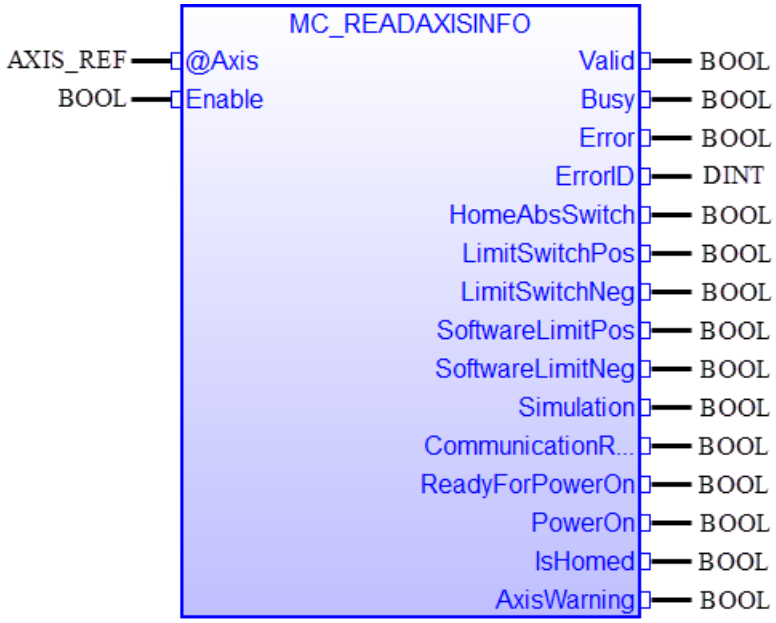
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Enable	BOOL	Continuously read the motion state while enabled.
VAR_OUTPUT	Valid	BOOL	True when all the output parameters are valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	ConstantVelocity	BOOL	Axis velocity is constant
	Accelerating	BOOL	Axis is accelerating
	Decelerating	BOOL	Axis is decelerating
	DirectionPositive	BOOL	Axis is moving in positive direction
	DirectionNegative	BOOL	Axis is moving in negative direction

Remarks:

- The output '*Valid*' turns TRUE when all the outputs of the axis motion status are valid.
- If '*Enable*' is FALSE, all the output data is invalid ('*Valid*' = FALSE)

4.20 MC_ReadAxisInfo

This MFB serves to read various state information on the axis.



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Enable	BOOL	Continuously read the axis information while enabled in each PLC cycle.
VAR_OUTPUT	Valid	BOOL	True when all the output parameters are valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	HomeAbsSwitch	BOOL	The home switch (digital input) is active
	LimitSwitchPos	BOOL	The positive hardware limit switch (digital input) is active
	LimitSwitchNeg	BOOL	The negative hardware limit switch (digital input) is active
	SoftwareLimitPos	BOOL	Axis reached the positive software limit
	SoftwareLimitNeg	BOOL	Axis reached the positive software limit
	Simulation	BOOL	Axis is in simulation mode (e.g. motor is simulated)
	CommunicationReady	BOOL	The EtherCAT master is initialized and set into operation mode.
	ReadyForPowerOn	BOOL	Drive is ready to be enabled. MC_Power can be enabled now.
	PowerOn	BOOL	If TRUE shows that the MC_Power has been enabled and the power stage is switched ON

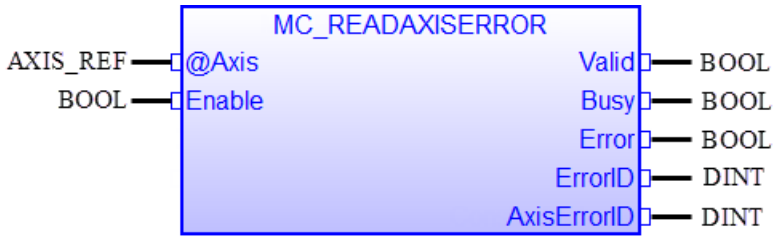
	IsHomed	BOOL	If TRUE shows that the MC_Home has been executed successfully and the absolute reference position is known to the axis.
	AxisWarning	BOOL	Warning(s) on the axis is present

Remarks:

- The output '*Valid*' turns TRUE when all the outputs of the axis motion status are valid.
- If '*Enable*' is FALSE, all the output data is invalid ('*Valid*' = FALSE)
- The MFB reads the axis information in each POU cycle as long as the input '*Enable*' is set to TRUE.
- The '*CommunicationReady*' parameter indicates whether any axis in the EtherCAT network is offline. As soon as the communication to any one of the axes fails the '*CommunicationReady*' output parameter for all axes will be set to TRUE. Therefore it does not represents the communication status of a single axis but rather of all axes.

4.21 MC_ReadAxisError

The MFB reads axis errors which caused a sudden stop or prevented the axis from reaching the target position.



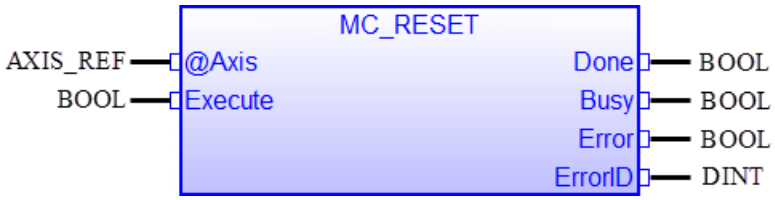
Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Enable	BOOL	Continuously read the current axis error state while enabled.
VAR_OUTPUT	Valid	BOOL	True when the output parameter 'AxisErrorID' is valid.
	Busy	BOOL	Function block is busy executing the read command
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification
	AxisErrorID	DINT	The value of the axis error. See axis error table

Remarks:

-

4.22 MC_Reset

This MFB has to be called if the axis is in '*ErrorStop*' state in order to clear and reset the axis error. A successful reset execution transitions the axis state from '*ErrorStop*' to '*Standstill*' (see state diagram).



Input/Output	Name	Type	Comments
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis structure
VAR_INPUT	Execute	BOOL	Reset all internal axis -related errors
VAR_OUTPUT	Done	BOOL	TRUE: All axis errors have been reset and the axis state has been transition into 'Standstill' or 'Disabled' state
	Busy	BOOL	Function block is busy resetting the axis error(s)
	Error	BOOL	An error has occurred within the FB
	ErrorID	DINT	Error identification

Remarks:

-

5 Coordinated Motion

5.1 Function Blocks Overview

The following table gives an overview of the supported multi-axis MFBs. They are divided into administrative and motion related categories. The administrative MFBs do not directly control the axis movement, but rather read the group status and assign axis to a group.

Administrative	Group Motion
MC_AddAxisToGroup	MC_GroupStop
MC_RemoveAxisFromGroup	MCV_GroupHalt
MC_UngroupAllAxes	MC_GroupInterrupt
MC_GroupEnable	MC_GroupContinue
MC_GroupDisable	MC_MoveLinearAbsolute
MC_GroupReadActualPosition	MC_MoveLinearrelative
MC_GroupReadActualVelocity	MC_MoveCircularAbsolute
MC_GroupReadStatus	MC_MoveCircularRelative
MC_GroupReadError	
MC_GroupReset	

Table 8: Group function blocks

5.2 State Diagram

The state diagram (Figure 11) describes the states of a group of axes. It determines which group MFBs are allowed to be executed under a specific state. The group state does not effect the administrative MFBs which returns the current group state, position, velocity, etc.. While axes are in group state, the single axis state diagram is also active per axis.

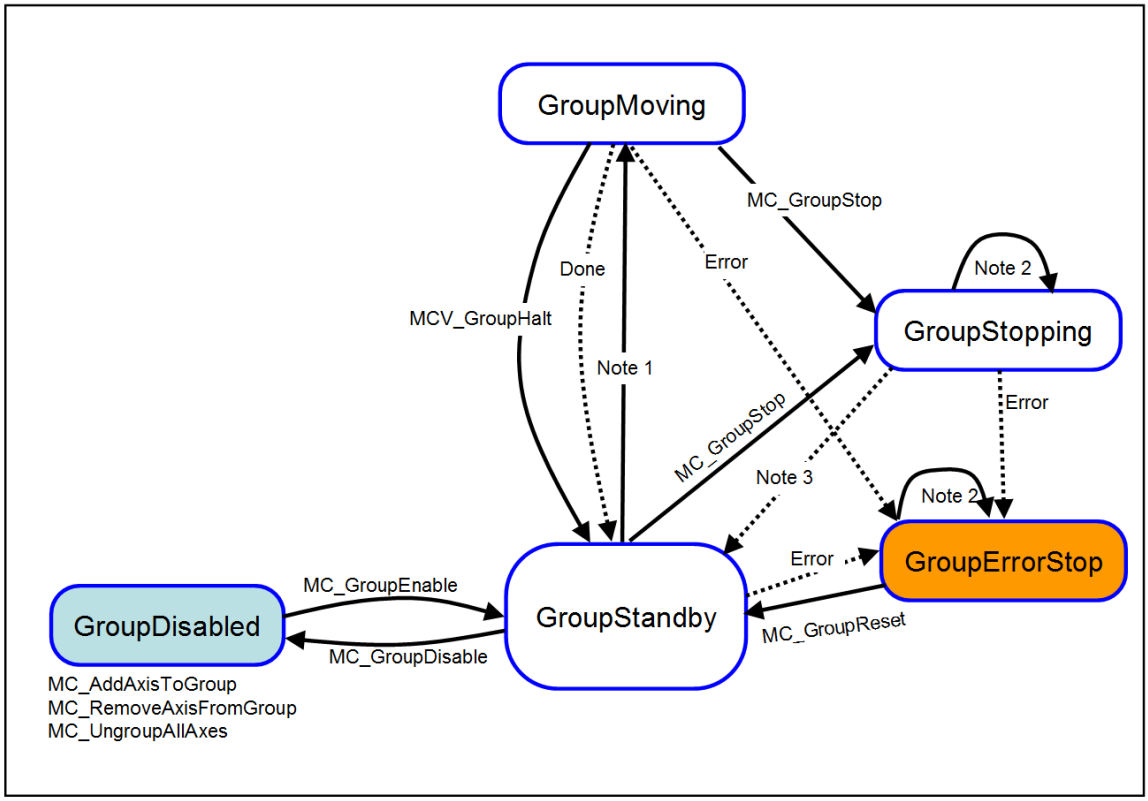


Figure 11: The group state diagram

Notes to the group state diagram (Figure 11):
Note to transitions: Continuous lines are commanded transition; dotted lines are automatic transitions.

- Note 1: Applicable for all group motion MFBs
- Note 2: In the '*GroupErrorStop*' and '*GroupStopping*' states all MFBs can be called, but the group movement MFBs will not be executed and return an error. It is necessary to first transition the state into '*GroupStandby*' by calling '*MC_GroupReset*' '*MC_GroupStop*' respectively before MFBs can take control of a group.
- Note 3: Group transitions from '*GroupStopping*' to '*GroupStandby*' occurs when the both '*MC_GroupStop.Done*' is TRUE and '*MC_GroupStop.Execute*' has been set to FALSE.
- Note 4 (not shown in the figure): '*MC_GroupDisable*' can be issued in all states and will change the state to '*GroupDisabled*'.

The following describes the group states and transition in more details:

5.2.1 GroupDisabled

- This is the initial state of the group after the axis assignment ('*MCV_AxisAssign*')

which assigns each servo/stepper slave with an axis number for easy identification. All the groups are empty directly after axis assignment and axes have to be added to the group ('*MC_AddAxisToGroup*'). To take control over a group it has to be enabled via '*MC_GroupEnable*' which transition the states from '*GroupDisabled*' to '*GroupStandby*' state.

- The group can only be enabled if all its member axes are in '*Standstill*'.
  i. If one of the axis is in '*ErrorStop*' state then the error has to be resolved and cleared ('*MC_Reset*').
  ii. If one of the axis is in '*Stopping*' state then release the '*MC_Stop*' command by setting '*MC_Stop.Execute*' to FALSE.
- If the group encounters an error or the '*MC_GroupStop*' is active while it is in '*GroupStandby*' then its states transition immediately into '*GroupErrorStop*' or '*GroupStopping*' respectively.
- Once a group is enabled the single axis motion command can not take control of the group axes. Single axis motion FBs like '*MC_Stop*', '*MC_MoveAbsolute*', etc. can not be issued. The single axis administrative FB are still allowed to read the axis status.
- Homing is not supported as a group command. Disable the group to do single axis home search.

5.2.2 GroupStandby

- After enabling the group it transition into '*GroupStandby*'. All the group axes are in standstill and are ready to execute group motion commands.
- In case an error arises in any of the group axis the state changes to '*GroupErrorStop*', which can only be left via issuing '*MC_ResetGroup*'.
- '*MC_GroupStop*' changes the state to '*GroupStopping*'
- In '*GroupStandby*' the single axis state of the member axis are in '*Standstill*'

5.2.3 GroupMoving

- The following motion commands transfers the state into '*GroupMoving*':
  i. MC_MoveLinearAbsolute
  ii. MC_MoveLinearRelative
  iii. MC_MoveCircAbs
  iv. MC_MoveCircRel
  v. MC_GroupMoveIncPath
- The group remains in this state until the command has finished ('*Done*'=TRUE), an error has been encountered (e.g. hardware limit has been triggered) or a stop command has been issued '*MC_GroupStop*' or '*MCV_Halt*'.

- Group motion command will always lead the single axis state to change to '*SynchronizedMotion*'.

5.2.4 GroupStopping

- This state is being controlled by the '*MC_GroupStop*' FB. The '*MC_GroupStop.Execute*' = TRUE triggers an group stop command and changes the state to '*GroupStopping*'. The group remains in this state as long as '*Execute*' is TRUE. The '*MC_GroupStop.Done*' = TRUE indicates that all axes have reach standstill. The group automatically transfers to the state '*GroupStandby*' as soon as the both conditions '*MC_GroupStop.Done*' = TRUE and '*MC_GroupStop.Execute*' = FALSE are met.
- The single axis state transitions from '*SynchronizedMotion*' to '*Standstill*' once the group axis has stopped.

5.2.5 GroupErrorStop

- Group error stops occur when one of the group axis triggers the hardware limit switch, exceeds the software limit or encounters a servo drive error. Faulty EtherCAT connection may also cause an error stop. In the most cases single axis '*ErrorStop*' results in an '*GroupErrorStop*' as the single error affects the group.
- After the error cause has been resolved the error has to be cleared via '*MC_GroupReset*'. The command not only clears the group error but also the single axis error of the axis members. For example: Once the software limit of a single axis in the group has been reached the axis and group state changes to error stop. Executing '*MC_GroupReset*' will simultaneously clear both the single axis and group error.

5.3 Relationship Single Axis and Group State Diagrams

Once a group has been enabled via '*MC_GroupEnable*' single axis motion commands, like '*MC_MoveAbsolute*' can not be issued. The single axis MFB error output will be set to TRUE to indicate that the command has not been accepted. Single axis command does not effect the group state and its movement, e.g. a moving group will continue without any interruption if a single axis movement commands is being called.

If the group is in an disabled state single axis motion commands can take control over

each axis in the group. While the single axis command is in control of one of the group member axis the group can not be enabled for group command control as long as the single axis command is busy.

To enable a group all its member axes have to be in '*StandStill*'.

General rules for the interaction between the single axis towards its groups:
- If at least one axis in the group is moving then the group is in the state of '*GroupMoving*'.
- If all axes are in '*Standstill*', the group is either in '*GroupStandby*', '*GroupDisabled*' or '*GroupErrorStop*'
- If one axis is in '*ErrorStop*', the entire group is in '*GroupErrorStop*'.
- The single axis '*MC_Home*' can not be issued if the group has been enabled.
- The single axis '*MC_Stop*' is can not be issued if the group has been enabled.
- Single axis can not be remove from group if the group has been enabled.

General rules for the interaction between a group and the single axis in it:
- If the group is commanded by a group moving command, all the single axes in the group are in the state '*SynchronizedMotion*'
- If the group is in a state '*GroupStandby*', the state of the single axes are in '*StandStill*'.

Overview of the influence of group motion commands on a single axis state:

Command	Group State	Axis State
MC_MoveLinear... MC_MoveCircular...	GroupMoving	SynchronizedMotion
MC_GroupStop	GroupStopping / GroupStandby	SynchronizedMotion / StandStill
MC_GroupReset	GroupErrorStop / GroupStandby	One or more axis in ErrorStop

Table 9: Influence of group motion commands on a single axis state

5.4 Blending and Buffering of Movements

5.4.1 Blending

Blending is required for applications requiring continuous motion between two different command moves. With single axis the end position of each consecutive motion command is always reached; the commanded end position of the first command forms the start position of the next command.
However, in blended interpolation motion, the path does not pass through all of the points in the original trajectory. Blending modifies the original path to archive a smooth trajectory without corners. The end position of the first interpolated command and the start position of the second command are modified in such a way to prevent the axis to come to standstill.
Currently only the blending mode '*TMStartAccNext*' is being supported (Figure 10). This mode determines during the deceleration phase the time at which the next command takes control of the axis group. The corner contour is defined by the '*TransitionPara*' which specifies the percentage value of the programmed acceleration time at which the next command takes control.

No.	MC_TRANSITION_MODE	Description
0	TMNone	Insert no transition curve
1	TMStartVelocity	Not supported
2	TMConstVelocity	Not supported
3	TMConerDistance	Not supported
4	TMConerDeviation	Not supported
5 - 9	Reserved by PLCopen	Not supported
10	TMStartAccNext	Set blending percentage

Table 10: Overview of Transition Modes

For example, a trajectory is specified as a sequence of more than one path segments whereby each segment is controlled by a separate command. A velocity profile is calculated for each of the path segments, wherein each velocity profile is divided into a blend-in region, a blend-out region and a remainder region. Each path segment is executed such that the blend-in region of its velocity profile overlaps only with the blend-out region of the previous profile. The '*TransitionPara*' determines on the deceleration curve the start time of the blend-in region of the next path segment (Table 11).

- '*TransitionPara*' = 0
  No blending takes place and the group axes will decelerate to zero at the end of each segment and then accelerate to the target velocity of the next segment.

- '*TransitionPara*' = 50
  Overlaps of the blend-in and -out region starts after 50 % of the deceleration time of the path segment has elapsed.
- '*TransitionPara*' = 100
  Overlaps of the blend-in and -out region starts immediately when the path segment starts to decelerate.

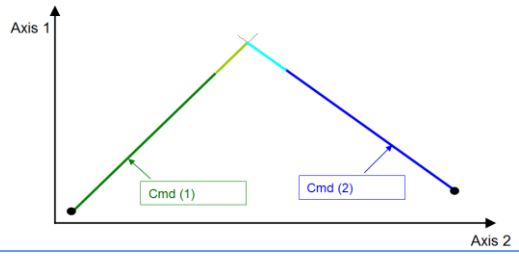
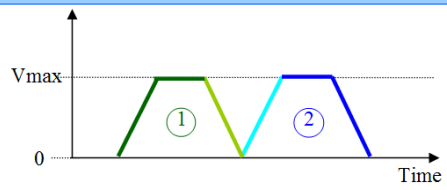
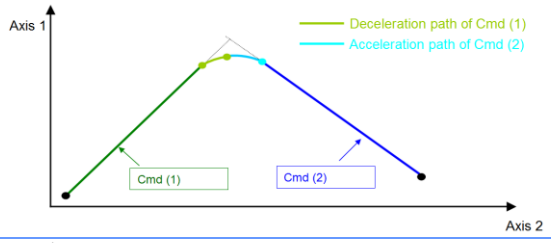
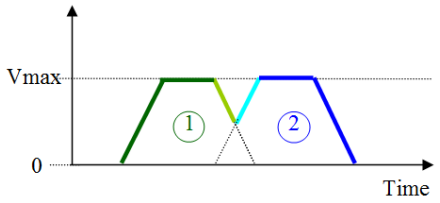
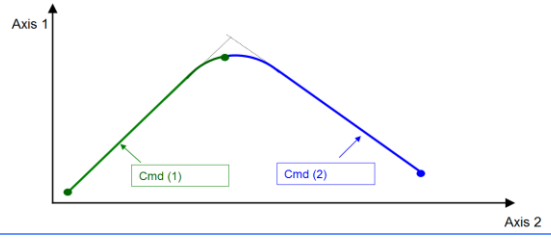
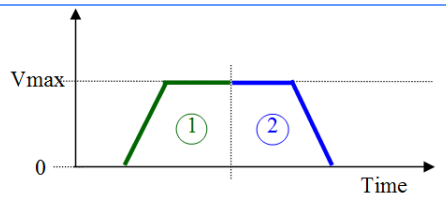
TransPara	AccDecTime	
0		
50		
100		

Table 11: Transition parameter setting for 'mcBlendingNext'

Note:
- The acceleration and deceleration time is equal for each movement command
- The acceleration and deceleration time can only be set for the first path segment before the movement starts. After the movement starts the time can not be modified and all the following path segments will refer to the same time setting.

Example:
Figure 12 shows the original path deviations due to different '*TransitionPara*' settings (percentage values).

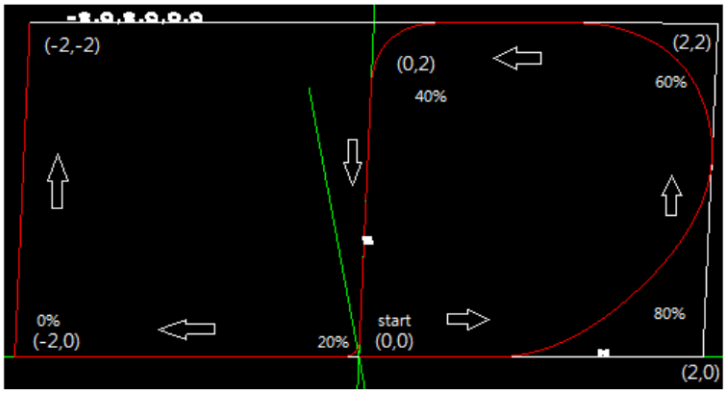


Figure 12: Contour curves with different '*TransitionPara*' settings

5.4.2 Buffering

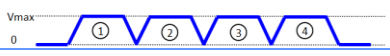
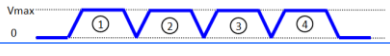
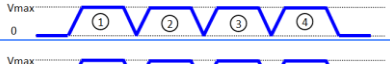
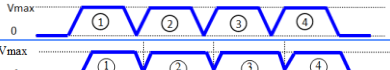
For axes group movement the same buffer modes are used as for single axis MFBs (Table 12).

No.	MC_BUFFER_MODE	Description
0	mcAborting	Start MFB immediately
1	mcBuffered	Start MFB after all the commands in the buffer has finished
2	mcBlendingLow	Not supported
3	mcBlendingPrevious	Not supported
4	mcBlendingNext	The velocity is blended with the velocity of the second FB
5	mcBlendingHigh	Not supported

Table 12:Overview of Buffer Modes

The supported buffer modes and its relationship with the transition modes and parameters are shown in Table 13. Each circled numbers in the velocity time graph represents a new group command.

The '*TransitionMode*' setting will be ignored if the '*BufferMode*' is set to '*mcAborting*' or 'mcBuffered'.

BufferMode	TransitionMode	TransPara	AccDecTime	Velocity-Time
mcBuffered	TMNone	0 ... 100	1...999	
	TMStartAccNext	0 ... 100	1...999	
mcBlendingNext	TMNone	0 ... 100	1...999	
	TMStartAccNext	0	1..999	
	TMStartAccNext	50	1..999	

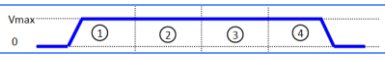
	TMStartAccNext	100	1..999	

Table 13: Buffer modes

6 Axis Group Motion Function Blocks

This chapter describes the Win-GRAF PLCopen library for group motion control function blocks such as linear and circular interpolation.

