Technical Specification
PLCopen - Technical Committee 2 – Task Force

Function blocks for motion control

(Formerly Part 1 and Part 2)

Version 2.0, Published

DISCLAIMER OF WARANTIES

THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS AND MAY BE SUBJECT TO FUTURE ADDITIONS, MODIFICATIONS, OR CORRECTIONS. PLCOPEN HEREBY DISCLAIMS ALL WARRANTIES OR CONDITIONS OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES, OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, FOR THIS DOCUMENT. IN NO EVENT WILL PLCOPEN BE RESPONSIBLE FOR ANY LOSS OR DAMAGE ARISING OUT OR RESULTING FROM ANY DEFECT, ERROR OR OMISSION IN THIS DOCUMENT OR FROM ANYONE'S USE OF OR RELIANCE ON THIS DOCUMENT.

March 17, 2011.

Function blocks for motion control

The following specification has been developed within the PLCopen Motion Control Task Force.
This specification was written by the following members of the Motion Control Task Force:

Nils Gotha	Baumüller
Klaus Bernzen	Beckhoff
Wilfried Plaß	Beckhoff
Joachim Unfried	B&R
Martin Schrott	B&R
Roland Schaumburg	Danfoss
Jan Braun	Eckelmann
Alfred Möltner	Elau/Schneider Electric
Ryszard Bochniak	2MC-Software (Eckelmann)
Djafar Hadiouche	GE
Juergen Hipp	ISG
Harald Buchgeher	Keba
Candido Ferrio	Omron
Josep Lario	Omron
Yoshikazu Tachibana	Omron
Klas Hellmann	Phoenix Contact
Jan Kosa	Phoenix Contact
Burkhard Werner	Phoenix Contact
Wolfgang Fien	Schneider Motion (former Berger Lahr)
Willi Gagsteiger	Siemens AG
Hilmar Panzer	3S-Smart Software Solutions
Edwin Schwellinger	3S-Smart Software Solutions
Lutz Augenstein	Stöber Antriebstechnik
Heiko Berner	Stöber Antriebstechnik
Eelco van der Wal	PLCopen

Change Status List:

Version number	Date	Change comment
V 0.1	September 26, 2008	First working draft. Merging of part 1 and 2, and the corrigendum.
V 0.2	January 6, 2010	As result of the meeting in Frankfurt on November 9 & 10, 2009
V 0.3	February 4, 2010	As result of the meeting in Bad Pyrmont, Feb. 3&4, 2010. Open issues discussed
V 0.4	April 1, 2010	As result of the meeting in Frankfurt, March 17 & 18, 2010. Feedback for Version 1.99 included. Basis for editorial corrections before release as V. 1.99. Document file errors cleaned via docx conversion.
V 1.99	May 21, 2010	Published as Release for Comments
V 1.99A	Dec. 14, 2010	As results of the feedback of several meetings
V 1.99B	Jan. 31, 2011	As a result of the feedback and the Jan. webmeeting
V 1.99C	Feb. 27, 2011	As a result of the Feb. webmeeting
V 1.99D	March 16, 2011	As result of the March webmeeting. Last version before release
V 2.0	March 17, 2011	Official release

Table of Contents

Table of Figures

Table of Tables

1. General

The motion control market displays a wide variety of incompatible systems and solutions. In businesses where different systems are used, this incompatibility induces considerable costs for the end-users, learning is confusing, engineering becomes difficult, and the process of market growth slows down.
Standardization would certainly reduce these negative factors. Standardization means not only the programming languages themselves, (as standardization is achieved using the worldwide IEC 61131-3 standard) but also standardizing the interface towards different motion control solutions. In this way the programming of these motion control solutions is less hardware dependent. The reusability of the application software is increased, and the costs involved in training and support are reduced.

Users have requested that PLCopen helps to solve this problem, which initiated the Motion Control Task Force. This Task Force has defined the programmer's interface by standardizing the Function Blocks for Motion Control.

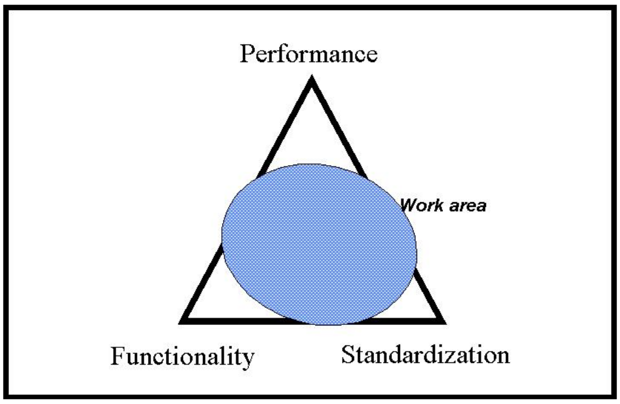


Figure 1: The triangle with user options

For the positioning of this activity, please check figure 1. This triangle has the following user options at its corners:

- Performance
- Functionality
- Standardization.

In practice, users write their programs very closely coupled to the hardware with dedicated functions, in order to get the highest performance possible as dictated by their environment. This limits the user in his options with respect to the target hardware and the reusability of the control software and raises the training investment.
The second user option enables a very broad range of software functionality to be offered. This can be very helpful to the user, but will seldom lead to high performance. Also the training costs are increased.
The third corner, standardization, is primarily focused on reusability across different systems from different suppliers, including integrated, distributed and networked systems, as well as reduction in training investments. Due to the general character of this definition, the performance on different architectures can be less optimal than hard coding. Due to this, standardization should not be expected to offer maximum performance but can closely approach maximum functionality, meaning that the bottom of the triangle is very short.

The first specification was released as an independent library of function blocks for motion control. It included motion functionality for single axes and multiple axes, several administrative tasks, as well as a state diagram. This specification provides the user with a standard command set and structure independent of the underlying architecture.
This structure can be used on many platforms and architectures. In this way one can decide which architecture will be used at a later stage of the development cycle. Advantages for the machine builder are, amongst others, lower costs for supporting the different platforms and the freedom to develop application software in a more independent way, without limiting the productivity of the machine. In addition to those benefits, system maintenance is easier and the education period is shorter. This is a major step forward, and is more and more accepted by users as well as suppliers.

With the release of part 1, it was understood that additional functionality was needed. Part 1 provides the basis for a set of inter-related specifications:

Part 1 - PLCopen Function Blocks for Motion Control
Part 2 - PLCopen Motion Control - Extensions, which in the new release 2.0 is merged with Part 1
Part 3 - PLCopen Motion Control - User Guidelines
Part 4 - PLCopen Motion Control – Coordinated Motion
Part 5 - PLCopen Motion Control - Homing Extensions
Part 6 - PLCopen Motion Control –Fluid Power Extensions

With the release of the underlying document, Part 1 – PLCopen Function Blocks for Motion Control version 2.0, Part 2 – PLCopen Motion Control Extensions has been integrated into the Basic document
The PLCopen Motion Control User Guidelines, Part 3, is an addition to the PLCopen Function Blocks for Motion Control, and should not be seen as stand alone document.

1.1. Objectives

The Motion Control Function Blocks are applicable in the IEC 61131-3 languages with following factors in consideration:

1	Simplicity	-	ease of use, towards the application program builder and installation & maintenance
2	Efficiency	-	in the number of Function Blocks, directed to efficiency in design (and understanding)
3	Consistency	-	conforming to IEC 61131-3 standard
4	Universality	-	hardware independent
5	Flexibility	-	future extensions / range of application
6	Completeness	-	not mandatory but sufficiently

1.1.1. Language context goals

- Focus on definition of Function Block interfaces and behavior and the data types according to the IEC 61131-3 specification.
- These Function Blocks and data types can be used in all IEC 61131-3 languages.
- The examples in this document are given informatively in textual and graphical IEC 61131-3 languages.
- The contents of the Function Blocks can be implemented in any programming language (e.g. IEC 61131-3 ST, C) or even in firmware or hardware. Therefore the content should not be expected to be portable across platforms.
- Reusable applications composed from these Function Blocks and data types are simplified using PLCopen exchange standards.
- This specification shall be seen as an open framework without hardware dependencies. It provides openness in the implementation on different platforms such as fully integrated, centralized or distributed systems. The actual implementation of the Function Blocks themselves is out of the scope of this standard.

1.1.2. Definition of a set of Function Blocks

A basic problem concerns the granularity or modularity of the standardized Function Blocks. The extremes are one Function Block per axis versus a command level functionality. The objectives stated above can be achieved more easily by a modular design of the Function Blocks. Modularity creates a higher level of scalability, flexibility and reconfigurability. Large-scale blocks (Derived Function Blocks) can then be created from these, e.g. the whole axis, for ease of application program building and browsing.
If feasible, a Function Block specified here could be implemented as a Function (for instance MC_ReadParameter).

1.1.3. Overview of the defined Function Blocks

The following table gives an overview of the defined Function Blocks, divided into administrative (not driving motion) and motion related sets.

Administrative		***Motion***	
Single Axis	***Multiple Axis***	***Single Axis***	***Multiple Axis***
MC_Power	MC_CamTableSelect	MC_Home	MC_CamIn
MC_ReadStatus		MC_Stop	MC_CamOut
MC_ReadAxisError		MC_Halt	MC_GearIn
MC_ReadParameter		MC_MoveAbsolute	MC_GearOut
MC_ReadBoolParameter		MC_MoveRelative	MC_GearInPos
MC_WriteParameter		MC_MoveAdditive	MC_PhasingAbsolute
MC_WriteBoolParameter		MC_MoveSuperimposed	MC_PhasingRelative
MC_ReadDigitalInput		MC_MoveVelocity	MC_CombineAxis
MC_ReadDigitalOutput		MC_MoveContinuousAbsolute	
MC_WriteDigitalOutput		MC_MoveContinuousRelative	
MC_ReadActualPosition		MC_TorqueControl	
MC_ReadActualVelocity		MC_PositionProfile	
MC_ReadActualTorque		MC_VelocityProfile	
MC_ReadAxisInfo		MC_AccelerationProfile	
MC_ReadMotionState			
MC_SetPosition			
MC_SetOverride			
MC_TouchProbe			
MC_DigitalCamSwitch			
MC_Reset			
MC_AbortTrigger			
MC_HaltSuperimposed			

Table 1: Overview of the defined Function Blocks

1.1.4. Compliance and Portability

The objective of this work is to achieve a level of portability for Motion Control Function Blocks acting on an axis, and providing the same functionality to the user as described within this document, with respect to user interface, input / output variables, parameters and units used.

The possibility of combining several MC libraries from different vendors within one application is left open to be solved by the systems integrator or end user.

An implementation which claims compliance with this PLCopen specification shall offer a set of (meaning one or more) Function Blocks for motion control with at least the **basic** input and output variables, marked as "**B**" in the defined tables in the definition of the Function Blocks in this document.

For higher-level systems and future extensions any subset of the **extended** input and output variables, marked as "**E**" in the tables can be implemented.

Vendor specific additions are marked with "**V**".

For more specific information on compliance and the usage of the PLCopen Motion Control logo, refer to Appendix B.

- **Basic** input/output variables are mandatory	Marked in the tables with the letter "**B**"
- **Extended** input /output variables are optional	Marked in the tables with the letter "**E**"
- **Vendor Specific** additions	Marked in the vendor's compliance documentation with "**V**"

Any vendor is allowed to add Vendor Specific parameters to any of the Function Blocks specified within this document.

Note:
According to the IEC 61131-3 specification, the input variables may be unconnected or not parameterized by the user. In this case the Function Block will use the value from the previous invocation of the Function Block instance or in case of the first invocation the initial value will be used. Each Function Block input has a defined initial value, which is typically 0.

The data type REAL listed in the Function Blocks and parameters (e.g. for velocity, acceleration, distance, etc.) may be

exchanged to SINT, INT, DINT or LREAL without being seen as incompliant to this standard, as long as it is consistent for the whole set of Function Blocks and parameters.
Implementation allows the extension of data types as long as the basic data type is kept. For example: WORD may be changed to DWORD, but not to REAL.
Any FBs and inputs that are no longer specified in this new version of the specification can be kept in the vendors' systems to keep compatibility, avoiding incompatible changes in existing FBs.

1.1.5. Length of names and ways to shorten them

There are systems that only support a limited number of significant characters in the name. For these rules for shorter names are provided here. These names are still seen as compliant, although have to be mentioned in the certification document.

List of rules to shorten names:

Command	Cmd
Position	Pos
Velocity	Vel
Acceleration	Acc
Deceleration	Decel
Absolute	Abs
Relative	Rel
Actual	Act
Superimposed	SupImp
Additive	Add
Parameter	Par
Continuous	Cont
GearRatioDenominatorM1	RatioDenM1
GearRatioNumeratorM1	RatioNumM1

Resulting compliant names as example:

CommandAborted	CmdAborted
MC_MoveContinuousRelative	MC_MoveContRel
MC_ReadParameter	MC_ReadPar

1.1.6. History

The first official release of Part 1 was made in November 2001. Since that time feedback has been received from both users and implementers. In 2004 it was decided to release a new version, Version 1.1, of Part 1, which includes the changes resulting from inclusion of the feedback into the specification. This update was published in April 2005.
In September 2005 the first official release of Part 2 – Extensions was published.
After that date, a corrigendum and addendum was maintained for both parts. During 2008 the proposal was accepted to merge both part 1 and 2 in one new part 1, to be released as version 2.0, the document you are looking at now.
Basically the two sets of function blocks have been merged. In addition, several overall changes were done. These changes include (however are not limited to):

- The simplification of the representation of the State Diagram, with a.o. the removal of the transition commands
- The new input 'ContinuousUpdate' extending the behaviour of the relevant motion related function blocks
- Adopted description resulting in a changed behaviour of the output 'Active'
- Aborting mode deleted in some FBs
- Changes in the mcAborting enum
- The split of MC_Phasing and MC_MoveContinuous FBs in to relative and absolute versions for both
- New FBs MC_ReadMotionState, MC_ReadAxisInfo, MC_CombineAxes and MC_HaltSuperimposed
- The description at Camming
- The functionality of MC_CamTableSelect is extended with input 'ExecutionMode'. Description of 'Periodic'defined more precise.
- The functionality of MC_CamIn is extended with inputs 'MasterStartDistance'and 'MasterSyncPosition'.
- New input 'MasterValueSource' and corresponding datatype in MC_CamIn, MC_GearIn, MC_GearInPos, MC_ReadMotionState, and MC_CombineAxes

- Input ‘Mode’ of MC_SetPosition now called ‘Relative’ (in line with Part 4)
- Unified naming conventions for Function Blocks, Enum elements, Data types, Structures, Inputs and Outputs for all PLCopen Motion Control specifications.
- The behaviour of the ‘InVelocity’, ‘InGear’, ‘InTorque’, and ‘InSync’outputs changed after the corresponding SET value is reached
- FBs MC_ReadAxisInfo, MC_PhasingRelative and MC_PhasingAbsolute added to Function Blocks which are not listed in the State Diagram
- Description of inputs ‘Axis’, ‘Master’ and ‘Slave’ changed
- Description of outputs ‘Busy’, ‘Error’ and ‘ErrorID’ changed

2. Model

The following Function Block (FB) library is designed for the purpose of controlling axes via the language elements consistent with those defined in the IEC 61131-3 standard. It was decided by the task force that it would not be practical to encapsulate all the aspects of one axis into only one function block. The retained solution is to provide a set of command oriented function blocks that have a reference to the axis, e.g. the abstract data type 'Axis', which offers flexibility, ease of use and reusability.
Implementations based on IEC 61131-3 (for instance via Function Blocks and SFC) will be focused towards the interface (look-and-feel / 'proxy') of the Function Blocks. This specification does not define the internal operation of the Function Blocks.
This leads to some consequences that are described in this chapter.

2.1. The State Diagram

The following diagram normatively defines the behavior of the axis at a high level when multiple Motion Control Function Blocks are «simultaneously» activated. This combination of motion profiles is useful in building a more complicated profile or to handle exceptions within a program. (In real implementations there may be additional states at a lower level defined).

The basic rule is that motion commands are always taken sequentially, even if the PLC had the capability of real parallel processing. These commands act on the axis' state diagram.

The axis is always in one of the defined states (see diagram below). Any motion command that causes a transition changes the state of the axis and, as a consequence, modifies the way the current motion is computed.
The state diagram is an abstraction layer of what the real state of the axis is, comparable to the image of the I/O points within a cyclic (PLC) program.
A change of state is reflected immediately when issuing the corresponding motion command. (Note: the response time of 'immediately' is system dependent, coupled to the state of the axis, or an abstraction layer in the software)

The diagram is focused on a single axis. The multiple axis Function Blocks, MC_CamIn, MC_GearIn and MC_Phasing, can be looked at, from a state diagram point of view, as multiple single-axes all in specific states. For instance, the CAM-master can be in the state 'ContinuousMotion'. The corresponding slave is in the state 'SynchronizedMotion'. Connecting a slave axis to a master axis has no influence on the master axis.

Arrows within the state diagram show the possible state transitions between the states. State transitions due to an issued command are shown by full arrows. Dashed arrows are used for state transitions that occur when a command of an axis has terminated or a system related transition (like error related). The motion commands which transit the axis to the corresponding motion state are listed above the states. These motion commands may also be issued when the axis is already in the according motion state.

Remarks on states:

Disabled	The state 'Disabled' describes the initial state of the axis. In this state the movement of the axis is not influenced by the FBs. Power is off and there is no error in the axis. If the MC_Power FB is called with 'Enable'=TRUE while being in 'Disabled', the state changes to 'Standstill'. The axis feedback is operational before entering the state 'Standstill'. Calling MC_Power with 'Enable'=FALSE in any state except 'ErrorStop' transfers the axis to the state 'Disabled', either directly or via any other state. Any on-going motion commands on the axis are aborted ('CommandAborted').
ErrorStop	'ErrorStop' is valid as highest priority and applicable in case of an error. The axis can have either power enabled or disabled and can be changed via MC_Power. However, as long as the error is pending the state remains 'ErrorStop'. The intention of the 'ErrorStop' state is that the axis goes to a stop, if possible. There is no further motion command accepted until a reset has been done from the 'ErrorStop' state. The transition to 'ErrorStop' refers to errors from the axis and axis control, and not from the Function Block instances. These axis errors may also be reflected in the output of the Function Blocks 'FB instances errors'.

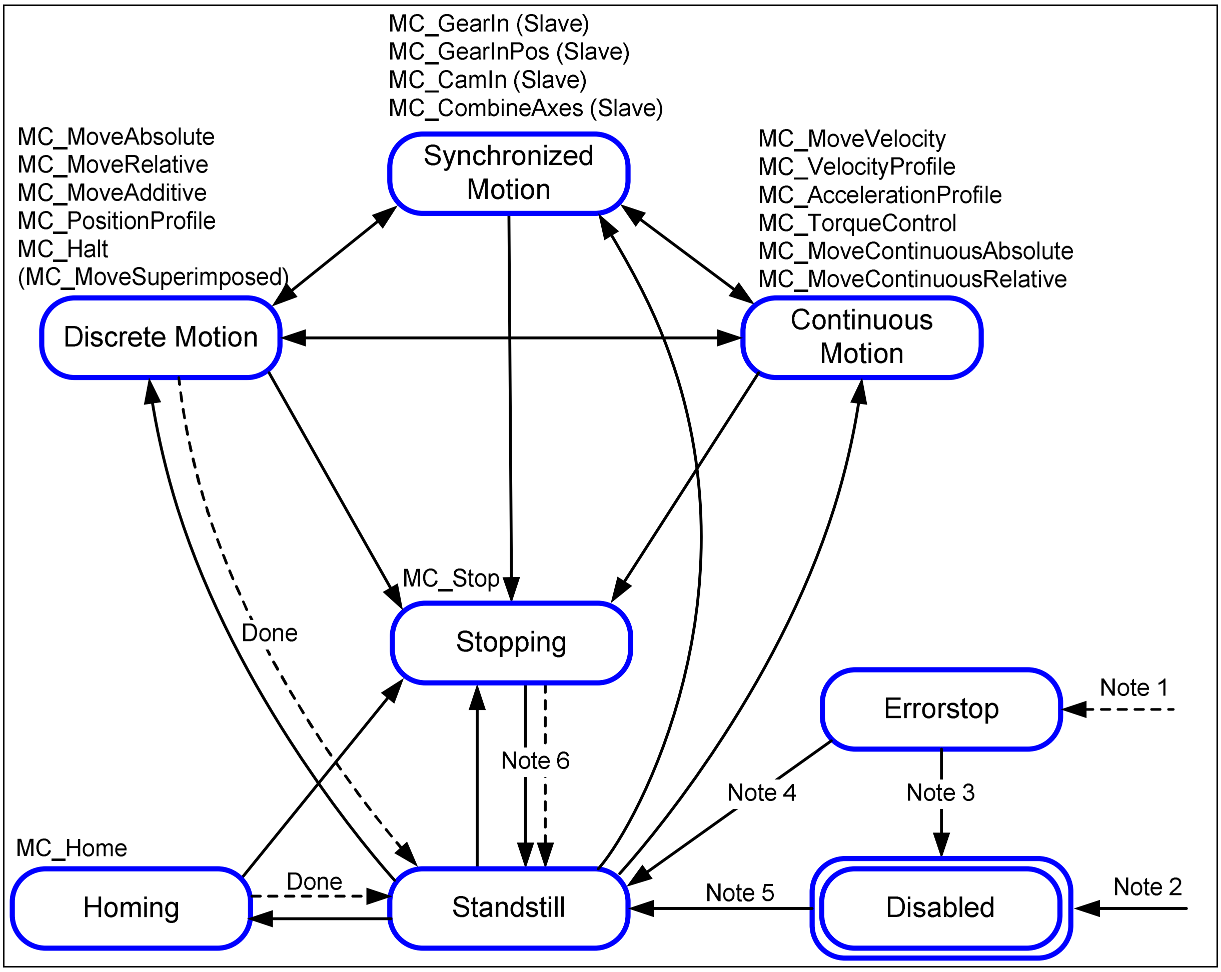
Standstill	Power is on, there is no error in the axis, and there are no motion commands active on the axis.

Remarks on commands:

MC_Stop	Calling the FB MC_Stop in state 'Standstill' changes the state to 'Stopping' and back to 'Standstill' when 'Execute' = FALSE. The state 'Stopping' is kept as long as the input 'Execute' is true. The 'Done' output is set when the stop ramp is finished.
MC_MoveSuperimposed	MC_MoveSuperimposed issued in state 'Standstill' brings the axis to state 'Discrete-Motion'. Issued in any other state the state of the axis is not influenced.
MC_GearOut, MC_CamOut	Change the state of the slave axis from 'SynchronizedMotion' to 'ContinuousMotion'. Issuing one of these FBs in any other state generates an error.

Function Blocks which are not listed in the State Diagram do not affect the state of the State Diagram, meaning that whenever they are called the state does not change. They are:

- MC_ReadStatus
- MC_ReadAxisError
- MC_ReadParameter
- MC_ReadBoolParameter
- MC_WriteParameter
- MC_WriteBoolParameter
- MC_ReadDigitalInput
- MC_ReadDigitalOutput
- MC_WriteDigitalOutput
- MC_ReadActualPosition
- MC_ReadActualVelocity
- MC_ReadActualTorque
- MC_ReadMotionState
- MC_SetPosition
- MC_SetOverride
- MC_AbortTrigger
- MC_TouchProbe
- MC_DigitalCamSwitch
- MC_CamTableSelect
- MC_ReadAxisInfo
- MC_PhasingRelative
- MC_PhasingAbsolute
- MC_HaltSuperimposed



Note 1:	From any state. An error in the axis occurred.
Note 2:	From any state. MC_Power.Enable = FALSE and there is no error in the axis.
Note 3:	MC_Reset AND MC_Power.Status = FALSE
Note 4:	MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE
Note 5:	MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
Note 6:	MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

Figure 2: FB State Diagram

2.2. Error handling

All access to the drive/motion control is via Function Blocks. Internally these Function Blocks provide basic error checking on the input data. Exactly how this is done is implementation dependent. For instance, if MaxVelocity is set to 6000, and the Velocity input to a FB is set to 10,000, either the system slows down or an error is generated. In the case where an intelligent drive is coupled via a network to the system, the MaxVelocity parameter is probably stored on the drive. The FB has to take care that it handles the error generated by the drive internally. With another implementation, the MaxVelocity value could be stored locally. In this case the FB will generate the error locally.

2.2.1. Centralized versus Decentralized

Both centralized and decentralized error handling methods are possible when using the Motion Control Function Blocks.

Centralized error handling is used to simplify programming of the Function Block. Error-reaction is the same independent of the instance in which the error has occurred.

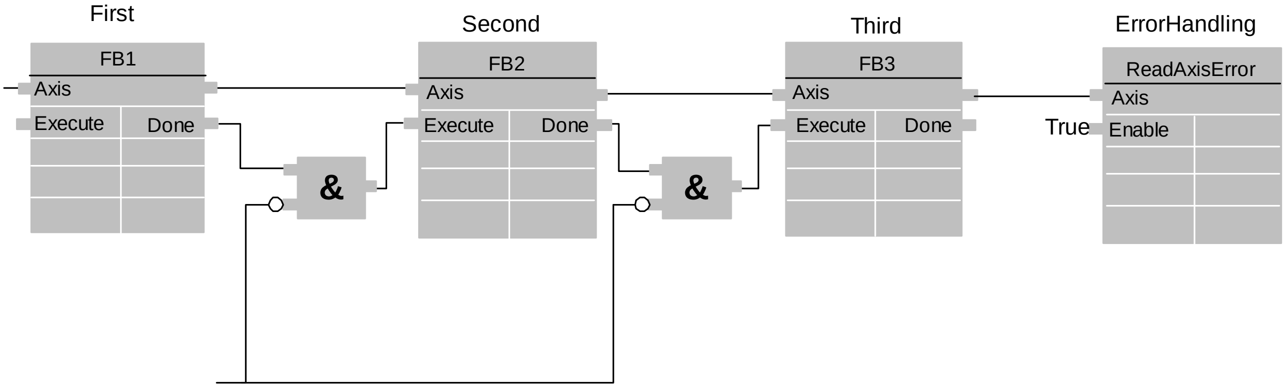


Figure 3: Function Blocks with centralized error handling

Decentralized error handling gives the possibility of different reactions depending on the Function Block in which an error occurred.

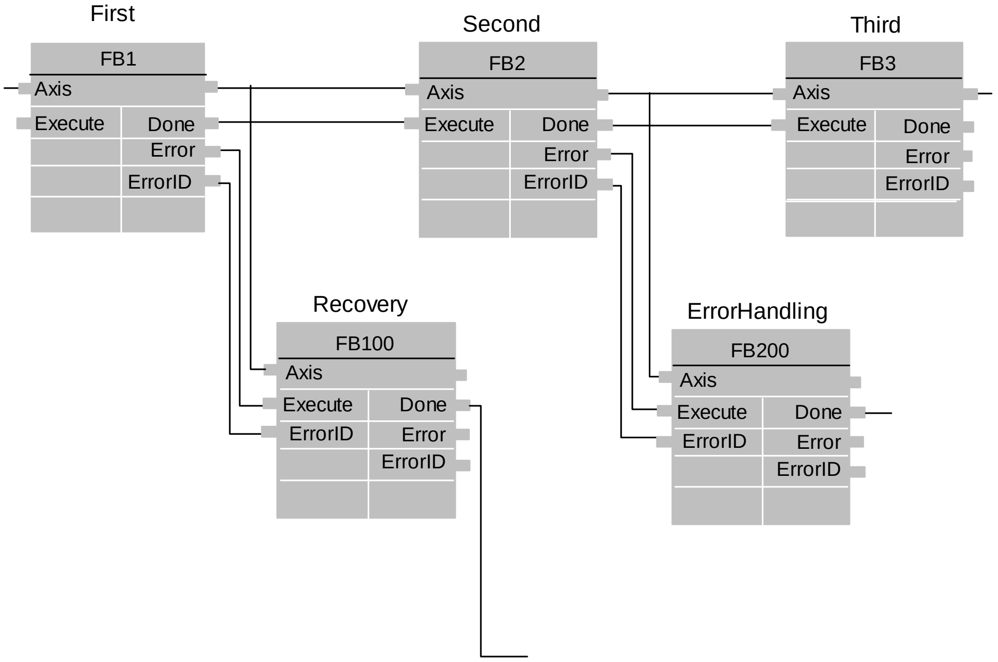


Figure 4: Function blocks with decentralized error handling

2.2.2. Buffered Commands

All buffered commands will be aborted if the applicable axis moves to the state 'ErrorStop'. The 'Error' output of applicable aborted FBs are SET. Any subsequent commands will be rejected and the error output is SET (action not allowed – see state diagram)
If a FB has an error (for instance due to a wrong set of parameters) the error output is set, and the behavior is depending on the application program. For instance, with two FBs, the first FB instance FB1 executes any motion command on an axis. Start a new command on a second FB instance FB2 in buffered mode on the same axis. This command is buffered and waits until FB1 is done. Before the first instance FB1 has finished its command, let one of the following situations occur:

1. The axis goes to state 'ErrorStop' (e.g. due to a following error or over-temperature). FB1 sets the output 'Error'. FB2 (as well as any other FB instance that is waiting to execute a buffered command on this axis) sets its 'Error' output and shows with the output 'ErrorID', that it cannot execute its job, because the axis is in a state that doesn't allow it. All buffered commands are cleared. After the axis error is reset by MC_Reset, it can be commanded again.

2. The FB1 sets its 'Error' output (e.g. due to an invalid parameterization). FB2 becomes active and executes the given command immediately afterwards, and the application should handle the error situation.

2.2.3. Timing example for the 'Enable' input

Example 1: On the left side of the picture the normal operation is shown. On the right side during the operation an error occurs. This error forces the 'Valid' output to be reset. The output 'Busy' stays high. After the error has been reset, the normal operation procedure is restored, possibly after some time.