6.1 MC_AddAxisToGroup

This MFB adds one axis to a group. Up to 32 axis can be added to one group. The ECAT-M801/M901 supports a maximum of 8 groups. Axis can only be added or removed from the group when it is in a disabled state ('*GroupStandby*'). Adding an axis to the group will not automatically enable the group. Before using a group motion command enable the group ('*MC_GroupEnable*') after all the axis has been assigned.

Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
	Axis	AXIS_REF	Reference to the axis to be added
VAR_INPUT	Execute	BOOL	Start the action at rising edge
VAR_OUTPUT	Done	BOOL	True when the axis has been successfully added to the group.
	Busy	BOOL	Function block is busy executing
	Error	BOOL	An error has occurred while adding the axis to the group
	ErrorID	DINT	Error identification -

Remarks:

- The MFB sets '*Done*' if the axis has been successfully added to the group or has already been part of the group before the MFB execution.
- This MFB can only be called while the group is in '*GroupDisabled*' state.
- All axes will be automatically removed from the all the groups once the '*MCV_AXISASSIGN.Execute*' is being triggered.
- The same axis can be assigned to more than one group, but only one group of the group that share the same axis can be enabled at a time

Example:

The following example shows an extract of a PLC program which uses '*MC_AddAxisToGroup*':

```
VAR
      Inst_MC_ADDAXISTOGROUP : ARRAY [0 .. 5] OF MC_AddAxisToGroup ;
      AxesGroup : ARRAY [0 .. 2] OF lib:AXES_GROUP_REF ;
      flgAddToGroup : BOOL := TRUE ;
      AxisRef : ARRAY [0 .. 5] OF lib:AXIS_REF ;
END_VAR
```

```
//=============================================================
//Add to Group:
//=============================================================
AxesGroup[0].GroupNo := 0;
AxesGroup[0].CardNo := AxisRef[0].CardNo;

AxesGroup[1].GroupNo := 1;
AxesGroup[1].CardNo := AxisRef[1].CardNo;

AxesGroup[2].GroupNo := 2;
AxesGroup[2].CardNo := AxisRef[2].CardNo;

//Add axis 0 to group 0:
Inst_MC_ADDAXISTOGROUP[0]( AxesGroup[0](*lib:AXES_GROUP_REF*),
AxisRef[0](*lib:AXIS_REF*), flgAddToGroup(*BOOL*) );

//Add axis 1 to group 0 after axis 0 has been added:
if Inst_MC_ADDAXISTOGROUP[0].Done = TRUE then
    Inst_MC_ADDAXISTOGROUP[1]( AxesGroup[0](*lib:AXES_GROUP_REF*),
AxisRef[1](*lib:AXIS_REF*), flgAddToGroup(*BOOL*) );
end_if;

//Add axis 2 to group 1:
if Inst_MC_ADDAXISTOGROUP[1].Done = TRUE then
```

```
    Inst_MC_ADDAXISTOGROUP[2]( AxesGroup[1](*lib:AXES_GROUP_REF*),
AxisRef[2](*lib:AXIS_REF*), flgAddToGroup(*BOOL*) );
end_if;


//Add axis 3 to group 1 after axis 2 has been added::
if Inst_MC_ADDAXISTOGROUP[2].Done = TRUE then
    Inst_MC_ADDAXISTOGROUP[3]( AxesGroup[1](*lib:AXES_GROUP_REF*),
AxisRef[3](*lib:AXIS_REF*), flgAddToGroup(*BOOL*) );
end_if;


//Add axis 4 to group 2:
if Inst_MC_ADDAXISTOGROUP[3].Done = TRUE then
    Inst_MC_ADDAXISTOGROUP[4]( AxesGroup[2](*lib:AXES_GROUP_REF*),
AxisRef[4](*lib:AXIS_REF*), flgAddToGroup(*BOOL*) );
end_if;


//Add axis 5 to group 2 after axis 4 has been added::
if Inst_MC_ADDAXISTOGROUP[4].Done = TRUE then
    Inst_MC_ADDAXISTOGROUP[5]( AxesGroup[2](*lib:AXES_GROUP_REF*),
AxisRef[5](*lib:AXIS_REF*), flgAddToGroup(*BOOL*) );
end_if;
```

6.2 MC_RemoveAxisFromGroup

This MFB removes one axis from the group. This MFB can only be called when the group is disabled ('*GroupDisabled*').

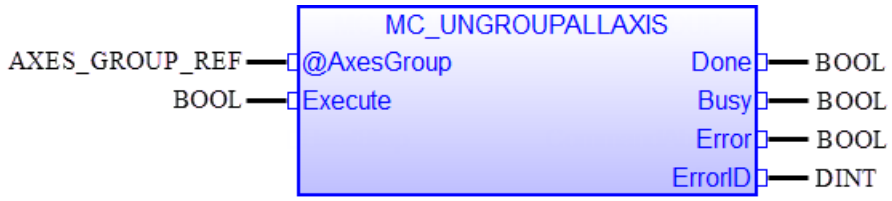
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
	Axis	AXIS_REF	Reference to the axis to be removed
VAR_INPUT	Execute	BOOL	Start the axis removal at rising edge
VAR_OUTPUT	Done	BOOL	True when the axis has been successfully removed from the group.
	Busy	BOOL	TRUE - Function block is busy executing. When Busy becomes FALSE again, the function block is ready for a new command.
	Error	BOOL	An error has occurred while adding the axis to the group
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The MFB returns '*Done'*=TRUE if the axis is not part of the group anymore. It will also return TRUE if the axis was not part of the group before calling this MFB.

6.3 MC_UngroupAllAxes

This MFB removes all axes from the group. This MFB can only be called when the group is disabled ('*GroupDisabled*').



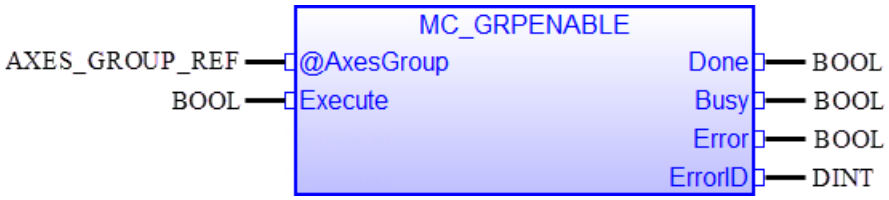
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start removing all axes at rising edge
VAR_OUTPUT	Done	BOOL	True when all axis has been successfully removed from the group.
	Busy	BOOL	TRUE - Function block is busy executing. When Busy becomes FALSE again, the function block is ready for a new command.
	Error	BOOL	An error has occurred while adding the axis to the group
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The MFB returns '*Done*'=TRUE if no axis is part of the group anymore. This means it also returns '*Done*'=TRUE if the group is already empty prior calling this FB.

6.4 MC_GroupEnable

This MFB changes the state of a group from '*GroupDisabled*' to '*GroupStandby*'. The group has to be enabled before calling and group moving MFBs.
More than one group can be enable at the same time, but it is not possible to enable more than group which includes the same axis. If the same axis is part of different groups then it is necessary to first disable the currently active group with the axis before enabling the next group containing the same axis. The group member axes all have to be in '*Standstill*' otherwise the group can not be enabled.



Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start enabling the group at rising edge
VAR_OUTPUT	Done	BOOL	True when the group has been successfully enabled.
	Busy	BOOL	TRUE - Function block is busy executing. When Busy becomes FALSE again, the function block is ready for a new command.
	Error	BOOL	An error has occurred while enabling the group
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The group cannot be enabled in the following cases:
  - If no axis has been added to the group (empty group) via '*MC_AddAxisToGrp*'
  - If the axis of the group to be enabled belongs to other enabled axis group
  - If a single axis of the group is executing a single axis motion command (e.g. the moving axis is controlled by a single axis command).
  - If a single axis of the group is in '*Disabled*', '*Stopping*' or '*ErrorStop*' state.
- ATTENTION:
  - No single axis motion command can be issued once the axis is part of an enabled group. Commands such as '*MC_Stop*', '*MC_MoveAbsolute*', '*MC_MoveRelative*' will return an error if they are called while the axis is assigned to an enabled group.
  - Single axis administrative MFBs can still be called while the axis is part of an enabled group.

- If the '*MC_GrpStop.Execute*' is TRUE when the group is being enabled then the group switches into '*GroupStopping*' state.
- The function '*MC_AddAxisToGrp*', '*MC_RemoveAxisfromGrp*' or 'MC_UngroupAllAxis' can not be executed on an enabled group.

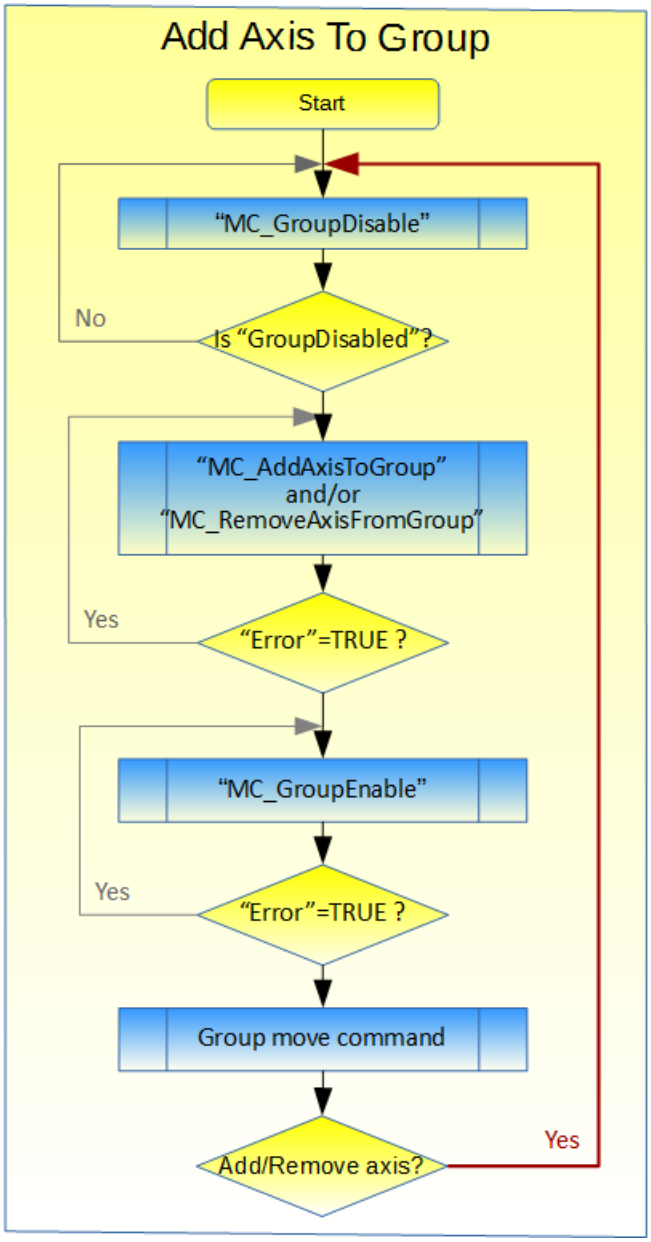
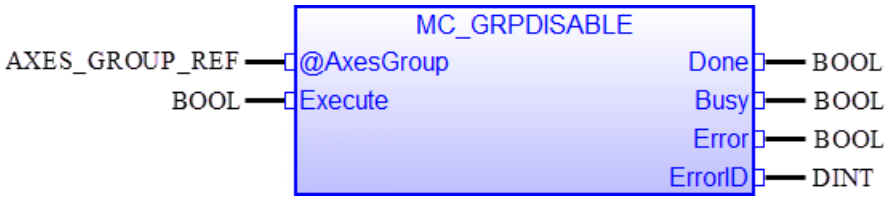


Figure 13: Adding/Removing axis to group

6.5 MC_GroupDisable

This MFB changes the state of the group to '*GroubDisabled*'. A moving group should be first stopped by calling '*MC_GrpHalt*' or '*MC_GrpStop*' before executing '*MC_GrpDisable*' otherwise it will result in an immediate stop.



Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start disabling the group at rising edge
VAR_OUTPUT	Done	BOOL	True when the group has been successfully disabled.
	Busy	BOOL	TRUE - Function block is busy executing. When Busy becomes FALSE again, the function block is ready for a new command.
	Error	BOOL	An error has occurred while disabling the group
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- A moving group will stop instantly by this MFB. The sudden stop may exceed the servo drive allowed deceleration limits. Depending of the drive hardware this might result in power peaks and runtime errors. To prevent a sudden stop it necessary to first decelerate a moving group to standstill by executing '*MC_GrpHalt*' or '*MC_GrpStop*' before disabling the group.
- The group can be disabled while in '*GroupMoving*', '*GroupStopping*', '*GroupErrorStop*' and '*GroupStandby*'.
- When the group is being disabled the following action are being taken internally:
  - The group errors '*GroupErrorStop*' will be automatically reset
  - The stop status '*GroupStopping*' will be cleared even if the '*MC_GrpStop.Execute*' is still active. The group returns into '*GroupStopping*' when it is being enabled while '*MC_GrpStop.Execute*' is TRUE.

6.6 MC_GroupReadActualPosition

This MFB returns for each axes in the group either the actual encoder positions of the stepper/servo drives or the current commanded positions of the motion control engine inside the ECAT-M801/e-M901 card.

Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Enable	BOOL	• TRUE: Continuously get the position of the axes group
	CmdPos	BOOL	• FALSE: Read the actual position • TRUE: Read the commanded position
VAR_OUTPUT	Valid	BOOL	• TRUE: Indicates that the position output values are valid.
	Busy	BOOL	• TRUE: Function block is busy getting the position values.
	Error	BOOL	• TRUE: An error has occurred while reading the positions of the axes
	ErrorID	DINT	Error identification Command-specific error code
	Position	ARRAY[1..N] OF LREAL	Current or commanded position values of the group ATTENTION: Although the parameter is shown on the input side of the function block it acts as an output parameter.

Remarks:

•

6.7 MC_GroupReadActualVelocity

This MFB returns for each axes in the group either the actual velocity of the stepper/servo drives or the commanded velocity of the motion control engine inside the ECAT-M801/e-M901 card.

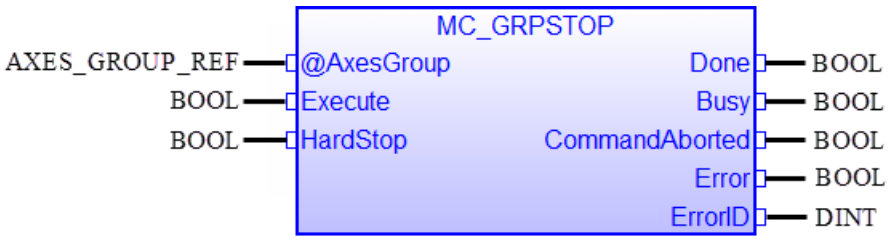
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Enable	BOOL	• TRUE: Continuously get the velocity of the axes group
	CmdPos	BOOL	• FALSE: Read the actual velocity • TRUE: Read the commanded velocity
VAR_OUTPUT	Valid	BOOL	• TRUE: Indicates that the velocity output values are valid.
	Busy	BOOL	• TRUE: FB is busy getting the velocity values.
	Error	BOOL	• TRUE: An error has occurred while reading the velocity
	ErrorID	DINT	Error identification Command-specific error code
	Velocity	ARRAY[1..N] OF LREAL	Current or commanded velocity values of the group ATTENTION: Although the parameter is shown on the input side of the function block it acts as an output.

Remarks:

•

6.8 MC_GroupStop

This MFB stops all the axes assigned to the group and transfers the axes group to the state '*GroupStopping*'. Two types of stop behaviors are supported: A sudden stop without deceleration or a controlled stop by decelerating to zero. While the group is in state '*GroupStopping*', no other MFB can move any group axis (see Group State Diagram). The '*MC_GrpStop.Done*' output is set to TRUE once the velocity reaches zero. The group axes can only be moved again once the '*MC_GrpStop.Execute*' input has been set to FALSE after the velocity reached zero. The axes group remains in the state '*GroupStopping*' as long as '*MC_GrpStop.Execute*' is TRUE or one of the axis in the group is still moving. The group transits to the state '*GroupStandBy*' when both conditions '*MC_GrpStop.Done*' is TRUE and '*MC_GrpStop.Execute*' is FALSE are met.

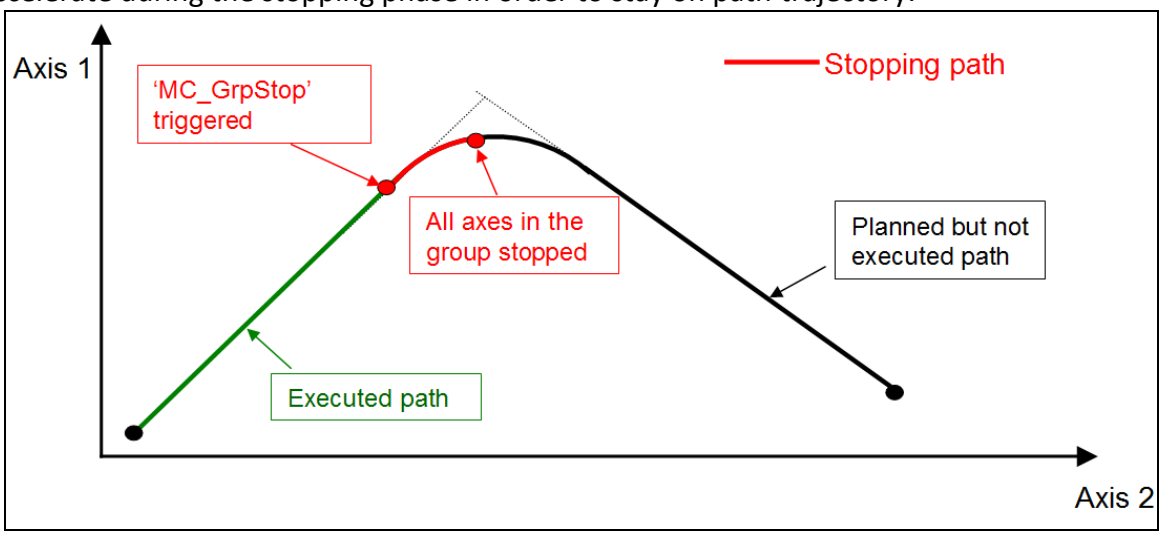


Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	• TRUE: Immediate stop (Hardstop=TRUE) or decelerates to stop (Hardstop=FALSE)
	HardStop	BOOL	• FALSE: Decelerate to stop within the deceleration time set by the MC_MoveXXX comand. • TRUE: Stop without deceleration
VAR_OUTPUT	Done	BOOL	• TRUE: The axes group has reached velocity zero.
	Busy	BOOL	• TRUE: Function block is busy decelerating the axes group.
	CommandAborted	BOOL	• Command is aborted by disabling 'MC_Power' of one or more axes in the group. The state changes to 'GroupDisabled'.
	Error	BOOL	• TRUE: An error occurred during the stopping process
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- For '*MC_GrpStop.HardStop*'= FALSE the group axes remain on the path during the deceleration phase. Therefore it may occur that single axis within the group may

accelerate during the stopping phase in order to stay on path trajectory.



- The deceleration time for '*MC_GrpStop.HardStop*'= FALSE is set by the current executing MFB (e.g. '*MC_MoveLinearAbsolute*', '*MC_MoveLinearRelative*'). The '*AccDecTime*' input of the first MFB in the buffer sets the acceleration and deceleration time for all following motion commands.
- A timing diagram with the state transitions for '*MC_GroupStop*' is shown below.
  - In case 1 the '*Done*' output is set (TRUE) before the '*Execute*' is reset (FALSE).
  - In case 2 the *'Execute*' is reset before '*Done*' turns true.

  The group stays in 'Group*Stopping*'
  - as long as '*Execute*' is set (TRUE)
  - until the condition '*Done*' is TRUE after '*Execute*' has been set to FALSE.

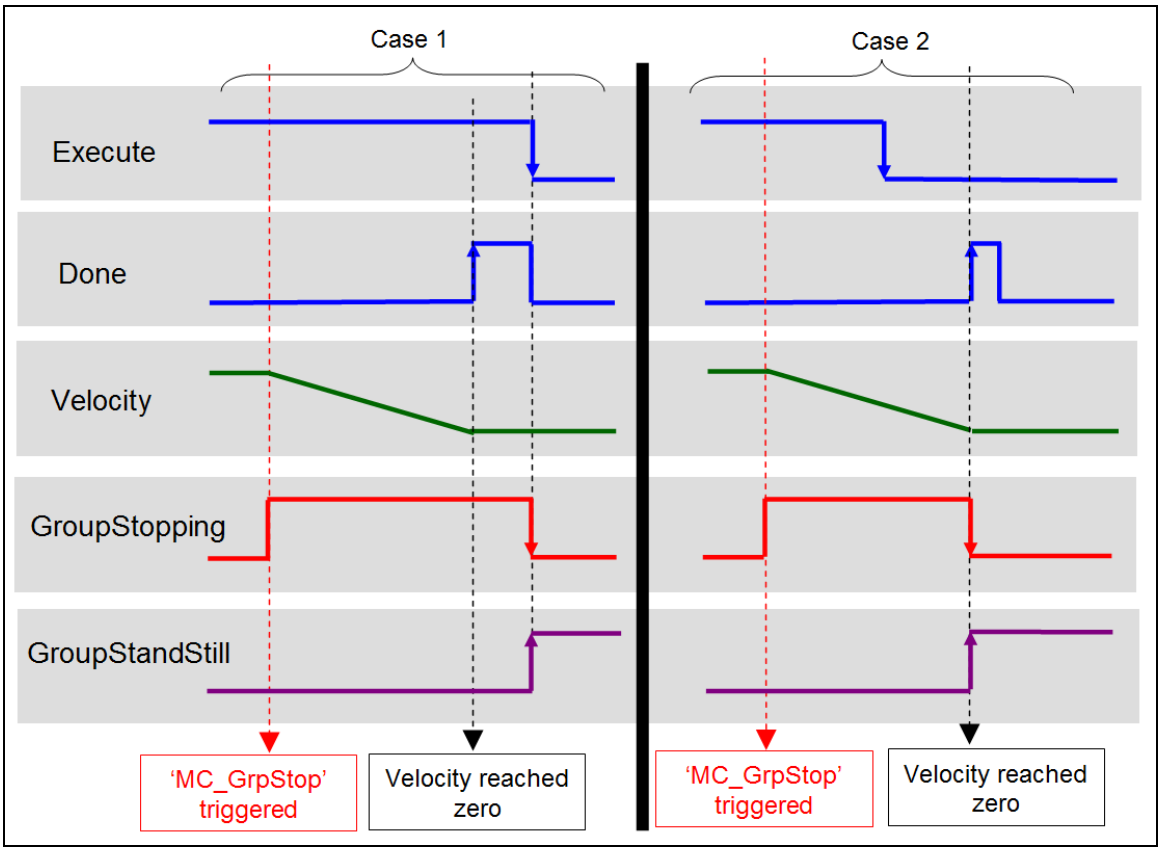


Figure 14: MC_GroupStop timing diagram

- The example below shows the behavior in combination with a '*MC_MoveLinearRelative*'.
  c) An axes group in linear movement is ramped down with '*MC_GroupStop*'. The group stops on the original path.
  d) The axes group rejects motion commands as long as '*MC_GroupStop. Execute*' is TRUE. '*MC_MoveLinearRelative*' reports an error indicating the enabled '*MC_GroupStop*' command. The returned error is an function block and not an axis error, which means the state remains in 'GroupStopping' and will not change to '*GroupErrorStop*'. At the 3rd “Exe_1” rising edge, the group starts the next movement.

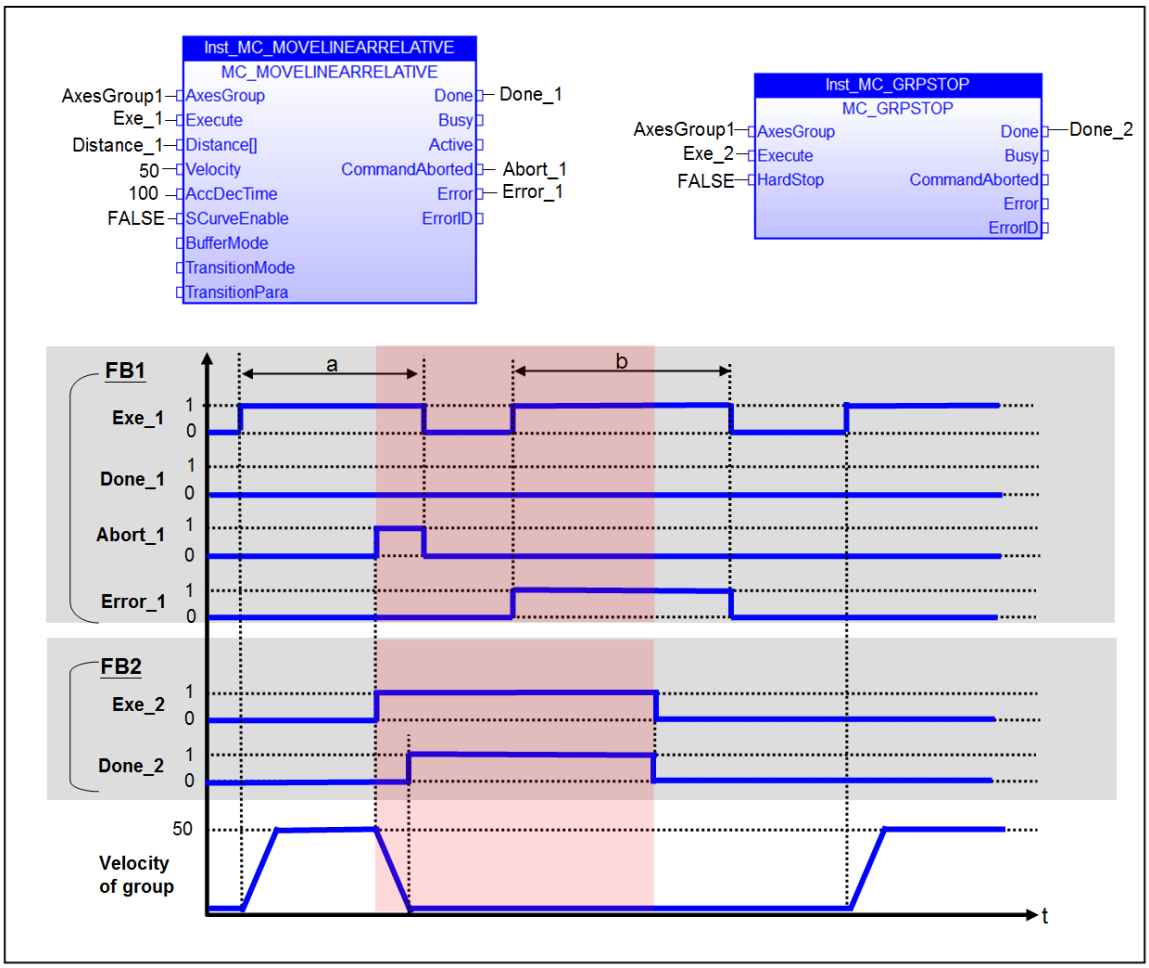
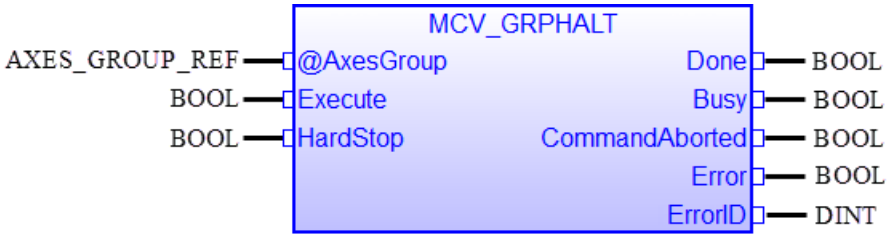


Figure 15: Behavior of '*MC_GroupStop*' in combination with '*MC_MoveLinearRelative*'

6.9 MCV_GroupHalt

This MFB commands a controlled motion stop and aborts any ongoing motion commands and clears any buffered commands. '*AxesGroup*' is transition to the state '*GroupStopping*', until the velocities of all group member axes are zero. With the '*Done*' output set (TRUE), the state changes to '*GroupStandby*'. The difference between '*MC_GrpStop*' and '*MC_GrpHalt*' is that the '*MC_GrpHalt*' automatically transitions the group state to '*GroupStandby*' once the velocity reached zero even when the '*Execute*' is still TRUE. On the contrary, the '*MC_GrpStop*' waits until the '*Done*' turns TRUE and the input '*Execute*' is FALSE before switching into '*GroupStandBy*' state.



Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	• TRUE: Immediate stop (Hardstop=TRUE) or decelerates to stop (Hardstop=FALSE)
	HardStop	BOOL	• FALSE: Decelerate to stop within the deceleration time set by the MC_MoveXXX command. • TRUE: Stop without deceleration
VAR_OUTPUT	Done	BOOL	• TRUE: The axes group has reached velocity zero.
	Busy	BOOL	• TRUE: Function block is busy decelerating the axes group.
	CommandAborted	BOOL	• Command is aborted by disabling '*MC_Power*' of one or more axes in the group. The state changes to '*GroupErrorStop*'.
	Error	BOOL	• TRUE: An error occurred during the stopping process
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The group axes remain on the path during the deceleration phase, therefore single axis of the group might accelerate in between due to the given path.
- '*MCV_GroupHalt*' does not support buffer mode. '*MCV_GroupHalt*' will be executed immediately and all the motion commands currently in the buffer will be aborted.

- After all the axes has reached standstill the group state automatically switches from '*GroupStopping*' to '*GroupStandby*'. Once the state has switch to '*GroupStandby*' the next motion command can be executed.
- The next group motion command can be issued even if '*MCV_GroupHalt .Execute*' is TRUE as long as the group state is '*GroupStandBy*'.

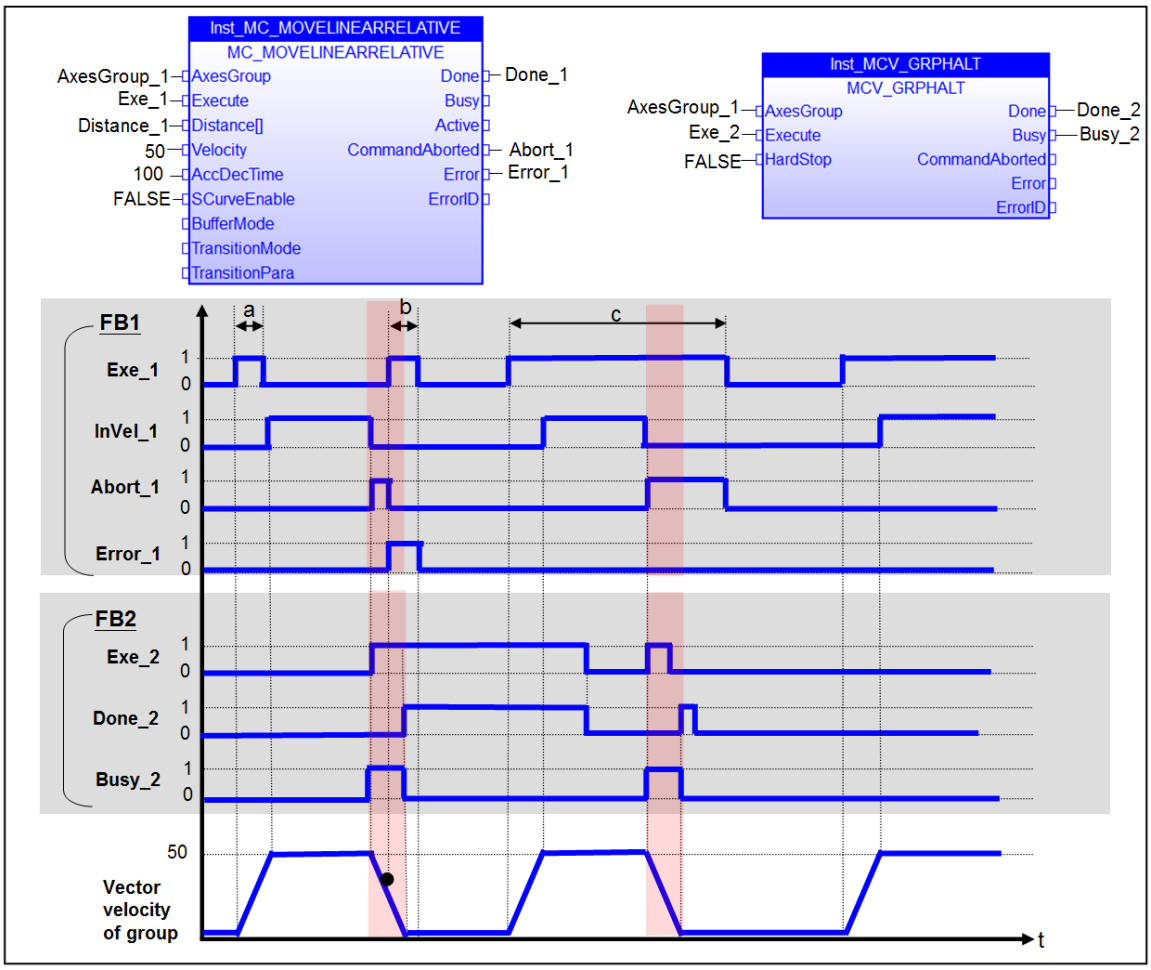


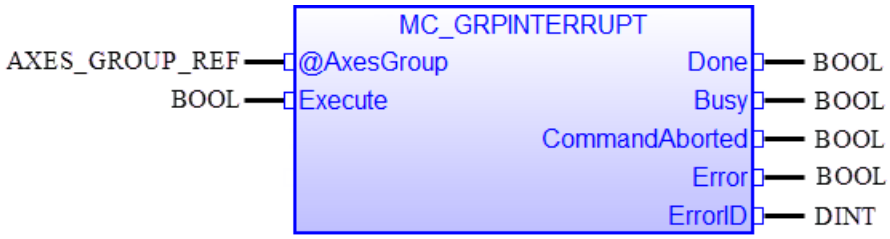
Figure 16: Behavior of '*MC_GroupHalt*' in combination with '*MC_MoveLinearRelative*'

Figure 16 shows an example of the behavior of the '*MC_GrpHalt*' in conjunction with '*MC_MoveLinearRelative*' :

- While the '*MCV_GrpHalt*' command is busy no other motion command for the group can be executed.
- Once '*MCV_GrpHalt.Done*' is TRUE for at least one cycle a new movement command can be executed; even if the '*MCV_GrpHalt.Execute*' is still TRUE. In contrast the '*MC_GroupStop*' only allows the execution of the next movement if the '*MC_GroupStop.Done*' = TRUE and '*MC_GroupStop.Execute*' = FALSE.

6.10 MC_GroupInterrupt

This MFB interrupts a move command and commands a controlled stop along the commanded trajectory. The interrupted move command will not be allowed to continue and reach its target position until the '*MC_GroupContinue.Execute*' is being triggered. The interrupted MFB commands is not being aborted and will continue once the '*MC_GroupContinue.Execute*' is being triggered. This MFB does not change the group state which means the '*AxesGroup*' state remains in the original state even after the all the member axes have stopped and the '*Done*' output set.



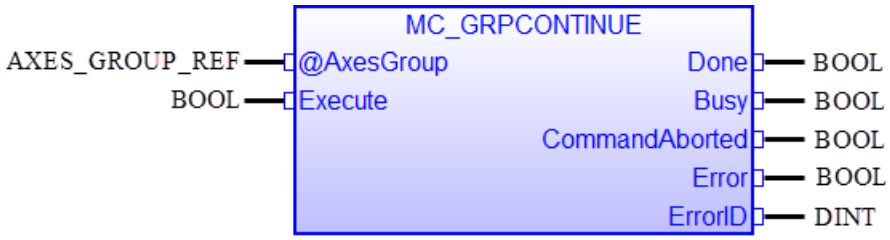
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	A rising edge triggers the MFB command
VAR_OUTPUT	Done	BOOL	TRUE: The axes group has reached velocity zero.
	Busy	BOOL	TRUE: Function block is busy decelerating the axes group.
	CommandAborted	BOOL	The MFB is being aborted by another command
	Error	BOOL	TRUE: An error occurred during the stopping process
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The deceleration is being determined by the MFB which accelerated the axes from standstill.
- The group axes remain on the commanded trajectory during the deceleration phase.
- When velocity zero is reached the '*Done*' output is set.
- The interrupted MFB commands is not being aborted and therefore the output '*CommandAborted*' will not turn TRUE. The '*Busy*' output remains TRUE and the '*Active*' is set to FALSE.
- The group remains in the interrupt state until the group velocity reaches zero and '*MC_GrpContinue*' is being executed.

6.11 MC_GroupContinue

This MFB transfers control over the group axes back to the original MFB which has been interrupted by the '*MC_GroupInterrupt*'.



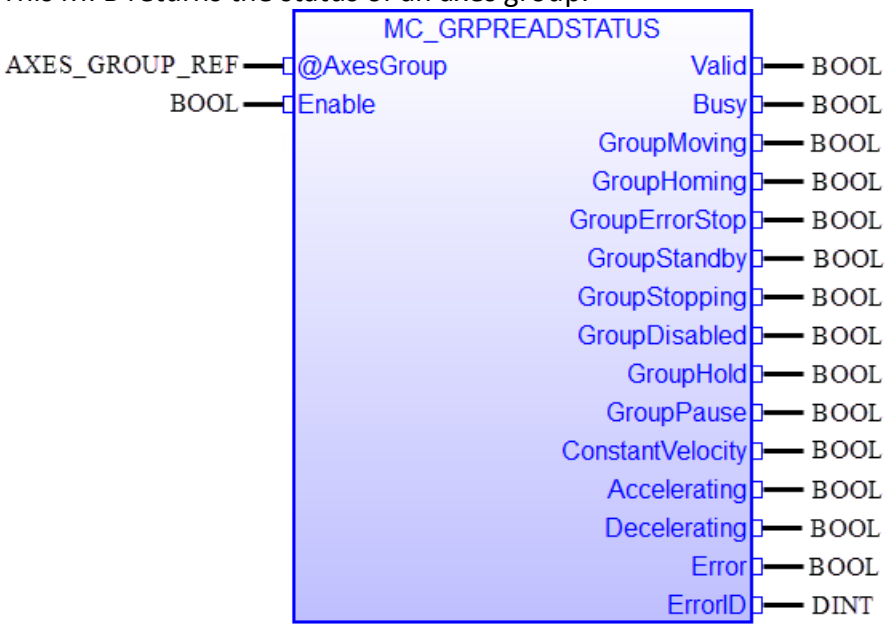
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	A rising edge triggers the FB execution
VAR_OUTPUT	Done	BOOL	TRUE: The interrupted MFB regain control of the group axes
	Busy	BOOL	TRUE: Function block is busy transferring control back to the interrupted MFB
	CommandAborted	BOOL	The FB is being aborted by another command
	Error	BOOL	TRUE: An error has occurred
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- Once the original MFB has full control of the group again
  - the '*MC_GroupContinue.Done*' is set to TRUE and
  - the original MFB '*Active*' output turns TRUE
- The group axes velocity has to be zero before the original MFB can take back control.
- The deceleration is being determined by the MFB which accelerated the axes from standstill.
- The group axes remain on the commanded trajectory during the deceleration phase.
- When velocity zero is reached the '*Done*' output is set.

6.12 MC_GroupReadStatus

This MFB returns the status of an axes group.



Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Enable	BOOL	TRUE: Continuously get the status of the axes group
VAR_OUTPUT	Valid	BOOL	TRUE: The status outputs are valid.
	Busy	BOOL	TRUE: Function block is busy getting the axes group status.
	GroupMoving	BOOL	See group state diagram
	GroupHoming	BOOL	See group state diagram
	GroupErrorStop	BOOL	See group state diagram
	GroupStandby	BOOL	See group state diagram
	GroupStopping	BOOL	See group state diagram
	GroupDisabled	BOOL	See group state diagram
	GroupHold	BOOL	Reserved
	GroupPause	BOOL	Reserved
	ConstantVelocity	BOOL	Moving with constant velocity on commanded path
	Accelerating	BOOL	'AxedGroup' is accelerating on commanded path
	Decelerating	BOOL	'AxedGroup' is decelerating on commanded path
	Error	BOOL	TRUE: An error occurred during reading of the group status
	ErrorID	DINT	Error identification Command-specific error code

6.13 MC_GroupReadError

This MFB returns the error code of the axes group. This error is not related to the function block error output parameter (e.g. invalid parameterization). If an group error occurred the group axes will stop and once all axes velocity is zero the group state is transition to '*GroupErrorStop*'.

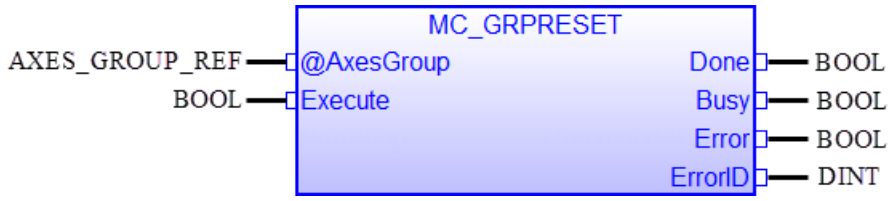
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Enable	BOOL	• TRUE: Continuously reads the "*GroupErrorID*" of the axes group
VAR_OUTPUT	Valid	BOOL	• TRUE: Outputs are valid.
	Busy	BOOL	• TRUE: Function block is busy reading the "*GroupErrorID*" .
	Error	BOOL	• TRUE: An error occurred during reading of the "*GroupErrorID*" .
	ErrorID	DINT	Error identification Command-specific error code
	GroupErrorID	DINT	The value of the axes group error. See error table in appendix

Remarks:

- If the '*GroupErrorID*' output is zero then there is no error on the axes group.
- The '*GroupErrorID*' output is valid only if the output '*Valid*' is TRUE.
- An axes group error can for example be:
  - Exceeding of the soft limit
  - Hardware limit switch has been activated
  - Driver emergency is active
  - Communication error: No cyclic data exchange with the remote servo drive is possible; EtherCAT master switched into SafeOP mode, etc..

6.14 MC_GroupReset

This MFB resets all the axes error in the group and transition from the state '*GroupErrorStop*' to '*GroupStandby*' once all the errors have been cleared. If the error could not be reset (e.g. communication error between EtherCAT master and slave) then the group state remains in '*GroupErrorStop*' state.



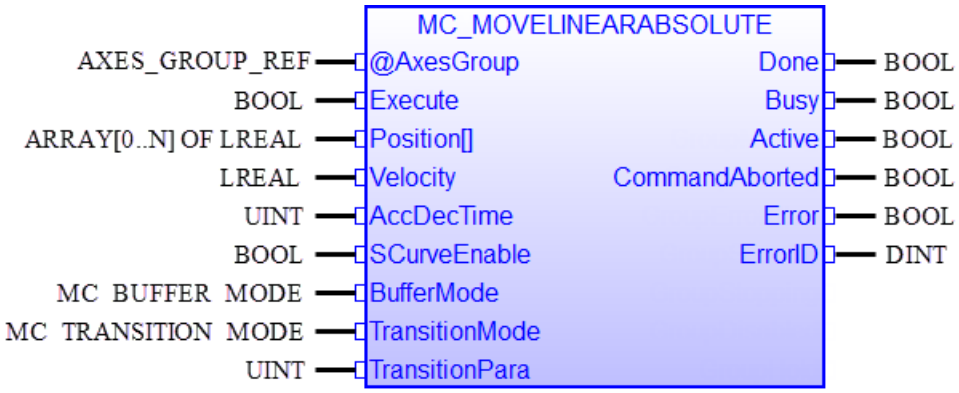
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Enable	BOOL	• TRUE: Continuously reads the "*GroupErrorID*" of the axes group
VAR_OUTPUT	Done	BOOL	• TRUE: All the axes inside the group have been reset.
	Busy	BOOL	• TRUE: Busy resetting all the axes of the group
	Error	BOOL	• TRUE: An error occurred during the reset process
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- This MFB also resets all axes in this group like the single axis FB '*MC_Reset*'.

6.15 MC_MoveLinearAbsolute

This MFB commands a linear interpolated movement from the current position to the specified absolute position.



Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start the motion at rising edge
	Position[]	ARRAY[0..15] OF LREAL	• Array [0..15] of absolute end positions for each dimension. • A maximum number of 16 axes are supported (N<=16) • If the axes group has got four axes then the array [0..3] must have at least four absolute target positions
	Velocity	LREAL	Maximum Velocity [u/s] for the path. Always positive.
	AccDecTime	UINT	The acceleration time [ms] to reach the maximum velocity. The deceleration time for decelerating from the maximum velocity to zero
	SCurveEnable	BOOL	Acceleration/deceleration curve • FALSE: Trapezoidal curve (T-curve) • TRUE: S-curve
	BufferMode	MC_BUFFER_MODE	See buffer mode definition
	TransitionMode	MC_TRANSITION_MODE	See transition mode definition
	TransitionPara	UINT	0 to 100
VAR_OUTPUT	Done	BOOL	• TRUE: Command has successfully reached the target position and finished executing
	Busy	BOOL	• TRUE: The Function Block is busy either waiting to be executed or executing.

	Active	BOOL	• TRUE: The Function Block has control of the axes group and is busy executing.
	CommandAborted	BOOL	• Command has been aborted by another command.
	Error	BOOL	• TRUE: An error occurred during the command execution.
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- While this MFB is active the group is in the '*GroupMoving*' state.
- When the absolute position is reached, the '*Done*' output will be set TRUE and the group state changes to the '*GroupStandby*' or starts to execute another move command.
  - This MFB completes with velocity zero if no further actions are pending.
  - In '*mcBuffered*' mode: When the motion is complete, the axis returns to the '*GroupStandby*' state.
- The input '*Velocity*' will not be reached if the distance to move is shorter than the distance needed for the acceleration and deceleration.
- The '*AccDecTime*' will be ignored if the group is already moving. The '*AccDecTime*' will only be considered if the group is starting from velocity zero (state '*GroupStandby*').
- The supported buffer modes and its relationship with the transition modes and parameters are shown in Table 13. Each circled numbers in the velocity time graph represents a new group command.
- '*TransitionPara*': Set the blending percent of a group. Blending will introduce a smooth transition from one command to another; however, it will produce corner error. In the 'mcBlendingNext' mode
  - "100" blending percent means to blend the next motion command from the starting of deceleration of the pervious motion command.
  - "0" blending percent means no blending part; and the behavior is similar to the 'mcBuffered' command mode.

  The red line in the graphs (Table 11) shows the interpolation path with different '*TransitionPara*' settings in 'mcBlendingNext' mode:
- While this MFB is active the group axes are in the '*SynchronizedMotion*' state.
- Timing diagram: see PLCopen: "Function blocks for motion control (Part4)" figure 15 page 57/119.