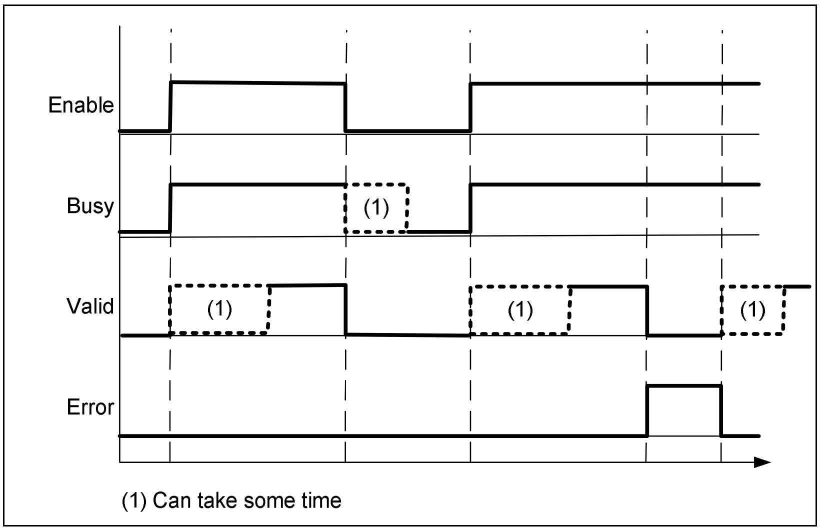


Figure 5: Example of error handling with 'Enable' input

The second example shows on the right side an error that cannot be automatically cleared. The outputs 'Busy' and 'Valid' are reset after the error is set. The FB needs a rising edge on the 'Enable' input to continue.

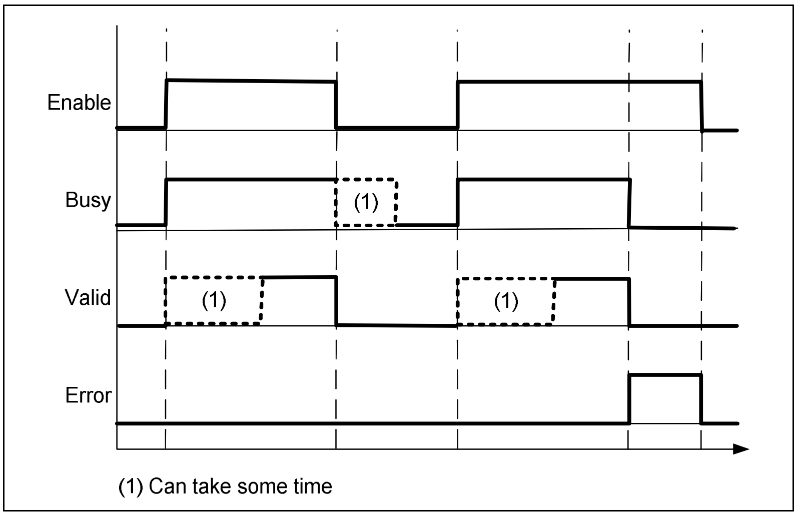


Figure 6: Second example of an error behavior with an 'Enable' input

2.3. Definitions

Within this document the following levels of values are used: Commanded/ Set/ Actual:

- *Commanded value* – is the value that is based on the inputs of the function blocks and can be used as (one of the) input to the profile generator.
- *Set value* – is at a 'lower' level, closer to the actuator. It is the latest value (generated by the profile generator) that is about to be send to the servo loop (e.g. actuator), e.g. the next value the actuator will use.
- *Actual value* – the latest value that is available in the system from the feedback system

2.4. FB interface

2.4.1. General rules

Input parameters	*With 'Execute' without 'ContinuousUpdate'*: The parameters are used with the rising edge of the 'Execute' input. To modify any parameter it is necessary to change the input parameter(s) and to trigger the 'Execute' input again. *With 'Execute' combined with 'ContinuousUpdate'*: The parameters are used with the rising edge of the 'Execute' input. The parameters can be modified continuously as long as the 'ContinuousUpdate' is SET. *With 'Enable'*: The parameters are used with the rising edge of the enable input and can be modified continuously.
Inputs exceeding application limits	If a FB is commanded with parameters which result in a violation of application limits, the inputs are limited by the system or the instance of the FB generates an error. The consequences of this error for the axis are application specific and thus should be handled by the application program.
Missing input parameters	According to IEC 61131-3, if any parameter of a function block input is missing ("open") then the value from the previous invocation of this instance will be used. In the first invocation the initial value is applied.
Acceleration, Deceleration and Jerk inputs	If the input 'Deceleration', 'Acceleration' or 'Jerk' is set to 0, the result is implementation dependent. There are several implementations possible, like one goes to the error state, one signals a warning (via a supplier specific output), one inhibits this in the editor, one takes the value as either specified in AxisRef or in the drive itself, or one takes a maximum value. Even if the 0 value input is accepted by the system, please use with caution especially if compatibility is targeted.
Output exclusivity	*With 'Execute'*: The outputs 'Busy', 'Done', 'Error', and 'CommandAborted' are mutually exclusive: only one of them can be TRUE on one FB. If 'Execute' is TRUE, one of these outputs has to be TRUE. Only one of the outputs 'Active', 'Error', 'Done' and 'CommandAborted' is set at the same time, except in MC_Stop where 'Active' and 'Done' can be set both at the same time *With 'Enable'*: The outputs 'Valid' and 'Error' are mutually exclusive: only one of them can be TRUE on one FB.
Output status	*With 'Execute'*: The 'Done', 'Error', 'ErrorID' and 'CommandAborted' outputs are reset with the falling edge of 'Execute' . However the falling edge of 'Execute' does not stop or even influence the execution of the actual FB. It must be guaranteed that the corresponding outputs are set for at least one cycle if the situation occurs, even if execute was reset before the FB completed. If an instance of a FB receives a new execute before it finished (as a series of commands on the same instance), the FB won't return any feedback, like 'Done' or 'CommandAborted', for the previous action. *With 'Enable'*: The 'Valid', 'Enabled', 'Busy', 'Error',and 'ErrorID' outputs are reset with the falling edge of 'Enable' as soon as possible.

Behavior of Done output	The ‘Done’ output is set when the commanded action has been completed successfully. With multiple Function Blocks working on the same axis in a sequence, the following applies: when one movement on an axis is interrupted with another movement on the same axis without having reached the final goal, ‘Done’ of the first FB will not be set.
Behavior of Busy output	*With ‘Execute’:* Every FB can have an output ‘Busy’, reflecting that the FB is not finished and new output values can be expected. ‘Busy’ is SET at the rising edge of ‘Execute’ and RESET when one of the outputs ‘Done’, ‘Aborted’, or ‘Error’ is set. *With ‘Enable’:* Every FB can have an output ‘Busy’, reflecting that the FB is working and new output values can be expected. ‘Busy’ is SET at the rising edge of ‘Enable’ and stays SET as long as the FB is performing any action. It is recommended that the FB should be kept in the active loop of the application program for at least as long as ‘Busy’ is true, because the outputs may still change.
Behavior of InVelocity, InGear, InTorque and InSync	The outputs ‘InVelocity’, ‘InGear’, ‘InTorque’, and ‘InSync’ (from now on referred to as ‘Inxxx’) have a different behavior than the ‘Done’ output. As long as the FB is Active, ‘Inxxx’ is SET when the set value equals the commanded value, and will be RESET when at a later time they are unequal. For example, the InVelocity output is SET when the set velocity is equal to the commanded velocity. This is similar for ‘InGear’, ‘InTorque’, and ‘InSync’ outputs in the applicable FBs. ‘Inxxx’ is updated even if ‘Execute’ is low as long as the FB has control of the axis (‘Active’ and ‘Busy’ are SET). The behavior of ‘Inxxx’ directly after ‘Execute’ is SET again while the condition of ‘Inxxx’ is already met, is implementation specific. ‘Inxxx’ definition does not refer to the actual axis value, but must refer to the internal instantaneous setpoint.
Output ‘Active’	The ‘Active’ output is required on buffered Function Blocks. This output is set at the moment the function block takes control of the motion of the according axis. For un-buffered mode the outputs ‘Active’ and ‘Busy’ can have the same value. For one axis, several Function Blocks might be busy, but only one can be active at a time. Exceptions are FBs that are intended to work in parallel, like MC_MoveSuperimposed and MC_Phasing’s, where more than one FB related to one axis can be active.
Behavior of CommandAborted output	‘CommandAborted’ is set, when a commanded motion is interrupted by another motion command. The reset-behavior of ‘CommandAborted’ is like that of ‘Done’. When ‘CommandAborted’ occurs, the other output-signals such as ‘InVelocity’ are reset.
Enable and Valid	The ‘Enable’ input is coupled to a ‘Valid’ output. ‘Enable’ is level sensitive, and ‘Valid’ shows that a valid set of outputs is available at the FB. The ‘Valid’ output is TRUE as long as a valid output value is available and the ‘Enable’ input is TRUE. The relevant output value can be refreshed as long as the input ‘Enable’ is TRUE. If there is a FB 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 again.
Position versus distance	‘Position’ is a value defined within a coordinate system. ‘Distance’ is a relative measure related to technical units. ‘Distance’ is the difference between two positions.
Sign rules	The ‘Acceleration’, ‘Deceleration’ and ‘Jerk’ are always positive values. ‘Velocity’, ‘Position’ and ‘Distance’ can be both positive and negative.

Error Handling Behavior	All blocks can have two outputs, which deal with errors that can occur while executing that Function Block. These outputs are defined as follow: **Error** Rising edge of 'Error' informs that an error occurred during the execution of the Function Block. **ErrorID** Error identification (Extended parameter) 'Done', 'InVelocity', 'InGear', 'InTorque', and 'InSync' mean successful completion so these signals are logically exclusive to 'Error'. Types of errors: • Function Blocks (e.g. parameters out of range, state machine violation attempted) • Communication • Drive Instance errors do not always result in an axis error (bringing the axis to 'ErrorStop') The error outputs of the relevant FB are reset with falling edge of 'Execute' and 'Enable'. The error outputs at FBs with 'Enable' can be reset during operation (without a reset of 'Enable').
FB Naming	In case of multiple libraries within one system (to support multiple drive / motion control systems), the FB naming may be changed to "MC_FBname_SupplierID".
Naming conventions ENUM types	Due to the naming constraints in the IEC standard on the uniqueness of variable names, the 'mc' reference to the PLCopen Motion Control namespace is used for the ENUMs. In this way we avoid the conflict that using the ENUM types 'positive' and 'negative' for instance with variables with these names throughout the rest of the project since they are called mcPositive and mcNegative resp.

Table 2: General Rules

The behavior of the 'Execute' / 'Done' style FBs is as follows:

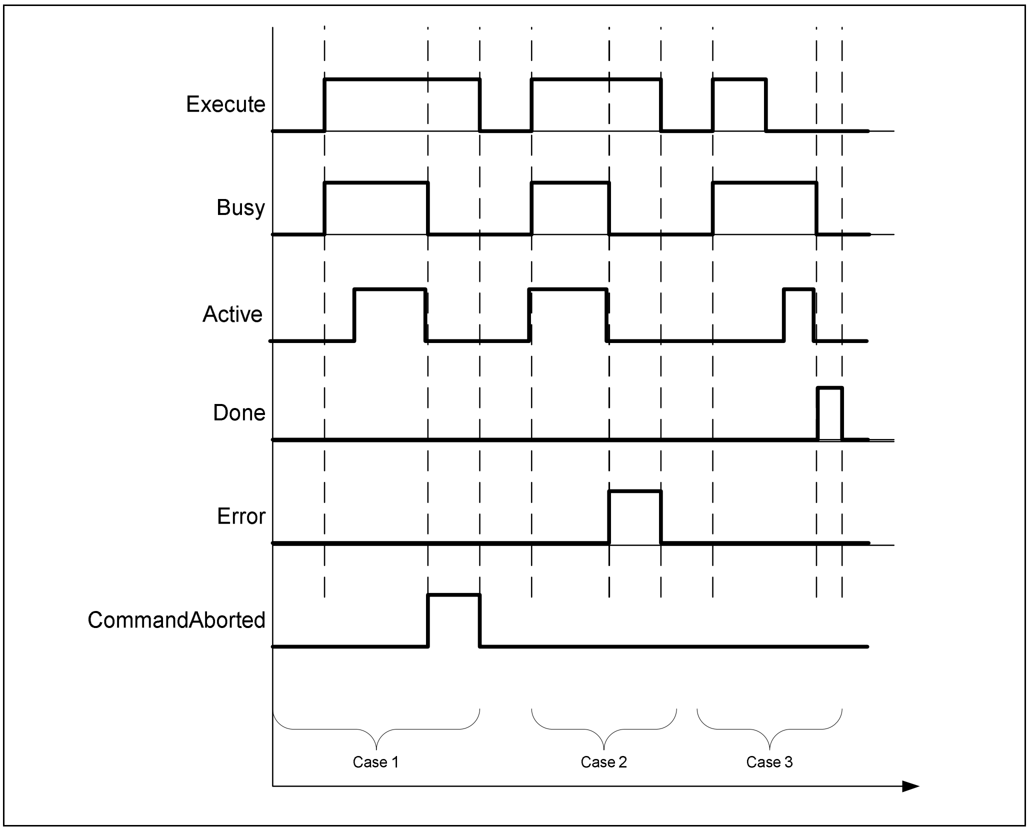


Figure 7: The behavior of the 'Execute' / 'Done' in relevant FBs

The behavior of the ‘Execute’ / ‘Inxxx’ style FBs is as follows:

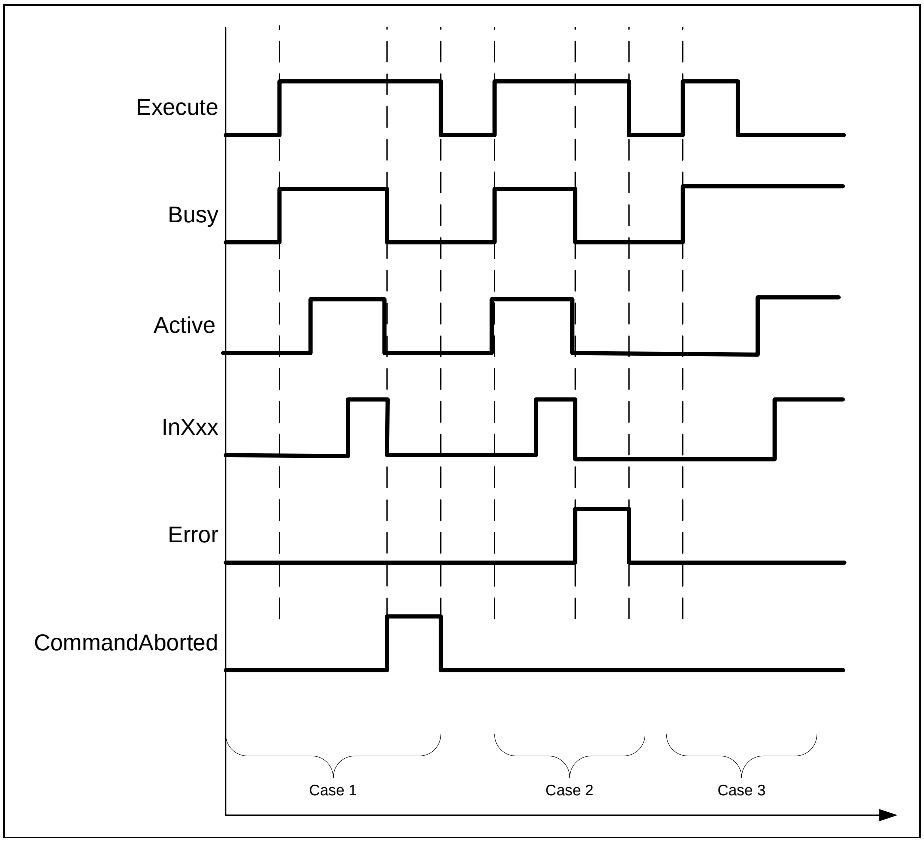


Figure 8: The behavior of the ‘Execute’ / ‘Inxxx’ in relevant FBs

2.4.2. Aborting versus Buffered modes

Some of the FBs have an input called ‘BufferMode’. With this input, the FB can either work in a ‘Non-buffered mode’ (default behavior) or in a ‘Buffered mode’. The difference between those modes is when they should start their action:

- A command in a non-buffered mode acts immediately, even if this interrupts another motion. The buffer is cleared.
- A command in a buffered mode waits till the current FB sets its ‘Done’ output (or ‘InPosition’, or ‘InVelocity’,..).

There are several options for the buffered mode. For this reason, this input is an ENUM of type MC_BUFFER_MODE. The following modes have been identified:

- Aborting — Default mode without buffering. The next FB aborts an ongoing motion and the command affects the axis immediately. The buffer is cleared.
- Buffered — The next FB affects the axis as soon as the previous movement is ‘Done’. There is no blending.
- BlendingLow — The next FB controls the axis after the previous FB has finished (equivalent to ‘Buffered’), but the axis will not stop between the movements. The velocity is blended with the lowest velocity of both commands (1 and 2) at the first end-position (1).
- BlendingPrevious — blending with the velocity of FB 1 at end-position of FB 1
- BlendingNext — blending with velocity of FB 2 at end-position of FB 1
- BlendingHigh — blending with highest velocity of FB 1 and FB 2 at end-position of FB1

The ENUM has been defined as follows:

No.	MC_BUFFER_MODE	Description
0	mcAborting	Start FB immediately (default mode)
1	mcBuffered	Start FB after current motion has finished
2	mcBlendingLow	The velocity is blended with the lowest velocity of both FBs
3	mcBlendingPrevious	The velocity is blended with the velocity of the first FB
4	mcBlendingNext	The velocity is blended with velocity of the second FB
5	mcBlendingHigh	The velocity is blended with highest velocity of both FBs

Table 3: The ENUM type MC_BUFFER_MODE

Supplier specific extensions are allowed after these defined Enums.

The examples as listed in Appendix A describe the different behavior of these modes.
The following table gives an overview of the effects on the defined function blocks:

Function block	**Can be specified as a buffered command**	**Can be followed by a buffered command**	**Relevant signal to activate the next buffered FB**
MC_Power	No	Yes	Status
MC_Home	Yes	Yes	Done
MC_Stop	No	Yes	Done AND NOT Execute
MC_Halt	Yes	Yes	Done
MC_MoveAbsolute	Yes	Yes	Done
MC_MoveRelative	Yes	Yes	Done
MC_MoveAdditive	Yes	Yes	Done
MC_MoveSuperimposed	No	No	--
MC_HaltSuperimposed	No	No	--
MC_MoveVelocity	Yes	Yes	InVelocity
MC_MoveContinuousAbsolute & MC_MoveContinuousRelative	Yes	Yes	InEndVelocity
MC_TorqueControl	Yes	Yes	InTorque
MC_PositionProfile	Yes	Yes	Done
MC_VelocityProfile	Yes	Yes	Done
MC_AccelerationProfile	Yes	Yes	Done
MC_CamIn	Yes	Yes - (in single mode)	EndOfProfile
MC_CamOut	No	Yes	Done
MC_GearIn	Yes	Yes	InGear
MC_GearOut	No	Yes	Done
MC_GearInPos	Yes	Yes	InSync
MC_PhasingRelative & MC_PhasingAbsolute	Yes	No	--
MC_CombineAxes	Yes	Yes	InSync

Table 4: Overview of the buffered commands on the relevant FBs

Note: The (administrative) FBs not listed here are basically not buffered, nor can be followed by a buffered FB. However, the supplier may choose to support the various buffering / blending modes.
If an on-going motion is aborted by another movement, it can occur that the braking distance is not sufficient due to deceleration limits.
In rotary axis, a modulo can be added. A modulo axis could go to the earliest repetition of the absolute position specified, in cases where the axis should not change direction and reverse to attain the commanded position.
In linear systems, the resulting overshoot can be resolved by reversing, as each position is unique and therefore there is no need to add a modulo to reach the correct position.

2.4.3. AXIS_REF Data type

The AXIS_REF is a structure that contains information on the corresponding axis. It is used as a VAR_IN_OUT in all Motion Control Function Blocks defined in this document. The content of this structure is implementation dependent and can ultimately be empty. If there are elements in this structure, the supplier shall support the access to them, but this is outside of the scope of this document. The refresh rate of this structure is also implementation dependent.
According to IEC 61131-3 it is allowed to switch the AXIS_REF for an active FB, for instance from Axis1 to Axis2. However, the behavior of this can vary across different platforms, and is not encouraged to do.

AXIS_REF data type declaration:

```
TYPE
   AXIS_REF : STRUCT
      (Content is implementation dependent)
    END_STRUCT
END_TYPE
```

Example:

```
TYPE
   AXIS_REF : STRUCT
      AxisNo: UINT;
      AxisName: STRING (255);
      …
   END_STRUCT
END_TYPE
```

2.4.4. Technical Units

The only specification for physical quantities is made on the length unit (noted as [u]) that is to be coherent with its derivatives i.e. (velocity [u/s]; acceleration [u/s^2]; jerk [u/s^3]). Nevertheless, the unit [u] is not specified (manufacturer dependent). Only its relations with others are specified.

2.4.5. Why the command input is edge sensitive

The ‘Execute’ input for the different Function Blocks described in this document always triggers the function with its rising edge. The reason for this is that with edge triggered ‘Execute’ new input values may be commanded during execution of a previous command. The advantage of this method is a precise management of the instant a motion command is performed. Combining different Function Blocks is then easier in both centralized and decentralized models of axis management. The ‘Done’ output can be used to trigger the next part of the movement.
The example given below is intended to explain the behavior of the Function Block execution. The figure illustrates the sequence of three Function Blocks “First”, “Second” and “Third” controlling the same axis. These three Function Blocks could be for instance various absolute or relative move commands. When “First” is completed the motion its rising output ‘First.Done’ triggers ‘Second.Execute’. The output ‘Second.Done’ AND ‘In13’ trigger the ‘Third.Execute’.

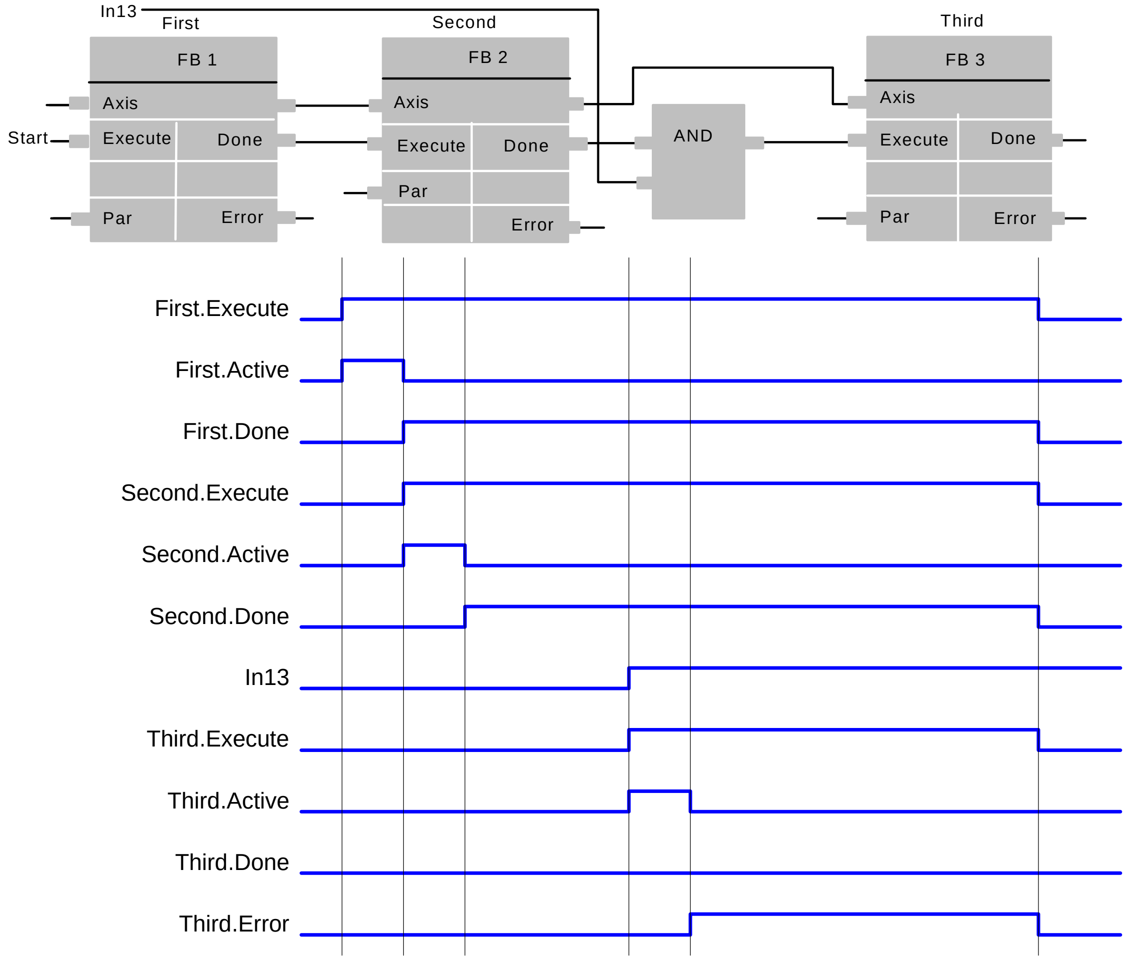


Figure 9: Function blocks to perform a complex movement

2.4.6. The input ‘ContinuousUpdate’

Like described in the previous chapter, the input ‘Execute’ triggers a new movement. With a rising edge of this input the values of the other function block inputs are defining the movement. Until version 1.1 there was the general rule that a later change in these input parameters doesn’t affect the ongoing motion.
Nevertheless, there are numerous application examples, where a continuous change of the parameters are needed. The user could retrigger the ‘Execute’ input of the FB, but this complicated the application.

Therefore, the input ‘ContinuousUpdate’ has been introduced. It is an extended input to all applicable function blocks. If it is TRUE, when the function block is triggered (rising ‘Execute’), it will - as long as it stays TRUE – make the function block use the current values of the input variables and apply it to the ongoing movement. This does not influence the general behavior of the function block nor does it impact the state diagram. In other words it only influences the ongoing movement and its impact ends as soon as the function block is no longer ‘Busy’ or the input ‘ContinuousUpdate' is set to FALSE. (Remark: it can be that certain inputs like 'BufferMode' are not really intended to change every cycle. However, this has to be dealt with in the application, and is not forbidden in the specification.)

If 'ContinuousUpdate' is FALSE with the rising edge of the 'Execute' input, a change in the input parameters is ignored during the whole movement and the original behavior of previous versions is applicable.

The 'ContinuousUpdate' is not a retriggering of the 'Execute' input of the function block. A retriggering of a function block which was previously aborted, stopped, or completed, would regain control on the axis and also modify its state diagram. Opposite to this, the 'ContinuousUpdate' only effects an ongoing movement.
Also, a 'ContinuousUpdate' of relative inputs (e.g. 'Distance' in MC_MoveRelative) always refers to the initial condition (at rising edge of 'Execute').
Example:

- MC_MoveRelative is started at 'Position' 0 with 'Distance' 100, 'Velocity' 10 and 'ContinuousUpdate' set TRUE. 'Execute' is Set and so the movement is started to position 100
- While the movement is executed (let the drive be at position 50), the input 'Distance' is changed to 130, 'Velocity' 20.
- The axis will accelerate (to the new 'Velocity' 20) and stop at 'Position' 130 and set the output 'Done' and does not accept any new values.

2.5. Example 1: the same Function Block instance controls different motions of an axis

Figure 10: Single FB usage with a SFC shows an example where the Function Block FB1 is used to control "AxisX" with three different values of 'Velocity'. In a Sequential Function Chart (SFC) the 'Velocity' 10, 20, and 0 is assigned to V. To trigger the 'Execute' input with a rising edge the variable E is stepwise set and reset.

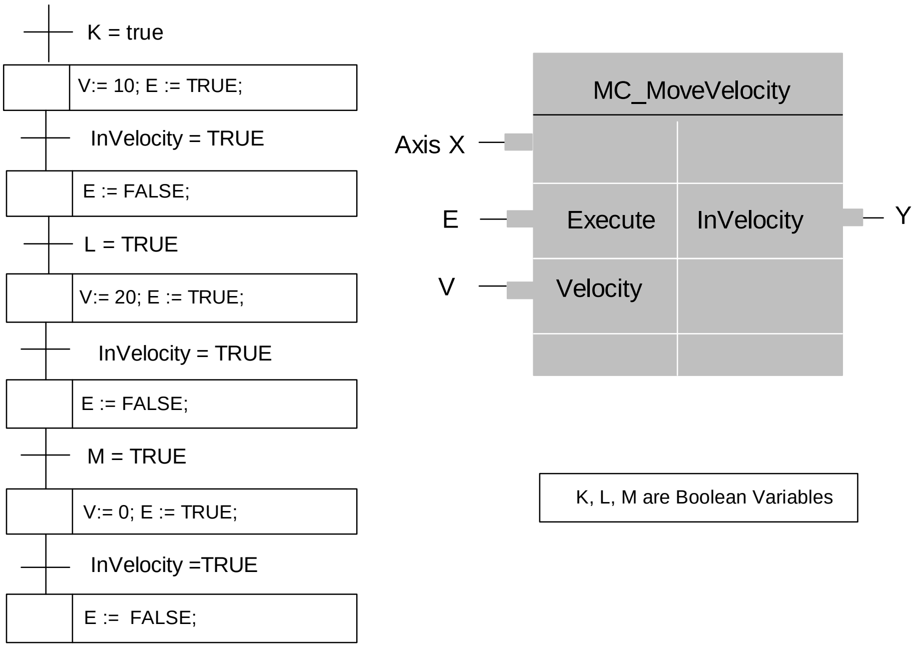


Figure 10: Single FB usage with a SFC

The following timing diagram explains how it works.