Example:

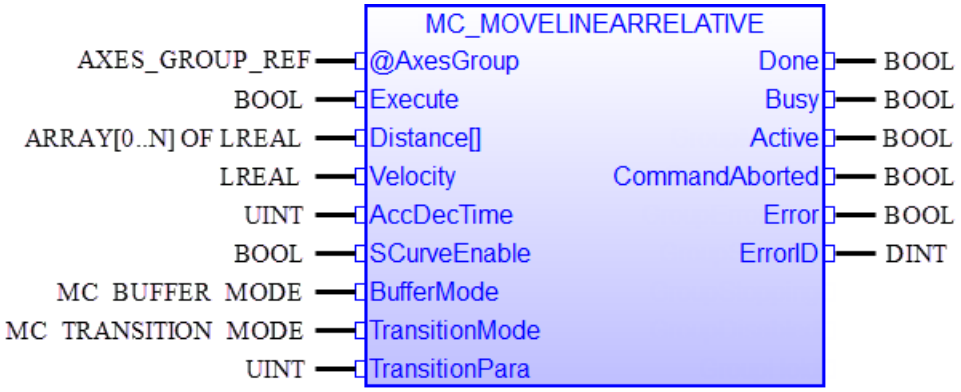
```
VAR
      flgAddToGroup : BOOL := TRUE ;
      flgMoveLinear : BOOL := FALSE ;
      AxesGroup: lib:AXES_GROUP_REF := [UINT#1,UINT#1] ;
      dfPosition : ARRAY [0 .. 5] OF LREAL :=   LREAL#10000,LREAL#20000,
```

```
                        LREAL#30000,LREAL#40000,LREAL#50000,LREAL#60000  ;
      dfVectorVelocity : LREAL := LREAL#1000 ;
      AccDecTime : UINT := UINT#999 ;
      wTransitionPara : UINT := UINT#50 ;
      Inst_MC_MoveLinearAbsolute : MC_MoveLinearAbsolute ;

      MovAbsDone: BOOL ;
      MovAbsBusy: BOOL ;
      MovAbsActive: BOOL ;
      MovAbsAborted: BOOL ;
      MovAbsError: BOOL ;
      MovAbsErrID: BOOL ;
END_VAR

(* Initialize the ECAT-M8000 *)
//...
(* Axis Assignment *)
//...


(* Execute an instance of MC_MOVELINEARABSOLUTE *)
flgMoveLinear := TRUE ;
Inst_MC_MOVELINEARABSOLUTE( AxesGroup(*lib:AXES_GROUP_REF*),
                            flgMoveLinear(*BOOL*),
                            dfPosition(*LREAL*),
                            dfVectorVelocity(*LREAL*),
                            AccDecTime(*UINT*),
                            FALSE(*SCurveEnable BOOL*),
                            MC_BUFFER_MODE#mcBuffered,
                            MC_TRANSITION_MODE#TMStartAccNext,
                            wTransitionPara(*UINT*) );

(* Get function block output status: *)
MovAbsDone := Inst_MC_MoveLinearAbsolute.Done;
MovAbsBusy := Inst_MC_MoveLinearAbsolute.Busy;
MovAbsActive := Inst_MC_MoveLinearAbsolute.Active;
MovAbsAborted := Inst_MC_MoveLinearAbsolute.CommandAborted;
```

```
MovAbsError := Inst_MC_MoveLinearAbsolute.Error;
MovAbsErrID := Inst_MC_MoveLinearAbsolute.ErrorID;
```

6.16 MC_MoveLinearRelative

This MFB executes a linear interpolated motion command and moves the group axes a relative distance from the current position. The parameter '*Distance[]*' contains the linear path of each axis in the group from the current position to the end position.



Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start the motion at rising edge
	Distance[]	ARRAY[0..N] OF LREAL	• Array [0..N] of relative distance for each dimension. • A maximum number of 16 axes are supported (N<16) • If the axes group has got four axes then the array [0..3] must have at least four relative distance entries
	Velocity	LREAL	Maximum velocity [u/s] for the path. Always positive.
	AccDecTime	UINT	The acceleration time (ms) to reach the maximum velocity. The deceleration time for decelerating from the maximum velocity to zero
	SCurveEnable	BOOL	Acceleration/deceleration curve • FALSE: Trapezoidal curve (T-curve) • TRUE: S-curve
	BufferMode	MC_BUFFER_MODE	See buffer mode definition
	TransitionMode	MC_TRANSITION_MODE	See transition mode definition
	TransitionPara	UINT	0 to 100
VAR_OUTPUT	Done	BOOL	• TRUE: Command has successfully reached the target position and

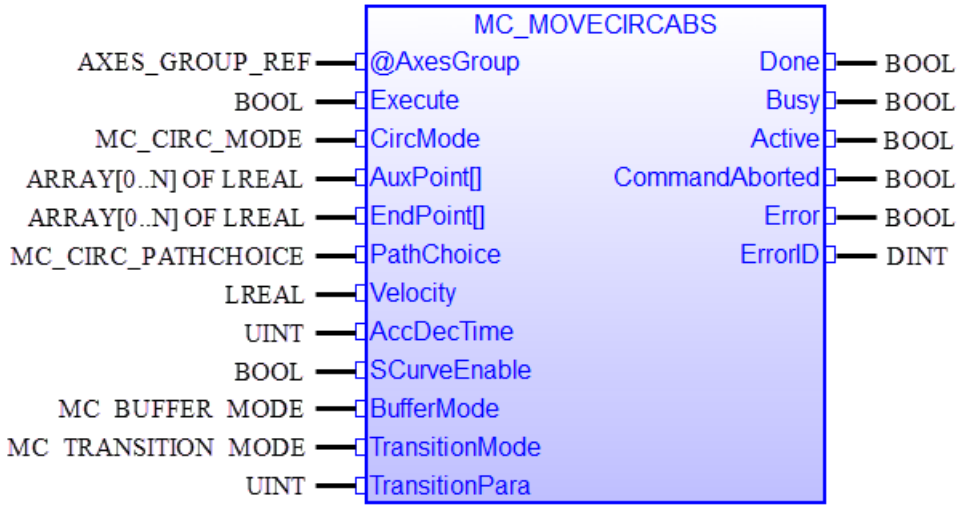
			finished executing
	Busy	BOOL	• TRUE: The MFB is busy either waiting to be executed or executing.
	Active	BOOL	• TRUE: The MFB has control of the axes group and is busy executing.
	CommandAborted	BOOL	• Command has been aborted by another command
	Error	BOOL	• TRUE: An error occurred during the command execution.
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- While this MFB is active the group is in the '*GroupMoving*' state.
- After the distance has been traveled, the group changes to '*GroupStandby*' state if no further motion commands are pending. The '*Done*' output will be set TRUE.
- The input '*Velocity*' will not be reached if the distance to move is shorter than the sum of distance needed to accelerate to the target velocity and deceleration decelerate to zero velocity.
- The '*AccDecTime*' will be ignored if the group is already moving. The '*AccDecTime*' will only be considered if the group is starting from velocity zero (state '*GroupStandby*').
- The supported buffer modes and its relationship with the transition modes and parameters are shown in Table 13. Each circled numbers in the velocity time graph represents a new group command.
- The '*TransitionPara*' sets the blending percent of a group (Table 11). Blending will introduce a smooth transition from one command to another; however, it will deviate from the original path.
- Timing diagram: see PLCopen: "Function blocks for motion control (Part4)" figure 16 page 60/119.
-

6.17 MC_MoveCircularAbsolute

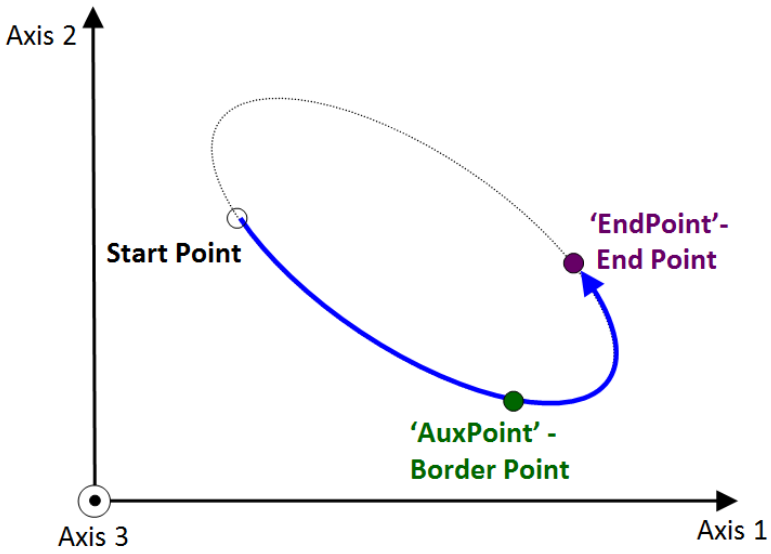
This MFB commands an interpolated circular movement starting from the actual position. The end point as well as the auxiliary point are defined as absolute position in the coordinate system.

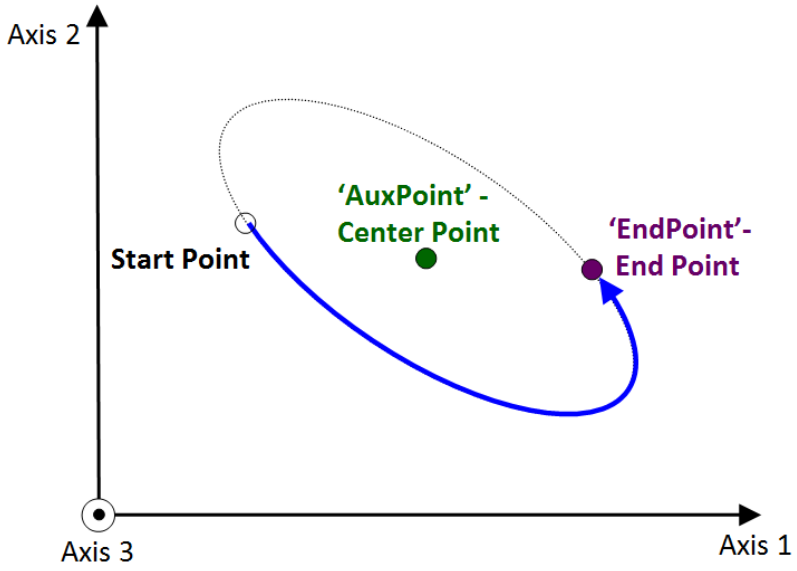


Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start motion at rising edge
	CircMode	MC_CIRC_MODE	This enumerate type has got the following elements: • BORDER, CENTER, RADIUS, ANGLE They determine the purpose (usage) of the '*AuxPoint*' input parameter
	AuxPoint[]	ARRAY[0..N] OF LREAL	• Parameter behavior depends on the 'CircMode' (see Table 14). • Array [0..2] of absolute positions • A maximum number of 3 axes in a group are supported (N<3) • If the axes group has got three axes then the array [0..2] must have at least three absolute positions entries (one for each axis)
	EndPoint[]	ARRAY[0..N] OF LREAL	• Parameter behavior depends on the 'CircMode' (see Table 14). • Array [0..3] of end positions. • A maximum number of 3 axes in a group are supported
	PathChoice	MC_CIRC_PATHCHOICE	Direction of the circle: • Clockwise or Counterclockwise
	Velocity	LREAL	Maximum Velocity [u/s] for the path. Always positive.

	AccDecTime	UINT	• Acceleration and deceleration time. Both parameter are set to the same value. • The acceleration time to reach the maximum velocity. • The deceleration time for decelerating from the maximum velocity to zero
	SCurveEnable	BOOL	Acceleration/deceleration curve • FALSE: Trapezoidal curve (T-curve) • TRUE: S-curve
	BufferMode	MC_BUFFER_MODE	See buffer mode definition
	TransitionMode	MC_TRANSITION_MODE	See transition mode definition
	TransitionPara	UINT	0 to 100
VAR_OUTPUT	Done	BOOL	• TRUE: Command has successfully reached the target position and finished executing
	Busy	BOOL	• TRUE: The MFB is busy either waiting to be executed or executing.
	Active	BOOL	• TRUE: The MFB has control of the axes group and is busy executing.
	CommandAborted	BOOL	• Command has been aborted by another command.
	Error	BOOL	• TRUE: An error occurred during the command execution.
	ErrorID	DINT	Error identification Command-specific error code

No.	MC_CIRC_MODE	Description
0	BORDER	Defines a point on the circle which is crossed on the path from the starting to the end point. • '*AuxPoint*': - The absolute position of the point on the circle (array [0..2]): ▪ For groups with two axes the index 0 and 1 represents the absolute point on a 2D circle border ▪ For group with three axes the index 0, 1 and 2 represents the absolute point on a 3D circle border - A maximum number of 3 axes in a group are supported (N<3) • '*EndPoint*' - Defines the absolute end point (array [0..2]) of the circle chord. ▪ For groups with two axes the index 0 and 1 represents the absolute end point of a 2D circle ▪ For group with three axes the index 0, 1 and 2 represents the absolute end point of a 3D circle - The MFB outputs an error if it fails to calculate a circular path on which both the '*AuxPoint*' and '*EndPoint*' points lies.

		
1	CENTER	Defines the center point and end position of the circle. The center point determines the radius of the arc which connect the start position and end position. • *AuxPoint*': - Absolute position of the circle center point (array [0..2]): ▪ For group with two axes the index 0 and 1 represents the absolute center point of a 2D circle. ▪ For group with three axes then the index 0, 1 and 2 represents the absolute center point on a 3D circle. - A maximum number of 3 axes in a group are supported (N<3) • '*EndPoint*' - Defines the absolute end point (array [0..2]) of the circle chord. ▪ For groups with two axes the index 0 and 1 represents the absolute end point of a 2D circle ▪ For group with three axes the index 0, 1 and 2 represents the absolute end point of a 3D circle - The FB outputs an error if it fails to calculate a circular path with the center point '*AuxPoint*' and end point '*EndPoint*'.

		
2	RADIUS	Not supported
3	ANGLE	Defines the center point, angle and normal vector (for 3D) of the circle sector. The end point is being calculated internally by the MFB. • *AuxPoint*': - Absolute position of the circle center point (array [0..2]): ▪ For group with two axes the index 0 and 1 represents the absolute center point of a 2D circle. ▪ For group with three axes then the index 0, 1 and 2 represents the absolute center point on a 3D circle. - A maximum number of 3 axes in a group are supported (N<3) • '*EndPoint*' - Defines the normal vector and angle (array [0..3]). ▪ Normal vector: The index 0 to 2 defines the normal vector of a 3D circle. The normal vector needs only to be defined if the group has three axes otherwise the values stored in array index 0 to 2 will be ignored. ▪ Angle: The index 3 of the array sets the angle between the end point and the start point (right-hand rule) (unit: degree) - The FB outputs an error if it fails to calculate a circular path with the center point '*AuxPoint*' and angle '*EndPoint[3]*'.

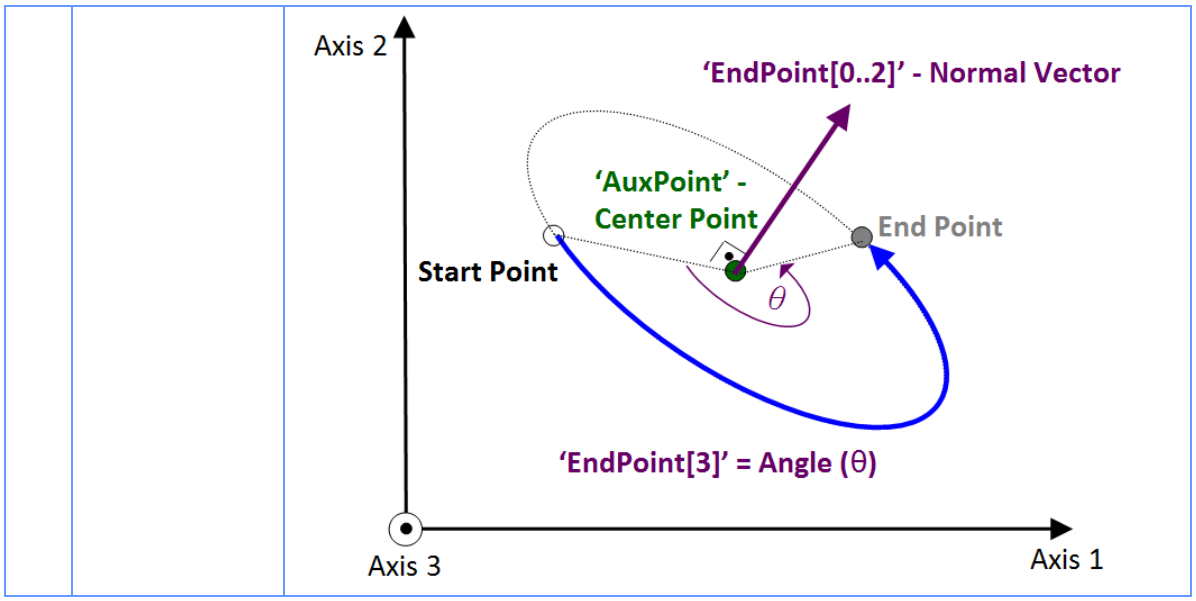
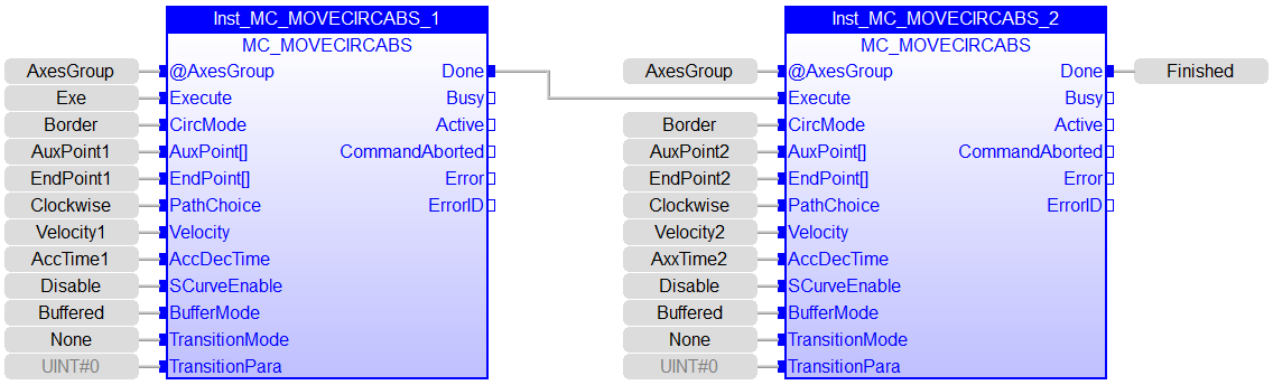


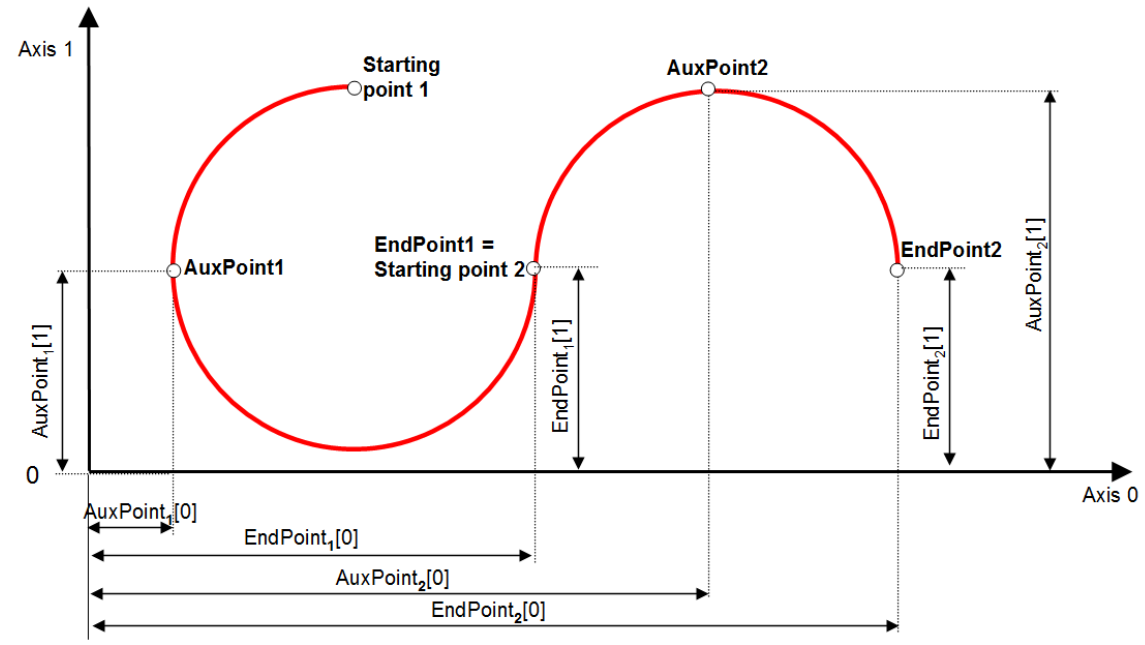
Table 14: Circular Mode for 'MC_MoveCircularAbsolute'

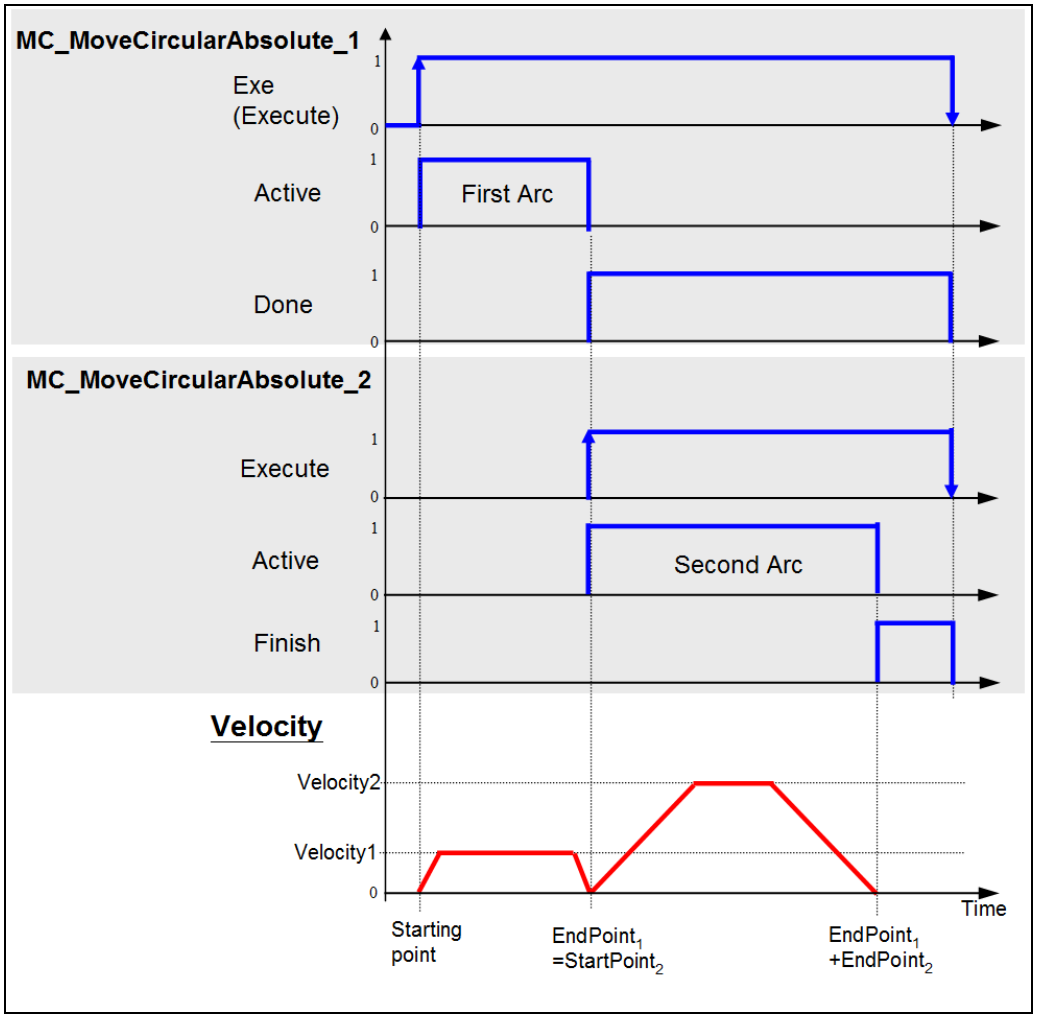
Example:

Example 1:

Sequence example of two '*MC_MoveCircularAbsolute*' MFBs:




Axis 1
Starting point 1
AuxPoint2
AuxPoint1
EndPoint1 = Starting point 2
EndPoint2
AuxPoint1[1]
EndPoint1[1]
EndPoint2[1]
AuxPoint2[1]
0
Axis 0
AuxPoint1[0]
EndPoint1[0]
AuxPoint2[0]
EndPoint2[0]


MC_MoveCircularAbsolute_1
Exe (Execute)
Active
First Arc
Done
MC_MoveCircularAbsolute_2
Execute
Active
Second Arc
Finish
Velocity
Velocity2
Velocity1
Time
Starting point
EndPoint1 =StartPoint2
EndPoint1 +EndPoint2

Example 2:

Path movements with '*MC_MoveLinear*' and '*MC_MoveCircular*':

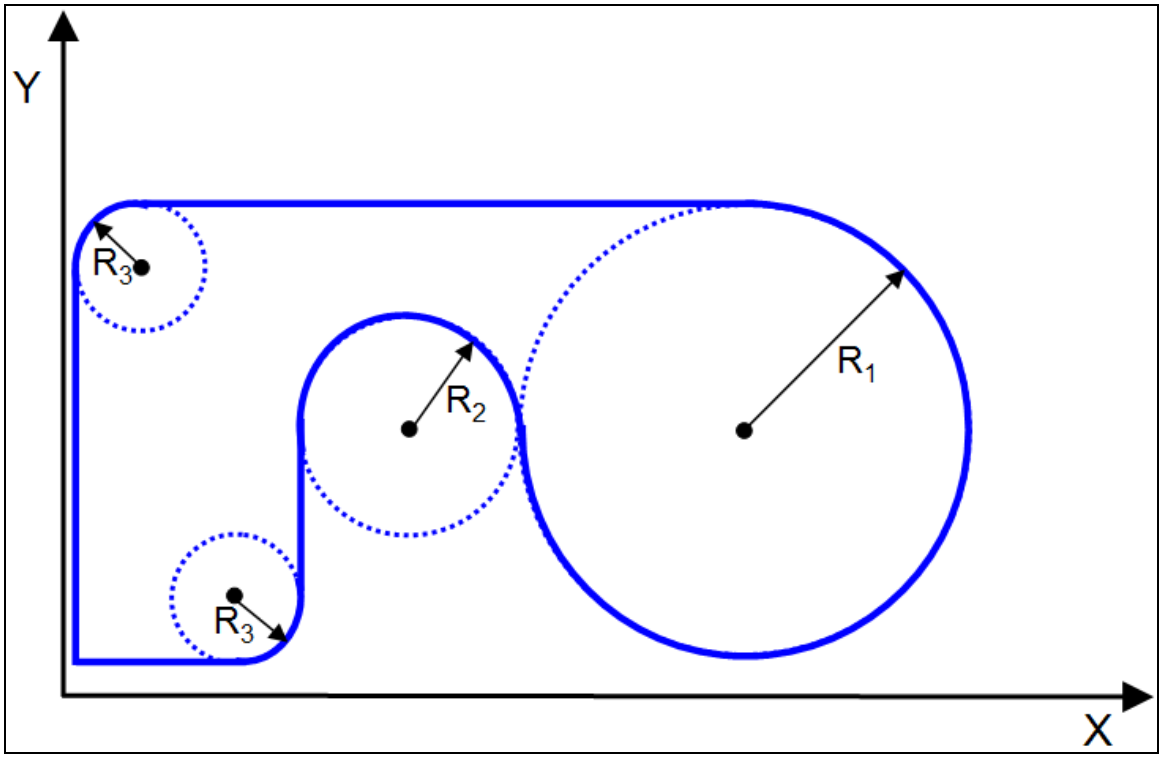
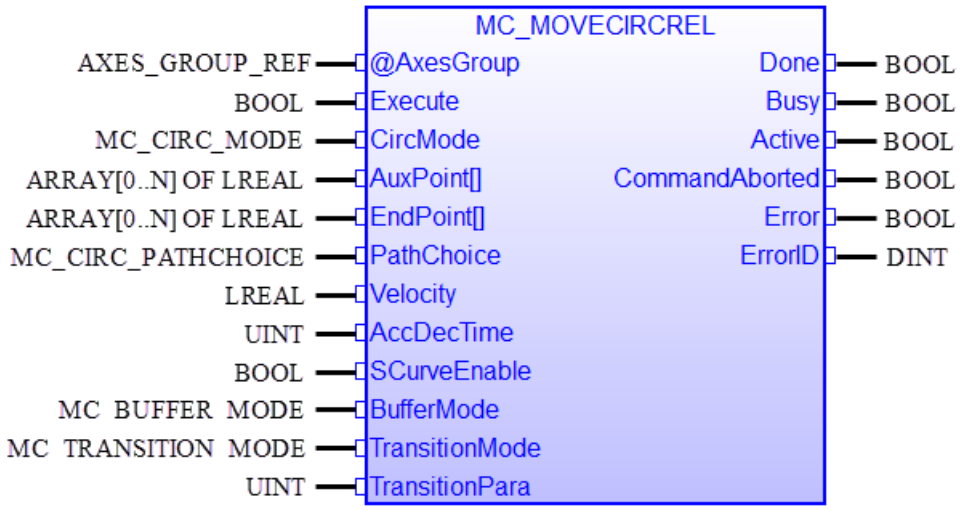


Figure 17: Path movements with single commands (MC_MoveLinear, MC_MoveCircular)

6.18 MC_MoveCircularRelative

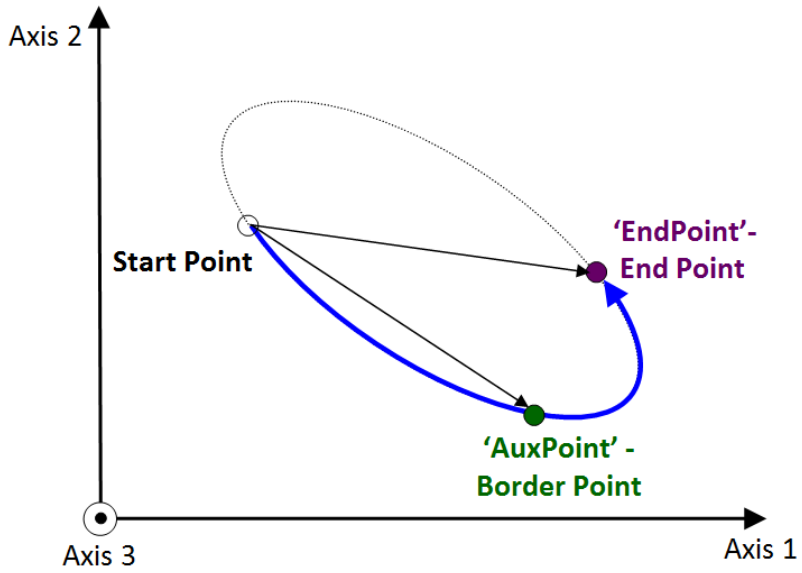
This MFB executes a circular movement starting from the actual position. In order for the MFB to calculate the circular path the auxiliary position has to be specified in addition to the end position. The end point as well as the auxiliary point are defined as relative position to the starting point in the Cartesian coordinate system.

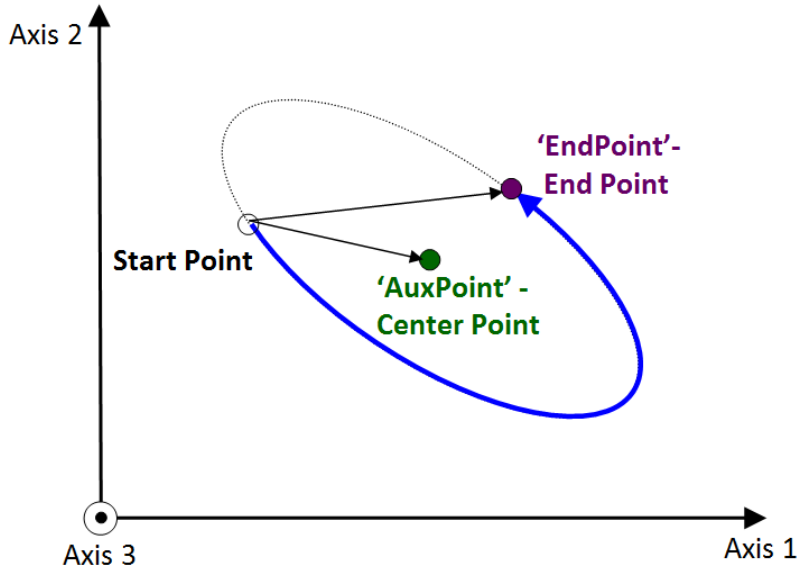


Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start motion at rising edge
	CircMode	MC_CIRC_MODE	This enumerate type has got the following elements: • BORDER, CENTER, RADIUS, ANGLE They determine the purpose (usage) of the 'AuxPoint' input parameter
	AuxPoint[]	ARRAY[0..N] OF LREAL	• Parameter behavior depends on the 'CircMode' (see Table 15). • Array [0..2] of relative positions • A maximum number of 3 axes in a group are supported (N<3) • If the axes group has got three axes then the array [0..2] must have three relative positions (one for each axis)
	EndPoint[]	ARRAY[0..N] OF LREAL	• Parameter behavior depends on the 'CircMode' (see Table 15). • Array [0..3] of relative end positions. • A maximum number of 3 axes in a group are supported
	PathChoice	MC_CIRC_PATHCHOICE	Direction of the circle: • Clockwise or Counterclockwise

	Velocity	LREAL	Maximum Velocity [u/s] for the path. Always positive.
	AccDecTime	UINT	• Acceleration and deceleration time. Both parameter are set to the same value. • The acceleration time to reach the maximum velocity. • The deceleration time for decelerating from the maximum velocity to zero
	SCurveEnable	BOOL	Acceleration/deceleration curve • FALSE: Trapezoidal curve (T-curve) • TRUE: S-curve
	BufferMode	MC_BUFFER_MODE	See buffer mode definition
	TransitionMode	MC_TRANSITION_MODE	See transition mode definition
	TransitionPara	UINT	0 to 100
VAR_OUTPUT	Done	BOOL	• TRUE: Command has successfully reached the target position and finished executing
	Busy	BOOL	• TRUE: The Function Block is busy either waiting to be executed or executing.
	Active	BOOL	• TRUE: The Function Block has control of the axes group and is busy executing.
	CommandAborted	BOOL	• Command has been aborted by another command
	Error	BOOL	• TRUE: An error occurred during the command execution.
	ErrorID	DINT	Error identification Command-specific error code

No.	MC_CIRC_MODE	Description
0	BORDER	Defines an intermediate point (border point) on the circle which is crossed on the path from the starting to the end point. The FB internally calculates the circular path by using the starting, intermediate and end points. Only circular paths less than 360° are valid. Both intermediate ('*AuxPoint*') and end ('*EndPoint*') points are defined relative to the starting point. • '*AuxPoint*': - Specifies the relative position of the intermediate point (array [0..2]) on the circular path, via which the end point is to be approached. It is specified relative to the starting point. ▪ For groups with two axes the index 0 and 1 represents the relative position of the point on a 2D circle path. Each index represents the relative position of one axis. ▪ For group with three axes the index 0, 1 and 2 represents the relative position of the point on a 3D circle path - A maximum number of 3 axes in a group are supported (N<3)

		- The intermediate position has to has to lie exactly on the circular path and cannot be approximated. • '*EndPoint*' - Defines the relative position of the end point (array [0..2]) of the circle chord. The position is defined relative to the starting point. ▪ For groups with two axes the index 0 and 1 represents the relative end point position of a 2D circle ▪ For group with three axes the index 0, 1 and 2 represents the relative end point position of a 3D circle - The FB outputs an error if it fails to calculate a circular path on which both the 'AuxPoint' and 'EndPoint' points lies. 
1	CENTER	Defines the center point and end position of the circle. The center point determines the radius of the arc which connect the start position and end position. Both center ('*AuxPoint*') and end ('*EndPoint*') points are defined relative to the starting point. • *AuxPoint*': - Relative position of the circle center point (array [0..2]). It is specified relative to the starting point. ▪ For group with two axes the index 0 and 1 represents the relative position of the center point of a 2D circle. Each index represents the relative position of one axis. ▪ For group with three axes then the index 0, 1 and 2 represents the relative position of the center point on a 3D circle. - A maximum number of 3 axes in a group are supported (N<3) • '*EndPoint*' - Defines the absolute end point (array [0..2]) of the circle chord. ▪ For groups with two axes the index 0 and 1 represents the relative end point position of a 2D circle ▪ For group with three axes the index 0, 1 and 2 represents the relative end point position of a 3D circle - The FB outputs an error if it fails to calculate a circular path with the

		relative center point 'AuxPoint' and end point 'EndPoint'. 
2	RADIUS	Not supported
3	ANGLE	Defines the center point, angle and normal vector (for 3D) of the circle sector. The end point is being calculated internally by the FB. The center point ('*AuxPoint*') is defined relative to the starting point. • *AuxPoint*': - Relative position of the circle center point (array [0..2]). It is specified relative to the starting point. ▪ For group with two axes the index 0 and 1 represents the relative center point of a 2D circle. ▪ For group with three axes then the index 0, 1 and 2 represents the relative center point of a 3D circle. - A maximum number of 3 axes in a group are supported (N<3) • '*EndPoint*' - Defines the normal vector and angle (array [0..3]). ▪ Normal vector: The index 0 to 2 defines the normal vector of a 3D circle. The normal vector needs only to be defined if the group has three axes otherwise the values stored in array index 0 to 2 will be ignored. ▪ Angle: The index 3 of the array sets the angle between the end point and the start point (right-hand rule) (unit: degree) - The FB outputs an error if it fails to calculate a circular path with the center point 'AuxPoint' and angle 'EndPoint[3]'.

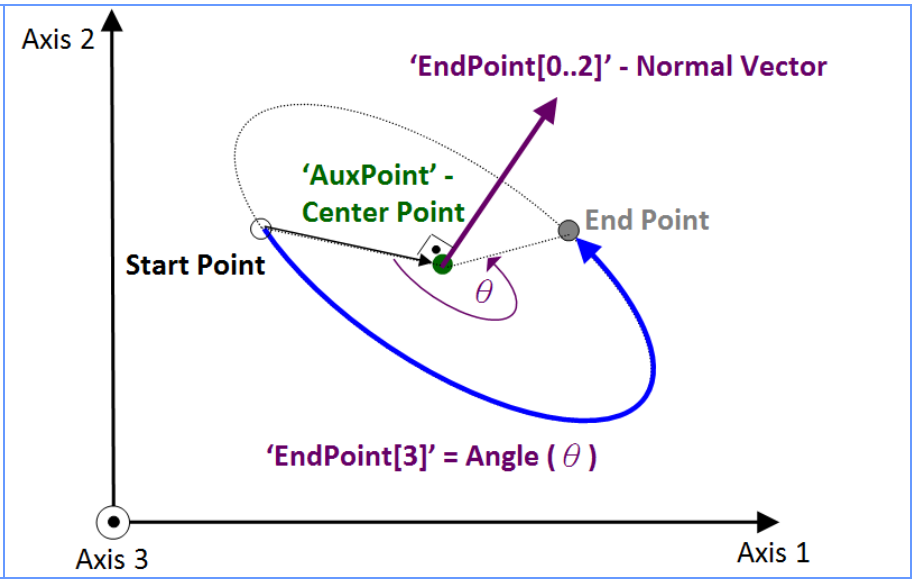
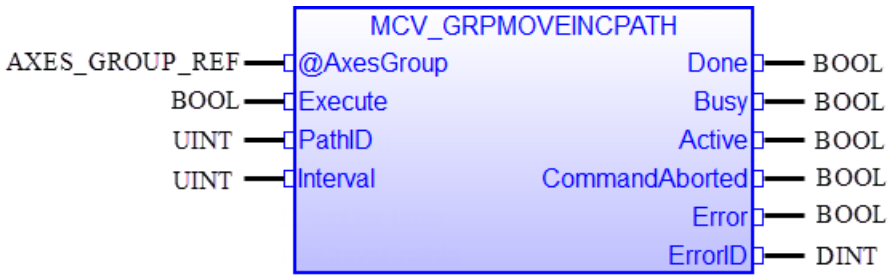
		

Table 15: Circular Mode for 'MC_MoveCircularAbsolute'

6.19 MCV_GroupMoveIncPath

This MFB commands a group of axis to move according to the incremental path specified in the '*PathID*'. The 'PathID' refers to a csv file inside the ECAT-M801/e-M9000 which stores a sequence of absolute distances for each group axis. In each interval the MFB reads the next absolute position from the csv file and executes an incremental interpolated linear movement from the current position to the next absolute position of the list. All axes synchronously start moving at the beginning of the interval and reach the absolute position together at the end of the interval. The input parameter 'Interval' determines the interval time and thereby the speed at which the axes moves to the next absolute position. A maximum of 16 axes in the group are supported.



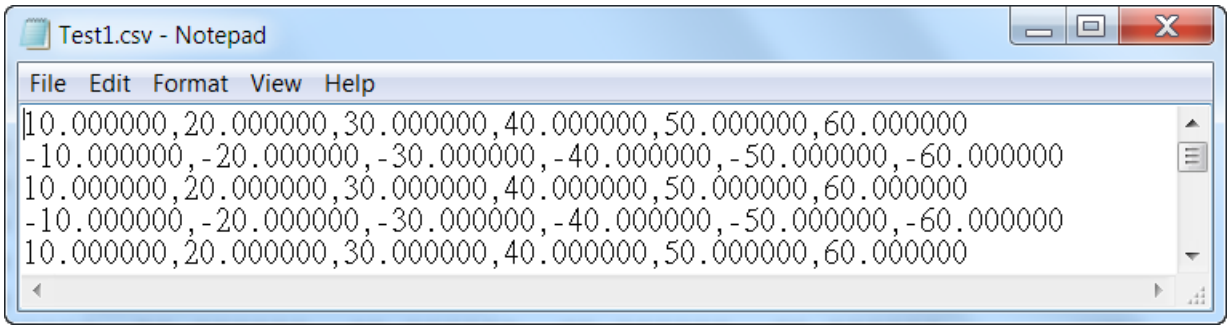
Input/Output	Name	Type	Comments
VAR_IN_OUT	AxesGroup	AXIS_GROUP_REF	Reference to a group of axes.
VAR_INPUT	Execute	BOOL	Start motion at rising edge
	PathID	UINT	• The handle number of the csv file stored in the ECAT-M801/e-M901 device. • This handle is being set via the input parameter '*PathID*' of the FB '*MCV_IncPathLoadFileCsv*'
	Interval	UINT	• The interval within a incremental position command in the csv file will be executed.
VAR_OUTPUT	Done	BOOL	• All the target position commands stored inside the csv file has been executed
	Busy	BOOL	• TRUE: The FB is busy either waiting to be executed or executing.
	Active	BOOL	• TRUE: The FB has control of the axes group and is busy executing.
	CommandAborted	BOOL	• Command has been aborted by another command
	Error	BOOL	• TRUE: An error occurred during the command execution.
	ErrorID	DINT	• Error identification • Command-specific error code

Remarks:

- The '*Interval*' set the speed at which the list of incremental absolute commands in the csv file will be executed. At the end of each interval all axes reaches the incremental absolute position set by the csv file simultaneously.
  - The interval refers to the EtherCAT master cycle. That means if the interval is being set to 1 then in every master cycle a new incremental command will be read from the csv file and be executed.
  - For example:
    - 1: read position array data for motion every cycle time (default setting)
    - 2: read position array data for motion every two cycle time.

6.19.1 Create Incremental CSV File

The csv file contains a list of absolute positions for each axis in the group. Each line represents the absolute target coordinates of the group for a cycle. The motion engine will start from the top line of the csv file and in each cycle sequentially read the values of a line to get the commanded absolute position for the group. The first value in a line always represents the absolute target position for the first axis in the group; the second value in a line the position of the second axis, etc.. The absolute positions of a group are separated by a comma except the value in the line which ends with a new line character.



The csv file can be created manually by the user or programmatically within the PLC program by calling the function '*MCV_CsvFile_Open*', '*MCV_ CsvFile_AddPoint*', '*MCV_ CsvFile_Close*' (Figure 18).

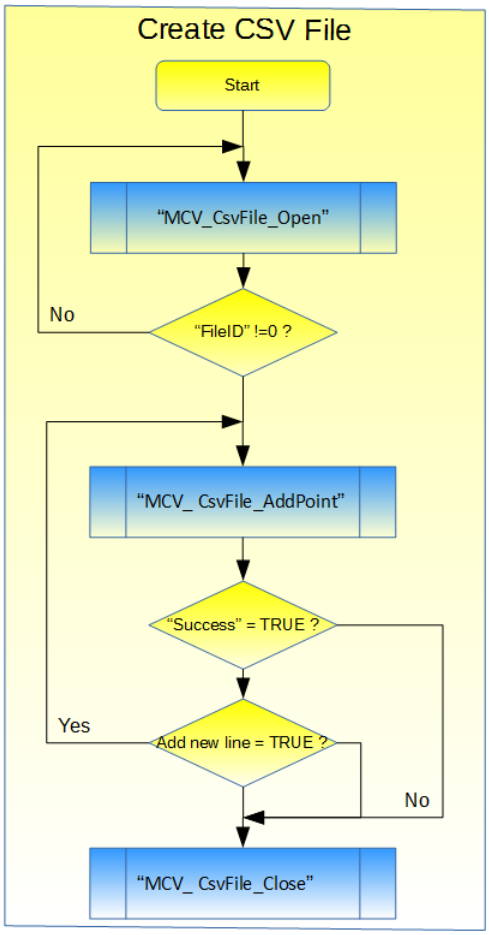


Figure 18: Creating and filling a CSV file for incremental motion execution

6.19.1.1 MCV_CsvFile_Open

Call this function to open the csv file before writing a sequence of absolute position data to it. If the file does not exist a new file with the name will be created and added to the directory '*./Motion/CSV*'.

Input/Output	Name	Type	Comments
VAR_INPUT	File	STRING	• File name without directory. • Do not add any directory name to the file name. • The file will be save to the directory: ./Motion/CSV
	Mode	MC_CSV_FILE_MOD	Sets the file saving mode

VAR_OUTPUT	FileID	DINT	• 0: Open the file failed • not zero: File identifier number

Remarks:

- The enumerate '*MCV_CSV_FILE_MODE*' is defined as follows:
  - '*fRead*' :
    - Opens for reading. If the file doesn't exist or cannot be found, the function call fails. Do not use this mode for the writing data to the file.
  - '*fClearWrite*':
    - Opens an empty file for writing. If the given file exists, its contents are destroyed.
  - '*fWriteAppend*':
    - Opens for writing at the end of the file (appending) without removing the end-of-file (EOF) marker before new data is written to the file. Creates the file if it doesn't exist.
  - ' fReadWrite':
    - Opens for both reading and writing. The file must exist.
  - '*fClearReadWrite*':
    - Opens an empty file for both reading and writing. If the file exists, its contents are destroyed.

6.19.1.2 MCV_CsvFile_AddPoint

This function adds a new group target position to the csv file. This function has to be called for each new position to add to the file. Each element of the input array '*Point[]*' has to contain the value of the next absolute target position for each axis in the group: the value of the first array element [0] sets the absolute target position for the first axis in the group, the second element [1] the second axis, etc. It is important to make sure that the array contains exactly the same elements as there are axes in the group. A maximum of sixteen axes in a group are supported.

Input/Output	Name	Type	Comments
VAR_INPUT	FileID	DINT	File identifier number returned by the '*MCV_CsvFile_Open*' function.
	Point[]	ARRAY[0..N] OF LREAL	• Array of absolute positions. • Each positions represents the next absolute target position of the axis in the group.