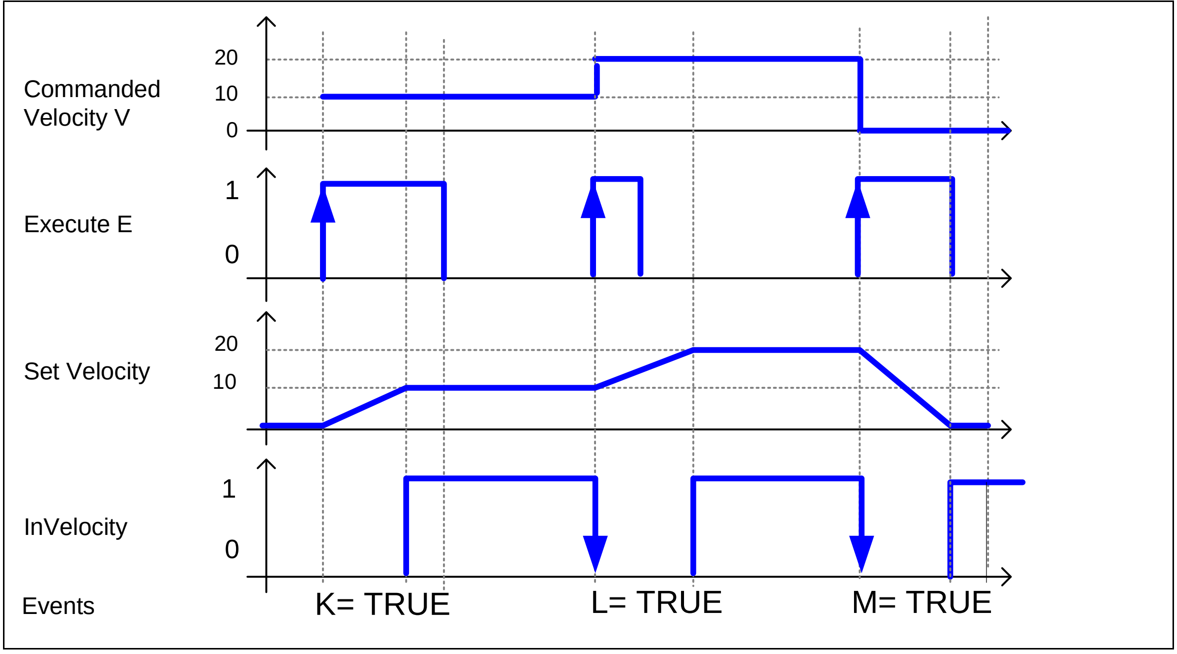


Figure 11: Timing diagram for a usage of a single FB

Note: if the execute input is retriggered with the same commanded velocity while 'InVelocity' is SET, the behavior of the output 'InVelocity' is implementation dependent (for instance: reset for one cycle or not reset at all)

2.6. Example 2: different Function Block instances control the motions of an axis

Different instances related to the same axis can control the motions on an axis. Each instance will then be «responsible» for one part of the global profile.

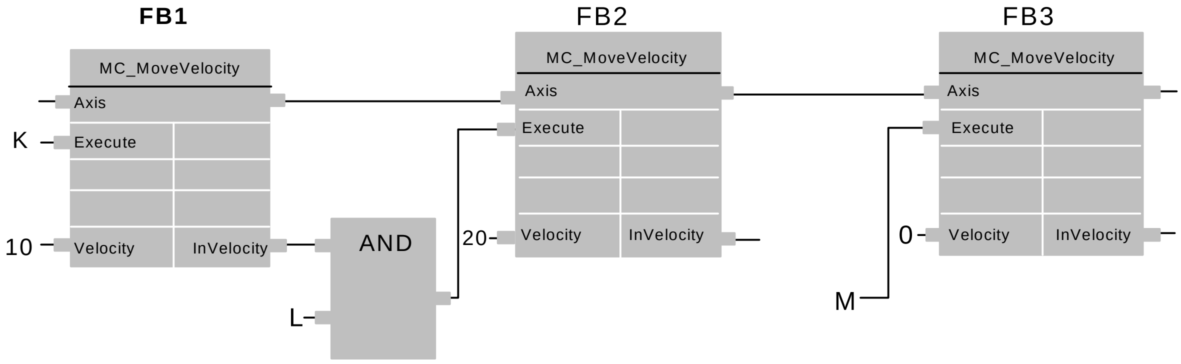


Figure 12: Example of cascaded Function Blocks

The corresponding timing diagram:

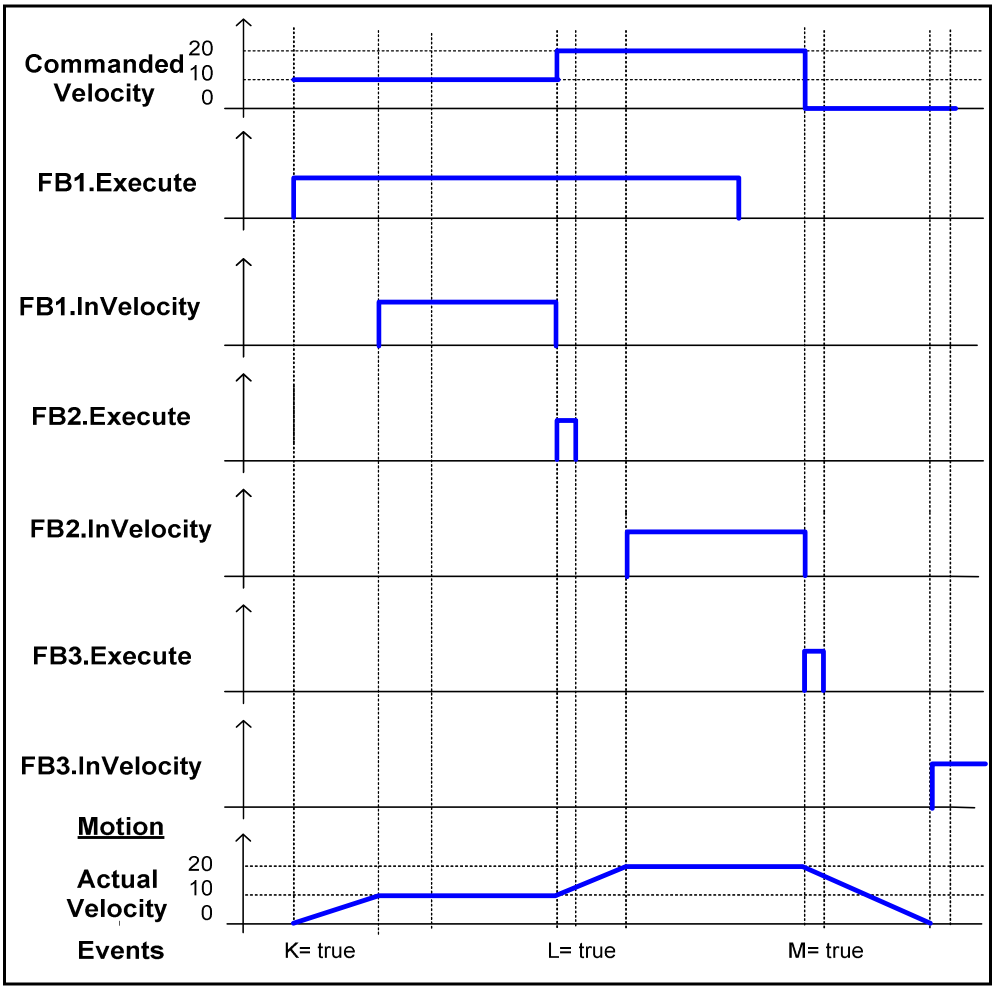


Figure 13: Timing diagram of example cascaded Function Blocks

A corresponding solution written in LD can look like this:

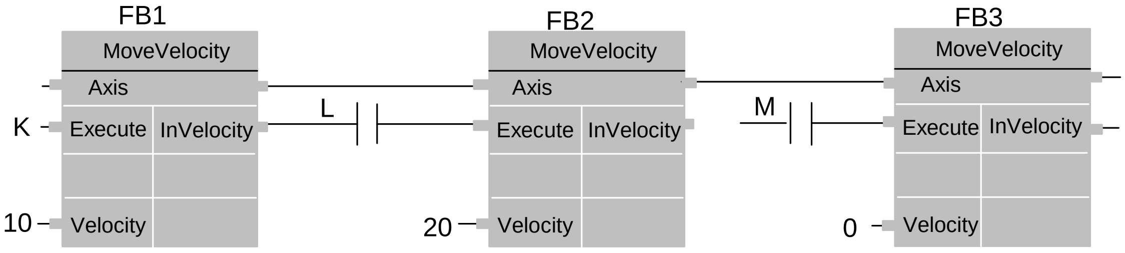
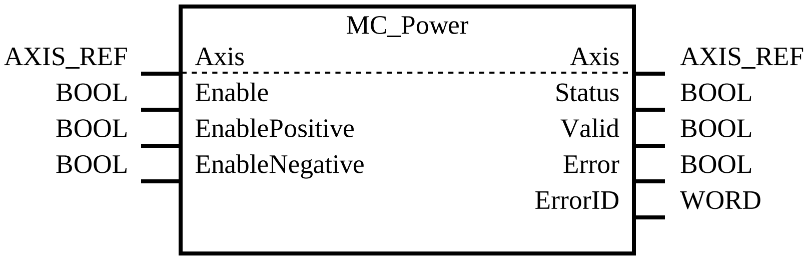


Figure 14: Example of cascaded Function Blocks with LD

3. Single-Axis Function Blocks

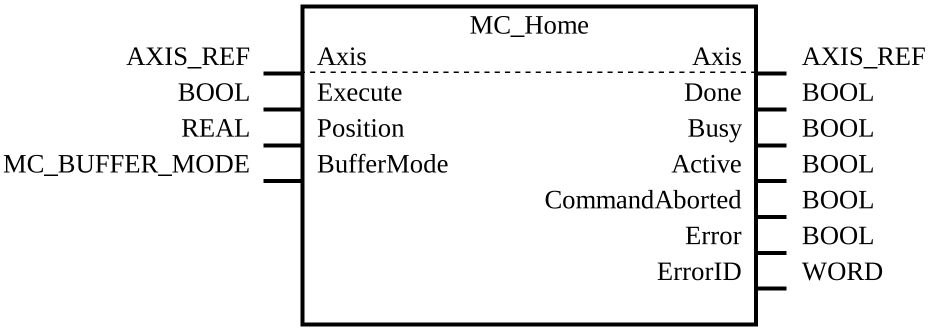
3.1. MC_Power

FB-Name		**MC_Power**		
This Function Block controls the power stage (On or Off).				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	As long as ‘Enable’ is true, power is being enabled.
	E	EnablePositive	BOOL	As long as ‘Enable’ is true, this permits motion in positive direction
	E	EnableNegative	BOOL	As long as ‘Enable’ is true, this permits motion in negative direction
VAR_OUTPUT				
	B	Status	BOOL	Effective state of the power stage
	E	Valid	BOOL	If true, a valid set of outputs is available at the FB
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
Notes: • The ‘Enable’ input enables the power stage in the drive and not the FB itself • If the MC_Power FB is called with the ‘Enable’ = TRUE while being in ‘Disabled’, the axis state changes to ‘Standstill’. • It is possible to set an error variable when the Command is TRUE for a while and the Status remains false with a Timer FB and an AND Function (with inverted Status input). It indicates that there is a hardware problem with the power stage. • If power fails (also during operation) it will generate a transition to the ‘ErrorStop’ state. • ‘EnablePositive’ and ‘EnableNegative’ are both level sensitive. • ‘EnablePositive’ & ‘EnableNegative’ can both be true. • Only 1 FB MC_Power should be issued per axis.				



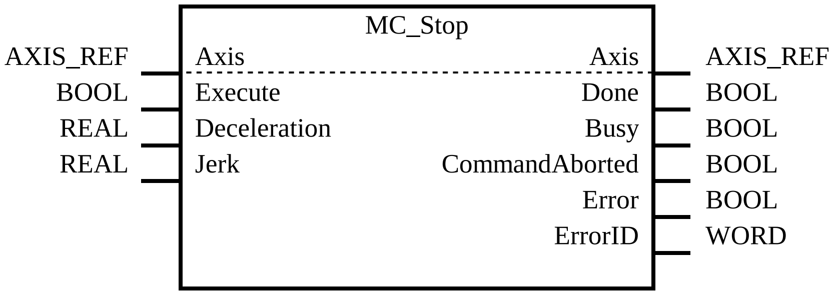
3.2. MC_Home

FB-Name			**MC_Home**	
This Function Block commands the axis to perform the «search home» sequence. The details of this sequence are manufacturer dependent and can be set by the axis' parameters. The 'Position' input is used to set the absolute position when reference signal is detected. This Function Block completes at 'Standstill' if it was started in 'Standstill'.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	BOOL	Start the motion at rising edge
	B	Position	REAL	Absolute position when the reference signal is detected [u]
	E	BufferMode	MC_BufferMode	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT				
	B	Done	BOOL	Reference known and set sucessfully
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	E	Active	BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	BOOL	'Command' is aborted by another command
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
Notes: MC_Home is a generic FB which does a system specified homing procedure which can be constructed by the StepHoming FBs as specified in Part 5 – Homing Procedures.				



3.3. MC_Stop

FB-Name	**MC_Stop**		
This Function Block commands a controlled motion stop and transfers the axis to the state 'Stopping'. It aborts any ongoing Function Block execution. While the axis is in state 'Stopping', no other FB can perform any motion on the same axis. After the axis has reached 'Velocity' zero, the 'Done' output is set to TRUE immediately. The axis remains in the state 'Stopping' as long as 'Execute' is still TRUE or 'Velocity' zero is not yet reached. As soon as 'Done' is SET and 'Execute' is FALSE the axis goes to state 'Standstill'.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the action at rising edge
E	Deceleration	REAL	Value of the 'Deceleration' [u/s^2]
E	Jerk	REAL	Value of the 'Jerk' [u/s^3]
VAR_OUTPUT			
B	Done	BOOL	Zero velocity reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	CommandAborted	BOOL	'Command' is aborted by switching off power (only possibility to abort)
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Note: 1. This FB is primarily intended for emergency stop functionality or exception situations 2. As long as 'Execute' is high, the axis remains in the state 'Stopping' and may not be executing any other motion command. 3. If 'Deceleration' = 0, the behavior of the function block is implementation specific			



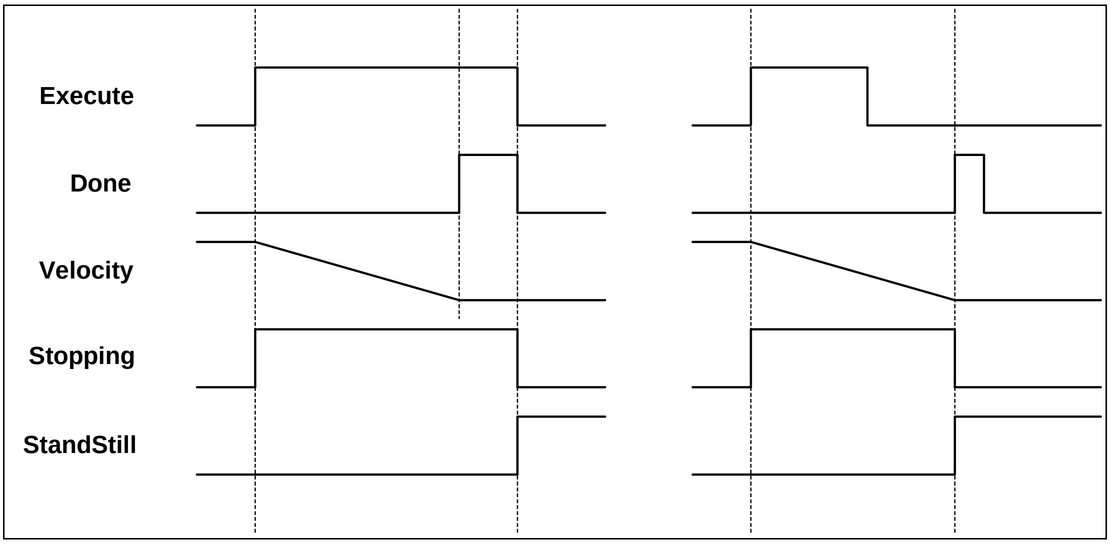


Figure 15: MC_Stop timing diagram

The example below shows the behavior in combination with a MC_MoveVelocity.
a) A rotating axis is ramped down with FB MC_Stop.
b) The axis rejects motion commands as long as MC_Stop parameter ‘Execute’ = TRUE. FB MC_MoveVelocity reports an error indicating the busy MC_Stop command.

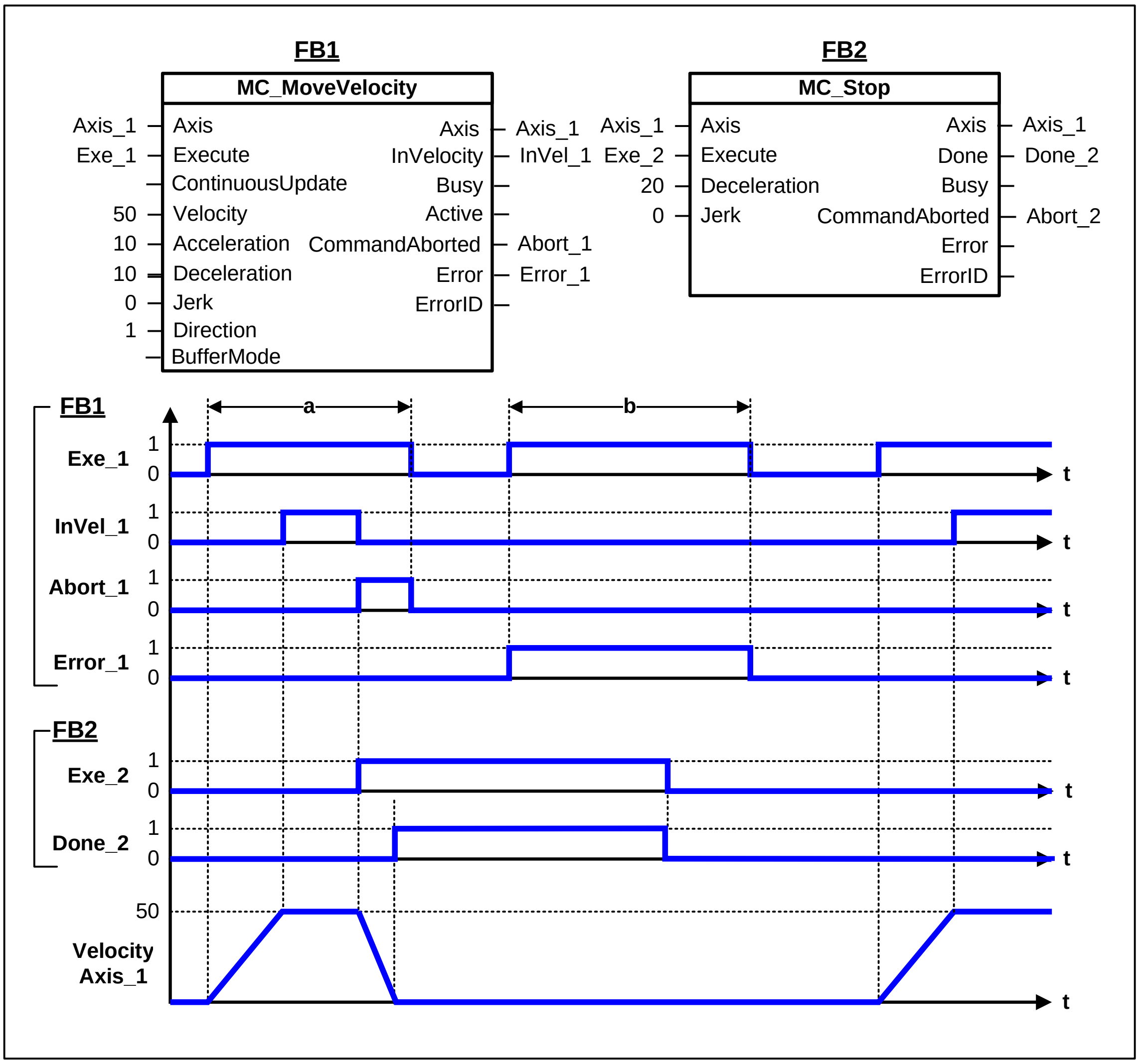
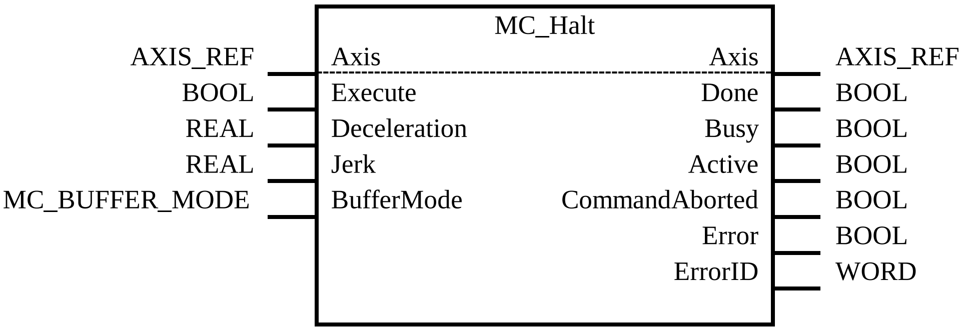


Figure 16: Behavior of MC_Stop in combination with MC_MoveVelocity

3.4. MC_Halt

FB-Name		**MC_Halt**	
This Function Block commands a controlled motion stop. The axis is moved to the state 'DiscreteMotion', until the velocity is zero. With the 'Done' output set, the state is transferred to 'Standstill'.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the action at rising edge
E	Deceleration	REAL	Value of the 'Deceleration' [u/s^2]
E	Jerk	REAL	Value of the 'Jerk' [u/s^3]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	Done	BOOL	Zero velocity reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: • MC_Halt is used to stop the axis under normal operation conditions. In non-buffered mode it is possible to set another motion command during deceleration of the axis, which will abort the MC_Halt and will be executed immediately. • If this command is active the next command can be issued. E.g. a driverless vehicle detects an obstacle and needs to stop. MC_Halt is issued. Before the 'Standstill' is reached the obstacle is removed and the motion can be continued by setting another motion command, so the vehicle does not stop.			



The example below shows the behavior in combination with a MC_MoveVelocity.

a) A rotating axis is ramped down with Function Block MC_Halt

b) Another motion command overrides the MC_Halt command. MC_Halt allows this, in contrast to MC_Stop. The axis can accelerate again without reaching 'Standstill'.

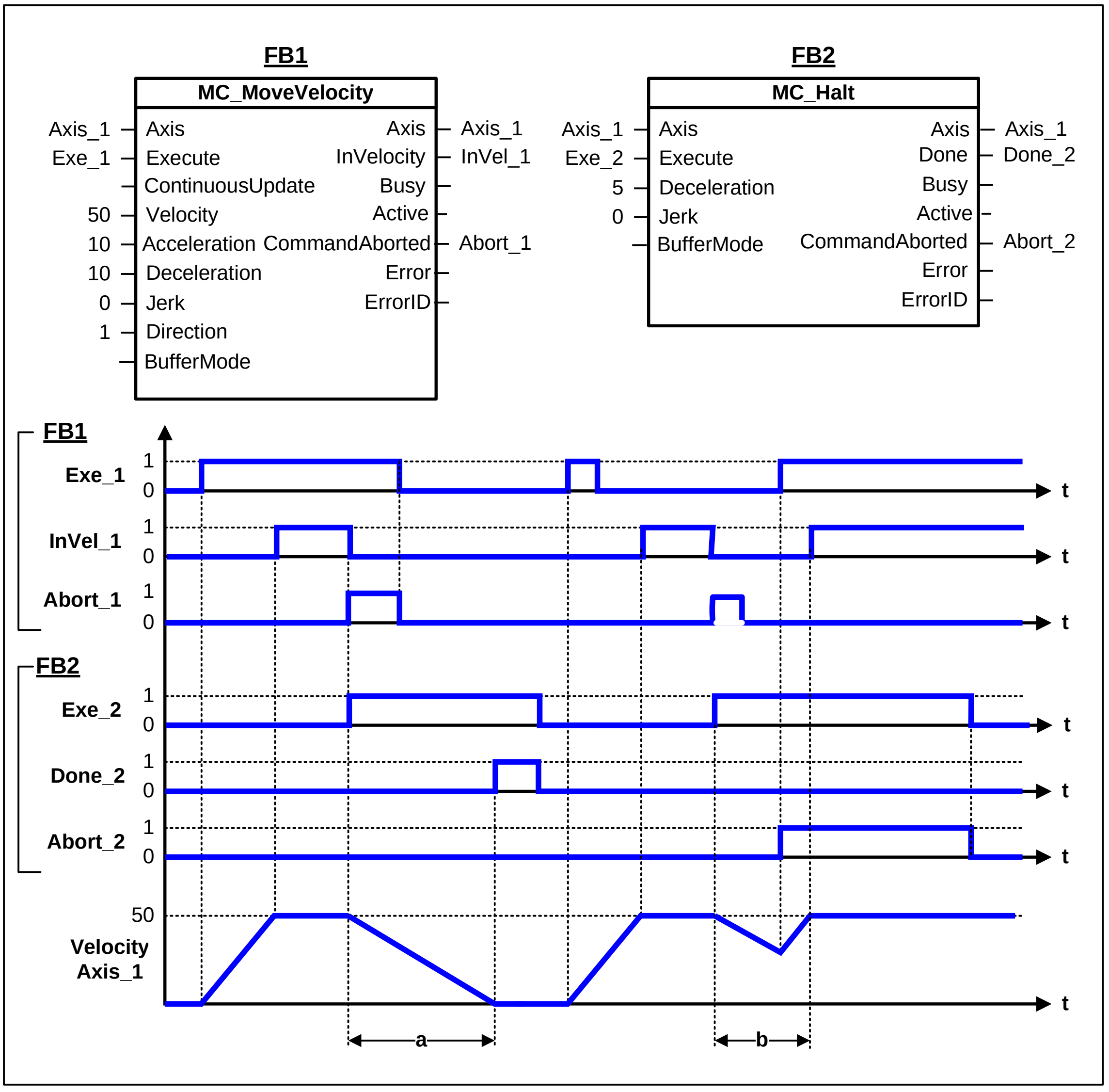


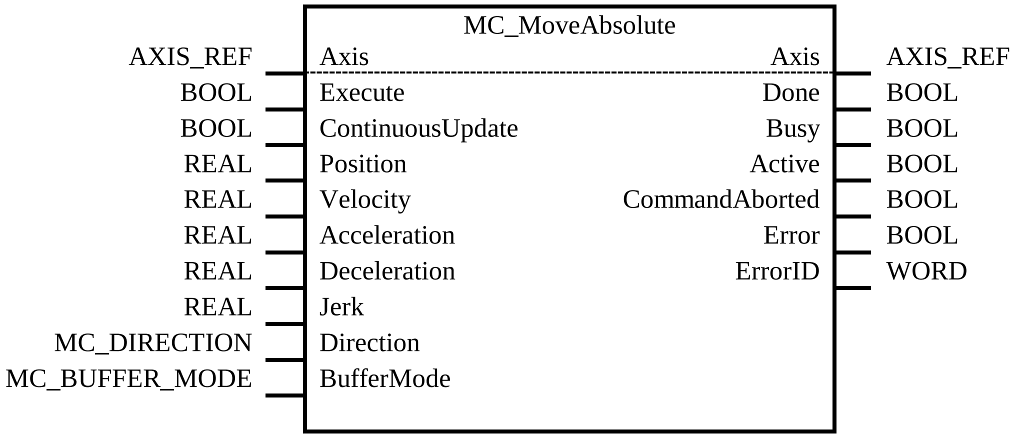
Figure 17: Example of MC_Halt

3.5. MC_MoveAbsolute

FB-Name		**MC_MoveAbsolute**	
This Function Block commands a controlled motion to a specified absolute position.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
B	Position	REAL	Commanded ‘Position’ for the motion (in technical unit [u]) (negative or positive)
B	Velocity	REAL	Value of the maximum ‘Velocity’ (not necessarily reached) [u/s].
E	Acceleration	REAL	Value of the ‘Acceleration’ (always positive) (increasing energy of the motor) [u/s^2]
E	Deceleration	REAL	Value of the ‘Deceleration’ (always positive) (decreasing energy of the motor) [u/s^2]
E	Jerk	REAL	Value of the ‘Jerk’ [u/s^3]. (always positive)
B	Direction	MC_DIRECTION	Enum type (1-of-4 values: mcPositiveDirection, mcShortestWay, mcNegativeDirection, mcCurrentDirection)
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 ‘Aborting versus Buffered modes’
VAR_OUTPUT			
B	Done	BOOL	Commanded position finally reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:

- This action completes with velocity zero if no further actions are pending
- If there is only one mathematical solution to reach the ‘CommandedPosition’ (like in linear systems), the value of the input ‘Direction’ is ignored
- For modulo axis - valid absolute position values are in the range of [0, 360[, (360 is excluded), or corresponding range. The application however may shift the ‘CommandedPosition’ of MC_MoveAbsolute into the corresponding modulo range.
- The Enum type ‘mcShortestWay’ is focused to a trajectory which will go through the shortest route. The decision which direction to go is based on the current position where the command is issued.



The following figure shows two examples of the combination of two absolute move Function Blocks:

1. The left part of the timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded position of 6000 (and the velocity is 0) then the output 'Done' causes the Second FB to move to the 'Position' 10000.

2. The right part of the timing diagram illustrates the case if the Second move Function Block starts the execution **while** the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB moves directly to the position 10000 although the position of 6000 is not yet reached.

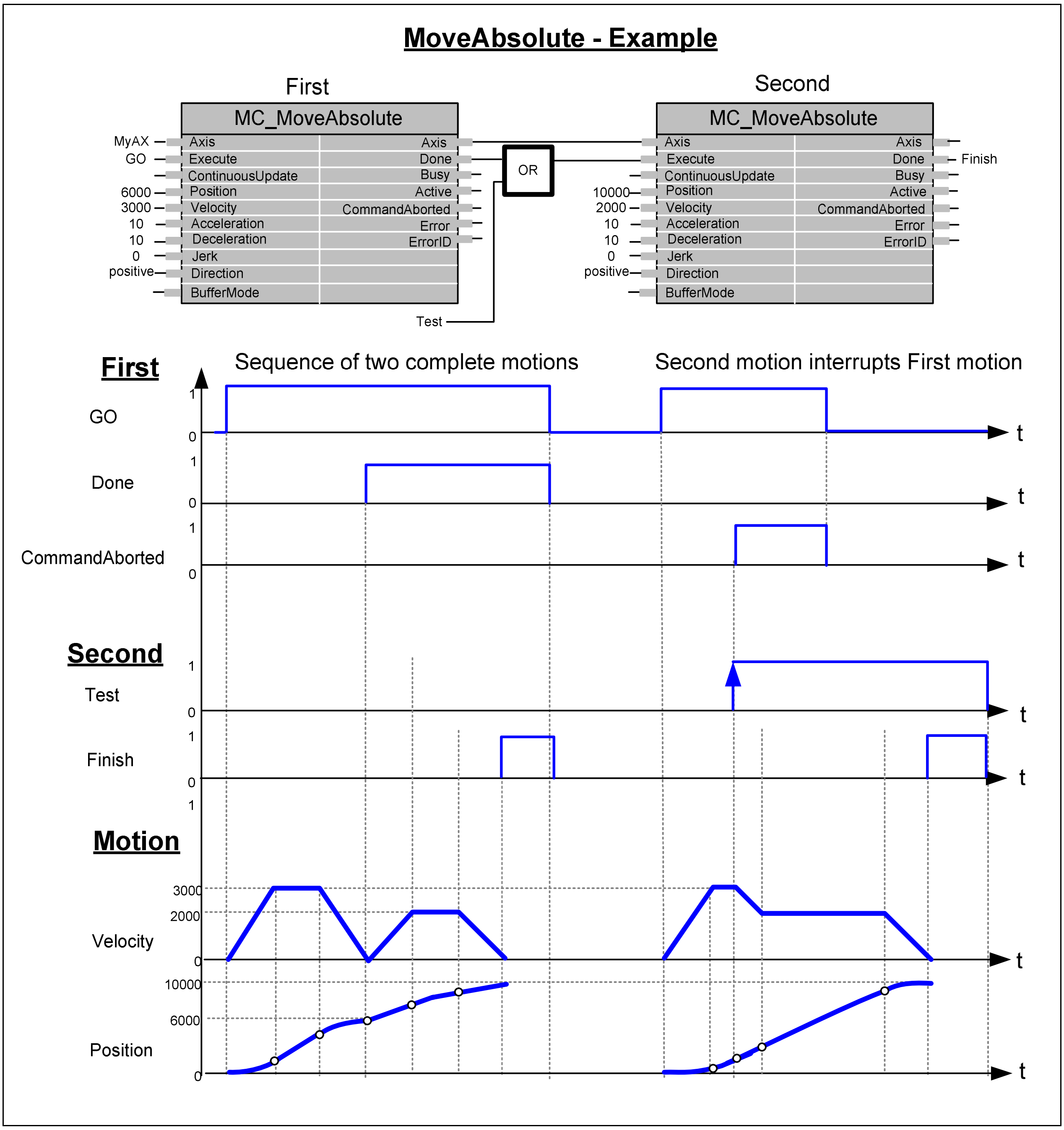
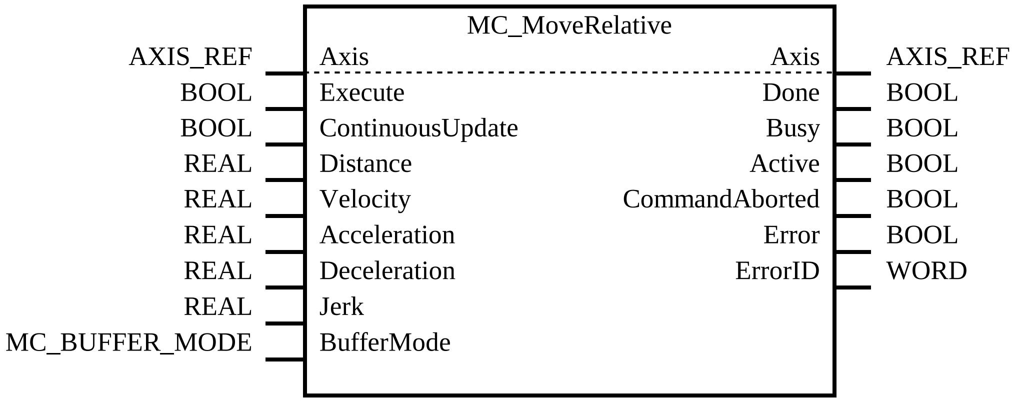


Figure 18: Timing diagram for MC_MoveAbsolute

Note to figure: the examples are based on two instances of the Function Block: instance "First" and "Second".

3.6. MC_MoveRelative

FB-Name		**MC_MoveRelative**	
This Function Block commands a controlled motion of a specified distance relative to the set position at the time of the execution.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
B	Distance	REAL	Relative distance for the motion (in technical unit [u])
E	Velocity	REAL	Value of the maximum velocity (not necessarily reached) [u/s]
E	Acceleration	REAL	Value of the acceleration (increasing energy of the motor) [u/s^2]
E	Deceleration	REAL	Value of the deceleration (decreasing energy of the motor) [u/s^2]
E	Jerk	REAL	Value of the Jerk [u/s^3]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	Done	BOOL	Commanded distance reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: This action completes with velocity zero if no further actions are pending.			



The following figure shows the example of the combination of two relative move Function Blocks

1. The left part of the timing diagram illustrates the case if the Second Function Block is called **after** the First one.
   If First reaches the commanded distance 6000 (and the velocity is 0) then the output ‘Done’ causes the Second FB to move the commanded distance 4000 and moves the axis to the resulting position of 10000.

2. The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution **while** the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB **adds on the actual position** of 3250 the distance 4000 and moves the axis to the resulting position of 7250.

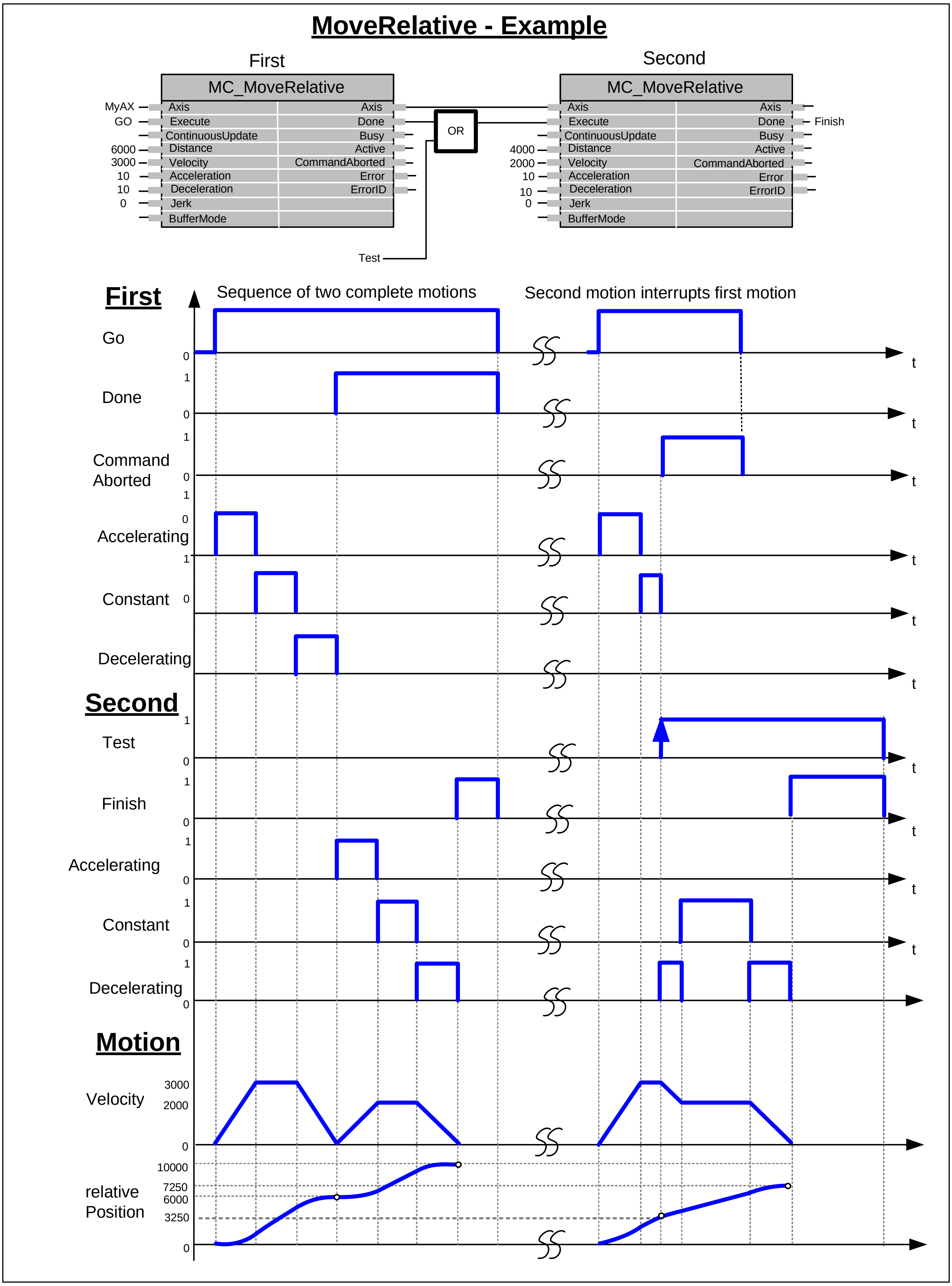
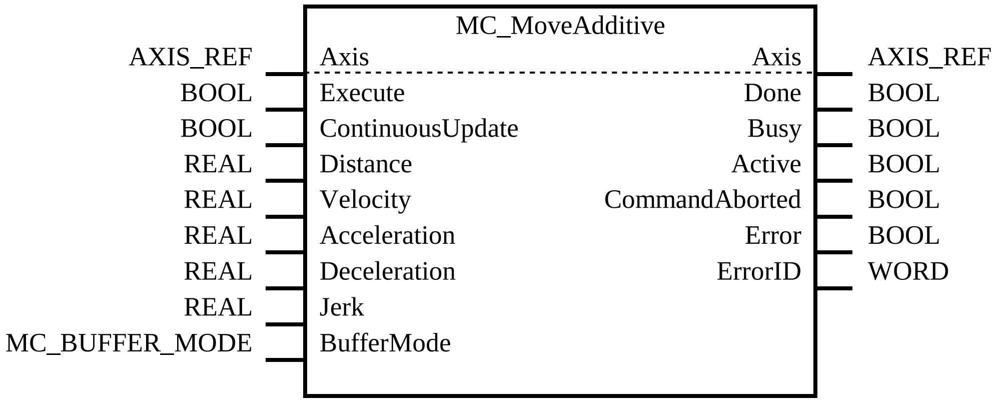


Figure 19: Timing diagram for MC_MoveRelative

3.7. MC_MoveAdditive

FB-Name	**MC_MoveAdditive**		
This Function Block commands a controlled motion of a specified relative distance additional to the most recent commanded position in the axis state 'DiscreteMotion'. The most recent commanded position may be the result of a previous MC_MoveAdditive motion which was aborted. If the FB is activated in the axis state 'ContinuousMotion', the specified relative distance is added to the set position at the time of the execution.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input 'ContinuousUpdate'
B	Distance	REAL	Relative distance for the motion (in technical unit [u])
E	Velocity	REAL	Value of the maximum velocity (not necessarily reached) [u/s]
E	Acceleration	REAL	Value of the acceleration (increasing energy of the motor) [u/s^2]
E	Deceleration	REAL	Value of the deceleration (decreasing energy of the motor) [u/s^2]
E	Jerk	REAL	Value of the Jerk [u/s^3]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	Done	BOOL	Commanded distance reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: -			



The following figure shows two examples of the combination of two Function Blocks while the axis is in 'DiscreteMotion' state:

1. The left part of the timing diagram illustrates the case if the Second Function Block is called **after** the First one.
   If First reaches the commanded distance 6000 (and the velocity is 0) then the output 'Done' causes the Second FB to move the commanded distance 4000 and moves the axis to the resulting position of 10000.

2. The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution **while** the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB **adds on the previous commanded position** of 6000 the distance 4000 and moves the axis to the resulting position of 10000.

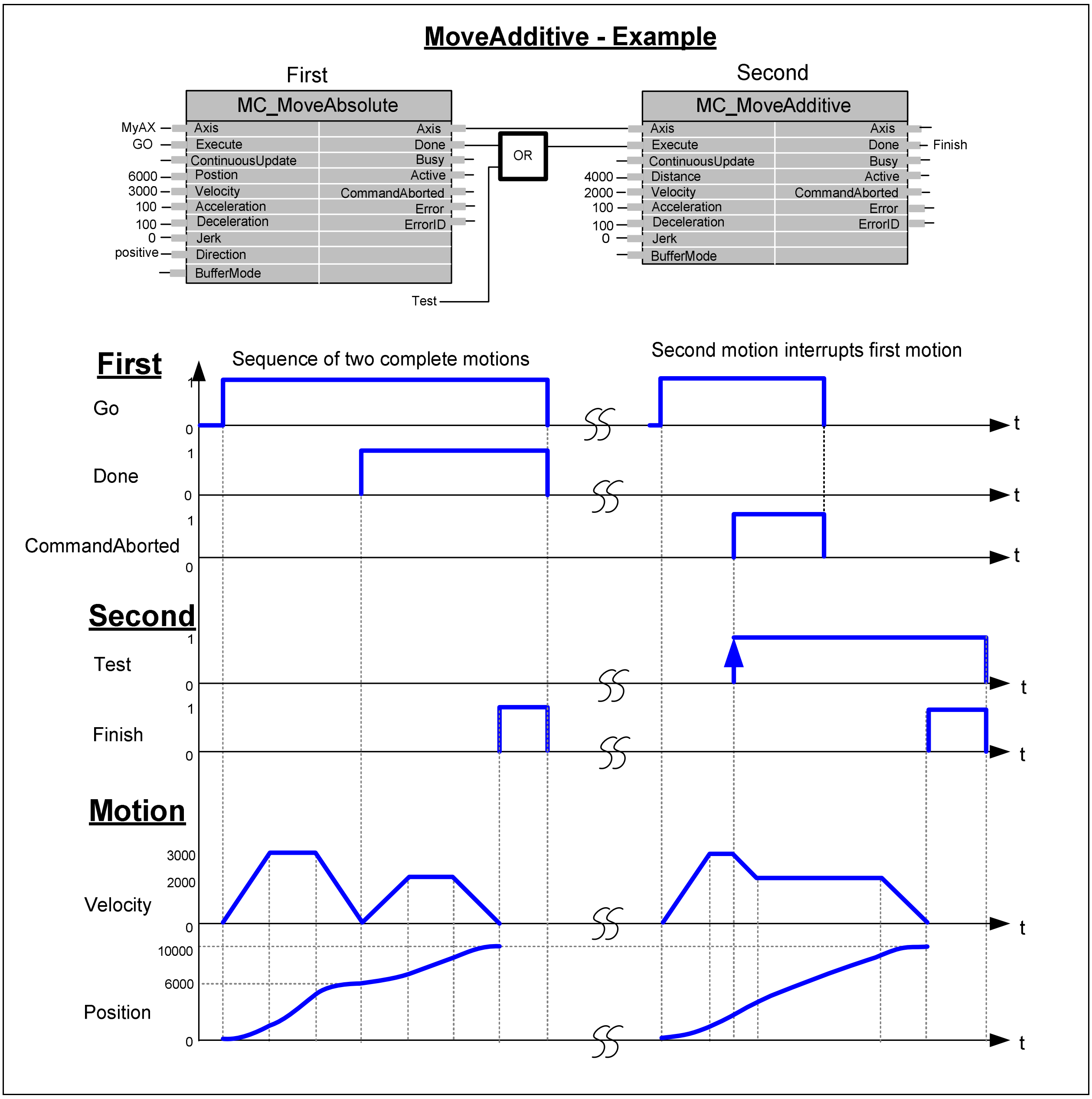


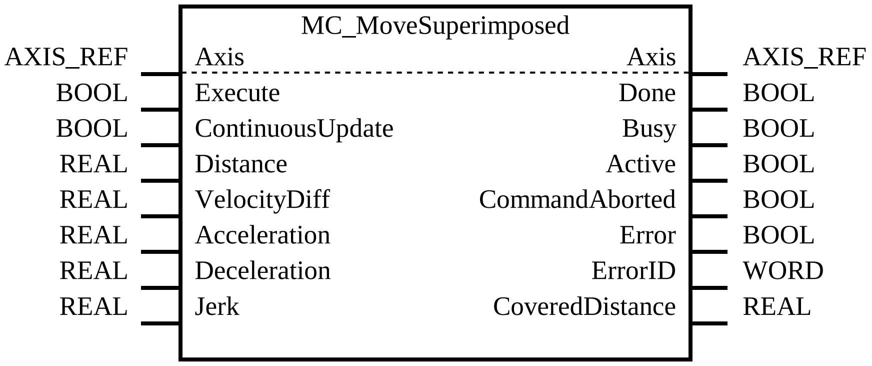
Figure 20: Timing diagram for MC_MoveAdditive

3.8. MC_MoveSuperimposed

FB-Name		**MC_MoveSuperimposed**	
This Function Block commands a controlled motion of a specified relative distance additional to an existing motion. The existing Motion is not interrupted, but is superimposed by the additional motion.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
B	Distance	REAL	Additional distance that is to be superimposed (in technical unit [u])
E	VelocityDiff	REAL	Value of the velocity difference of the additional motion (not necessarily reached) [u/s]
E	Acceleration	REAL	Value of the acceleration (increasing energy of the motor) [u/s^2]
E	Deceleration	REAL	Value of the deceleration (decreasing energy of the motor) [u/s^2]
E	Jerk	REAL	Value of the Jerk [u/s^3]
VAR_OUTPUT			
B	Done	BOOL	Additional distance superimposed to the ongoing motion
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
E	CoveredDistance	REAL	Displays continuously the covered distance contributed by this FB since it was started

Note:

- If MC_MoveSuperimposed is active, then any other command in aborting mode except MC_MoveSuperimposed will abort both motion commands: both the MC_MoveSuperimposed and the underlying motion command. In any other mode, the underlying motion command is not aborted
- If MC_MoveSuperimposed is active and another MC_MoveSuperimposed is commanded, only the on-going MC_MoveSuperimposed command is aborted, and replaced by the new MC_MoveSuperimposed, but not the underlying motion command
- The FB MC_MoveSuperimposed causes a change of the velocity and, if applicable, the commanded position of an ongoing motion in all relevant states
- In the state ‘Standstill’ the FB MC_MoveSuperimposed acts like MC_MoveRelative
- The values of ‘Acceleration’, ‘Deceleration’, and ‘Jerk’ are additional values to the on-going motion, and not absolute ones. With this, the underlying FB always finishes its job in the same period of time regardless of whether a MC_MoveSuperimposed FB takes place concurrently.
- The output ‘Active’ has a different behavior as in buffered FBs.



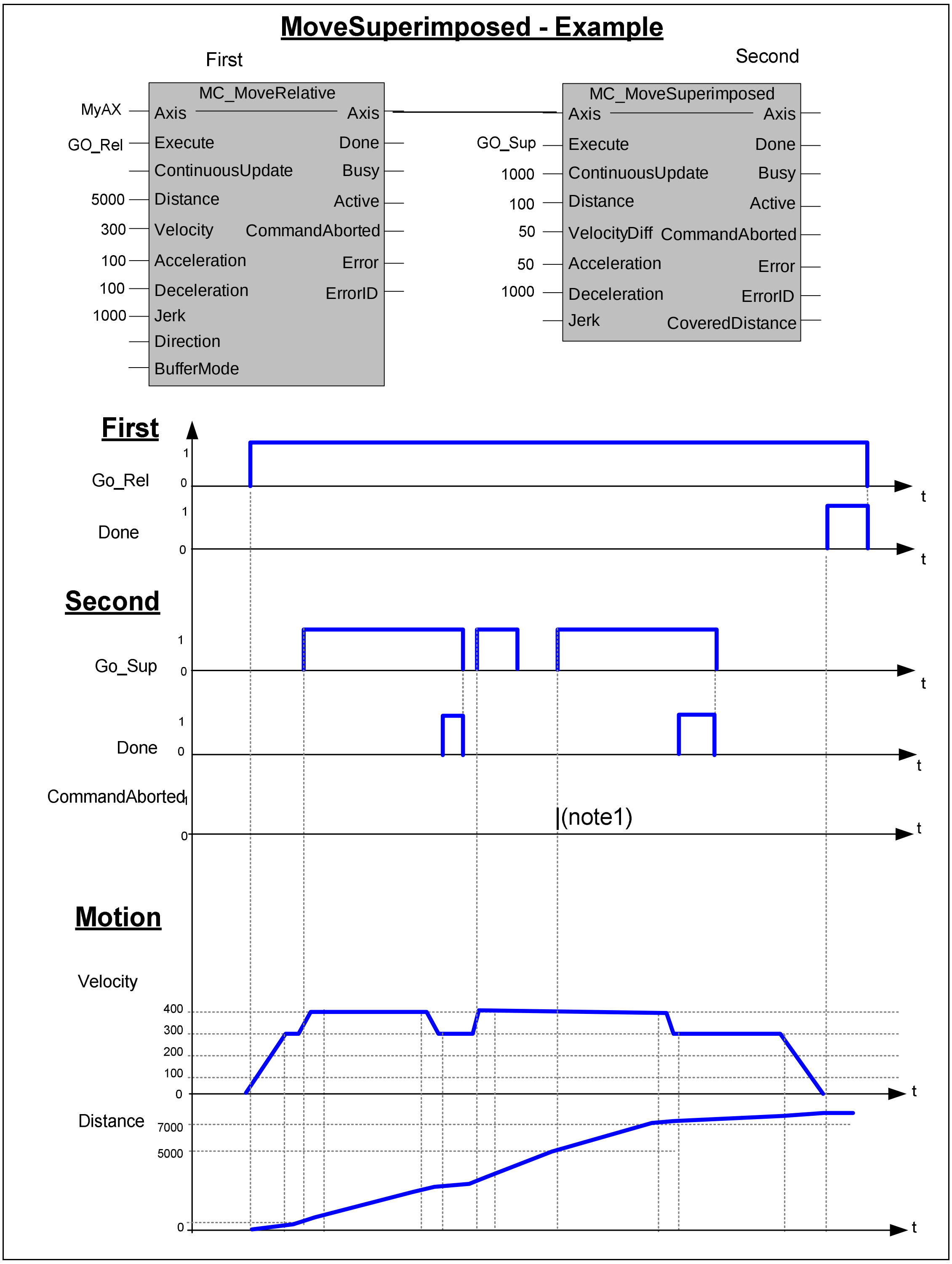


Figure 21: Timing diagram for MC_MoveSuperimposed

Note 1: the 'CommandAborted' is not visible here, because the new command works on the same instance (see general rules 2.4.1)

Note 2: the end position is between 7000 and 8000, depending on the timing of the aborting of the second command set for the MC_MoveSuperimposed

Example of MC_MoveSuperimposed during Camming with modulo axes. In green color the slave position is shown both with and without MC_MoveSuperimposed:

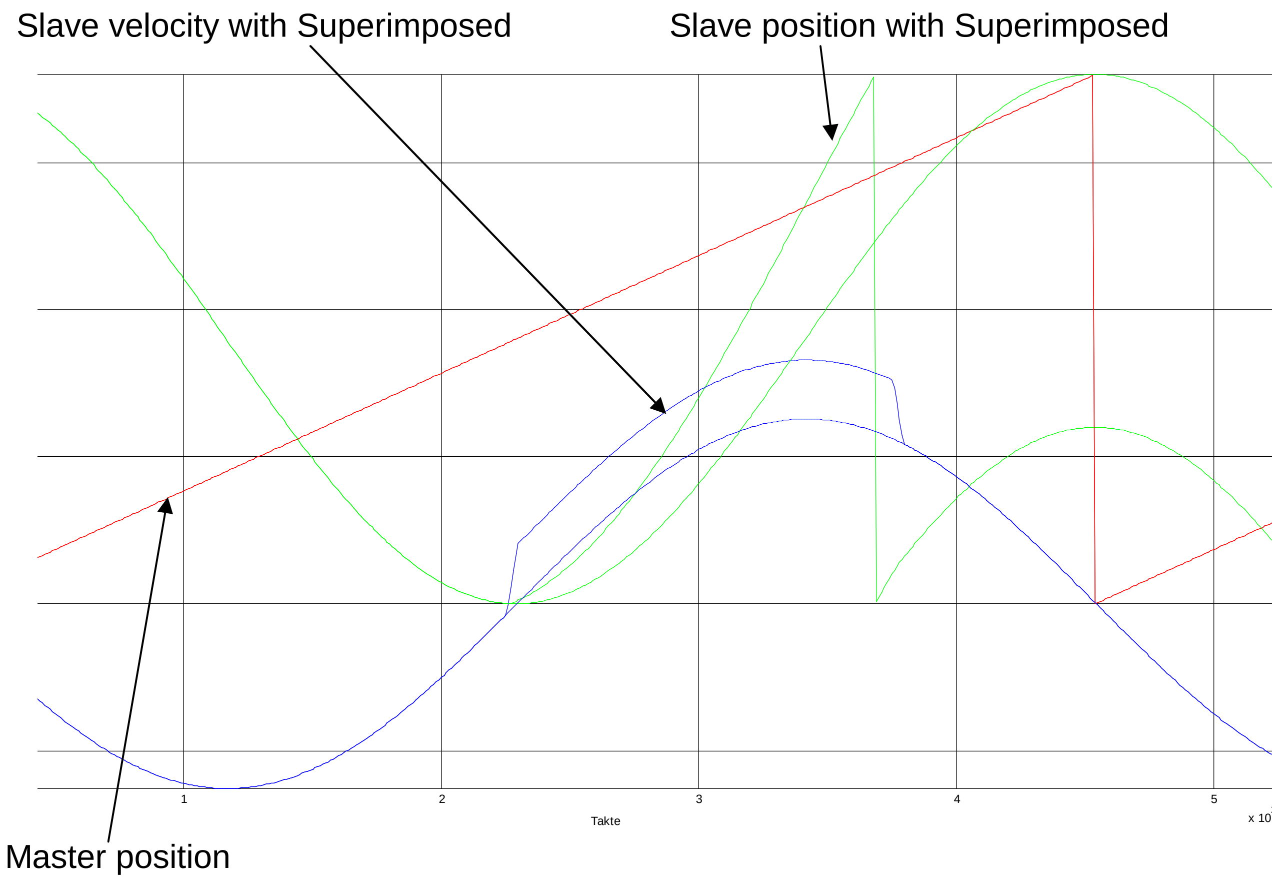


Figure 22: Example of the effect of MC_MoveSuperimposed on a slave axis

Note: at Slave velocity, the double line shows the effect of MoveSuperimposed while in synchronized motion during Camming. The same is valid for the related slave position.

The next example shows MC_MoveSuperimposed working on MC_MoveAbsolute. MC_MoveSuperimposed continues its movement even after the underlying discrete motion is finished.

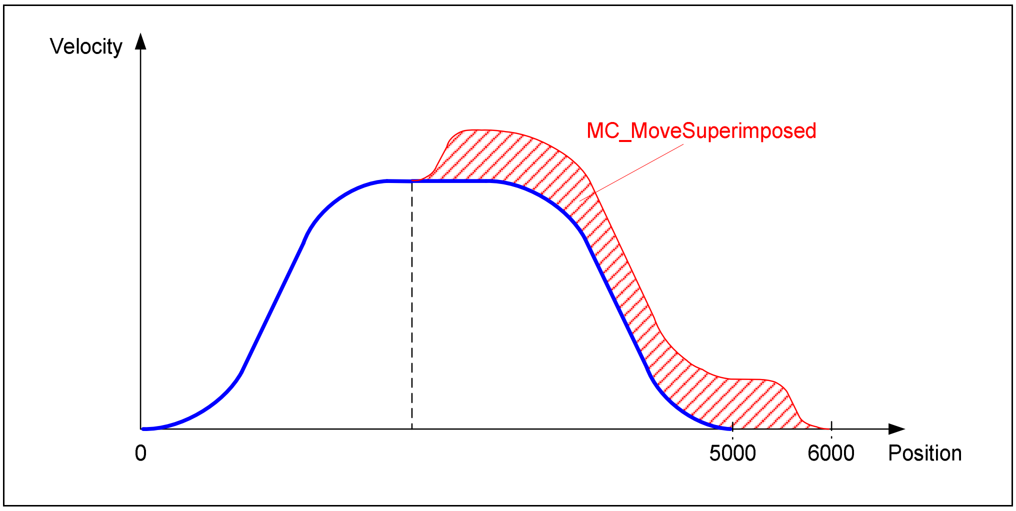
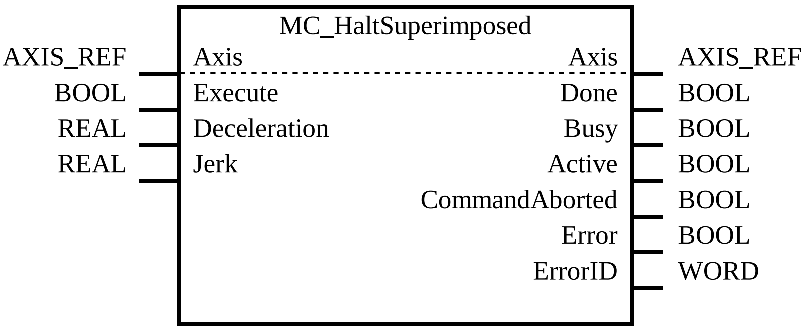


Figure 23: Example of the effect of MC_MoveSuperimposed on MC_MoveAbsolute

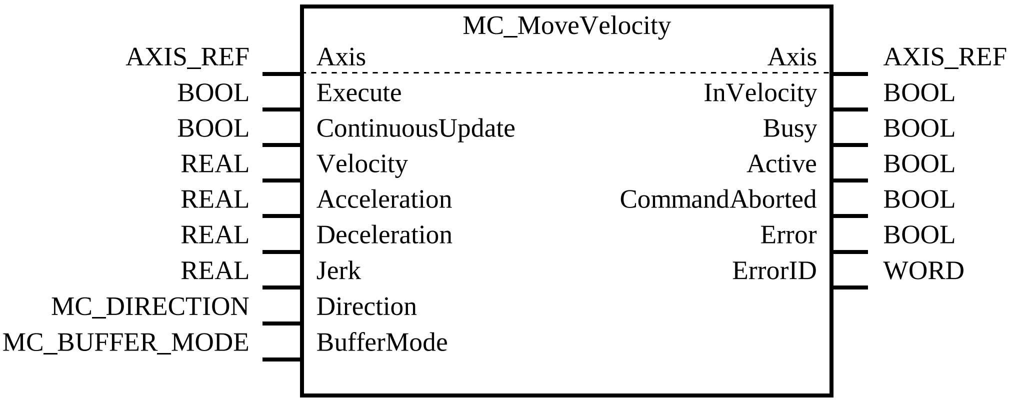
3.9. MC_HaltSuperimposed

FB-Name			**MC_HaltSuperimposed**	
This Function Block commands a halt to all superimposed motions of the axis. The underlying motion is not interrupted.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	BOOL	Start the action at rising edge
	E	Deceleration	REAL	Value of the deceleration [u/s^2]
	E	Jerk	REAL	Value of the Jerk [u/s^3]
VAR_OUTPUT				
	B	Done	BOOL	Superimposed motion halted
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	E	CommandAborted	BOOL	‘Command’ is aborted by another command
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
-				



3.10. MC_MoveVelocity

FB-Name		**MC_MoveVelocity**	
This Function Block commands a never ending controlled motion at a specified velocity.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input 'ContinuousUpdate'
B	Velocity	REAL	Value of the maximum velocity [u/s]. Can be a signed value.
E	Acceleration	REAL	Value of the acceleration (increasing energy of the motor) [u/s^2]
E	Deceleration	REAL	Value of the deceleration (decreasing energy of the motor) [u/s^2]
E	Jerk	REAL	Value of the Jerk [u/s^3]
E	Direction	MC_DIRECTION	Enum type (1-of-3 values: mcPositiveDirection, mcNegativeDirection, and mcCurrentDirection. Note: shortest way not applicable)
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	InVelocity	BOOL	Commanded velocity reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: • To stop the motion, the FB has to be interrupted by another FB issuing a new command • The signal 'InVelocity' has to be reset when the block is aborted by another block. • Negative velocity * negative direction = positive velocity • In combination with MC_MoveSuperimposed, the output 'InVelocity' is SET as long as the contribution of this FB (MC_MoveVelocity) to the set velocity is equal to the commanded velocity of this FB.			