			• The array should not have more or less elements than the group has axes • Each group can contain a maximum of 16 axis. Therefore array should not be longer than 16 elements
VAR_OUTPUT	Success	BOOL	• TRUE: Array data has been save to the file successfully. • FALSE: Failed to save the array data

6.19.1.3 MCV_CsvFile_Close

This function closes the csv file after all the axes positions have been written to it. It is required to close the file before it can be accessed by other function blocks.

Input/Output	Name	Type	Comments
VAR_INPUT	FileID	DINT	File identifier number returned by the '*MCV_CsvFile_Open*' function.
VAR_OUTPUT	Success	BOOL	• TRUE: File has been closed successfully. • FALSE: Failed to close the file

Example:

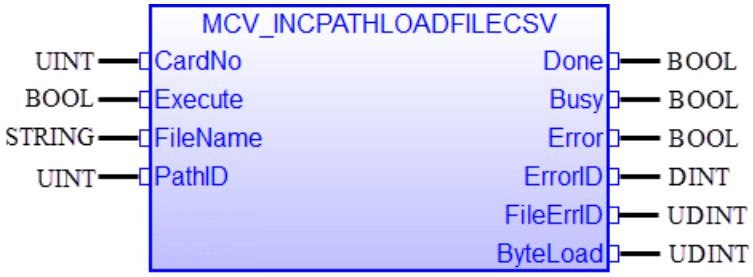
```
VAR
   flgOpen : BOOL := FALSE ;
   FileMode : MCV_CSV_FILE_MODE := fClearWrite ;
   FileID : DINT ;
   flgRet : BOOL ;
   dfPoint : ARRAY [0 .. 5] OF LREAL ;
   dfVal : LREAL := LREAL#0 ;
END_VAR
```

```
//============================================================
// Create a CSV file and fill it with incremental absolute
// position data
//============================================================
IF flgOpen = TRUE then
```

```
    //-------------------------------------------------
    //Create or open a CSV file in the directory ".\Motion\CSV\":
    //-------------------------------------------------
    FileID := MCV_CSVFILE_OPEN('Test1.CSV'(*STRING*), FileMode );

    //-------------------------------------------------
    //Write the following data to the file
    //-------------------------------------------------
    for i:=1 to 100 do
        if dfVal>LREAL#0.0 then
            dfVal := LREAL#-10.0;
        else
           dfVal := LREAL#10.0;
        end_if;

        for j:=0 to 5 do
           dfPoint[j] := dfVal * ANY_TO_LREAL(j+1);
        end_for;

        Ret := MCV_CSVFILE_ADDPOINT( FileID(*DINT*), dfPoint(*LREAL*) );
    end_for;

    //-------------------------------------------------
    //Close the file:
    //-------------------------------------------------
    flgRet := MCV_CSVFILE_CLOSE( FileID(*DINT*) );
    flgOpen := FALSE;
end_if;
```

6.19.2 Download CSV File

The MFB '*MCV_IncPathLoadFileCsv*' downloads the csv file to the ECAT-M801/e-M901 and stores it in a non-volatile memory. Depending on the csv file size it may take up to several task cycles before all the data are downloaded.



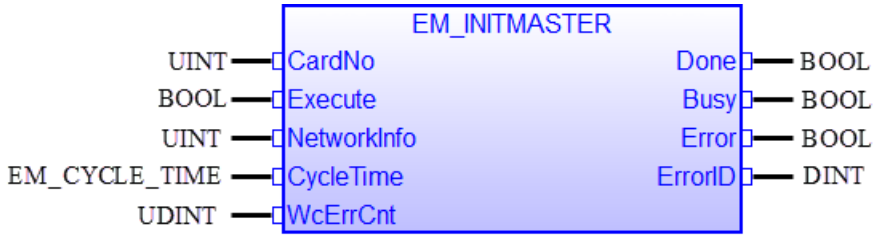
Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901.
	Execute	BOOL	Start the csv file download at rising edge
	FileName	STRING	The name of the csv file in the '*./Motion/CSV*' directory.
	PathID	UINT	• This number represents the handle to the downloaded file. Any non zero UNIT value can be assigned to the csv file inside the ECAT-M801/e-M901. • The command '*MC_GroupMoveIncPath*' needs this handle to determine which csv file to use for motion execution
VAR_OUTPUT	Done	BOOL	• TRUE: The file has been downloaded successfully
	Busy	BOOL	• TRUE: The FB block is still busy downloading the file
	Error	BOOL	• TRUE: Error occurred during the download process
	ErrorID	DINT	Error identification Command-specific error code
	FileErrID	UDINT	• The file error number: Special file errors returned by the OS. The ID are defined by the OS or the 'stdio.h' library
	ByteLoad	UDINT	The number of bytes downloaded. This value should be the same as the file size in bytes

7 ECAT-M801/e-M901 Related Function Blocks

The FBs introduced in this chapter are device specific. Their main purpose is to initialize the communication between the host (PC or PAC) and the ECAT-M801/e-M901 cards, initialize the EtherCAT network and set the EtherCAT master into operational mode. In addition slave to axis assignment has to be completed before the PLCopen defined FB can take control and access any axis.

7.1 EM_InitMaster

This FB establish the communication between the host (PC or PAC) and the ECAT-M801/e-M901 card(s); checks whether the EtherCAT network setup is correct by making sure that the slaves listed in the configuration file (created by the EtherCAT utility) are online in the EtherCAT network; and sets the EtherCAT master into operational mode (OP).



Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	Execute	BOOL	Set the master into OP mode at rising edge
	NetworkInfo	UINT	Network information file number (Configured by the EtherCAT utility)
	CycleTime	EM_CYCLE_TIME	Cycle time number (Table 16)
	WcErrCnt	UDINT	Counts of working counter errors
VAR_OUTPUT	Done	BOOL	• TRUE: Master is in OP mode • FALSE: Failed to initialize the EtherCAT network. Master is not in OP mode.
	Busy	BOOL	• TRUE: The FB block is still busy downloading

Input/Output	Name	Type	Comments
			the file
	Error	BOOL	• TRUE: Error occurred during the download process
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- At least one network information file must be pre-loaded into the M801/e-M901 card before calling the '*EM_InitMaster*' FB. It is necessary to use the ECAT Utility to create and download the configuration file. The purpose of the configuration file is to check whether the real EtherCAT network slave setup is identical with the configured network and map the process data(PDO).
- The execution of the '*EM_InitMaster*' FB takes several task cycles to finish, as it takes some time to initialize and configure each slave in the network and to switch the EtherCAT master into OP mode.
- After the master is in OP mode the other FB introduce in this chapter can be called.
- The enumerate EM_CYCLE_TIME is defined as follows:

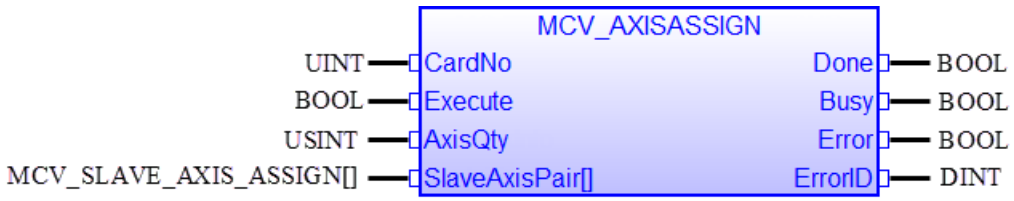
No.	EM_CYCLE_TIME	Description
0	CYCLE_TIME_1MS	1 millisecond cycle time
1	CYCLE_TIME_2MS	2 millisecond cycle time
2	CYCLE_TIME_3MS	3 millisecond cycle time
3	CYCLE_TIME_4MS	4 millisecond cycle time
4	CYCLE_TIME_5MS	5 millisecond cycle time
5	CYCLE_TIME_6MS	6 millisecond cycle time
6	CYCLE_TIME_7MS	7 millisecond cycle time
7	CYCLE_TIME_8MS	8 millisecond cycle time
8	CYCLE_TIME_9MS	9 millisecond cycle time
9	CYCLE_TIME_10MS	10 millisecond cycle time
10	CYCLE_TIME_11MS	11 millisecond cycle time
11	CYCLE_TIME_12MS	12 millisecond cycle time
12	CYCLE_TIME_13MS	13 millisecond cycle time
13	CYCLE_TIME_14MS	14 millisecond cycle time
14	CYCLE_TIME_15MS	15 millisecond cycle time

Table 16: EtherCAT cycle time defintions

- In OP mode the master accesses the all the slave in each cycle. If one of the slave is physically disconnected from the EtherCAT network while in OP mode the master will try to access it in each cycle. Once the slave is connected again to the network the master will automatically initialize it again and go into normal data exchange mode.

7.2 MCV_AxisAssign

This MFB assigns an axis number to the stepper/servo drive slave in the EtherCAT network so that it can be accessed by the PLCopen defined MFBs. This MFB basically does axis - slave mapping. Normally a EtherCAT slave does only control one axis, but some stepper/servo slaves on the market control more than one axis (e.g. ECAT-2094S from ICPDAS). In this case the sub-axis number '*SubAxisNo*' of the '*MCV_SLAVE_AXIS_ASSIGN*' structure has to be set as well.



Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	Execute	BOOL	At rising edge the specified servo/step drive slaves are mapped to an axis number
	AxisQty	USINT	The number of axes to map
	SlaveAxisPair[]	MCV_SLAVE_AXIS_ASSIGN	An array of an slave- axis pair assignment.
VAR_OUTPUT	Done	BOOL	• TRUE: Axis mapping was successful • FALSE: Axis assignment failed
	Busy	BOOL	• TRUE: The MFB block is still busy with the axis assignment
	Error	BOOL	• TRUE: Error occurred during the mapping process
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The axis assignment may take up to several cycle times as the EtherCAT master will automatically reset the axis and try to clear all axis error once the mapping procedure completed.
- The axes will be removed from all groups once the '*MCV_AXISASSIGN.Execute*' is being triggered. All groups will be empty after an successful execution of '*MCV_AXISASSIGN*' and the axes have to be added to the group again.
- Virtual axis: The MFB support virtual axis assignment. In this case the slave number has to be set to 65535.
- Definition of the '*MCV_SLAVE_AXIS_ASSIGN*' structure:

```
STRUCT
```

```
    SlaveNo : UINT;
    AxisNo  : UINT;
    SubAxisNo: UINT;
END_STRUCT
```

- The 'SubAxisNo' parameter has to be set when the slave controls more than one axis otherwise it has to be set to zero.
- Example 1:

  Slave 4, 5 and 6 each controls one servo drive. In addition two virtual axes should be assigned

No.	EM_CYCLE_TIME	Description

SlaveAxisPair[0]	`SlaveNo = 4;`
	`AxisNo = 0;`
	`SubAxisNo = 0;`
		Map the slave 4 to the axis number 0

SlaveAxisPair[1]	`SlaveNo = 5;`
	`AxisNo = 1;`
	`SubAxisNo = 0;`
		Map the slave 5 to the axis number 1

SlaveAxisPair[2]	`SlaveNo = 6;`
	`AxisNo = 2;`
	`SubAxisNo = 0;`
		Map the slave 6 to the axis number 2

SlaveAxisPair[3]	`SlaveNo = 65535;`
	`AxisNo = 3;`
	`SubAxisNo = 0;`
		Map the virtual slave to the axis number 3

SlaveAxisPair[4]	`SlaveNo = 65535;`
	`AxisNo = 4;`
	`SubAxisNo = 0;`
		Map the virtual slave to the axis number 4

  Remarks:
  - As the slaves only control one axis each it is necessary to set the 'SubAxisNo' parameter to zero.
  - In this example only three axes are mapped therefore set the axis quantity to three ('AxisQty' = 3).

- Example 2:

  Slave 3 control 4 axes; slave 4 controls one servo drive and slave 5 controls 2 axes.

No.	EM_CYCLE_TIME	Description

SlaveAxisPair[0]	`SlaveNo = 3;`
	`AxisNo = 0;`
	`SubAxisNo = 0;`
		Map the first axis (`SubAxisNo=0`) of slave 3 to the axis number 0

SlaveAxisPair[1]	`SlaveNo = 3;`
	`AxisNo = 1;`
	`SubAxisNo = 1;`
		Map the second axis (`SubAxisNo=1`) of slave 3 to the axis number 1

SlaveAxisPair[2]	`SlaveNo = 3;`
	`AxisNo = 2;`
	`SubAxisNo = 2;`
		Map the third axis (`SubAxisNo=2`) of slave 3 to the axis number 2

SlaveAxisPair[3]	`SlaveNo = 3;`
	`AxisNo = 3;`
	`SubAxisNo = 3;`
		Map the fourth axis (`SubAxisNo=3`) of

		slave 3 to the axis number 3
SlaveAxisPair[4]	SlaveNo = 4; AxisNo = 4; SubAxisNo = 0;	Map the slave 4 to the axis number 4
SlaveAxisPair[5]	SlaveNo = 5; AxisNo = 5; SubAxisNo = 0;	Map the first axis (SubAxisNo=0) of slave 5 to the axis number 5
SlaveAxisPair[6]	SlaveNo = 5; AxisNo = 6; SubAxisNo = 1;	Map the second axis (SubAxisNo=1) of slave 5 to the axis number 6

Remarks:

- In this example only seven axes are mapped therefore set 'AxisQty'= 7.

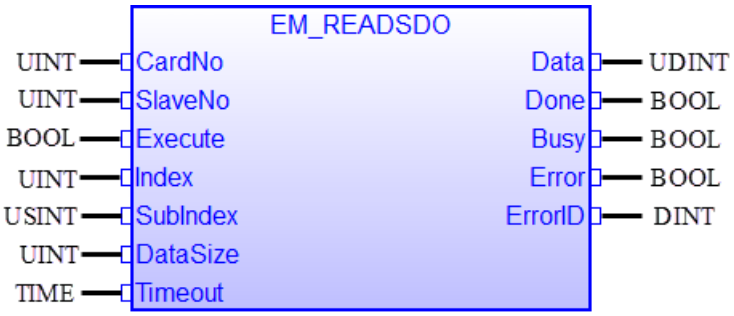
Example:

```
VAR
   SlaveAxisPair : ARRAY [0 .. 9] OF lib:MCV_SLAVE_AXIS_ASSIGN ;
   flgAxisAssign : BOOL := FALSE ;
   diAxisQty : DINT ;
   Inst_MCV_AXISASSIGN : MCV_AxisAssign ;
   CardNo : UINT := UINT#1 ;
END_VAR
```

```
//============================================================
//Assign each axis number to a servo driver slave:
//============================================================
IF (flgAxisAssign = TRUE) THEN
    diAxisQty := 6;
    FOR i:=0 TO diAxisQty-1 DO
        SlaveAxisPair[i].SlaveNo := any_to_uint(i);
        SlaveAxisPair[i].AxisNo := any_to_uint(i);
        SlaveAxisPair[i].SubAxisNo  := 0;
    END_FOR;
END_IF;

Inst_MCV_AXISASSIGN( CardNo, flgAxisAssign, any_to_usint(diAxisQty),
SlaveAxisPair );
```

7.3 EM_ReadSdo

This FB reads Service Data Object (SDO) data from an EtherCAT slave. This requires the slave to have a mailbox and support the CoE protocol. The '*Index*' and '*Subindex*' input parameters select the object to be read. SDO communication is acyclic, has a lower priority as the PDO communication and therefore has a longer response time. A maximum of four bytes can be read per request.



Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	SlaveNo	UINT	EtherCAT slave address
	Execute	BOOL	Read the SDO at rising edge
	Index	UINT	• Index (decimal value) of the object to be read. • Consult the slave device manual for available Indexes with Read access.
	SubIndex	USINT	• Sub-index (decimal value) of the object to be read. • Consult the slave device manual for available Subindexes with Read access.
	DataSize	UINT	• The number of byte to read. Maximum supported data size is 4 bytes
	Timeout	TIME	Maximum time [milliseconds] allowed for the execution of the FB .
VAR_OUTPUT	Data	UDINT	The read SDO data
	Done	BOOL	• TRUE: SDO data has been read
	Busy	BOOL	• TRUE: Still waiting for the slave respond frame
	Error	BOOL	• TRUE: Error occurred during the SDO read procedure.
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- Can only be called after a successful '*EM_InitMaster*' execution.

7.4 EM_WriteSdo

This FB is used for writing a specific object in the object dictionary of a slave. This requires the slave to have a mailbox and support the CoE protocol. The '*Index*' and '*Subindex*' input parameters select the object to be written to. SDO communication is acyclic, has a lower priority as the PDO communication and therefore has a longer response time. SDO data types of up to a length of four bytes can be written to.

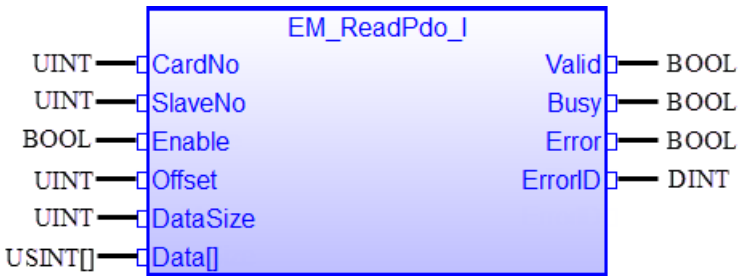
Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	SlaveNo	UINT	EtherCAT slave address
	Execute	BOOL	Write to the SDO at rising edge
	Index	UINT	• Index (decimal value) of the object to be written. • Consult the slave device manual for available Indexes with Write access.
	SubIndex	USINT	• Sub-index (decimal value) of the object to be written. • Consult the slave device manual for available Subindexes with Write access.
	DataSize	UINT	• The number of bytes to write. Maximum supported data size is 4 bytes
	Data	UDINT	Data to be written to the SDO
	Timeout	TIME	Maximum time [milliseconds] allowed for the execution of the FB .
VAR_OUTPUT	Done	BOOL	• TRUE: SDO data has been written
	Busy	BOOL	• TRUE: Still waiting for the slave to respond
	Error	BOOL	• TRUE: Error occurred during the SDO write operation.
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- Can only called after a successful '*EM_InitMaster*' execution.

7.5 EM_ReadPdo

This FB reads Process Data Object (PDO) data received from an EtherCAT slave. This requires the EtherCAT master to be in DC mode.



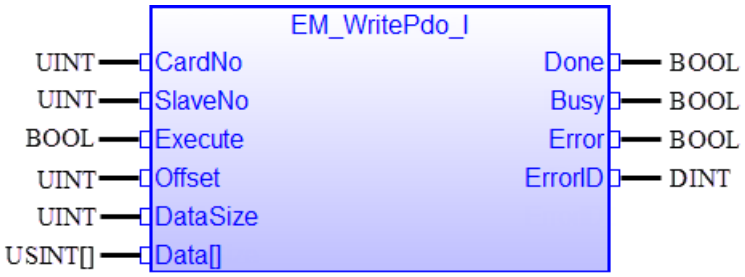
Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	SlaveNo	UINT	EtherCAT slave address
	Enable	BOOL	• TRUE: Read the PDO in each cycle
	Offset	UINT	• Slave PDO offset start position to read from
	DataSize	UINT	• The number of bytes to read. Maximum supported data size is 512 bytes
VAR_OUTPUT	Data[]	USINT[]	Data array read from the PDO received by the master from the slave. ATTENTION: Although the parameter is shown on the input side of the function block it acts as an output.
	Valid	BOOL	• TRUE: The output data "Data[]" is valid
	Busy	BOOL	• TRUE: Still reading the PDO frame received from the slave
	Error	BOOL	• TRUE: Error occurred during the PDO read procedure.
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- Can only be called after a successful '*EM_InitMaster*' execution.

7.6 EM_WritePdo

This FB directly writes process data to the frame send by the EtherCAT master to the specified slave. PDO data communication is cycle and has the highest priority.



Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	SlaveNo	UINT	EtherCAT slave address
	Execute	BOOL	Write to the PDO at rising edge
	Offset	UINT	• The offset position of the process data • The process data length send by the master to the slave is set during the configuration phase and is fixed when in OP mode The offset position allows the user to enter/modify data at any position of the process data frame assigned to the specific slave. • The process data object structure (e.g. the number of entries with its offset and size) is set by the network configuration file.
	DataSize	USINT	• The number of bytes to write. • Maximum supported data size is 512 bytes
	Data[]	USINT[]	• A USINT array of data to write to the PDO
VAR_OUTPUT	Done	BOOL	• TRUE: PDO data has been written
	Busy	BOOL	• TRUE: Still writing to the PDO
	Error	BOOL	• TRUE: Error occurred during the PDO write operation.
	ErrorID	DINT	Error identification Command-specific error code

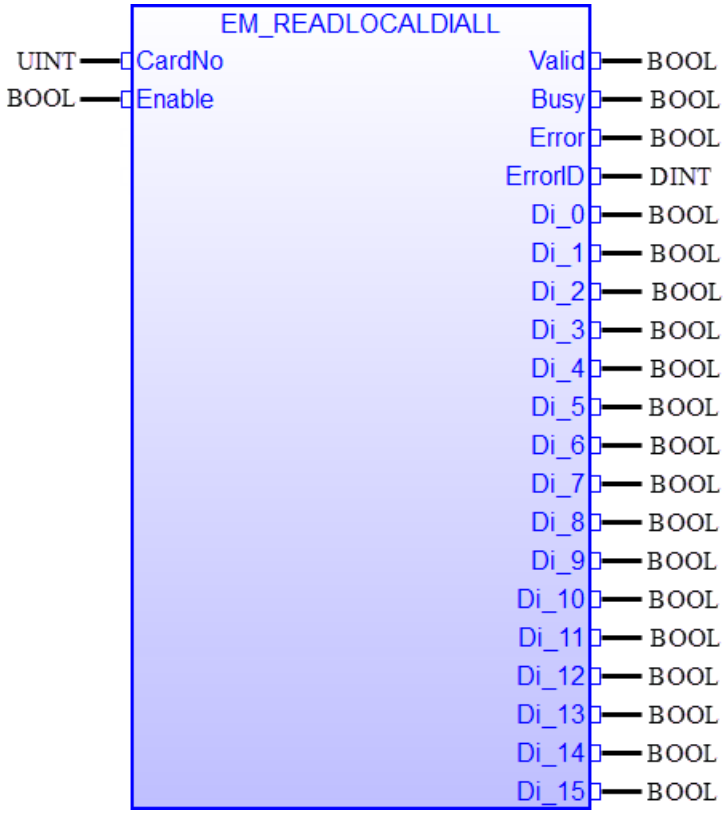
Remarks:

- Can only be called after a successful '*EM_InitMaster*' execution.

7.7 EM_ReadLocalDiAll

This FB reads the local digital input channel of the ECAT-M801/ e-M901

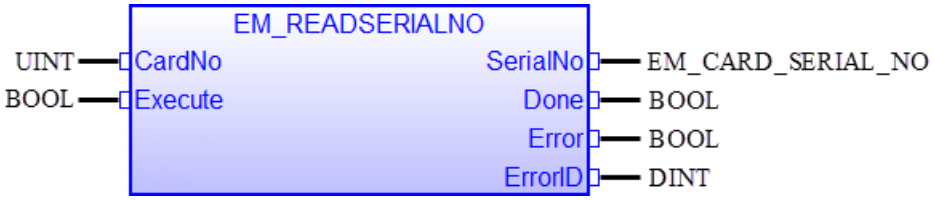
- ECAT-M801 has 13 onboard DI channels
- e-M901 has 8 onboard DI channels



Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number.
	Enable	BOOL	• Continuously read the status of the state diagram while enabled.
VAR_OUTPUT	Valid	BOOL	• TRUE: The digital input states are valid • FALSE: Failed to read the digital input states
	Busy	BOOL	• TRUE: Still busy reading the DI channels
	Error	BOOL	• TRUE: Error occurred during the read operation.
	ErrorID	DINT	Error identification Command-specific error code
	Di_0 - Di_15	BOOL	• Digital input state • For ECAT-M801 only channel Di_0 to Di_12 are valid • For e-M901 only channels Di_0 to Di_7 are valid

7.8 EM_ReadSerialNo

This FB reads the serial number of the ECAT-M801/e-M901 PCIe card.