The following figure shows two examples of the combination of two MC_MoveVelocity Function Blocks:

1. The left part of the timing diagram illustrates the case if the Second Function Block is called **after** the First one is completed. If First reaches the commanded velocity 3000 then the output 'First.InVelocity' AND the signal Next causes the Second FB to move to the velocity 2000. In the next cycle 'First.InVelocity' is Reset and

‘First.CommandAborted’ is Set. Therefore the ‘Execute’ of the 2nd FB is Reset. And as soon as the axis reaches ‘Velocity’ 2000 the ‘Second.InVelocity’ is set.

2. The right part of the timing diagram illustrates the case if the Second move Function Block starts the execution **while** the First FB is not yet ‘InVelocity’.
   The following sequence is shown: The First motion is started again by GO at the input ‘First.Execute’. While the First FB is still accelerating to reach the velocity 3000 the First FB will be interrupted and aborted because the Test signal starts the Run of the Second FB. Now the Second FB runs and decelerates the velocity to 2000.

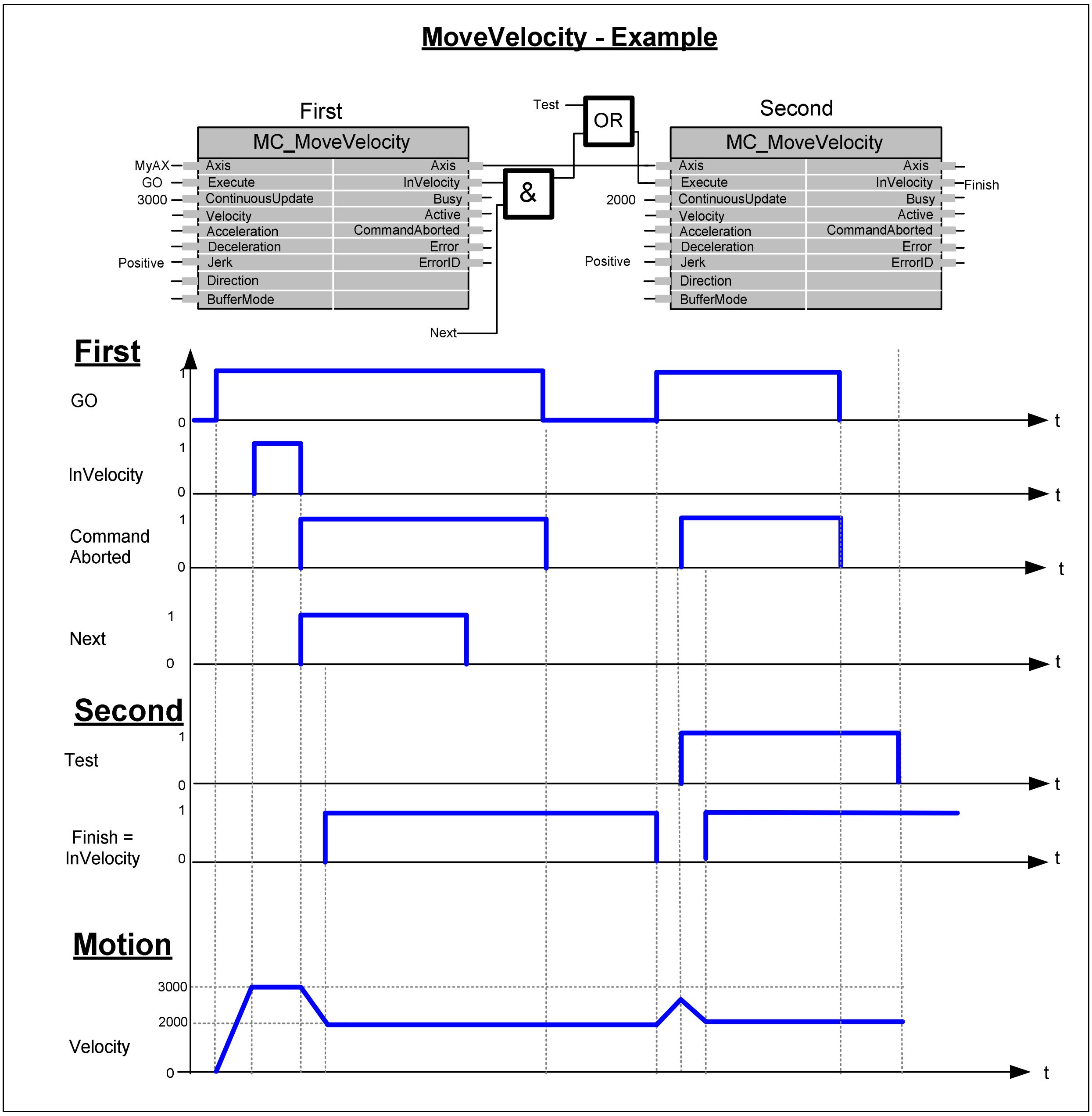
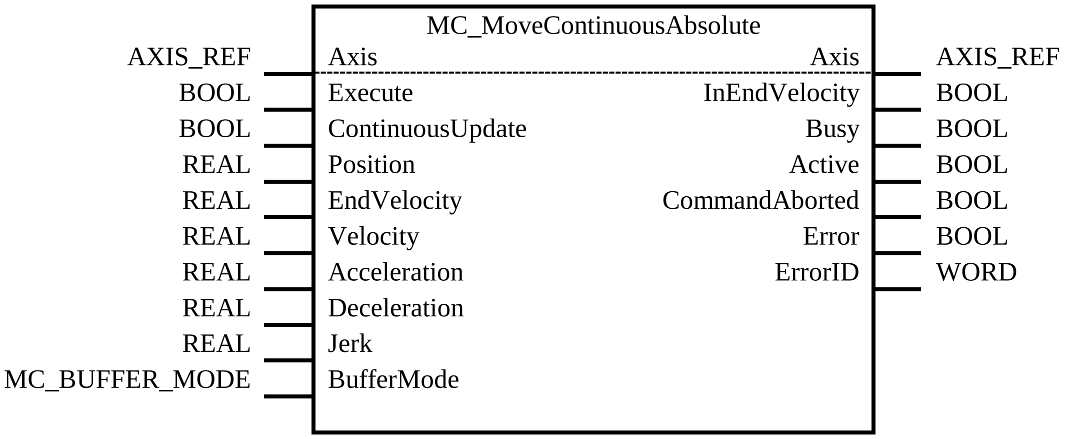


Figure 24: MC_MoveVelocity timing diagram

Note: 2nd FB in mode ‘Aborting’ (If in buffered mode the velocity would reach 3000 before actuating the next FB).

3.11. MC_MoveContinuousAbsolute

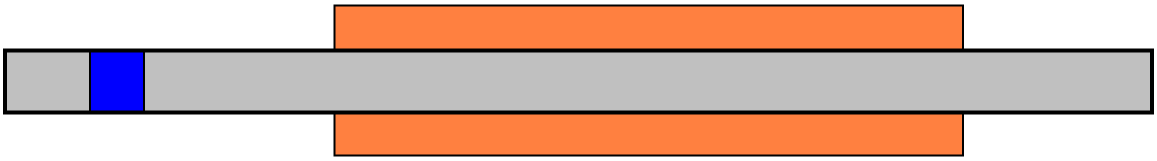
FB-Name		MC_MoveContinuousAbsolute	
This Function Block commands a controlled motion to a specified absolute position ending with the specified velocity.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input 'ContinuousUpdate'
B	Position	REAL	Commanded position for the motion (in technical unit [u]) (negative or positive)
B	EndVelocity	REAL	Value of the end velocity [u/s]. Signed value
B	Velocity	REAL	Value of the maximum velocity [u/s]
E	Acceleration	REAL	Value of the acceleration [u/s^2]
E	Deceleration	REAL	Value of the deceleration [u/s^2]
E	Jerk	REAL	Value of the Jerk [u/s^3]
E	Direction	MC_DIRECTION	Enum type (1-of-4 values: mcPositiveDirection, mcNegativeDirection, mcCurrentDirection and mcShortestWay
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	InEndVelocity	BOOL	Commanded distance reached and running at requested end velocity
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
B	ErrorID	WORD	Error identification
Notes: • If the commanded position is reached and no new motion command is put into the buffer, the axis continues to run with the specified 'EndVelocity'. • State 'ContinuousMotion' (meaning: it will not stop by itself). • This FB can be replaced by the combination of MC_MoveAbsolute and MC_MoveVelocity if BufferMode is implemented on those FBs			



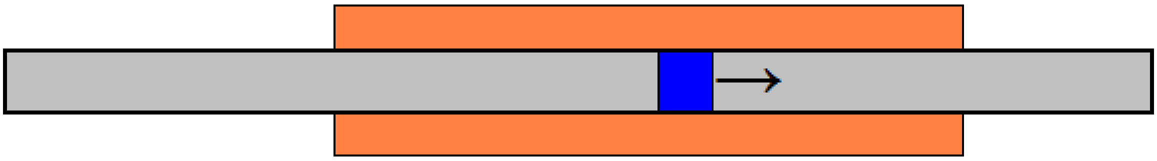
One use case for MC_MoveContinuousAbsolute is a linear cutter:

One linear axis that is carrying a laser device that is used to cut a workpiece.

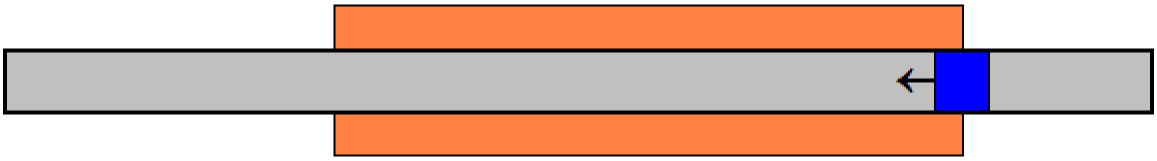
Starting from lrIdlePos the working chain is this:



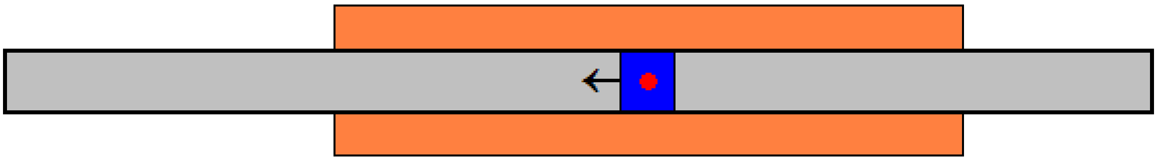
1. Move the laser with fast velocity over the position lrStartCutPos. The laser is off during this movement:



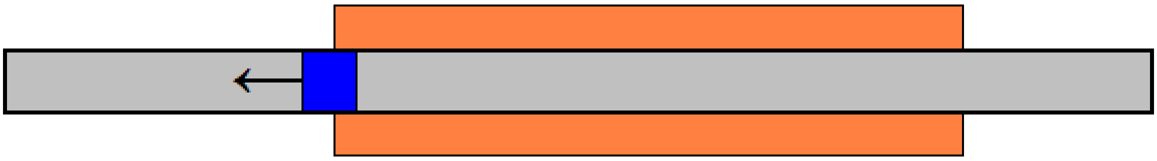
2. Turn back and make sure to have the speed lrCutVelocity when at lrStartCutPos. At this position, switch the laser on:



3. Travel over the work piece with this constant speed while the laser is on:



4. When reaching lrEndCutPos switch off the laser and move back to idle position with fast velocity:



During the cutting process the laser must be moved with a fix velocity, no acceleration or deceleration phase can be tolerated. The laser must be moved to its waiting position after the cutting was done.
This can be achieved with the FB MC_MoveContinuousAbsolute in the following way:

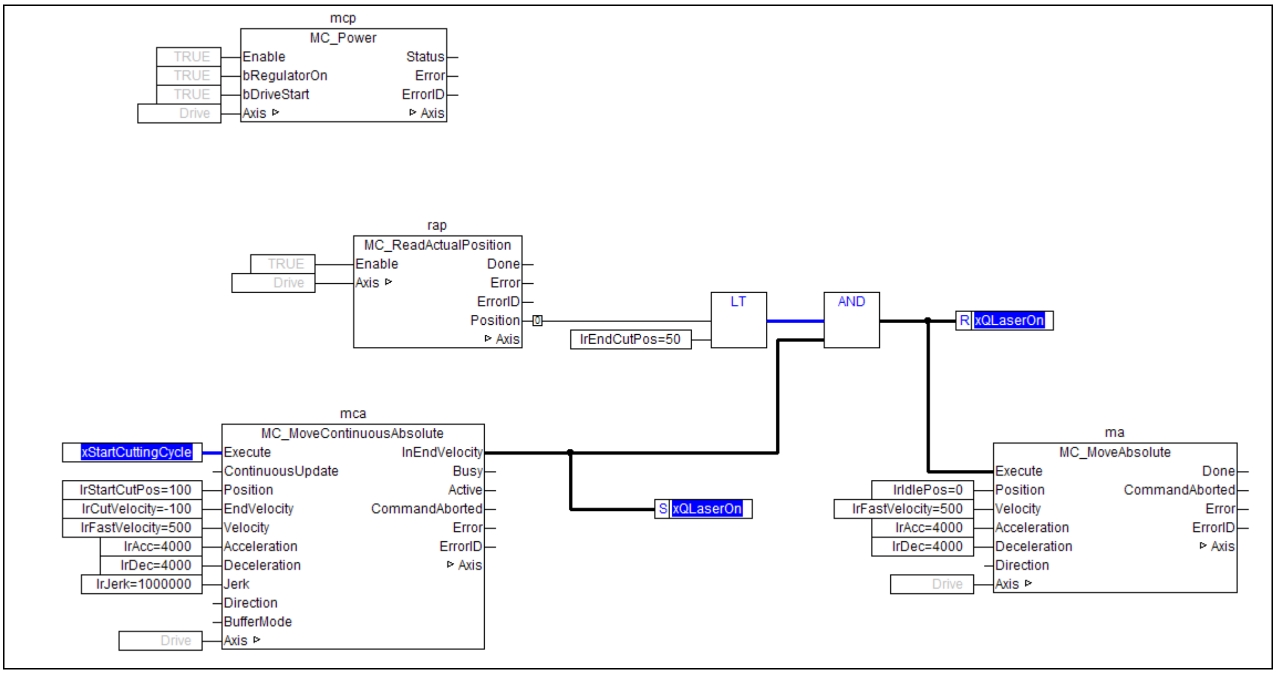


Figure 25: Example MC_MoveContinuousAbsolute

Started with a rising edge of xStartCuttingCycle, the instance ‘mca’ of MC_MoveContinuousAbsolute will move the axis with lrFastVelocity over lrStartCutPos, turn back and have the speed lrCutVelocity when reaching lrStartCutPos again in negative direction. In this point in time, ‘InEndVelocity’ is set, and the laser is switched on. As no other motion FB interrupts this movement, MC_MoveContinuousAbsolute will keep travelling in negative direction with the current speed. After the axis has overstepped the position lrEndPos, where the laser is switched off, the MC_MoveAbsolute instance ‘ma’ moves the axis with high speed to its idle position:

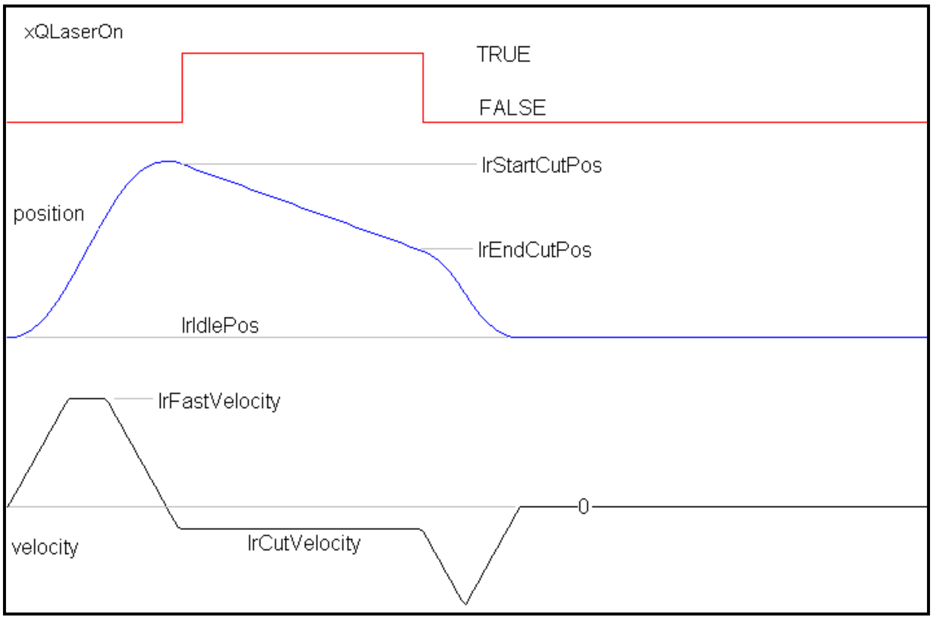


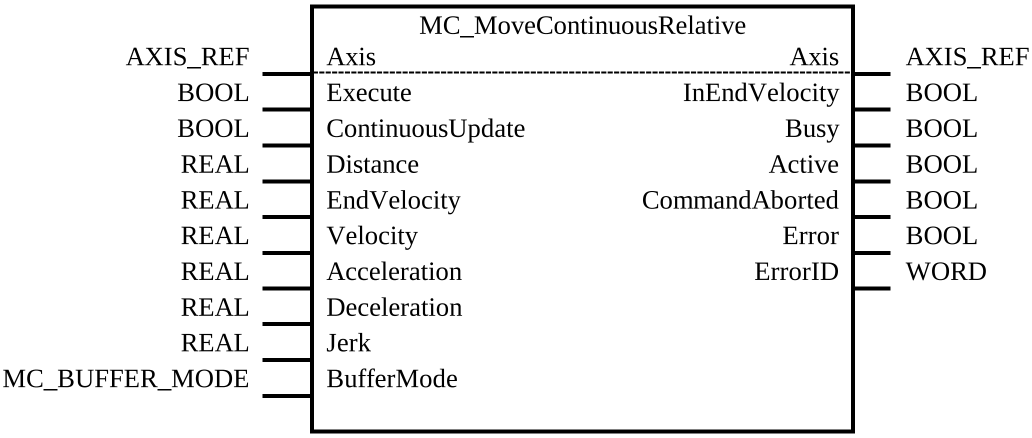
Figure 26: MC_MoveContinuousAbsolute timing diagram for example above

3.12. MC_MoveContinuousRelative

FB-Name		MC_MoveContinuousRelative	
This Function Block commands a controlled motion of a specified relative distance ending with the specified velocity.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
B	Distance	REAL	Relative distance for the motion [u]
B	EndVelocity	REAL	Value of the end velocity [u/s]. Signed value
B	Velocity	REAL	Value of the maximum velocity [u/s]
E	Acceleration	REAL	Value of the acceleration [u/s^2]
E	Deceleration	REAL	Value of the deceleration [u/s^2]
E	Jerk	REAL	Value of the Jerk [u/s^3]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	InEndVelocity	BOOL	Commanded distance reached and running at requested end velocity
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
B	ErrorID	WORD	Error identification

Notes:

- If the commanded position is reached and no new motion command is put into the buffer, the axis continues to run with the specified ‘EndVelocity’.
- State ‘ContinuousMotion’ (meaning: it will not stop by itself).
- This FB is specified here for systems without the support for the ‘BufferMode’.
- This FB can be replaced by the combination of MC_MoveAbsolute and MC_MoveVelocity if BufferMode is implemented on those FBs



These two sampling traces show the effect of the sign of the value of the input ‘EndVelocity’:
1. ‘EndVelocity’ with positive direction:

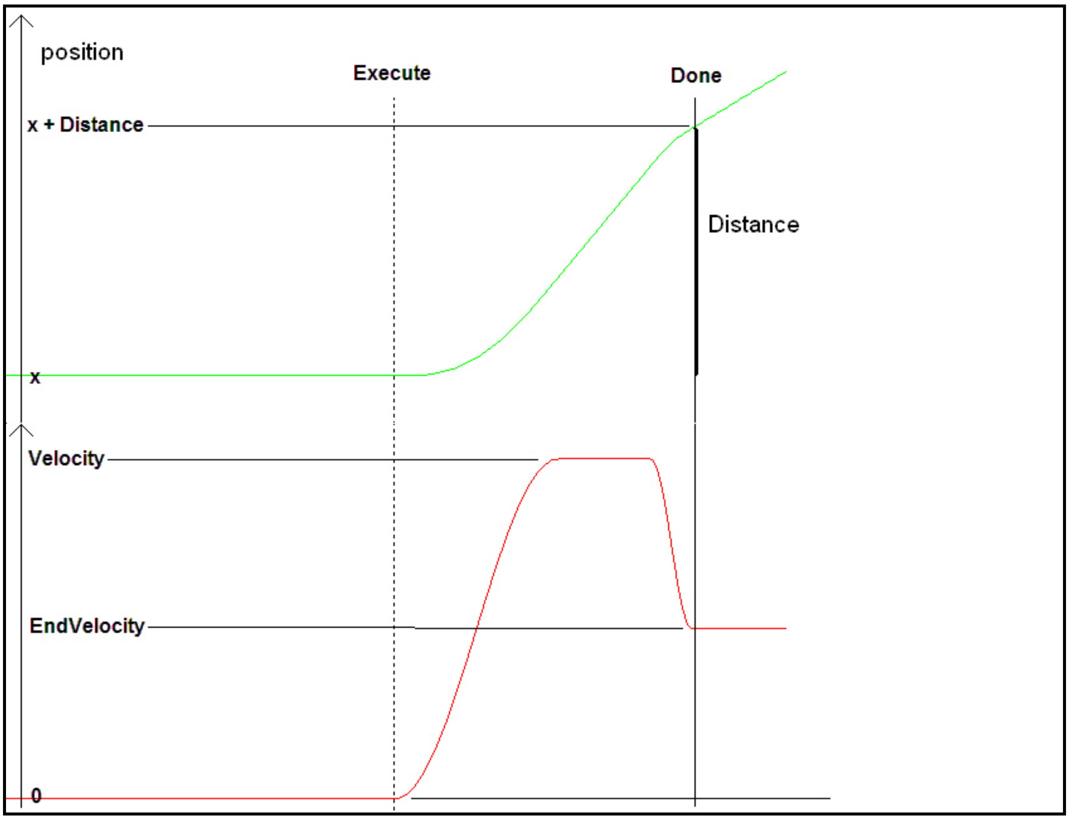


Figure 27: MC_MoveContinuousRelative timing diagram with positive direction

2. ‘EndVelocity’ with negative direction:

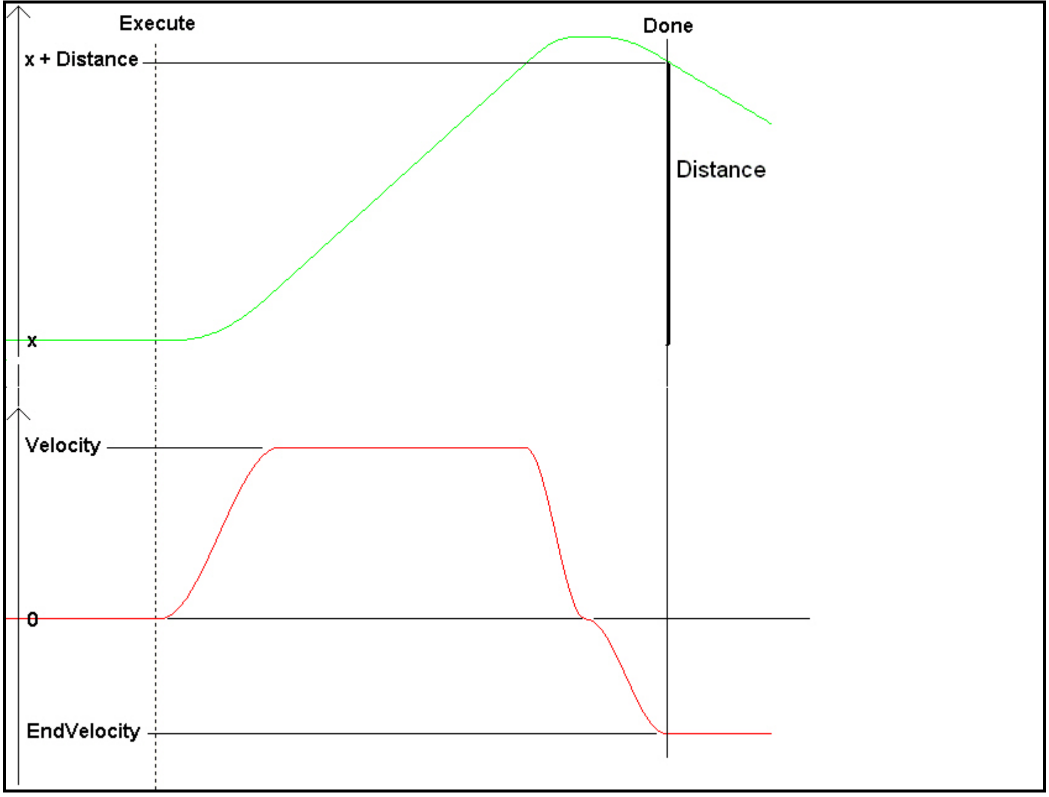


Figure 28: MC_MoveContinuousRelative timing diagram with negative direction

Example of MC_MoveContinuousRelative:

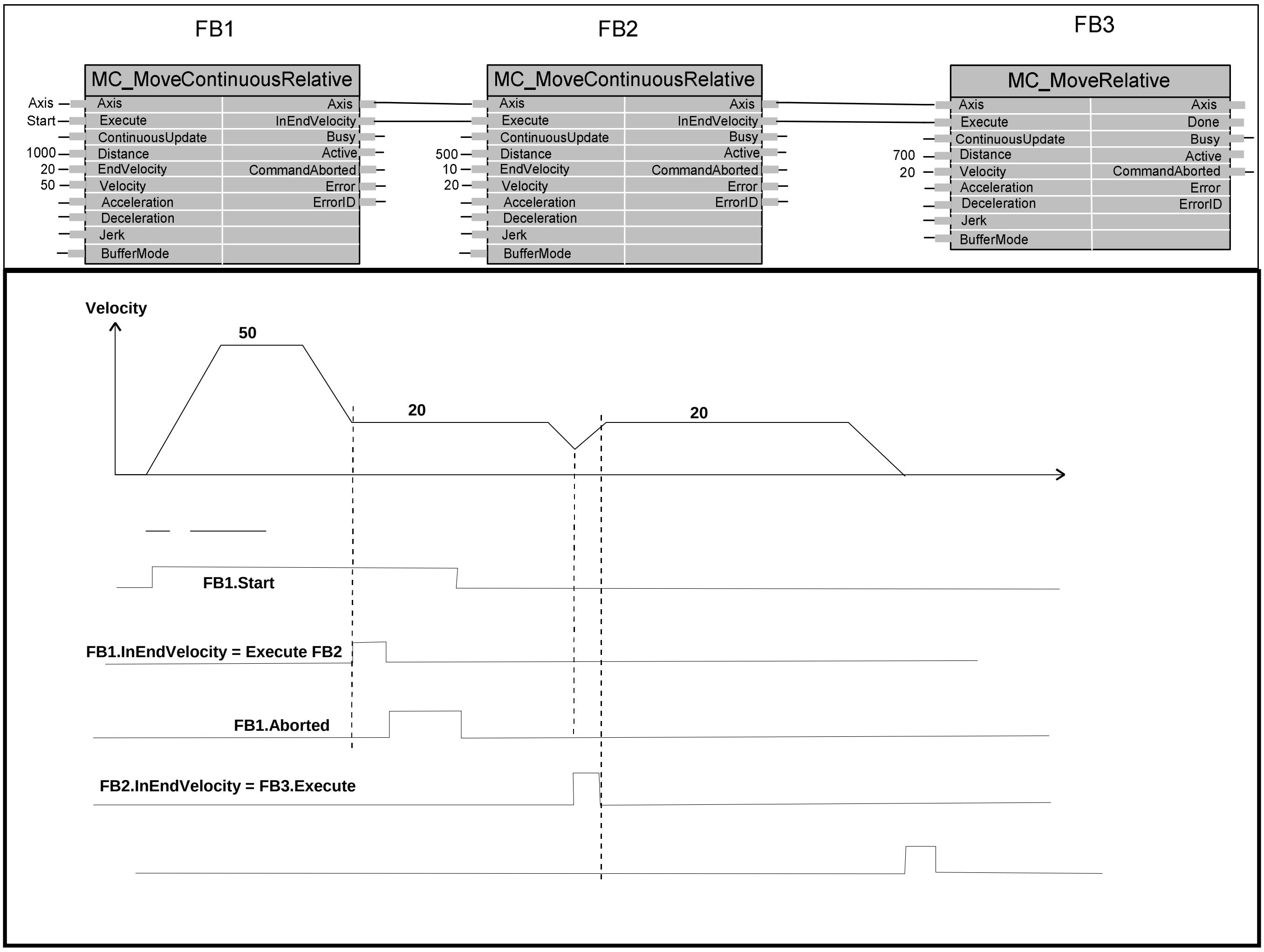


Figure 29: Example of MC_MoveContinuousRelative

3.13. MC_TorqueControl

FB-Name	**MC_TorqueControl**		
This Function Block continuously exerts a torque or force of the specified magnitude. This magnitude is approached using a defined ramp (‘TorqueRamp’), and the Function Block sets the ‘InTorque’ output if the commanded torque level is reached. This function block is applicable for force and torque. When there is no external load, force is applicable. Positive torque is in the positive direction of velocity.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Starts the motion on a rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
B	Torque	REAL	Value of the torque (Torque or force in technical unit [u])
E	TorqueRamp	REAL	The maximum time derivative of the set value of the torque or force (in technical unit per sec. [u/s])
E	Velocity	REAL	Absolute value of the maximum velocity.
E	Acceleration	REAL	Value of the maximum acceleration (acceleration is applicable with same sign of torque and velocity)
E	Deceleration	REAL	Value of the maximum deceleration (deceleration is applicable with opposite signs of torque and velocity)
E	Jerk	REAL	Value of the maximum jerk
E	Direction	MC_DIRECTION	Enum type (1 of 2 values: mcPositiveDirection, mcNegativeDirection or mcCurrentDirection). Specifies the direction of the torque. (Note: Torque input can be signed value).
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	InTorque	BOOL	Setpoint value of torque or force equals the commanded value
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:

1. The movement is limited by velocity, acceleration / deceleration, and jerk, or by the value of the torque, depending on the mechanical circumstances.
2. Specific additional tests are outside this FB. For instance, checking on the traveled distance could be done via tracing the actual positions during the action.
3. ‘Velocity’ is a limit input and is always a positive value. The direction is dependent on the torque and load.
4. The axis ceases to be in ‘Torque’ control mode when any motion control (not administrative) Function Block is accepted on the same axis.

MC_TorqueControl

AXIS_REF	Axis	Axis	AXIS_REF
BOOL	Execute	InTorque	BOOL
BOOL	ContinuousUpdate	Busy	BOOL
REAL	Torque	Active	BOOL
REAL	TorqueRamp	CommandAborted	BOOL
REAL	Velocity	Error	BOOL
REAL	Acceleration	ErrorID	WORD
REAL	Deceleration		
REAL	Jerk		
MC_DIRECTION	Direction		
MC_BUFFER_MODE	BufferMode		

The example below shows the typical behavior of an intermediate “resistive” load (see ‘Deceleration’ limit) with some “inertia” (see ‘TorqueRamp’ limit).

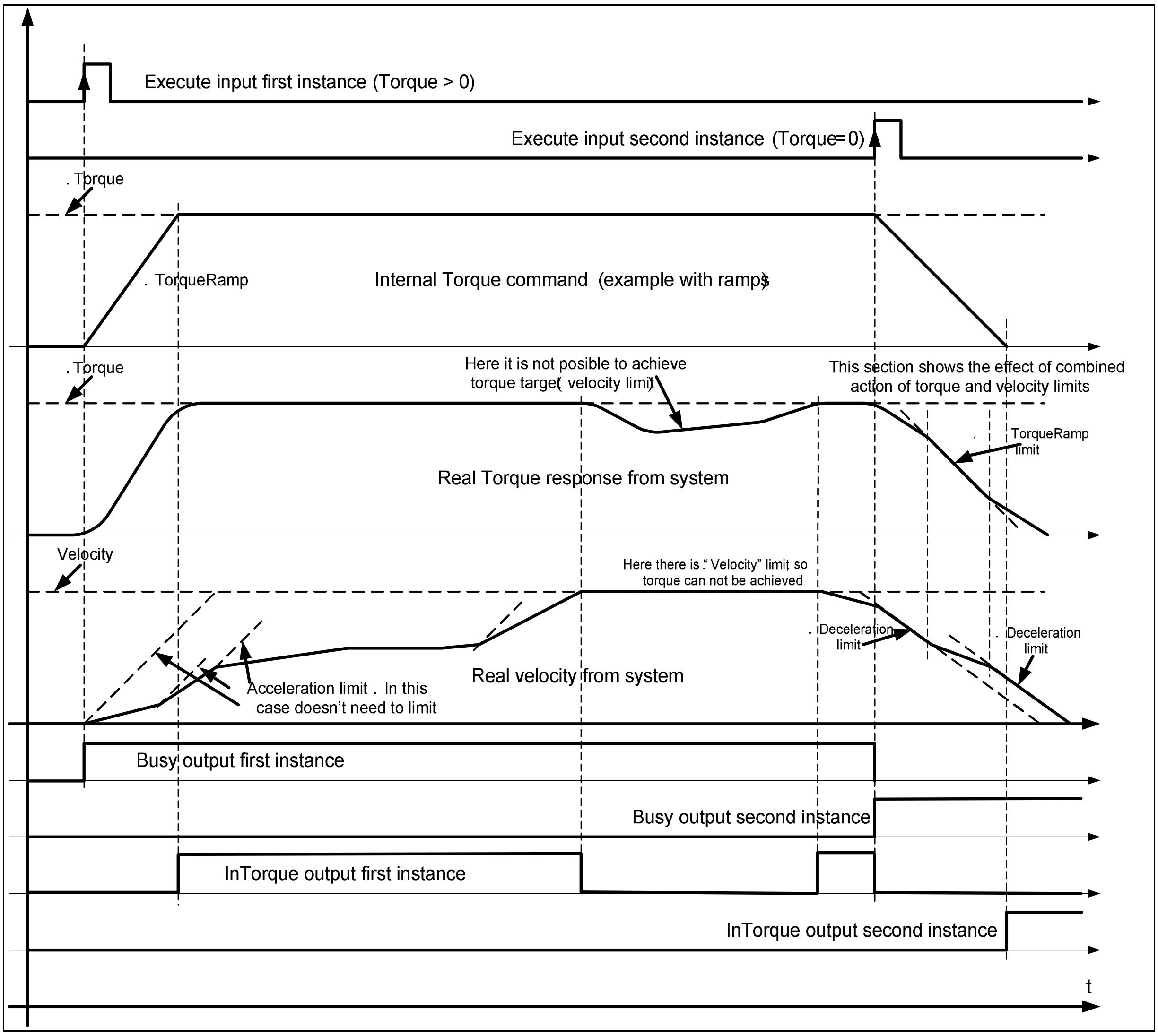

Execute input first instance (Torque > 0)
Execute input second instance (Torque=0)
. Torque
. TorqueRamp
Internal Torque command (example with ramps)
. Torque
Here it is not posible to achieve torque target(velocity limit)
This section shows the effect of combined action of torque and velocity limits
. TorqueRamp limit
Real Torque response from system
Velocity
Here there is ." Velocity" limit so torque can not be achieved
. Deceleration limit
. Deceleration limit
Acceleration limit . In this case doesn't need to limit
Real velocity from system
Busy output first instance
Busy output second instance
InTorque output first instance
InTorque output second instance
t

Figure 30: First example of MC_TorqueControl

This example could be implemented in a Function Block Diagram like:

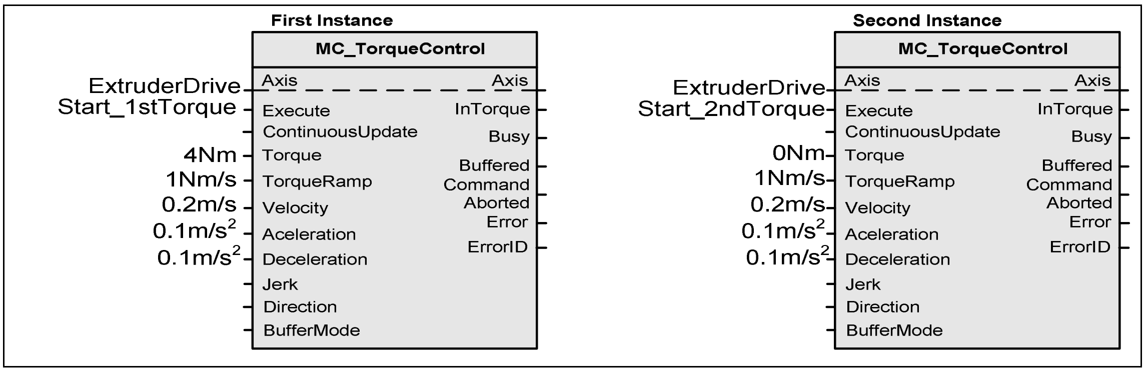


Figure 31: Program of example of MC_TorqueControl

The second example (below) opposite signs for ‘Direction’ & ‘Torque’ are used (e.g. Retention or brake control). (In the FB: +Direction –Torque). It is like an unwinding application with torque on the material, and a break in the material. When the material breaks, as shown in the middle of the picture, this causes a drop in the real Torque value (in absolute terms): the velocity will decrease, limited by the fastest “deceleration” limit specified by the ‘Deceleration’ VAR_INPUT down to zero velocity (with no tension there is a risk of having shock breakings, so we have to limit to the fastest). In this case the torque setpoint might not be achieved.

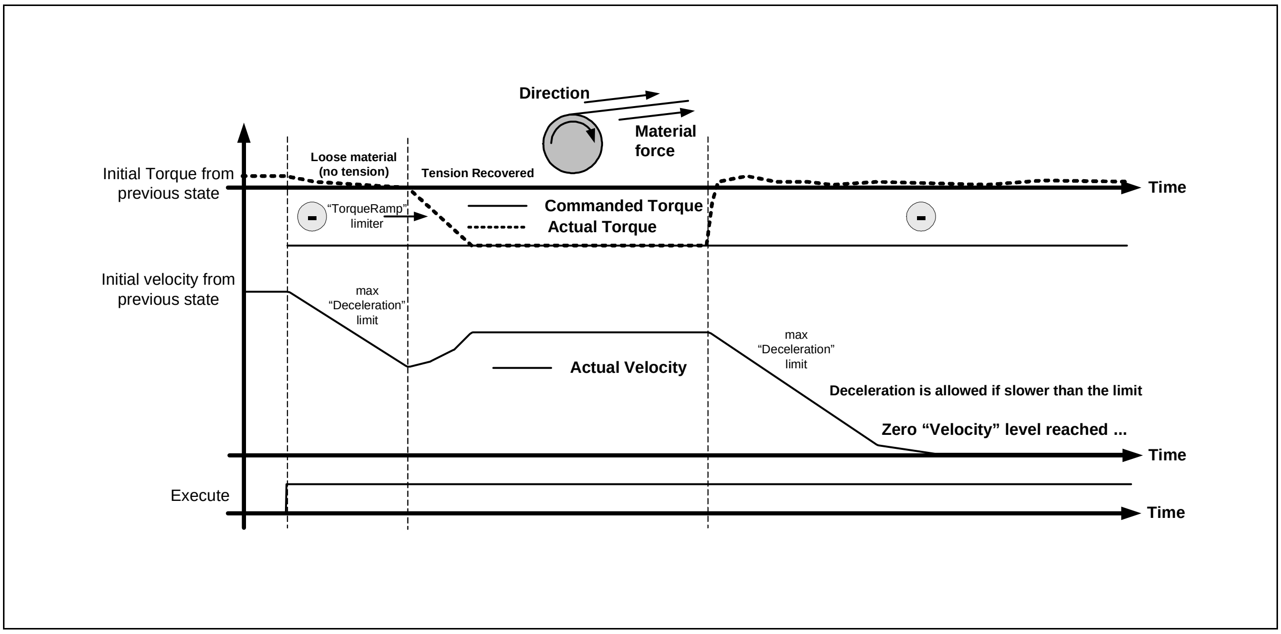


Figure 32: Second example of MC_TorqueControl

NOTE: In an unwinding application (derived from this brake control) material tension is the target, not motor torque. The instantaneous diameter of the roll should be taken into account to transform the “User tension setpoint”. Also additional inertia compensation by modification of the torque setpoint for acceleration / deceleration is common from instantaneous weight data (weight is commonly estimated from diameter). Additionally in unwinding applications, in the case of loose material (same condition as material break), a negative slow velocity reference is usually applied in order to “rewind” the loose material. In this case, this has to be provided by external programming.

3.14. MC_PositionProfile

FB-Name		**MC_PositionProfile**	
This Function Block commands a time-position locked motion profile			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
B	TimePosition	MC_TP_REF	Reference to Time / Position. Description - see note below
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
E	TimeScale	REAL	Overall time scaling factor of the profile
E	PositionScale	REAL	Overall Position scaling factor
E	Offset	REAL	Overall offset for profile [u]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	Done	BOOL	Profile completed
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function block
E	ErrorID	WORD	Error identification

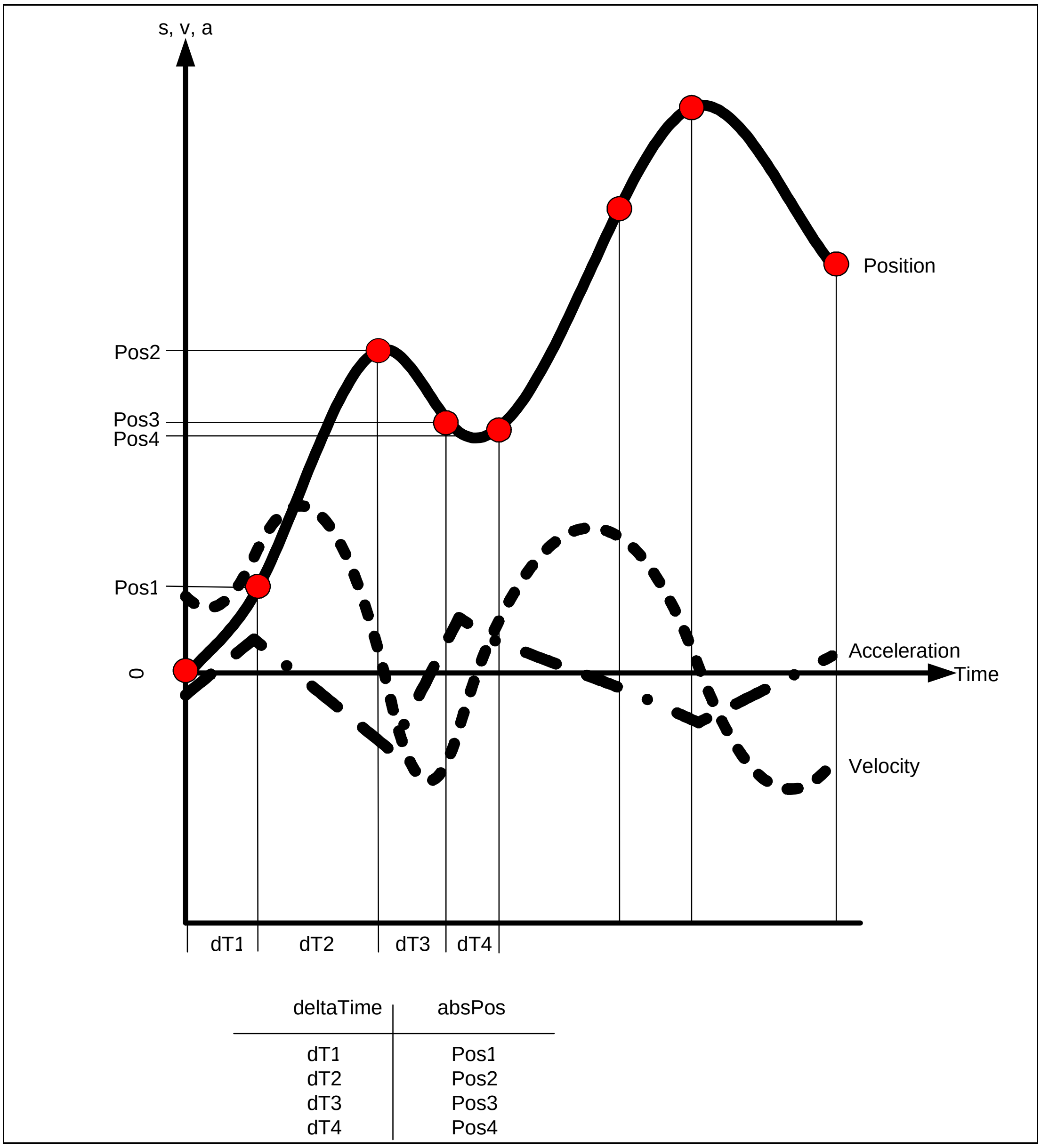
Notes:

- MC_TP_REF is a supplier specific data type. An example for this datatype is given below:
  - The content of a Time/Position pair may be expressed in DeltaTime/Pos, where Delta could be the difference in time between two successive points.

  -
    ```
    TYPE
      MC_TP : STRUCT
        DeltaTime : TIME;
        Position : REAL;
      END_STRUCT;
    END_TYPE
    ```

  -
    ```
    TYPE
      MC_TP_REF : STRUCT
        NumberOfPairs : WORD;
        IsAbsolute : BOOL;
        MC_TP_Array : ARRAY [1..N] OF MC_TP;
      END_STRUCT;
    END_TYPE
    ```
- This functionality does not mean it runs one profile over and over again: it can switch between different profiles
- Alternatively to this FB, the FB MC_CamIn coupled to a virtual master can be used

	MC_PositionProfile		
AXIS_REF	Axis	Axis	AXIS_REF
MC_TP_REF	TimePosition	TimePosition	MC_TP_REF
BOOL	Execute	Done	BOOL
BOOL	ContinuousUpdate	Busy	BOOL
REAL	TimeScale	Active	BOOL
REAL	PositionScale	CommandAborted	BOOL
REAL	Offset	Error	BOOL
MC_BUFFER_MODE	BufferMode	ErrorID	WORD



deltaTime	absPos
dT1	Pos1
dT2	Pos2
dT3	Pos3
dT4	Pos4

Figure 33: Example of Time / MC_PositionProfile

Note: The Time / Velocity and Time / Acceleration Profiles are similar to the 'Position' Profile, with sampling points on the 'Velocity' or 'Acceleration' lines.

3.15. MC_VelocityProfile

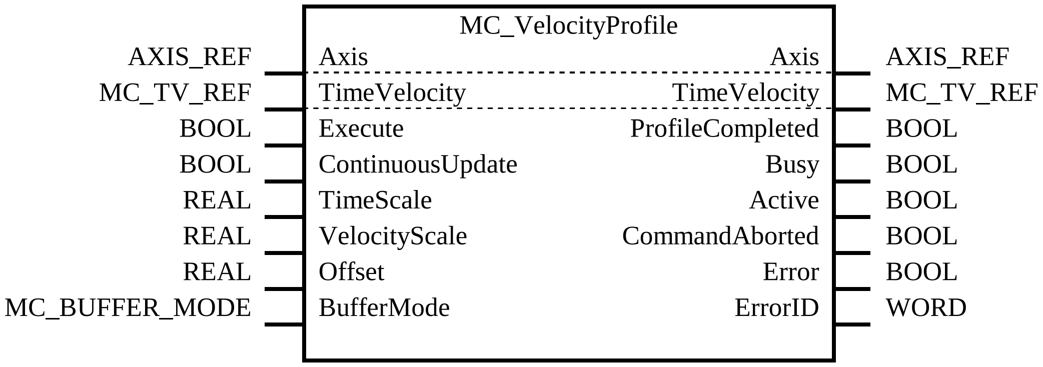
FB-Name		**MC_VelocityProfile**	
This Function Block commands a time-velocity locked motion profile. The velocity in the final element in the profile should be maintained. The state remains 'ContinuousMotion'.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
B	TimeVelocity	MC_TV_REF	Reference to Time / Velocity. Description - see note below
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input 'ContinuousUpdate'
E	TimeScale	REAL	Overall time scaling factor of the profile
E	VelocityScale	REAL	Overall velocity scaling factor of the profile
E	Offset	REAL	Overall offset for profile [u/s]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	ProfileCompleted	BOOL	End of profile reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:

- MC_TV_REF is a supplier specific datatype. An example for this datatype is given here below:
  - The content of Time/Velocity pair may be expressed in DeltaTime/Velocity, where Delta could be the difference in time between two successive points. Velocity can be a signed value.

  -
    ```
    TYPE
      MC_TV : STRUCT
        DeltaTime : TIME;
        Velocity : REAL;
      END_STRUCT;
    END_TYPE
    ```

  -
    ```
    TYPE
      MC_TV_REF : STRUCT
        NumberOfPairs : WORD;
        MC_TV_Array : ARRAY [1..N] of MC_TV;
      END_STRUCT;
    END_TYPE
    ```
- This functionality does not mean it runs one profile over and over again: it can switch between different profiles
- Alternatively to this FB, the CAM FB coupled to a virtual master can be used

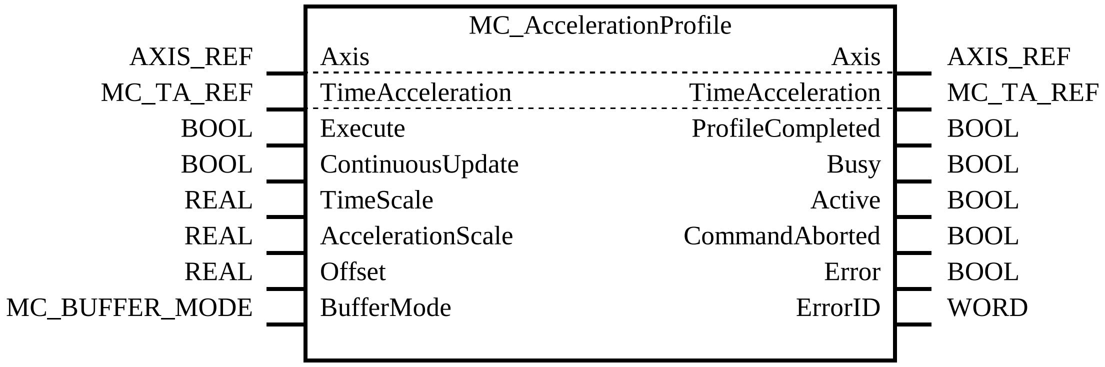

MC_VelocityProfile
AXIS_REF
Axis
Axis
AXIS_REF
MC_TV_REF
TimeVelocity
TimeVelocity
MC_TV_REF
BOOL
Execute
ProfileCompleted
BOOL
BOOL
ContinuousUpdate
Busy
BOOL
REAL
TimeScale
Active
BOOL
REAL
VelocityScale
CommandAborted
BOOL
REAL
Offset
Error
BOOL
MC_BUFFER_MODE
BufferMode
ErrorID
WORD

3.16. MC_AccelerationProfile

FB-Name	**MC_AccelerationProfile**			
This Function Block commands a time-acceleration locked motion profile. After finalizing the acceleration profile, the acceleration goes to 0 (and typically the final velocity is maintained). It stays in the state 'ContinuousMotion'.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
	B	TimeAcceleration	MC_TA_REF	Reference to Time / Acceleration. Description – see note below
VAR_INPUT				
	B	Execute	BOOL	Start the motion at rising edge
	E	ContinuousUpdate	BOOL	See 2.4.6 The input 'ContinuousUpdate'
	E	TimeScale	REAL	Overall time scaling factor of the profile
	E	AccelerationScale	REAL	Scale factor for acceleration amplitude
	E	Offset	REAL	Overall offset for profile [u/s^2]
	E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT				
	B	ProfileCompleted	BOOL	End of profile reached
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	E	Active	BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	BOOL	'Command' is aborted by another command
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification

Notes:

- MC_TA_REF is a supplier specific datatype. An example for this datatype is given here below:
  - The content of Time/Acceleration pair may be expressed in DeltaTime/Acceleration, where Delta could be the difference in time between two successive points.
  - TYPE
    ```
      MC_TA : STRUCT
          DeltaTime : TIME;
          Acceleration : REAL;
      END_STRUCT;
    END_TYPE
    ```
  - TYPE
    ```
      MC_TA_REF : STRUCT
          NumberOfPairs : WORD;
          MC_TA_Array : ARRAY [1..N] of MC_TA;
      END_STRUCT;
    END_TYPE
    ```
- alternatively to this FB, the CAM FB coupled to a virtual master can be used



Example of an acceleration profile:
A profile is made from a number of sequential “A to B” positioning points. It is simple to visualize, but requires a lot of sequences for a smooth profile. These requirements are often beyond the capability of low-end servos.
Alternatively, by using a modest amount of constant acceleration segments it is possible to define a well-matching motion profile. With this method the capability range of low-end servos can be extended.
It is possible to make matching to either:

1. Position versus time profile
2. Master versus slave axis

Advantages:

- Compact description of a profile
- Smooth profile properties by nature
- Low processor power requirements

Disadvantages

- Higher programming abstraction level with existing tools

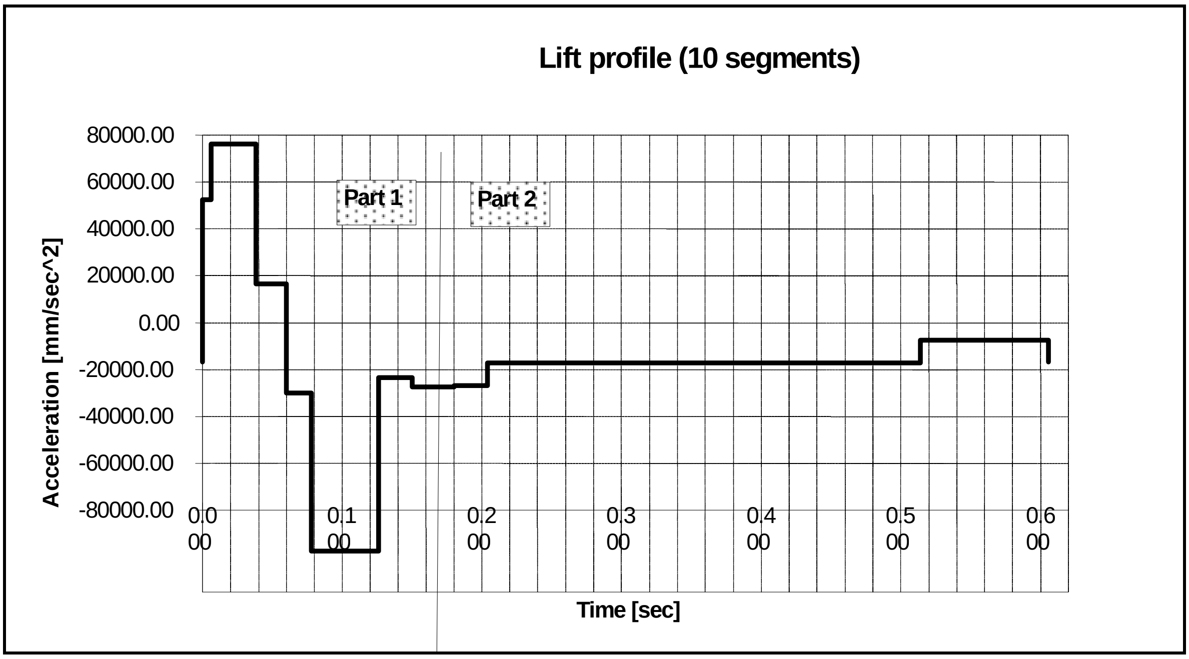


Figure 34: MC_AccelerationProfile, 10 segments only

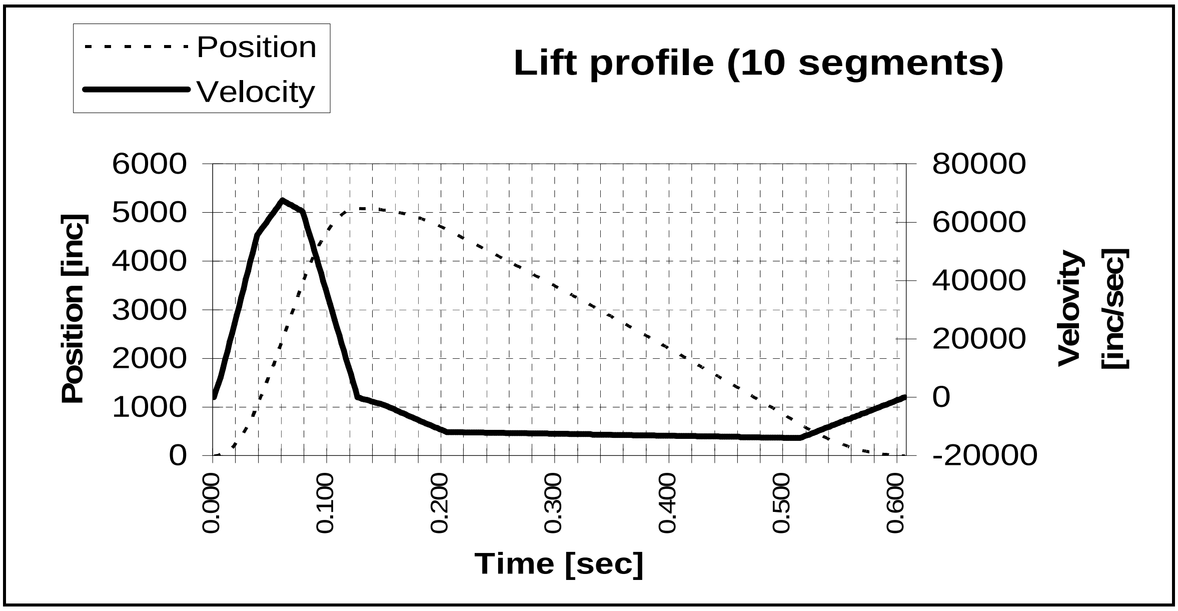
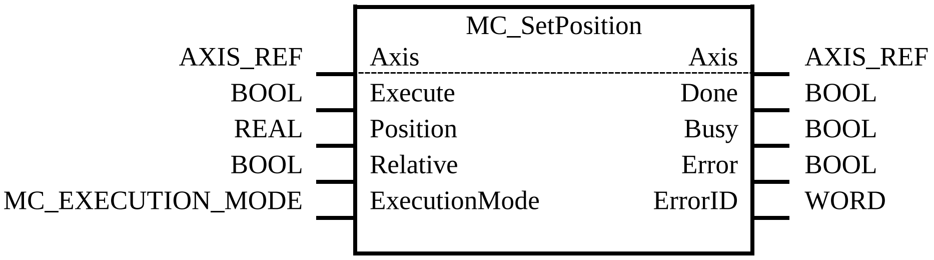


Figure 35: Resulting MC_PositionProfile

3.17. MC_SetPosition

FB-Name	**MC_SetPosition**		
This Function Block shifts the coordinate system of an axis by manipulating both the set-point position as well as the actual position of an axis with the same value without any movement caused. (Re-calibration with same following error). This can be used for instance for a reference situation. This Function Block can also be used during motion without changing the commanded position, which is now positioned in the shifted coordinate system.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Start setting position in axis
B	Position	REAL	Position unit [u] (Means ‘Distance’ if ‘Relative’= TRUE)
E	Relative	BOOL	‘Relative’ distance if True, ‘Absolute’ position if False (= Default)
E	ExecutionMode	MC_EXECUTION_MODE	ENUM. Defines the chronological sequence of the FB. *mcImmediately* - the functionality is immediately valid and may influence the on-going motion but not the state (note: is the default behaviour) *mcQueued* - Same functionality as buffer mode ‘Buffered’
VAR_OUTPUT			
B	Done	BOOL	‘Position’ has new value
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Note: ‘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 in the ‘Position’ parameter.			