Input/Output	Name	Type	Comments
VAR_INPUT	CardNo	UINT	• The ECAT-M801/e-M901 card number. • The card number has to be set via a dip switch on the ECAT-M801/e-M901 card.
	Execute	BOOL	Read the SDO at rising edge
VAR_OUTPUT	SerialNo	EM_CARD_SERIAL_NO	Serial number of the ECAT-M801/e-M901 PCIe card
	Done	BOOL	• TRUE: Finished reading the serial number
	Busy	BOOL	• TRUE: Still busy reading
	Error	BOOL	• TRUE: Error occurred
	ErrorID	DINT	Error identification Command-specific error code

Remarks:

- The structure '*EM_CARD_SERIAL_NO*' is defined as follows:

```
STRUCT
   Number : ARRAY[0..7] OF USINT;
END_STRUCT
```

The serial number length of 8 bytes is written to the USINT array of 8 elements.

8 ECAT Utility

The ECAT Utility is a configuration tool for creating network configuration files, initializing and monitoring the EtherCAT network, and supports simple motion control functions for testing purposes. The utility is created by ICPDAS specifically for the ECAT-M801 and EMP9K (EMP-9xx8-xx) EtherCAT masters and can only be used together with these two masters. The master and slaves must be online for the utility to function. It scans the EtherCAT topology and lists all slaves that are currently online in a tree view.

In the remainder of the manual, the abbreviation EMP9K is used for EMP-9xx8-xx.

8.1 Installation

The ECAT Utility is pre-installed on the EMP9K platform and no additonal installation is required.

The following installation process is only required for the Win-GRAF PC runtime:
The installation files and manual for the ECAT Utility has to be downloaded from the ICPDAS website ' https://www.icpdas.com/'. Enter the name 'ECAT-M801' or ' EMP-9' into the search box to get to the product website and click the 'Download Center' button

- Download the software and hardware manuals from the 'User Manual' category
- Download the setup file listed in the 'ECAT Master Driver' category.

8.1.1 Software Installation

Follow the installation instructions described in the 'EtherCAT Master Software Manual' to install all the driver necessary for the host to PCI communication and the ECAT-Utility.

8.1.2 Hardware Installation

The 'EtherCAT Master Hardware Manual' describes how to set the card ID number, install the PCIe card, wire the external connections and connect the device to the EtherCAT slaves. By factory default the card ID is set to zero. If only one ECAT-M801 card is being used the card ID does not need to be set.

8.2 EtherCAT Network Configuration

This chapter describes the procedure to setup a EtherCAT network and the basic function of the ECAT-Utility needed to integrate a servo/ step control slave into the network in order to allow the PLCopen defined function to command control over the axes.

Preparation for PC-Runtime only:

Before starting the ECAT-Utility make sure the ECAT-M801 card is plugged into the PCIs slot and the EtherCAT slaves are online.

Step 1: Set the card ID of the ECAT-M801/e-M901 via the onboard dipswitch. The default card ID is zero. It is not necessary to set the ID if only one EtherCAT master card is being used.

Step 2: Plug the ECAT-M801 into the PCI slot and make sure all the slaves are online.

For EMP9K and PC-Runtime:

In the first step it is necessary to scan the EtherCAT network for all the slaves online.

Step 1: Add the ESI file of each slave in the EtherCAT network to the ESI directory of the ECAT-Utility:
- ECAT-M801 device: '*C:\icpdas\Ecat-M801\ESI*'

The ESI file is provided by the manufacturer or distributor of the EtherCAT slave.

Step 2: Run the ECAT-Utility.
The '*ECAT_Utility.exe*' can be found in the following directory:
- ECAT-M801 device: '*C:\icpdas\Ecat-M801\Utility*'

Step 3: The popup window requires you to do the following configurations (Figure 19):

1. **MDevice Card No:** Set the 'MDevice Number' of the card ID you want to access. By default the ID number is set to zero. The ID of the ECAT-M801 device is set via the dip switch. For the EMP-9000 series, the ID is fixed at zero. 'MDevice' stands for EtherCAT master.
2. **Addressing Method:** Select the station addressing method. Two options are supported:
   - POSITION: At the startup the EtherCAT master automatically assign each slave an address according to their node position in the network. The slave node closest to the master is assigned the address 0 and the address of each following slave node is increased by one. Only the master sets the address and the slave can not change it. Switching the node position in the network causes the master to

assign a new address according to the new slave offset position in the next startup. It is therefore not possible to changed node position after the network configuration as been done otherwise the application program will access the wrong slave station.

- ALIAS (ID): A station alias is a value (with a range from 0 to 65535) designating a station. The user can a assign each slave a fixed address via the master or by directly setting the dip or rotary address switch of the slave. The advantage is that the master can always access the slave even if its station position in the network has been changed. If this option is selected it is required to set the alias for each slave. Make sure to assign each slave with a unique alias value in the network.

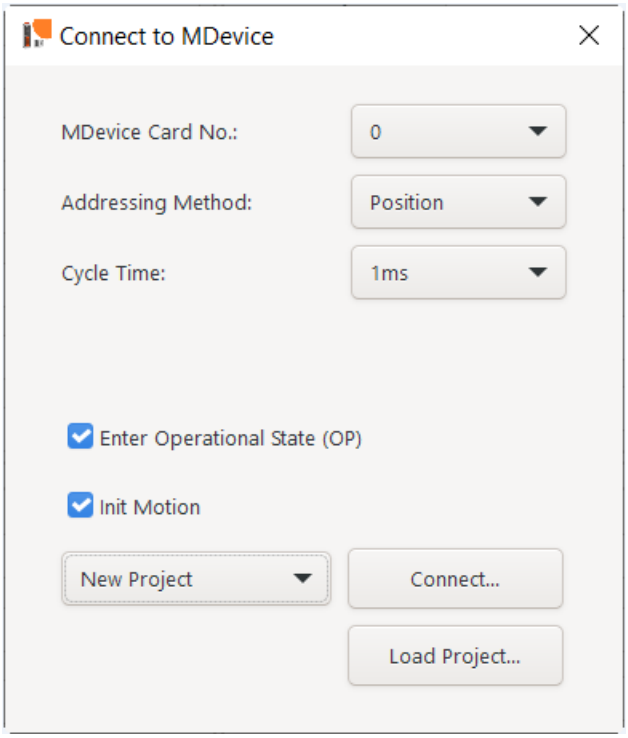


Figure 19: Project configuration

3. **Cycle Time:** Set the cycle time of the EtherCAT master. The EtherCAT master will at the set time interval exchange data with all the slaves in the EtherCAT network.
4. **Enter Operational State (OP):** If enabled the utility will automatically set the EtherCAT master in operational state after all the motion configurations have been completed.
5. **Init Motion:** If enabled the utility will automatically do all the motion initialization after each utility start.
6. Select Project: The settings made through the utility are saved as a project file on your local host. Two options are available:

- A new project can be selected. Select '*New Project*' from the drop box.
- An existing utility project can be open. Select a project from the drop box or use the 'Load Project' button to open an existing project.

7. **Connect...**: Click the '*Connect...*' button to connect to the EtherCAT master and motion control engine.
   - For a new project the utility will automatically scan the EtherCAT network and do the necessary network and motion configuration.
   - For an existing project, the utility uses the project file setting to configure the EtherCAT network and Motion Engine.

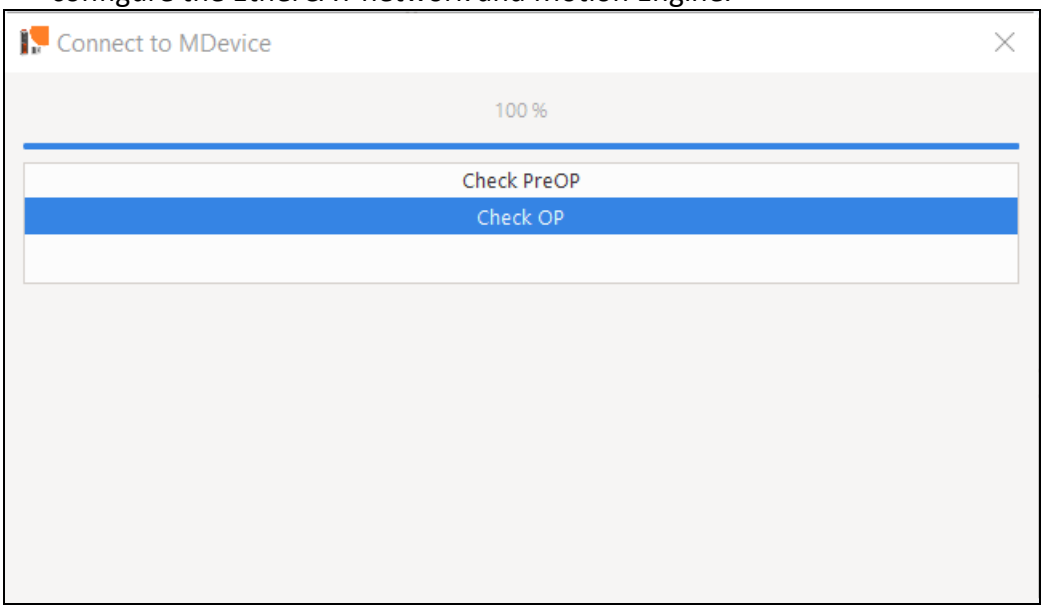


Figure 20: Master is scanning the EtherCAT network for online slaves

After the scan process successfully completed and if the '*Enter Operational State (OP)*' and '*Init Motion*' is enabled (Figure 19), then the utility automatically executes the following steps (Figure 21)

1. Creates a network configuration file and saves it under network information file number 0 in the EtherCAT master (tempupload).
2. Generates a motion configuration file (auto.motcfg) for the connected stepper and servo drives.
3. Displays all the detected slaves in a list in the user interface. (Six servo drives named MADLN05BE from Panasonic were detected here.)
4. Shows the number of slaves in the network (here: 6)
5. Switches the EtherCAT state machine into OP mode.

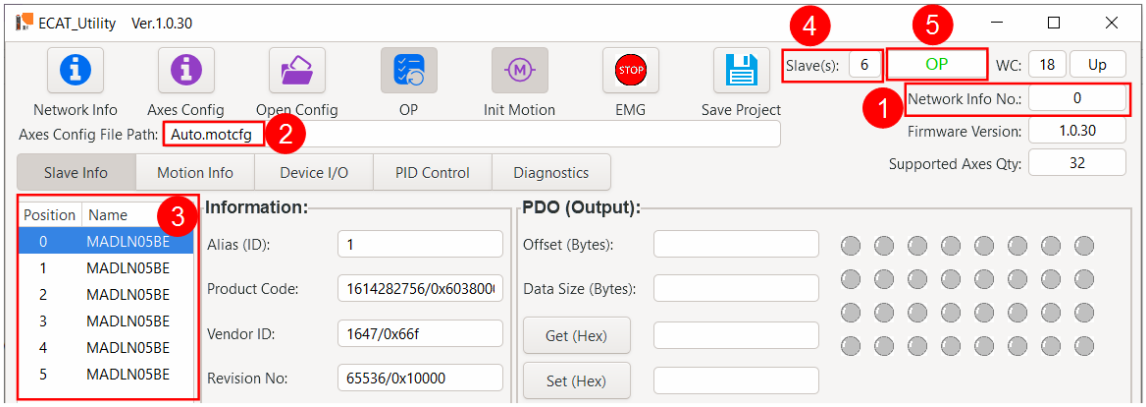


Figure 21: Utility in OP mode

Now the user can operate and control each slave directly through the utility. For example motion commands can be send via the 'Motion Info' tab or digital and analog in- and outputs can be monitored and controlled via the 'Device I/O' tab.

8.2.1 Manual Network Configuration

The utility allows you to configure network settings manually. The following steps are necessary if the 'Enter Operational State (OP)' (Figure 19) has been disabled or you want to change the automatically generated configurations.

Step 1: Scan the EtherCAT network and list all the online slave in a tree view:
Click the network information button to open the network configuration dialog.

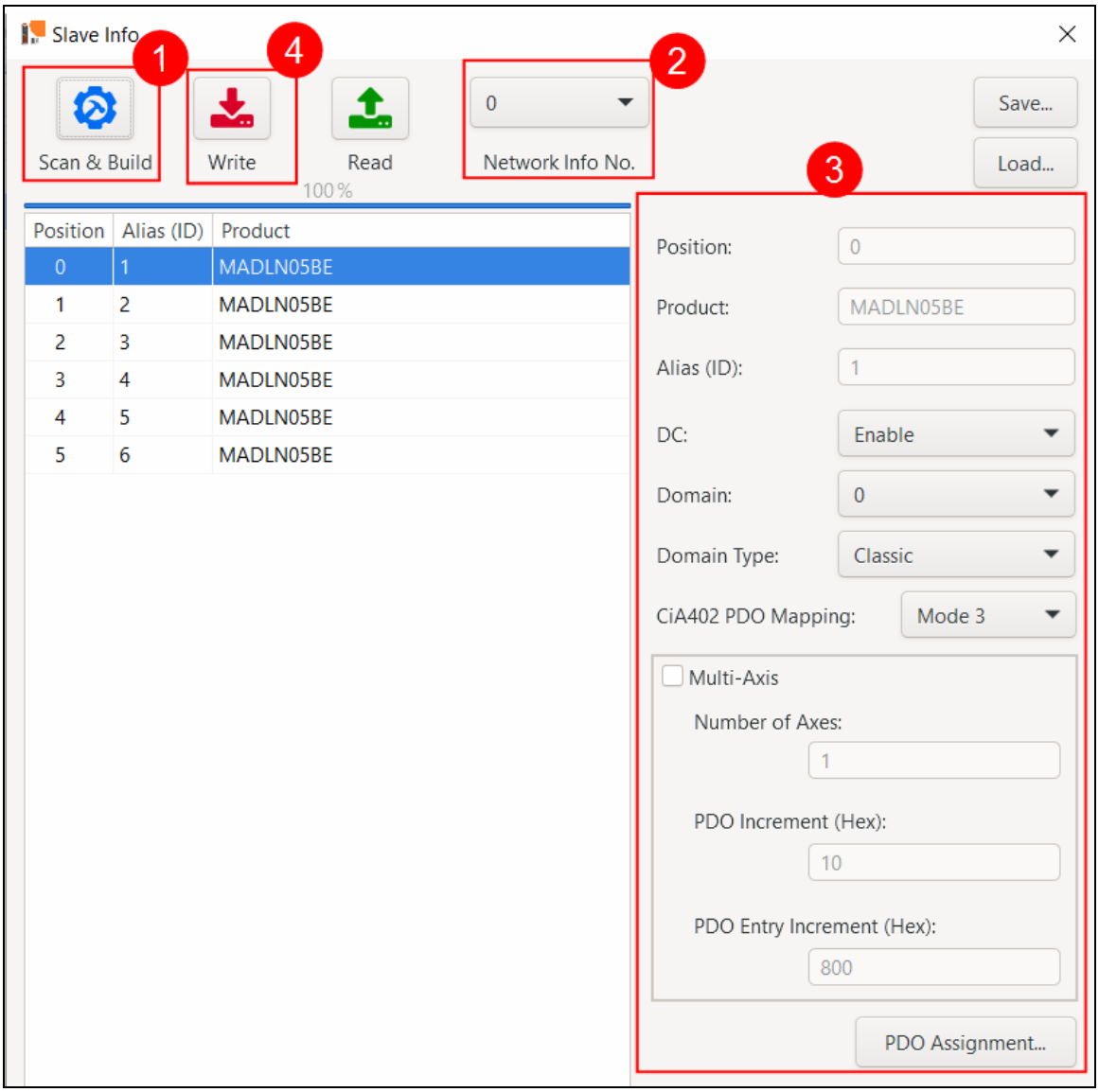


Figure 22: Manual configuration of the EtherCAT network

Step 2: Create a network configuration file.

1. Build a network configuration file by clicking the 'Scan & Build' button. The EtherCAT master scans the network for all the station online and list them in a tree view according to their node position in the network. The slave node closest to the master card is listed at the top, the second closest slave at the second position in the tree view list, etc.. The value of sub item 'Position' in the tree view indicates the node position in the network. The smaller the position value the closer the station node is to the master.
2. Set the network information ID number. The ECAT-M801/EMP9K can store several network information files. It is therefore important to identify the file with an ID number.

Your PLC application program has to tell the master which network information file to use for the initialization and network configuration. This is being done by assigning the input parameter 'NetworkInfo' of the function block '*EM_InitMaster*' to one of the ID numbers of the network information files. During the start-up phase the master initializes and configures the EtherCAT network according to assigned network information files.

3. Setup and configuration:
Only the motion control relevant configuration will be introduced. For a more detail description consult the ECAT-Utility manual.

Setup items:

i) **DC**: If the slave supports DC communication and meets the master DC cycle setting of your system then enable the DC mode. The master DC cycle time setting has to be set by the function block '*EM_InitMaster*' via the '*CycleTime*' input parameter in your PLC application. If the DC cycle time is faster than the minimum cycle time supported by the slave, select one of the options:

- Option 1: do not enable the DC mode of the selected slave otherwise the master will not be able to enter the operation (OP) mode
- Option 2: set the '*CycleTime*' input parameter of '*EM_InitMaster*' to a time value equal or greater the supported cycle time of the slowest slave in the network.

ii) **CiA 402 PDO Mapping**:
CiA 402 object dictionary defines a number of process data objects (PDO). According to the CiA402 specification not all objects are mandatory and therefore some CiA402 slaves only support the basic PDOs.

Using predefined PDO mapping:
To simplify the process of data mapping procedure a list of already mapped PDO (Mode 0, Mode 1, ..) are provided (Table 17). You can directly select one of the mode. Before selecting a mode you have to make sure that the CiA402 slave supports all the PDO entries of this mode, otherwise the master will not enter operation mode or work abnormally.

Mode	RxPDO		TxPDO	
Mode 0	6040	Controlword	6041	Statusword
	6060	Modes of operation	603F	Error code
	607A	Target Position	6061	Modes of operation display

Mode	RxPDO		TxPDO	
	60FF	Target Velocity	6064	Position actual value
	60B8	Touch probe function	606C	Velocity actual value
	60B1	Velocity offset	60FD	Digital inputs
Mode1	6040	Controlword	6041	Statusword
	6060	Modes of operation	603F	Error code
	607A	Target Position	6061	Modes of operation display
			6064	Position actual value
			606C	Velocity actual value
			60FD	Digital inputs
Mode2	6040	Controlword	6041	Statusword
	6060	Modes of operation	603F	Error code
	607A	Target Position	6061	Modes of operation display
	60FF	Target Velocity	6064	Position actual value
	6071	Target Torque	606C	Velocity actual value
	60B8	Touch probe function	60FD	Digital inputs
	60B0	Position offset	6077	Torque actual value
	60B1	Velocity offset		
	60B2	Torque offset		
Mode3	6040	Controlword	6041	Statusword
	6060	Modes of operation	603F	Error code
	607A	Target Position	6061	Modes of operation display
	60FF	Target Velocity	6064	Position actual value
			606C	Velocity actual value
			60FD	Digital inputs

Table 17: CiA402 PDO mapping modes

User defined PDO mapping:

The user is allowed to do the process data mapping manually by selection the '*User Define*' from the dropdown list. This method is error prone and a good understanding of the mapping procedure is required. The user defined mapping has to include the CiA 402 objects listed in Table 18.

Mapping procedure: After selecting '*User Define*' click the '*PDO Assignment...*' button to open the PDO configuration window (Figure 23). Select a PDO from the '*Unassigned PDOs*' list and move it to the '*Assigned PDOs*' list. The object entries of each PDO are shown in the '*PDO List*'. Click the '*Insert*' button to either create a new PDO or add a new CiA402entry to an existing PDO.

Close the '*PDO Assignment*' to apply the PDO assignment.

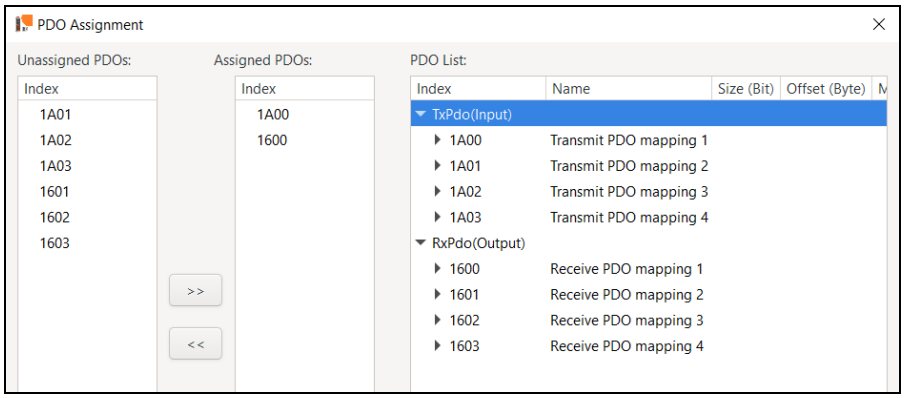


Figure 23: Manual PDO assignment

RxPDO		TxPDO	
6040	Controlword	6041	Statusword
6060	Modes of operation	6061	Modes of operation display
607A	Target Position	6064	Position actual value
		60FD	Digital inputs

Table 18: 'User Define' basic PDO entries

iii) **Multi-axis**:
Most of the CiA 402 servo/ stepper slaves just control one axes. Check the '*Multi-axis*' checkbox if the CiA 402 slave control more than one axis. Do not check the box for the four axes ECAT-2094S device because it is not a CiA 402 slave.
Set the number of axes. Consult the slave user manual to determine the PDO and PDO entry increment.

4. Download the network information file to the ECAT-M801/EMP2K MDevice by clicking the '*Write*' button. The network configuration has been completed now.
In the PLC application, call the '*EM_InitMaster*' function block and set the '*NetworkInfo*' input parameter to the number specified in the '*Network Info No*' text field.

8.3 Motion Control Test

This chapter describes how to set up the ECAT utility to directly control a single axis or a group of axes. Basic motion command like home search, jog, linear/circular movement are supported.