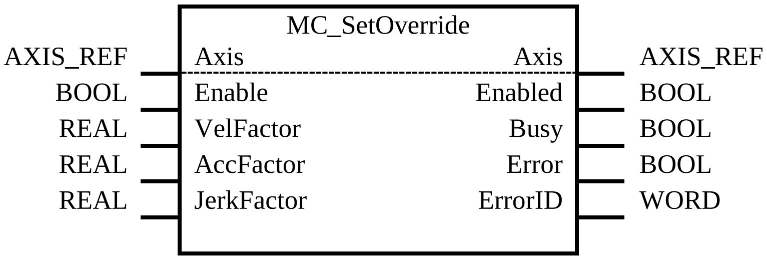


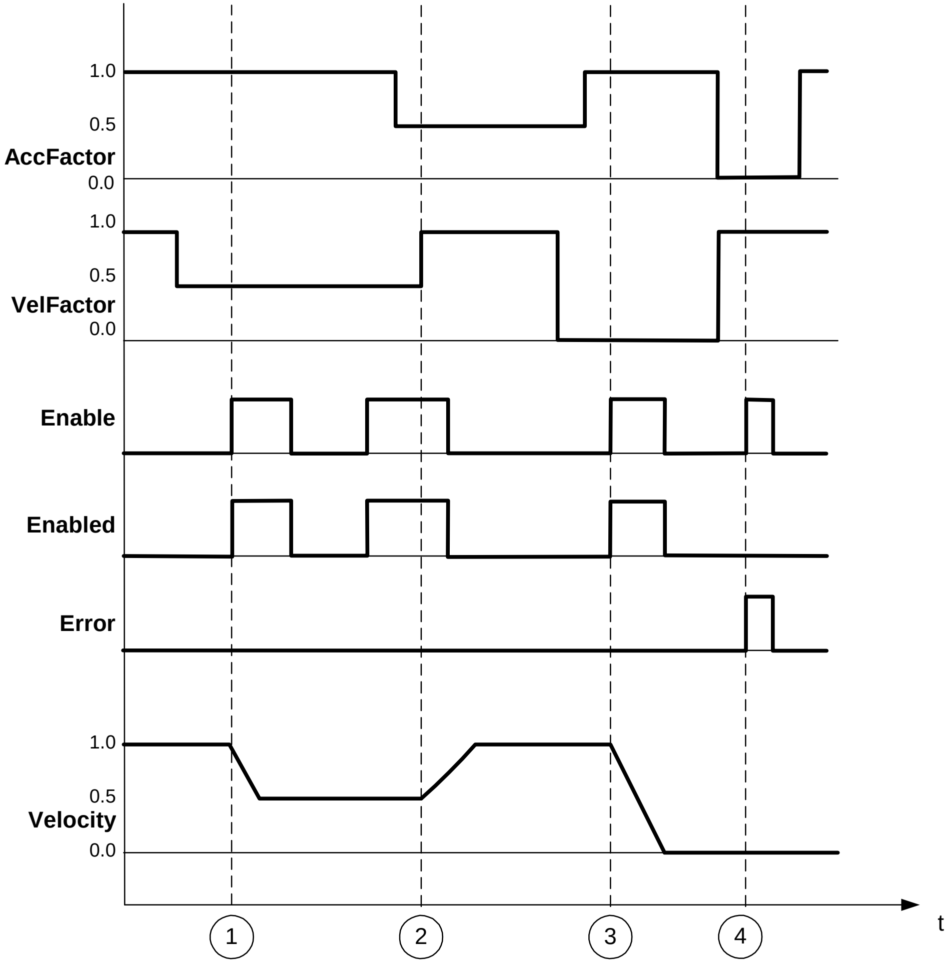
3.18. MC_SetOverride

FB-Name	**MC_SetOverride**		
This Function Block sets the values of override for the whole axis, and all functions that are working on that axis. The override parameters contribute as a factor to the calculation of the commanded velocity, acceleration and jerk of the motion.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Enable	BOOL	If SET, it writes the value of the override factor continuously. If RESET it should keep the last value.
B	VelFactor	REAL	New override factor for the velocity
E	AccFactor	REAL	New override factor for the acceleration/deceleration
E	JerkFactor	REAL	New override factor for the jerk
VAR_OUTPUT			
B	Enabled	BOOL	Signals that the override factor(s) is (are) set successfully
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:

1. The Input AccFactor acts on positive and negative acceleration (deceleration).
2. This Function Block sets the factor. The override factor is valid until a new override is set.
3. The default values of the override factor are 1.0.
4. The value of the overrides can be between 0.0 and 1.0. The behavior of values > 1.0 is vendor specific. Values < 0.0 are not allowed. The value 0.0 is not allowed for ‘AccFactor’ and ‘JerkFactor’.
5. The value 0.0 set to the ‘VelFactor’ stops the axis without bringing it to the state ‘Standstill’.
6. Override does not act on slave axes. (Axes in the state synchronized motion).
7. The Function Block does not influence the state diagram of the axis.
8. ‘VelFactor’ can be changed at any time and acts directly on the ongoing motion.
9. If in ‘Discrete’ motion, reducing the ‘AccFactor’ and/or ‘JerkFactor’ can lead to a position overshoot – a possible cause of damage
10. Activating this Function Block on an axis that is under control of MC_PositionProfile, MC_VelocityProfile, or MC_AccelerationProfile, is vendor specific.



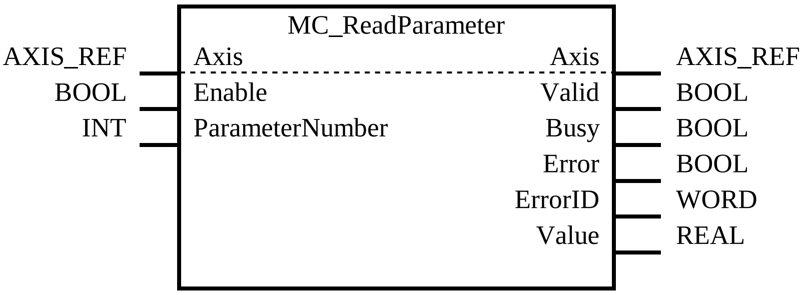


1. Axis Velocity changes to 50% with 100% of deceleration
2. Axis Velocity changes back to 100% with 50% acceleration
3. Axis Velocity moves to 0% with 100% deceleration
4. No Change, because AccFactor 0.0 is not allowed; Error is set

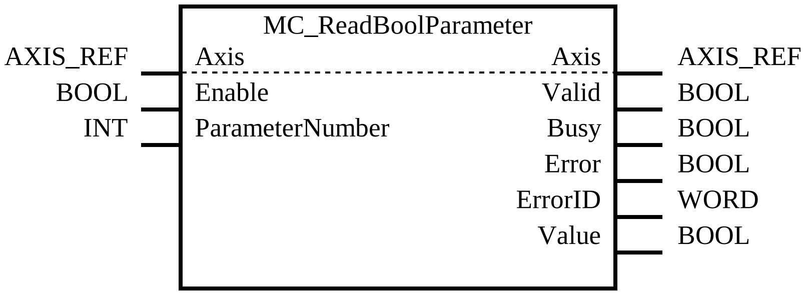
Figure 36: Graphical explanation of MC_SetOverride

3.19. MC_ReadParameter & MC_ReadBoolParameter

FB-Name			**MC_ReadParameter**	
This Function Block returns the value of a vendor specific parameter. The returned Value has to be converted to Real if necessary. If not possible, the vendor has to supply a vendor specific FB to read the parameter.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
	B	ParameterNumber	INT	Number of the parameter. One can also use symbolic parameter names which are declared as VAR CONST.
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	B	Value	REAL	Value of the specified parameter in the datatype, as specified by the vendor
Note: The parameters are defined in the table below.				



FB-Name			**MC_ReadBoolParameter**	
This Function Block returns the value of a vendor specific parameter with datatype BOOL.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
	B	ParameterNumber	INT	Number of the parameter. One can also use symbolic parameter names which are declared as VAR CONST.
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function block
	E	ErrorID	WORD	Error identification
	B	Value	BOOL	Value of the specified parameter in the datatype, as specified by the vendor
Note: The parameters are defined in the table below				



The parameters defined below have been standardized by the task force. Suppliers should use these parameters if they are offering this functionality.
All read-only parameters as defined may be writable during the initialization phase (supplier dependent).

These parameters are available for use in the application program, and typically are not intended for commissioning tools like operator panels, etc. (the drive is not visible – only the axis position)
Note: that the most used parameters are accessible via Function Blocks, and are not listed here.

(Note: PN is Parameter Number see FB MC_ReadParameter / MC_WriteParameter and Boolean versions)

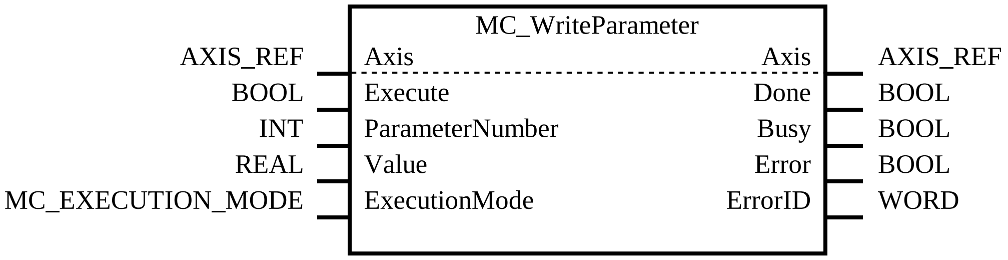
PN	Name	Datatype	B/E	R/W	Comments
1	CommandedPosition	REAL	B	R	Commanded position
2	SWLimitPos	REAL	E	R/W	Positive Software limit switch position
3	SWLimitNeg	REAL	E	R/W	Negative Software limit switch position
4	EnableLimitPos	BOOL	E	R/W	Enable positive software limit switch
5	EnableLimitNeg	BOOL	E	R/W	Enable negative software limit switch
6	EnablePosLagMonitoring	BOOL	E	R/W	Enable monitoring of position lag
7	MaxPositionLag	REAL	E	R/W	Maximal position lag
8	MaxVelocitySystem	REAL	E	R	Maximal allowed velocity of the axis in the motion system
9	MaxVelocityAppl	REAL	B	R/W	Maximal allowed velocity of the axis in the application
10	ActualVelocity	REAL	B	R	Actual velocity
11	CommandedVelocity	REAL	B	R	Commanded velocity
12	MaxAccelerationSystem	REAL	E	R	Maximal allowed acceleration of the axis in the motion system
13	MaxAccelerationAppl	REAL	E	R/W	Maximal allowed acceleration of the axis in the application
14	MaxDecelerationSystem	REAL	E	R	Maximal allowed deceleration of the axis in the motion system
15	MaxDecelerationAppl	REAL	E	R/W	Maximal allowed deceleration of the axis in the application
16	MaxJerkSystem	REAL	E	R	Maximum allowed jerk of the axis in the motion system
17	MaxJerkAppl	REAL	E	R/W	Maximum allowed jerk of the axis in the application

Table 5: Parameters for MC_ReadParameter and MC_WriteParameter

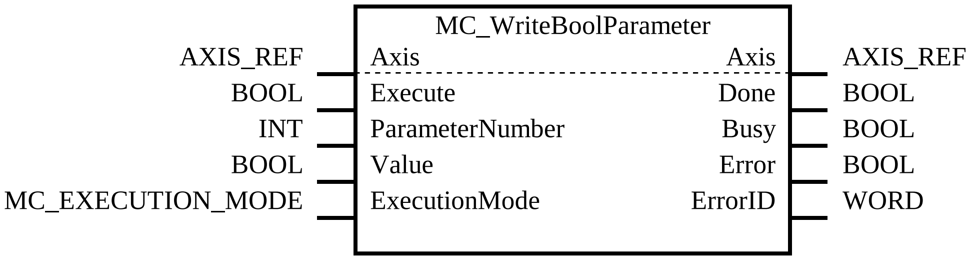
Extensions by any supplier or user are also allowed at the end of the list, although this can affect portability between different platforms. Parameter-numbers from 0 to 999 are reserved for the standard. Numbers greater than 999 indicate supplier-specific parameters.

3.20. MC_WriteParameter & MC_WriteBoolParameter

FB-Name		**MC_WriteParameter**	
This Function Block modifies the value of a vendor specific parameter.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	BOOL	Write the value of the parameter at rising edge
B	ParameterNumber	INT	Number of the parameter (correspondence between number and parameter is specified in the table above)
B	Value	REAL	New value of the specified parameter
E	ExecutionMode	MC_EXECUTION _MODE	Defines the chronological sequence of the FB. *mcImmediately* - the functionality is immediately valid and may influence the on-going motion but not the state (note: is the default behaviour) *mcQueued* - Same functionality as buffer mode 'Buffered'
VAR_OUTPUT			
B	Done	BOOL	Parameter successfully written
E	Busy	BOOL	The FB is not finished and new output values are to be expected.
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: The parameters are defined in the table above (under MC_ReadParameter, writing allowed)			

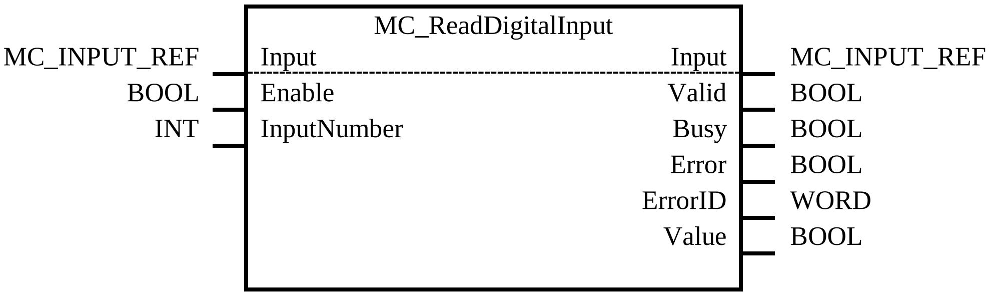


FB-Name			**MC_WriteBoolParameter**	
This Function Block modifies the value of a vendor specific parameter of type BOOL.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	BOOL	Write the value of the parameter at rising edge
	B	ParameterNumber	INT	Number of the parameter (correspondence between number and parameter is specified in the table above)
	B	Value	BOOL	New value of the specified parameter
	E	ExecutionMode	MC_EXECUTION _MODE	Defines the chronological sequence of the FB. *mcImmediately* - the functionality is immediately valid and may influence the on-going motion but not the state (note: is the default behaviour) *mcQueued* - Same functionality as buffer mode 'Buffered'.
VAR_OUTPUT				
	B	Done	BOOL	Parameter successfully written
	E	Busy	BOOL	The FB is not finished and new output values are to be expected.
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
Notes: The parameters are defined in the table above (under MC_ReadParameter, writing allowed)				



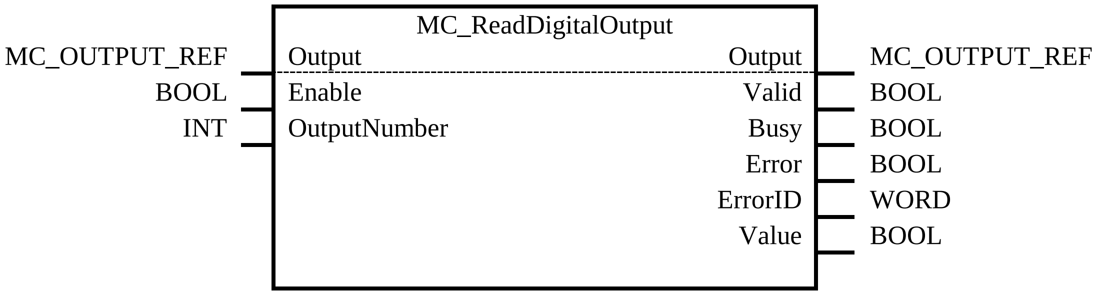
3.21. MC_ReadDigitalInput

FB-Name			**MC_ReadDigitalInput**	
This Function Block gives access to the value of the input, referenced by the datatype MC_INPUT_REF. It provides the value of the referenced input (BOOL)				
VAR_IN_OUT				
	B	Input	MC_INPUT_REF	Reference to the input signal source
VAR_INPUT				
	B	Enable	BOOL	Get the value of the selected input signal continuously while enabled
	E	InputNumber	INT	Selects the input. Can be part of MC_INPUT_REF, if only one single input is referenced.
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	B	Value	BOOL	The value of the selected input signal
Note: It is not guaranteed that the digital signal will be seen by the FB: a short pulse on the digital input could be over before the next Function Block cycle occurs.				



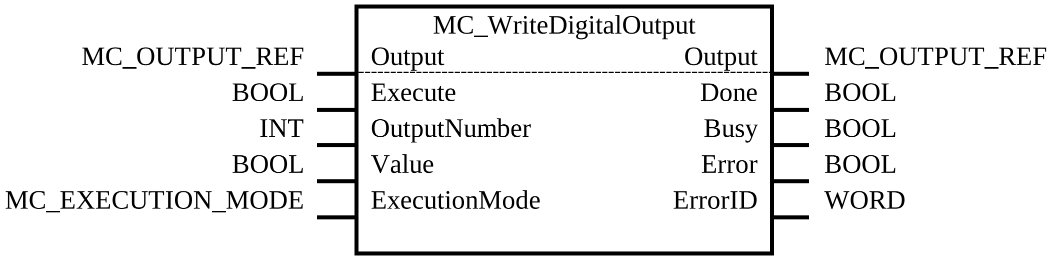
3.22. MC_ReadDigitalOutput

FB-Name			**MC_ReadDigitalOutput**	
This Function Block provides access to the value of a digital output, referenced by the datatype MC_OUTPUT_REF. It provides the value of the referenced output (BOOL).				
VAR_IN_OUT				
	B	Output	MC_OUTPUT_REF	Reference to the signal outputs
VAR_INPUT				
	B	Enable	BOOL	Get the value of the selected output signal continuously while enabled
	E	OutputNumber	INT	Selects the output. Can be part of MC_OUTPUT_REF, if only one single output is referenced.
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the the Function Block
	E	ErrorID	WORD	Error identification
	B	Value	BOOL	The value of the selected output signal
Note: It is not guaranteed that the digital signal will be seen by the FB: a short pulse on the digital output could be over before the next Function Block cycle occurs.				



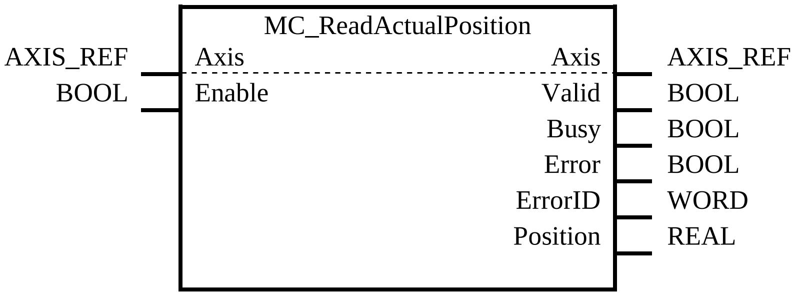
3.23. MC_WriteDigitalOutput

FB-Name		**MC_WriteDigitalOutput**	
This Function Block writes a value to the output referenced by the argument ‘Output’ once (with rising edge of Execute).			
VAR_IN_OUT			
B	Output	MC_OUTPUT_REF	Reference to the signal output
VAR_INPUT			
B	Execute	BOOL	Write the value of the selected output
E	OutputNumber	INT	Selects the output. Can be part of MC_OUTPUT_REF, if only one single input is referenced.
B	Value	BOOL	The value of the selected output
E	ExecutionMode	MC_EXECUTION_ MODE	Defines the chronological sequence of the FB. *mcImmediately* - the functionality is immediately valid and may influence the on-going motion but not the state (note: is the default behaviour) *mcQueued* - Same functionality as buffer mode ‘Buffered’
VAR_OUTPUT			
B	Done	BOOL	Writing of the output signal value is done
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: -			



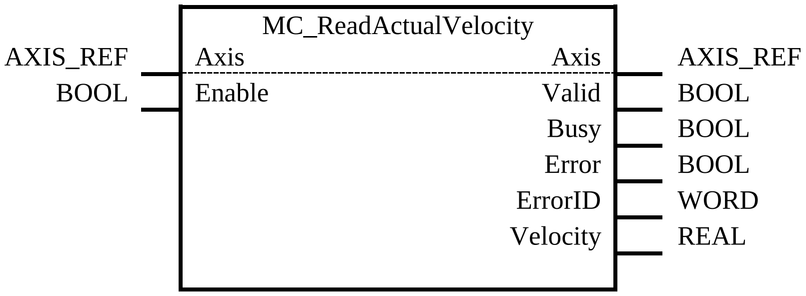
3.24. MC_ReadActualPosition

FB-Name			**MC_ReadActualPosition**	
This Function Block returns the actual position.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	B	Position	REAL	New absolute position (in axis' unit [u])



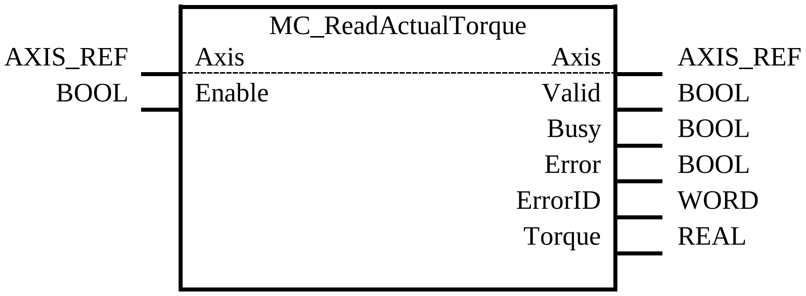
3.25. MC_ReadActualVelocity

FB-Name			**MC_ReadActualVelocity**	
This Function Block returns the value of the actual velocity as long as 'Enable' is set. 'Valid' is true when the data-output 'Velocity' is valid. If 'Enable' is Reset, the data loses its validity, and all outputs are reset, no matter if new data is available.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	B	Velocity	REAL	The value of the actual velocity (in axis' unit [u/s])
Notes: The output 'Velocity' can be a signed value				



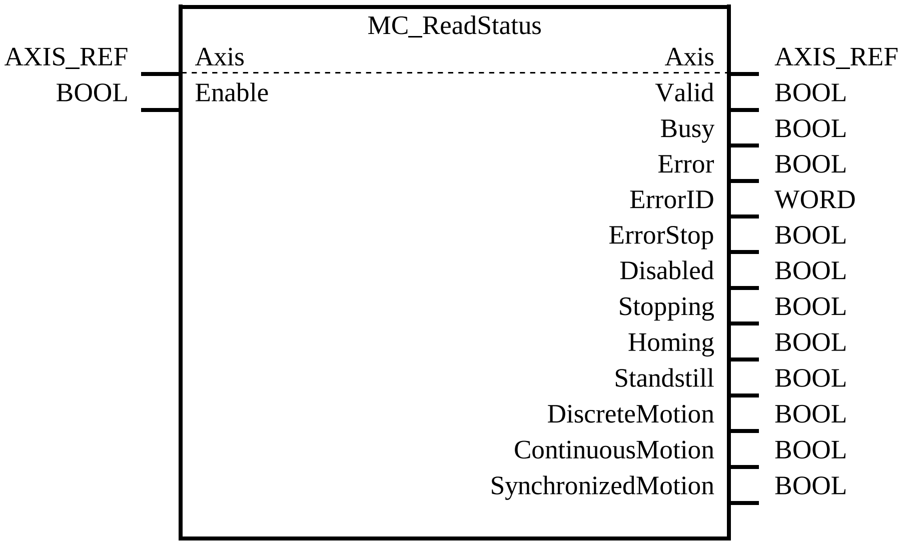
3.26. MC_ReadActualTorque

FB-Name		**MC_ReadActualTorque**		
This Function Block returns the value of the actual torque or force as long as 'Enable' is set. 'Valid' is true when the data-output 'Torque' is valid. If 'Enable' is Reset, the data loses its validity, and 'Valid' is also reset, no matter if new data is available.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	BOOL	A valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	B	Torque	REAL	The value of the actual torque or force (in technical units)
Notes: The output 'Torque' can be a signed value				



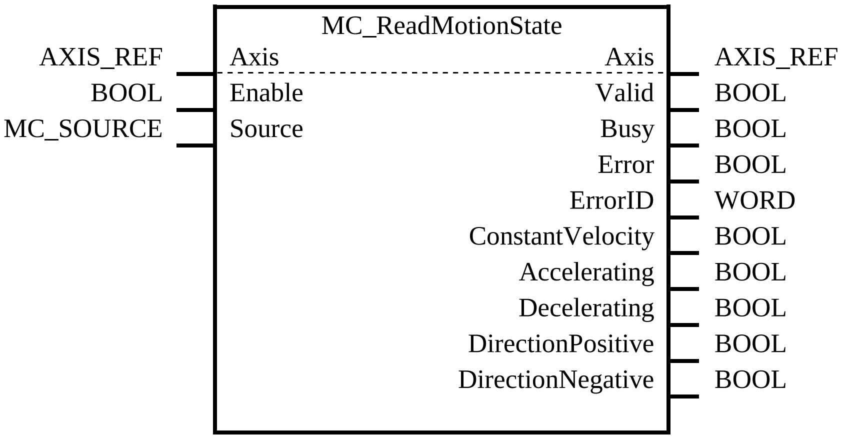
3.27. MC_ReadStatus

FB-Name		**MC_ReadStatus**	
This Function Block returns in detail the status of the state diagram of the selected axis.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT			
B	Valid	BOOL	A valid set of outputs is available at the FB
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
B	ErrorStop	BOOL	See state diagram
B	Disabled	BOOL	See state diagram
B	Stopping	BOOL	See state diagram
E	Homing	BOOL	See state diagram
B	Standstill	BOOL	See state diagram
E	DiscreteMotion	BOOL	See state diagram
E	ContinuousMotion	BOOL	See state diagram
E	SynchronizedMotion	BOOL	See state diagram



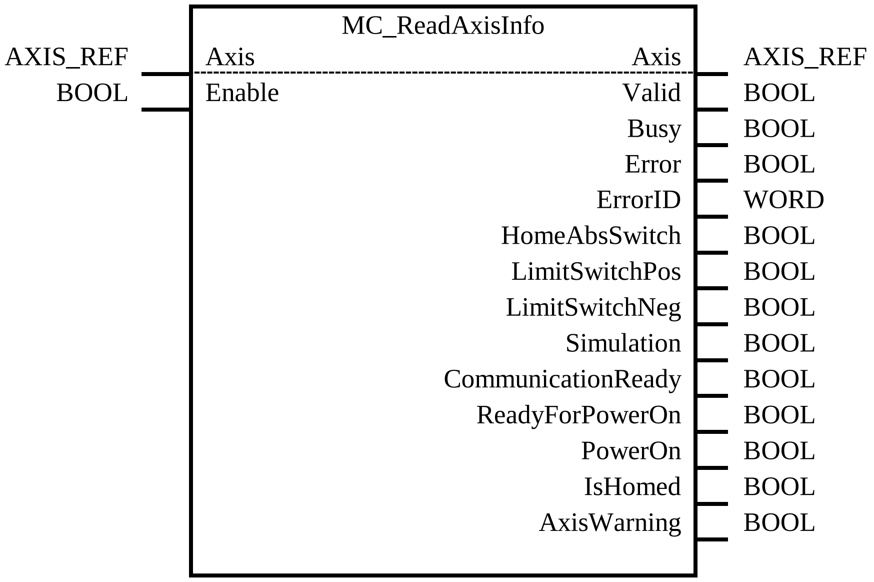
3.28. MC_ReadMotionState

FB-Name		**MC_ReadMotionState**	
This Function Block returns in detail the status of the axis with respect to the motion currently in progress.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT			
B	Enable	BOOL	Get the value of the parameter continuously while enabled
E	Source	MC_SOURCE	Defines the source of the relevant data: mcCommandedValue; mcSetValue, mcActualValue.
VAR_OUTPUT			
B	Valid	BOOL	True if a valid set of outputs available
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function block
E	ErrorID	WORD	Error identification
E	ConstantVelocity	BOOL	Velocity is constant. Velocity may be 0. For the actual value a window is applicable (window is vendor specific)
E	Accelerating	BOOL	Increasing the absolute value of the velocity
E	Decelerating	BOOL	Decreasing the absolute value of the velocity
E	DirectionPositive	BOOL	Signals that the position is increasing
E	DirectionNegative	BOOL	Signals that the position is decreasing



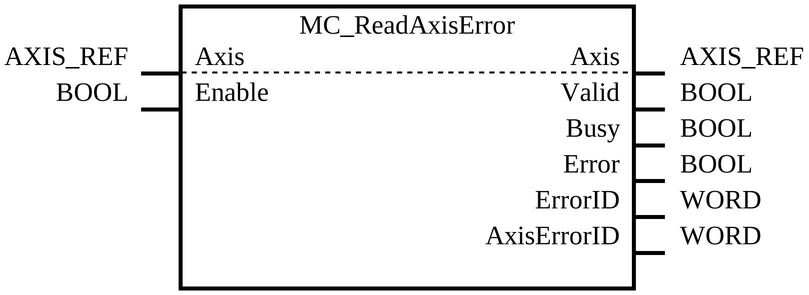
3.29. MC_ReadAxisInfo

FB-Name			**MC_ReadAxisInfo**	
This Function Block reads information concerning an axis, like modes, inputs directly related to the axis, and certain status information.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the axis information constantly while enabled
VAR_OUTPUT				
	B	Valid	BOOL	True if a valid set of outputs is available
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	E	HomeAbsSwitch	BOOL	Digital home switch input is active
	E	LimitSwitchPos	BOOL	Positive hardware end switch is active
	E	LimitSwitchNeg	BOOL	Negative hardware end switch is active
	E	Simulation	BOOL	Axis is in simulation mode (e.g. motor is simulated)
	E	CommunicationReady	BOOL	“Network” is initialized and ready for communication
	E	ReadyForPowerOn	BOOL	Drive is ready to be enabled (power on)
	E	PowerOn	BOOL	If TRUE shows that the power stage is switched ON
	E	IsHomed	BOOL	The absolute reference position is known for the axis (axis is homed)
	E	AxisWarning	BOOL	Warning(s) on the axis is present