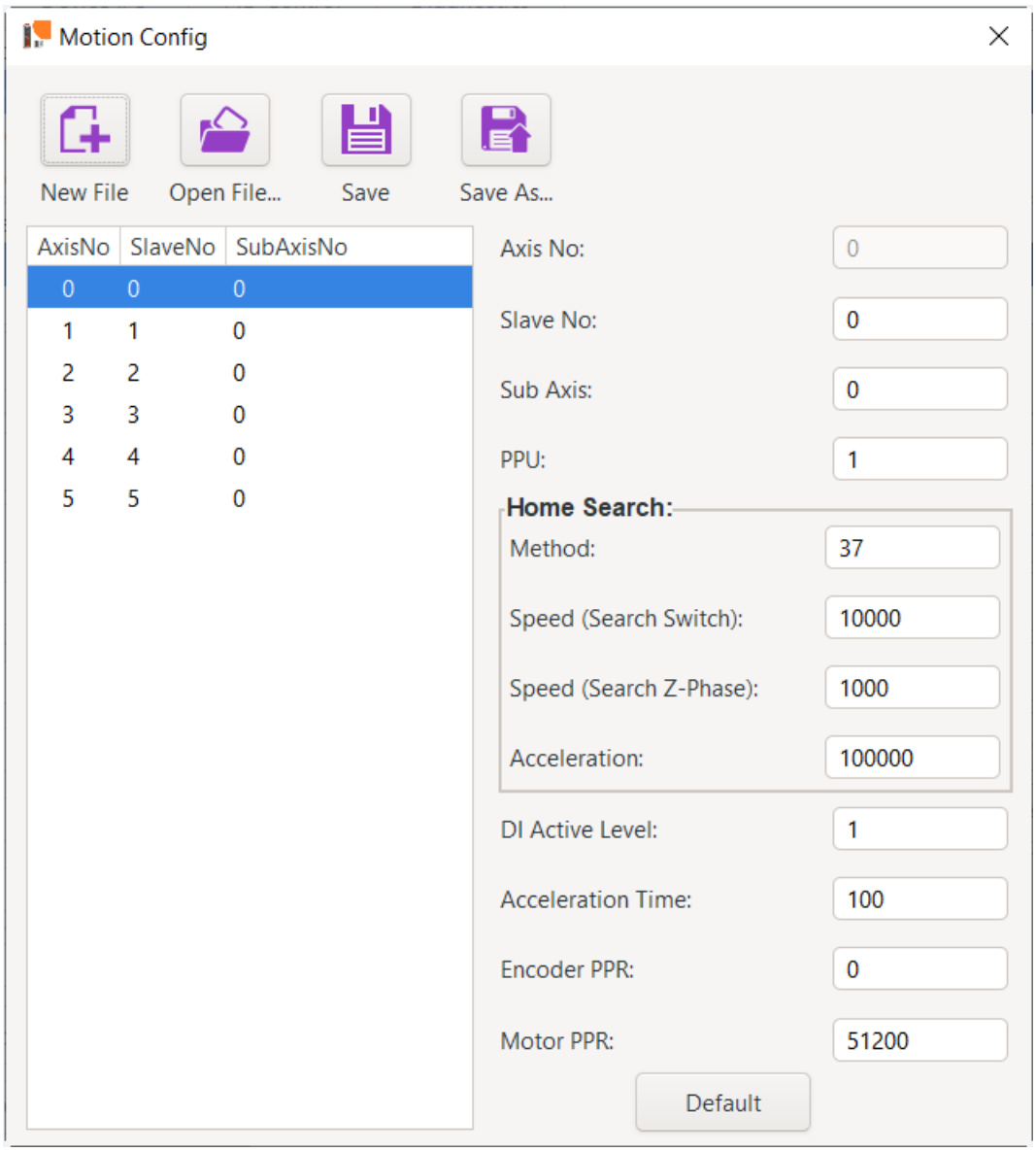
8.3.1 Axis Configuration

If the '*Operational State (OP)*' (Figure 19) is enabled the ECAT-Utility during startup automatically scans and configures the EtherCAT network and switches the EtherCAT state machine into OP mode. If '*Init Motion*' (Figure 19) is enable the utility will automatically do the basic motion configuration if no axis configuration has been done prior. Every stepper/servo slave on the network can now be controlled directly from the utility by using the default axis settings.

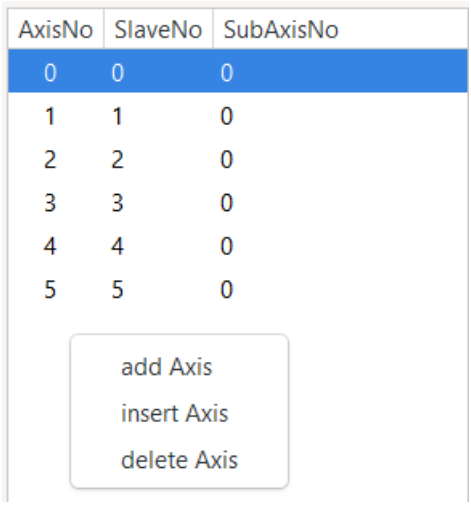
If the default values does not meets your requirement this section will describe how to set the parameters for each axis. For example to assign each stepper/servo slave with an axis number, set the pulse resolution, home search method, etc.. It is important to remember that the settings made in this chapter are used only for the ECAT-Utility and have no effect on the PLC logic control application

The following step describes how to create a new motion configuration file or modify the configuration setting done automatically by the utility.

Step 1: Create an axis configuration file:
Click on the '*Axis Config*' button to open the axis configuration dialog.



- Create a new file by clicking the '*New file*' button
- Add/insert a new axis to the list by right clicking the list view and selecting '*add Axis*' or '*insert Axis*'.



- Set the axis properties:
  Select an axis from the list view by clicking on it, then set all the parameters listed on the right side of the window. For example: Set the axis number for the servo/stepper drive, the pulse per unit (PPU) resolution, home speed, etc.
- Save the configuration file to the local PC as a '**.motcfg*' format. This step completes the axis configuration procedure.

Step 2: Load the axis configuration file into the ECAT-Utility memory:
Click the '*Open Config*' button and select the axis configuration file '*.motcfg*' created in the previous step.

Step 3: Set the EtherCAT master into operation OP mode:
Click the '*Start/Stop Operation*' button and wait until the master entered successfully the OP. The wait time depends on the number of slaves in the network and the configuration time needed for each slave. The status bar at the window top indicate whether the master is in OP mode. The info field at the window bottom displays the error type when the master could not enter the OP mode. The error type are described in the user manual of the EtherCAT master '*EtherCAT_Master_software_manual_en.pdf*'

Step 4: Initialize the axis with the configuration data:
Click the '*Init Motion*' button to initialize the motion structure inside the ECAT-Utility with the configuration file data.

Step 5: The ECAT-Utility can directly control each axis now. Click the 'Motion Info' tab to show the current status of all axes and display all the motion command supported by the utility.

8.3.2 Manual Axis Control Setting

Every time after starting the ECAT-Master utility the steps described below have to be taken to directly access each servo/stepper . If the '*Enter Operational State*' and '*Init Motion*' (Figure 19) has been selected, the utility executes the steps automatically.

1. Set the master into operation mode by clicking the '*Start/Stop Operation*' button on the toolbar. The master will use the selected network information file number stored in the ECAT-M801/EMP9K device as described in chapter 8.2.1. If the master can not find a network information file or the slave listed in the file can not be found in the network then the master will not go into operational mode.

2. Load the axis configuration file into the ECAT-Utility memory. This file basically assigns each servo/stepper motor slave an axis number through which the utility can access it.
   - Click the '*Open Config*' and select a previously created ".motcfg" file.
3. Initialize the motion engine and take control over the axis.
   - Click the '*Init Motion*' button
   - Click the '*Motion Info*' tab (Figure 24) to show the current status of each axis
   - Click the '*Axis*' tab in the lower half of the window to display the window for all the single axis commands.

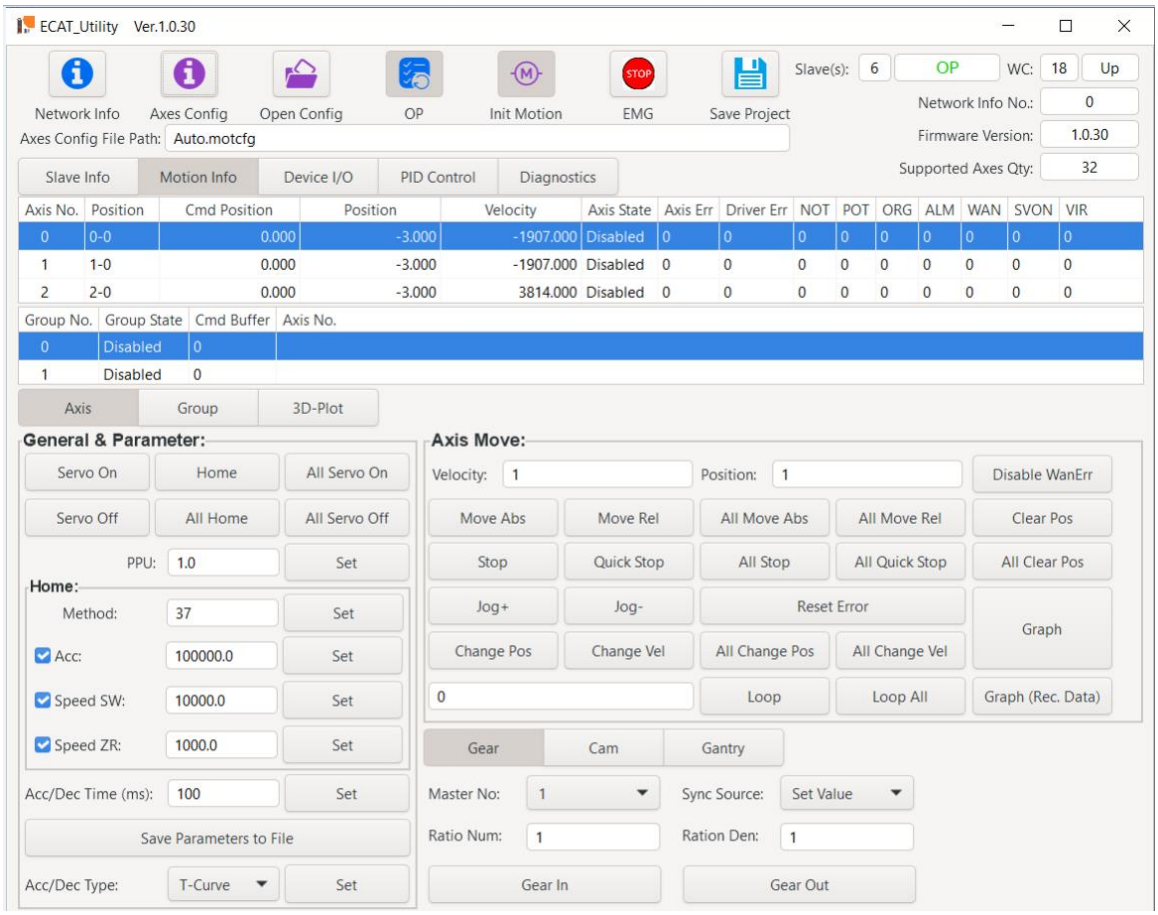


Figure 24: 'Motion Info' tab

8.3.3 Single-Axis Control

Single-axis parameter settings:

1. Choose an axis by clicking an axis number in the single-axis motion information panel ('*Motion Info*'). An axis is selected when the axis line is shown in a blue background color.
2. 2. The basic movement parameters have already been set in the '*.motcfg*' file. It is possible to change these settings. Settings such as the home search, acceleration/deceleration time and curve, pulse resolution, etc. can be changed at any time. Each setting has to be confirmed by clicking the '*Set*' button. It is important to note that these settings will not be saved to the '*.motcfg*' file.
3. Before any movement command other than the home search is issued, the *Velocity*' and '*Position*' parameters must be set.The position value can be a relative or absolute position depending on the motion command.

Motion control:

1. Click the '*Servo ON/OFF*' button to enable or disable the drive. For virtual axis, the motion control can be started without this operation.
2. Home search: Click the '*Home*' button to start homing of this axis.
3. Absolute distance move: Set the absolute position in the '*Position*' field and click the '*MoveAbs*' button to command the selected axis to the new position.
4. Relative distance move: Set the relative distance to travel in the '*Position*' field and click the 'MoveRel' button to drive the set distance.
5. Stop: Clicking the '*Stop*' or '*QuickStop*' button to stop the axis.
   - The '*QuickStop*' command will stop the axis immediately without deceleration.
   - The '*Stop*' command controls a decelerates stop using the 'Acc/Dec Time' value.
6. Jog: Start a movement with the set '*Velocity*' speed. Clicking the '*Jog+*' or '*Jog-*' button. The movement continues while the left mouse button is held down. The movement slows down and stops as soon as the mouse button is released.
7. Emergency stop: Click the '*EMG*' button to immediately stop all axes.

For more information about single and multi-axis commands refer to the manual 'EtherCAT_Master_software_manual_en.pdf '

9 Appendix

9.1 Abbreviation

Abbreviation	Description
FB	Function Block
MFB	Motion Function Block
CoE	CAN application protocol over EtherCAT
DC	Distributed Clock
ENI	EtherCAT Network Information
ESC	EtherCAT Slave Controller
OP	EtherCAT master or slave state 'Operational'
PreOP	EtherCAT device state 'Pre-Operational'
SafeOP	EtherCAT device state 'Safe-Operational'
WKC	Working Counter
EMP9K	EMP-9xx8-xx series (e.g. EMP-9058-16, EMP-9258-32, EMP-9098-16, etc.)

9.2 Error Codes

Error ID	Error Code	Description
ECAT_ERR_REQUEST_MASTER	-1001	Failed to request master
ECAT_ERR_ETHERNET_LINK_DOWN	-1002	Ethernet network link status is down
ECAT_ERR_SLAVES_STATE	-1003	Not all slaves are in state OPERATIONAL
ECAT_ERR_WORKING_COUNTER	-1004	Working counter mismatch
ECAT_ERR_SLAVE_CNT_EXCEEDED	-1005	Connected slave count exceeds maximum support slave count
ECAT_ERR_CREATE_DOMAIN	-1006	Failed to create domain data
ECAT_ERR_ALLOCATE_SLAVE_DATA	-1007	Failed to allocate slave data
ECAT_ERR_CONFIG_SLAVE	-1008	Failed to configure slaves
ECAT_ERR_NETWORK_MISMATCH	-1009	Currently connected bus topology does not match configured one
ECAT_ERR_MASTER_ACTIVATE	-1010	Failed to activate master

ECAT_ERR_GET_PROCESS_DATA	-1011	Failed to get domain process data
ECAT_ERR_CONFIG_CYCLIC_TASK	-1012	Failed to configure cyclic task
ECAT_ERR_RUN_CYCLIC_TASK	-1013	Failed to run cyclic task
ECAT_ERR_INVALID_SLAVE_TYPE	-1014	Invalid slave type
ECAT_ERR_SAME_SLAVE_NO	-1015	Same slave number
ECAT_ERR_INVALID_SLAVE_NO	-1016	Invalid slave number
ECAT_ERR_INVALID_PARAM	-1017	Invalid parameter
ECAT_ERR_INVALID_DATA_SIZE	-1018	Invalid size of data
ECAT_ERR_SDO_REQUEST_BUSY	-1019	SDO request is being processed
ECAT_ERR_SDO_REQUEST_ERROR	-1020	SDO request processing failed
ECAT_ERR_ALLOCATE_PDO_QUEUE	-1021	Failed to allocate PDO queue data
ECAT_ERR_INVALID_OFFSET	-1022	Invalid data offset
ECAT_ERR_INIT_MOTION	-1023	Failed to initialize motion
ECAT_ERR_GET_SLAVE_INFO	-1024	Failed to get slave information
ECAT_ERR_OPEN_FILE	-1025	Failed to open file
ECAT_ERR_WRITE_FILE	-1026	Failed to write data to file
ECAT_ERR_READ_FILE	-1027	Failed to read data from file
ECAT_ERR_FUNC_NOT_SUPPORT	-1028	Function is not supported
ECAT_ERR_INVALID_CHANNEL	-1029	Invalid channel parameter
ECAT_ERR_EMG_HAPPENED	-1030	Emergency happened
ECAT_ERR_INVALID_PID_NO	-1031	Invalid PID number
ECAT_ERR_TIMER_NOT_ACTIVATED	-1032	Timer is not activated
ECAT_ERR_ALL_EVENT_CREATE	-1033	All event created
ECAT_ERR_EVENT_NOT_CREATE	-1034	Event is not created
ECAT_ERR_INVALID_EVENTID	-1035	Invalid event id
ECAT_ERR_INVALID_FILTER_TYPE	-1036	Invalid filter type
ECAT_ERR_SLAVES_ALIAS	-1037	repeating alias or alias == 0
ECAT_ERR_SLAVES_ALIAS_NOT_EXIST	-1038	alias is not exist
ECAT_ERR_OPTASK	-1039	Master are in state OPERATIONAL

ECAT_ERR_MC_NOT_ENABLE_DC	-1100	Not enable DC
ECAT_ERR_MC_TIME_OUT	-1101	Call motion function time out
ECAT_ERR_MC_AXIS_CNT_EXCEEDED	-1102	Initialized axis count exceeds maximum support axis count
ECAT_ERR_MC_NOT_INITIALIZED	-1103	Motion is not initialized
ECAT_ERR_MC_INVALID_AXIS_NO	-1104	Invalid axis number
ECAT_ERR_MC_NOT_AXIS_SERVO_ON	-1105	Axis is not servo-on
ECAT_ERR_MC_INVALID_AXIS_STATE	-1106	Invalid axis state
ECAT_ERR_MC_DRIVE_FAULT	-1107	Drive fault
ECAT_ERR_MC_DRIVE_WARNING	-1108	Drive warning
ECAT_ERR_MC_INVALID_PARAM	-1109	Invalid motion parameter
ECAT_ERR_MC_HOMING	-1110	An error occurs when the homing
ECAT_ERR_MC_LIMIT_ACTIVE	-1111	Limit switch is active
ECAT_ERR_MC_INVALID_ACC_TIME	-1112	Invalid acceleration time
ECAT_ERR_MC_INVALID_GROUP_NO	-1113	Invalid group number
ECAT_ERR_MC_INVALID_GROUP_STATE	-1114	Invalid group state
ECAT_ERR_MC_AXIS_WAS_IN_GROUP	-1115	Axis is already in group
ECAT_ERR_MC_AXIS_IN_OTHER_GROUP	-1116	Axis is already in other group
ECAT_ERR_MC_GROUP_CMD_ALLOCATE	-1117	Failed to allocate group command
ECAT_ERR_MC_GROUP_CMD_BUFFER_OVERFLOW	-1118	Group command is overflow
ECAT_ERR_MC_INVALID_AXIS_SYNC_MODE	-1119	Invalid axis synchronization mode
ECAT_ERR_MC_INVALID_PROFILE_NO	-1120	Invalid profile number
ECAT_ERR_MC_INVALID_GROUP_MOVE_CMD	-1121	Invalid group command
ECAT_ERR_MC_GROUP_CMD_MODE_NOT_SUPPORT	-1122	The function does not support the current group command mode
ECAT_ERR_MC_INVALID_ACC_DEC_TYPE	-1123	Invalid acceleration type parameter
ECAT_ERR_MC_INVALID_VEL	-1124	Invalid velocity parameter
ECAT_ERR_MC_INVALID_ANGLE	-1125	Invalid angle parameter
ECAT_ERR_MC_INVALID_RADIUS	-1126	Invalid radius parameter

ECAT_ERR_MC_INVALID_END_POS	-1127	Invalid end position parameter
ECAT_ERR_MC_INVALID_ECAM_TABLE_NO	-1128	Invalid E-CAM table number
ECAT_ERR_MC_INVALID_NORMAL_VECTOR	-1129	Invalid normal vector parameter
ECAT_ERR_MC_NOT_SETUP	-1130	Not setup
ECAT_ERR_MC_GREATER_THAN_MAX_RODLENGTH	-1131	Calculated value is greater than maximum rod length
ECAT_ERR_MC_LESS_THAN_RODLENGTH	-1132	Calculated value is less than rod length
ECAT_ERR_MC_GREATER_THAN_RECORD_COUNT	-1133	Exceed maximum record count
ECAT_ERR_MC_SOFTWARE_LIMIT_ACTIVATE	-1134	Software limit is active
ECAT_ERR_MC_GANTRY_POS_EXCESSIVE_DEVIATION	-1135	Position excessive deviation of gantry control
ECAT_ERR_MC_GROUP_NO_NOT_SUPPORT	-1136	Group number not support
ECAT_ERR_MC_INVALID_MOVE_CMD	-1137	Invalid move command
ECAT_ERR_MC_QUEUE_IS_FULL	-1138	Queue is full
ECAT_ERR_MC_COORDINATE_TRANS_ON	-1139	Coordinate conversion is active
ECAT_ERR_IPC_INVALID_DEVICE_NO	-1201	Invalid device number
ECAT_ERR_IPC_DEVICE_IS_OPEN	-1202	Device is open
ECAT_ERR_IPC_DEVICE_NOT_OPEN	-1203	Device is not open
ECAT_ERR_IPC_CREATE_HANDLE	-1204	Failed to create IPC handle
ECAT_ERR_IPC_BUSY	-1205	IPC is busy
ECAT_ERR_IPC_TIME_OUT	-1206	IPC is time out
ECAT_ERR_IPC_INVALID_CMD	-1207	Invalid IPC command
ECAT_ERR_IPC_WRITE_SHM	-1208	Failed to write data to shard memory
ECAT_ERR_IPC_READ_SHM	-1209	Failed to read data from shard memory
ECAT_ERR_IPC_RUN_DOWN_UP_LOAD	-1210	Failed to process download / upload data

ECAT_ERR_IPC_INVALID_SHM	-1211	Invalid shard memory
ECAT_ERR_IPC_DEVICE_NOT_READY	-1212	Device is not ready
ECAT_ERR_DRV_GET_INFO	-1301	Failed to get driver information
ECAT_ERR_DRV_CREATE_HANDLE	-1302	Failed to create driver handle
ECAT_ERR_DRV_IOCTL	-1303	Call driver IO control error
ECAT_ERR_DRV_DEVICE_NOT_FOUND	-1304	Device not found
ERR_MC_DISABLED	-2001	Single axis is disabled and therefore can not be moved
ERR_MC_STOPPING	-2004	Single is in stopping state and can not be moved
ERR_MC_HOMING	-2005	Single is in homing state and can not be interrupted by another move command
ERR_MC_GS_DISABLED	-2011	Group is disabled and therefore can not execute and movement command
ERR_MC_GS_ERRORSTOP	-2013	Group is in 'Group*ErrorStop*' state and will not execute any movement commands
ERR_INPUT_PARA	-2021	Incorrect or not supported MFB input parameter value
FB_ERR_AXIS_REF_INVALID	-2022	Either card number or axis number is out of supported range
ERR_MULTI_INSTANCE_CALL	-2024	More than one instance of the same MFB has been activated. For some MFBs only one instance can control a card or/and axis at a time. For example '*EM_InitMaster*': Only one instance can take control of the ECAT-M801 master card. If two instance of '*EM_InitMaster*' try to take control an error will be generated.
ERR_GRP_AXIS_NOT_IN_GROUP	-2030	No axis has been added to the group
ERR_GRP_AXIS_MISMATCH	-2031	Internal group assignment error detected

ERR_ARRAY_OVERFLOW	-2035	The size of the input array of a MFB is too small. Increase the array size (elements) to solves this error. For example: The number of array elements does not correspondent to the number of axis in the group
ERR_AXES_GROUP_REF_INVALID	-2036	The card number or the group number is out of range
ERR_CARD_NO_MISMATCH	-2037	The card number of the group and axis are not the same
ERR_POWER_ON_MULTI_AXIS	-2039	• More than one '*MC_Power*' instance tried to take controls the same axis, which is not allowed according to PLCopen (Part1 page 29/14) • The same '*MC_Power*' instance tried to take control of more than one axis
ERR_FILE	-2041	File operation error
ERR_FILE_DIRECTORY_NOT_FOUND	-2042	File directory not found
ERR_PATH_ID_VALUE_NOT_SUPPORTED	-2043	Incorrect file path ID
ERR_STRING_LENGTH	-2044	String is too short or long
ERR_INTERVAL_VALUE_NOT_SUPPORTED	-2045	Interval value not supported
ERR_INC_MOVE_FILE_EXECUTING	-2046	The incremental path file could not be executed because it is being executed by another MFB.
ERR_INVALID_DATA_SIZE	-2049	The data sized is not being supported
ERR_BUFFER_MODE_NOT_SUPPORTED	-2102	Command buffer mode not supported
ERR_TRANSITION_MODE_NOT_SUPPORTED	-2104	The group transition mode is not being supported
ERR_CIRC_ANGLE_VALUE_MISSING	-2105	Angle value for the circular command is missing
ERR_CIRC_MODE_NOT_SUPPORTED	-2106	
ERR_CIRC_TOO_MANY_AXES_IN_GROUP	-2107	
ERR_CIRC_DIMENSION	-2108	only dimension 2D and 3D are supported for circular motion

9.3 Table of Figures

9.4 Table of Tables