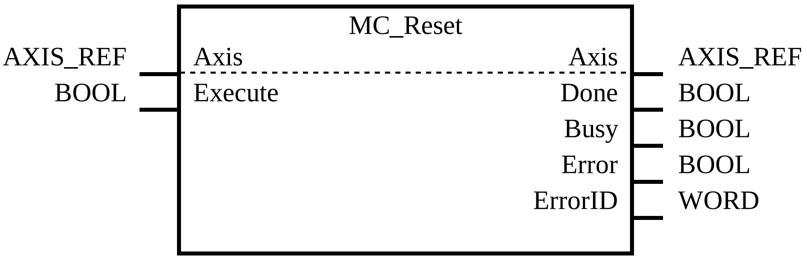
3.30. MC_ReadAxisError

FB-Name			**MC_ReadAxisError**	
This Function Block presents general axis errors not relating to the Function Blocks. (for instance axis errors, drive errors, communication errors)				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	BOOL	True if a valid output is available at the FB
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	B	ErrorID	WORD	Error identification
	E	AxisErrorID	WORD	The value of the axis error. These values are vendor specific
Notes: -				



3.31. MC_Reset

FB-Name		**MC_Reset**		
This Function Block makes the transition from the state ‘ErrorStop’ to ‘Standstill’ or ‘Disabled’ by resetting all internal axis-related errors – it does not affect the output of the FB instances.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	BOOL	Resets all internal axis-related errors
VAR_OUTPUT				
	B	Done	BOOL	‘Standstill’ or ‘Disabled’ state is reached
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
Note: the application of MC_Reset in other states then the state ‘ErrorStop’ is vendor specific				

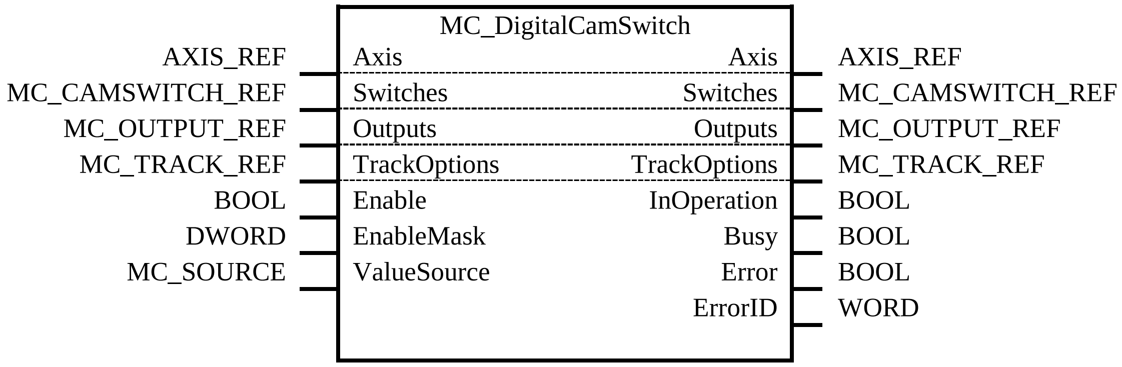


3.32. MC_DigitalCamSwitch

FB-Name	**MC_DigitalCamSwitch**		
This Function Block is the analogy to switches on a motor shaft: it commands a group of discrete output bits to switch in analogy to a set of mechanical cam controlled switches connected to an axis. Forward and backward movements are allowed.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
B	Switches	MC_CAMSWITCH_REF	Reference to the switching actions.
E	Outputs	MC_OUTPUT_REF	Reference to the signal outputs, directly related to the referenced tracks. (max. 32 per function block) (First output = first TrackNumber)
E	TrackOptions	MC_TRACK_REF	Reference to structure containing track related properties, e.g. the ON and OFF compensations per output/track.
VAR_INPUT			
B	Enable	BOOL	Enables the 'Switches' outputs
E	EnableMask	DWORD	32 bits of BOOL. Enables the different tracks. Least significant data is related to the lowest TrackNumber. With data SET (to '1' resp. TRUE) the related TrackNumber is enabled.
E	ValueSource	MC_SOURCE	Defines the source for axis values (e.g. positions): mcSetValue - Synchronization on set value mcActualValue - Synchronization on actual value
VAR_OUTPUT			
B	InOperation	BOOL	The commanded tracks are enabled
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:

- MC_CAMSWITCH_REFis a vendor specific reference to the pattern data.
- MC_OUTPUT_REF is a vendor specific structure linked to the (physical) outputs
- MC_TRACK_REF is vendor specific structure containing the track properties, e.g. the compensation per track (A track is a set of switches related to one output). It can contain the reference to the output also.
- This functionality is sometimes called PLS – Phase or Position or Programmable Limit Switch



Basic elements within the structure of MC_CAMSWITCH_REF

B/E	Parameter	Type	Description
B	TrackNumber	INT	TrackNumber is the reference to the track
B	FirstOnPosition [u]	REAL	Lower boundary where the switch is ON
B	LastOnPosition [u]	REAL	Upper boundary where the switch is ON
E	AxisDirection	INT	Both (=0; Default); Positive (1); Negative (2)
E	CamSwitchMode	INT	Position based (=0; Default); Time based (=1)
E	Duration	TIME	Coupled to time based CamSwitchMode

Basic elements within the array structure of MC_TRACK_REF

B/E	Parameter	Type	Description
E	OnCompensation	TIME	Compensation time with which the switching on is advanced or delayed in time per track.
E	OffCompensation	TIME	Time compensation the switching off is delayed per track.
E	Hysteresis [u]	REAL	Distance from the switching point (in positive and negative direction) in which the switch is not executed until the axis has left this area, in order to avoid multiple switching around the switching point.

This definition of a cam has a start and an end position, so the user can define each single cam, which has a **FirstOnPosition** and a **LastOnPosition** (or time). This Function Block is similar to a mechanical cam but has the additional advantages that the outputs can be set for a certain time, and to give it a time-compensation and a hysteresis.

CamSwitchMode can be Position, Time or other additional vendor specific types.

Duration: Time, the output of a time cam is ON

The time compensation (**OnCompensation** and **OffCompensation**) can be positive or negative. Negative means the output changes before the switching position is reached.

Hysteresis: This parameter avoids the phenomenon where the output continually switches if the axis is near the switching point and the actual position is jittering around the switching position. Hysteresis is part of MC_TRACK_REF, which means that a different hysteresis is possible for each track.

Example of MC_CAMSWITCH_REF

Parameter	Type	Switch01	Switch02	Switch03	Switch04	…	SwitchN
TrackNumber	INTEGER	*1*	*1*	*1*	*2*		
FirstOnPosition [u]	REAL	*2000*	*2500*	*4000*	*3000*		
LastOnPosition [u]	REAL	*3000*	*3000*	*1000*	*--*		
AxisDirection	INTEGER	*1=Pos*	*2=Neg*	*0=Both*	*0=Both*		
CamSwitchMode	INTEGER	*0=Position*	*0=Position*	*0=Position*	*1=Time*		
Duration	TIME	*--*	*--*	*--*	*1350*		

Note: Values are Examples

The example below uses the values from the example for MC_CAMSWITCH_REF above. It uses neither On/OffCompensation, nor hysteresis.
This is the behavior of the outputs, when the axis is moving continuously in the positive direction. The axis is a modulo axis with a modulo length of 5000 u.

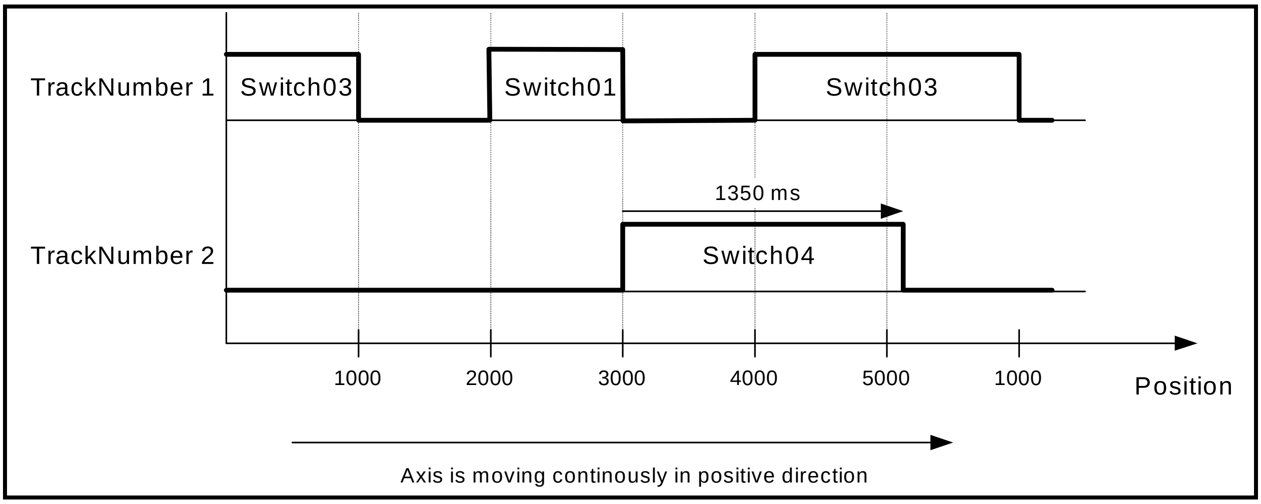


Figure 37: Example of MC_DigitalCamSwitch

Detailed description of Switch01.
This example additionally uses OnCompensation -125ms and OffCompensation +250ms.

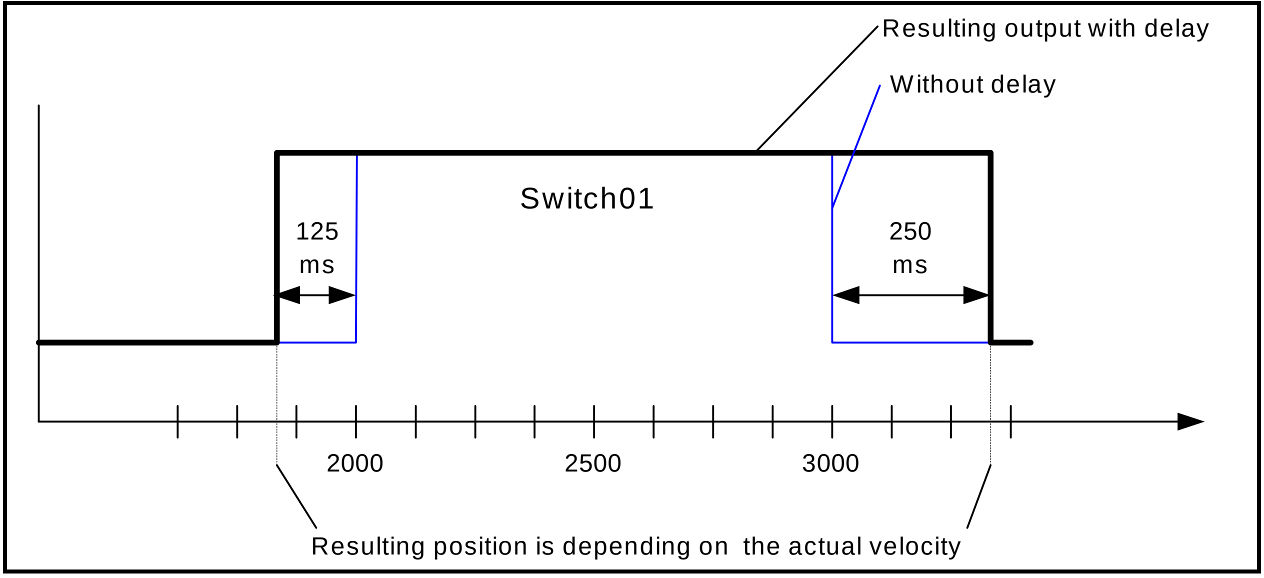

Resulting output with delay
Without delay
Switch01
125
ms
250
ms
2000
2500
3000
Resulting position is depending on the actual velocity

Figure 38: Detailed description of Switch01.

Below the behavior of the outputs, when the axis is moving continuously in the negative direction without On/OffCompensation and without Hysteresis.

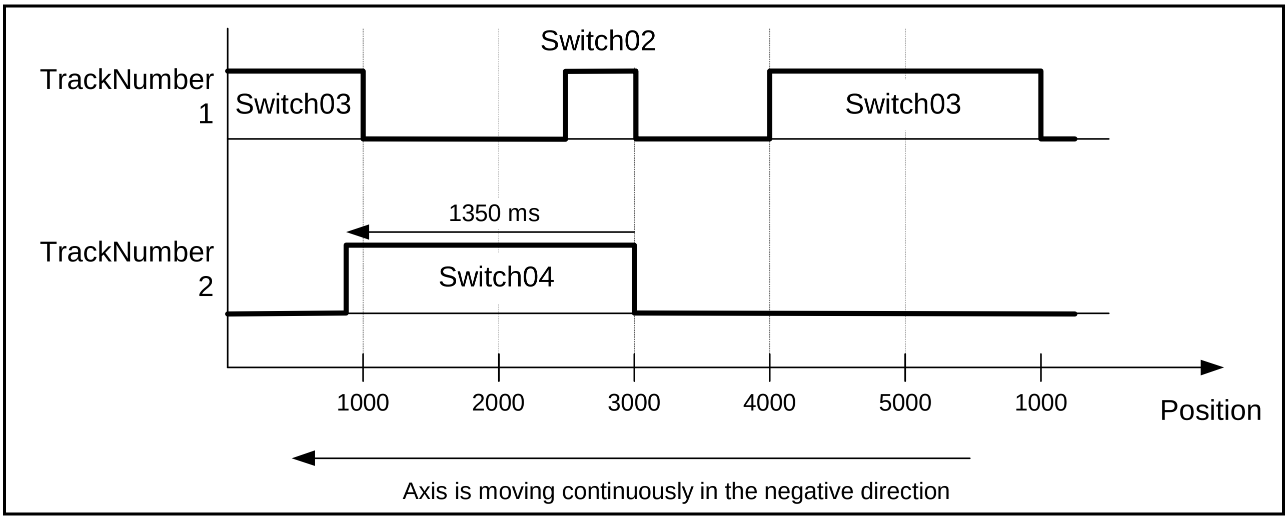
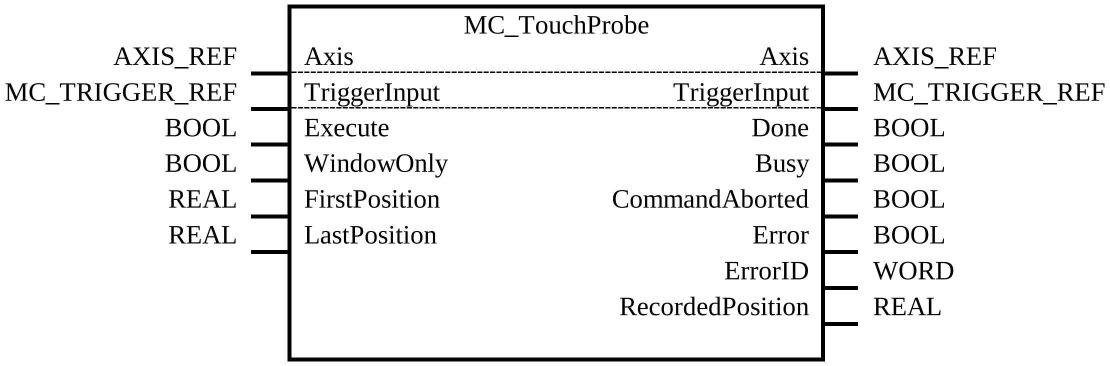


Figure 39: Example in negative direction

3.33. MC_TouchProbe

FB-Name		**MC_TouchProbe**		
This Function Block is used to record an axis position at a trigger event				
VAR_IN_OUT				
	B	Axis	AXIS_REF	Reference to the axis
	E	TriggerInput	MC_TRIGGER_REF	Reference to the trigger signal source. Trigger input may be specified by the AXIS_REF.
VAR_INPUT				
	B	Execute	BOOL	Starts touch probe recording at rising edge
	E	WindowOnly	BOOL	If SET, only use the window (defined hereunder) to accept trigger events
	E	FirstPosition	REAL	Start position from where (positive direction) trigger events are accepted (in technical units [u]). Value included in window.
	E	LastPosition	REAL	Stop position of the window (in technical units [u]). Value included in window.
VAR_OUTPUT				
	B	Done	BOOL	Trigger event recorded
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	E	CommandAborted	BOOL	‘Command’ is aborted by another command (MC_AbortTrigger)
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
	B	RecordedPosition	REAL	Position where trigger event occurred (in technical units [u])
Notes: 1. Intended for single shot operation, that is the first event after the rising edge at ‘Execute’ is valid for recording only. Possible following events are ignored 2. One Function Block instance should represent exactly one probing command 3. In case of multiple instances on the same probe and axis, the elements of MC_TRIGGER_REF should be extended with TouchProbeID - Identification of a unique probing command – this can be linked to MC_AbortTrigger				



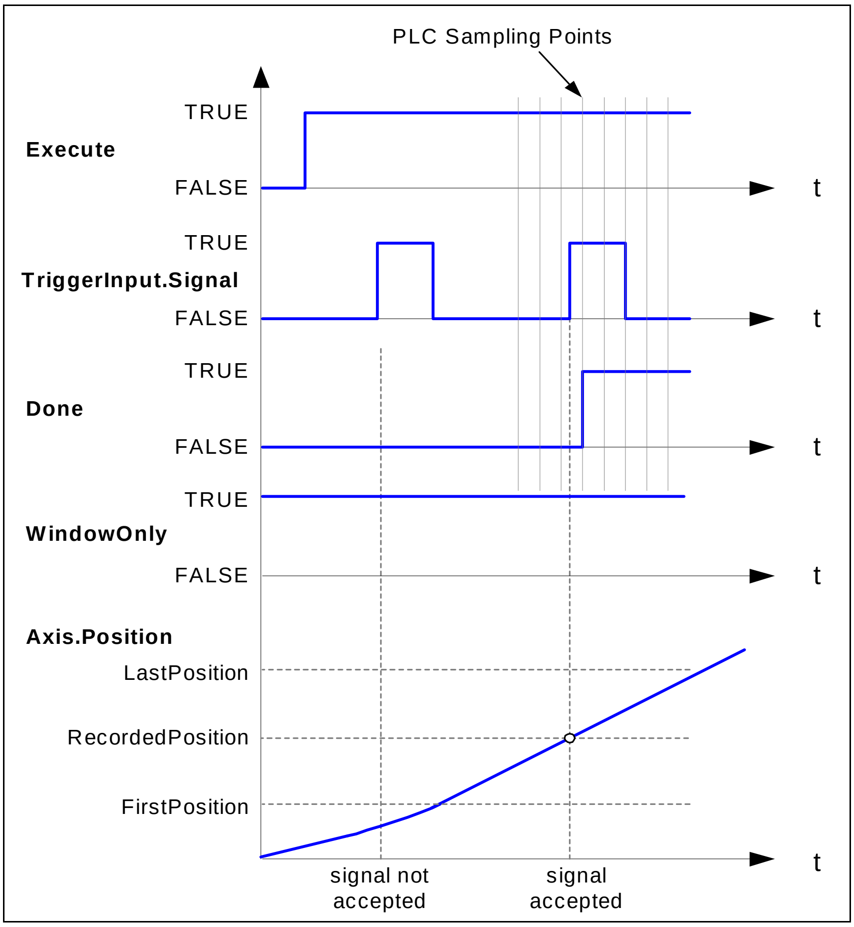


Figure 40: Timing example MC_TouchProbe

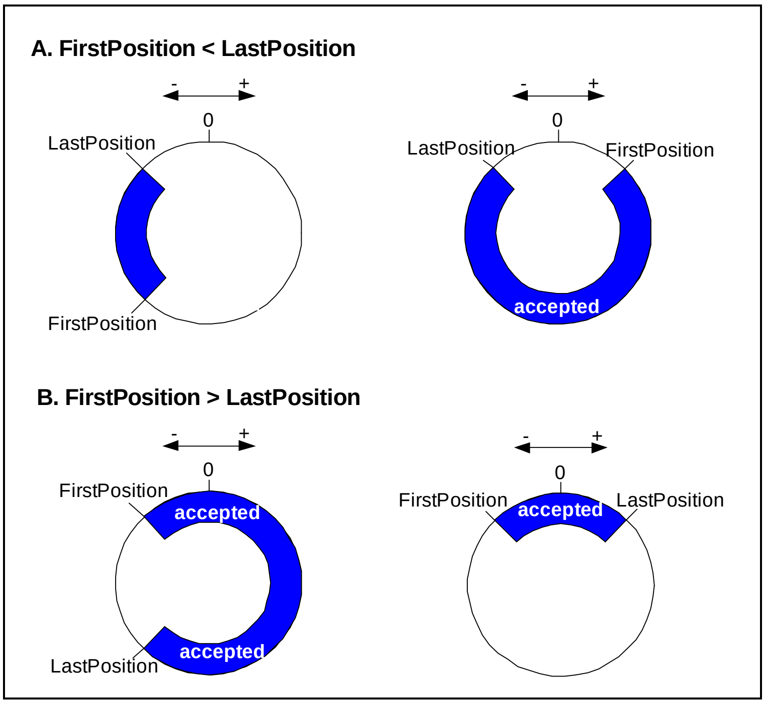
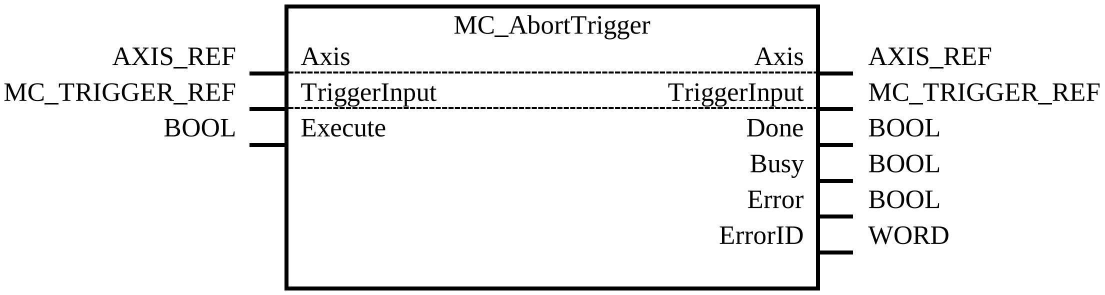


Figure 41: Examples of windows, where trigger events are accepted (for modulo axes)

3.34. MC_AbortTrigger

FB-Name		**MC_AbortTrigger**	
This Function Block is used to abort function blocks which are connected to trigger events (e.g. MC_TouchProbe)			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to the axis
E	TriggerInput	MC_TRIGGER_REF	Reference to the trigger signal source. 'TriggerInput' may be specified by the AXIS_REF. See Chapter 3.33 MC_TouchProbe
VAR_INPUT			
B	Execute	BOOL	Aborts trigger event at rising edge
VAR_OUTPUT			
B	Done	BOOL	Trigger functionality aborted
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: -			



4. Multi-Axis Function Blocks

With Multi-Axis Function Blocks a synchronized relationship exists between two or more axes. The synchronization can be related to time or position. Often this relationship is between a master axis and one or more slave axes. A master axis can be a virtual axis.
From the state diagram point of view, the multi-axis Function Blocks related to Camming and Gearing can be looked at as a master axis in one state (for instance: MC_MoveContinuous) and the slave axis in a specific synchronized state, called 'SychronizedMotion' (see State Diagram, chapter 2.1).

4.1. Remarks to Camming

A mechanical cam is a rotating or sliding piece in a mechanical linkage used especially in transforming rotary motion into linear motion or vice versa. It is often a part of a rotating wheel (e.g. an eccentric wheel) or shaft (e.g. a cylinder with an irregular shape) that strikes a lever at one or more points on its circular path. The cam can be a simple tooth, as is used to deliver pulses of power to a steam hammer, for example, or an eccentric disc or other shape that produces a smooth reciprocating (back and forth) motion in the follower, which is a lever making contact with the cam.
As such a cam creates a link between a master and one or more slaves in a position / position mode (see figure hereunder).
With motors and drives one can create the same position / position relationship but in this case via a so called Cam table listing the positions. So the relationship is converted to software and control.

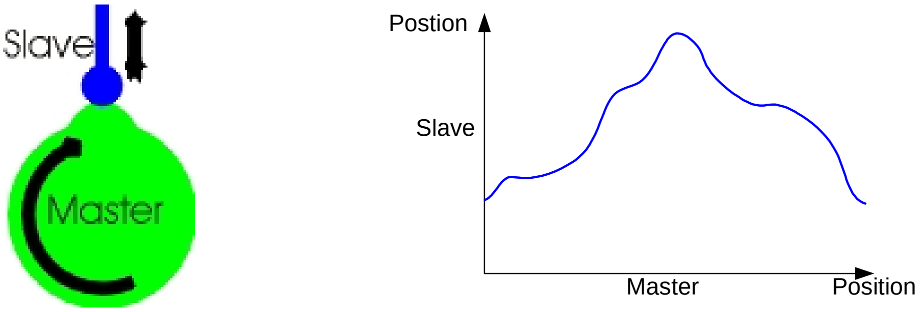


Figure 42: CAM profile illustration

Basically, one can differentiate between two types of Camming for both modulo and linear (or finite) master axes:

- **Periodic mode** - repeats the execution of the Cam profile on a continuous basis, even if the CAM profile does not match the modulo. This means that for a modulo axis with modulo is 360 degrees, and the CAM profiles is specified for 90 degrees it will be executed 4 times in a modulo. In reverse mode the profile is executed the inverse way.

- **Non-periodic mode** – the CAM profile is run only once. If the master position is outside of the Cam profile, the slave axis stays in synchronized motion and keeps the last position. In reverse mode, the CAM profile is not executed after having reached the 'EndOfProfile' position. The 90 degrees example above will be run only once.

Camming may be done with several combined cam tables which are executed sequentially, like a ramp-in, a production cycle, and a ramp-out. Between the different cam curves may be a gap (wait for trigger) in the execution. However, one could the buffered mode or use the output 'EndOfProfile' to start the next profile.

CAM table
Camming is done with one table (two dimensional – describing master and slave positions together) or two tables - for master and slave positions separately. The table should be strictly monotonic rising or falling, going both reverse and forward with the master.
It is allowed and possible to change tables while CAM is running and to change elements in the table while the CAM is running.
The generation and filling of the CAM table (master, slave) is performed by an external tool, which is supplier specific. The coupling of the FB MC_CamIn to the table is also supplier-specific.

Value presentation types

Master and slave axes may have different presentations:

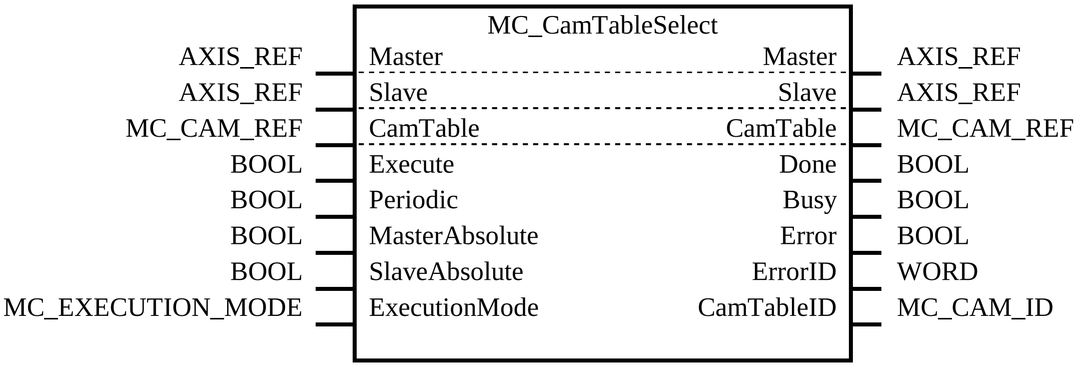
- Absolute values
- Relative to a starting position
- Relative steps (difference to the previous position)
- Equidistant or non-equidistant values.
- Polynomial Format. In this case the cam is described completely in the slave-table. The master-table is zero.

CAM Function Blocks

The advantages of having different Function Blocks for the camming functionality are a more transparent program execution flow and better performance in execution.

4.2. MC_CamTableSelect

FB-Name	**MC_CamTableSelect**		
This Function Block selects the CAM tables by setting the connections to the relevant tables			
VAR_IN_OUT			
E	Master	AXIS_REF	Reference to the master axis
E	Slave	AXIS_REF	Reference to the slave axis
B	CamTable	MC_CAM_REF	Reference to CAM description
VAR_INPUT			
B	Execute	BOOL	Selection at rising edge
E	Periodic	BOOL	1 = periodic, 0 = non periodic (single-shot)
E	MasterAbsolute	BOOL	1 = absolute; 0 = relative coordinates
E	SlaveAbsolute	BOOL	1 = absolute; 0 = relative coordinates
E	ExecutionMode	MC_EXECUTION_ MODE	Defines the chronological sequence of the FB. *mcImmediately* - the functionality is immediately valid and may influence the on-going motion but not the state (note: is the default behaviour) *mcQueued* - Same functionality as buffer mode 'Buffered'.
VAR_OUTPUT			
B	Done	BOOL	Pre-selection done
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
E	CamTableID	MC_CAM_ID	Identifier of CAM Table to be used in the MC_CamIn FB
Notes: • A virtual axis can be used as master axis • MC_CAM_REF is a supplier specific data type • MC_CAM_ID is a supplier specific data type • MC_CamTableSelect makes data available. This can include: 1. Starting point of a download of a profile 2. Start to generate a CAM profile • When the Done output is SET, the CamTableID is valid and ready for use in a MC_CamIn FB. • Possible parameters within the structure CAM_TABLE_REF are: o E MasterPositions REAL, List of expressions of the MasterValues for the 'CamTable' o E SlavePositions REAL, List of expressions of the SlaveValues for the 'CamTable'			



4.3. MC_CamIn

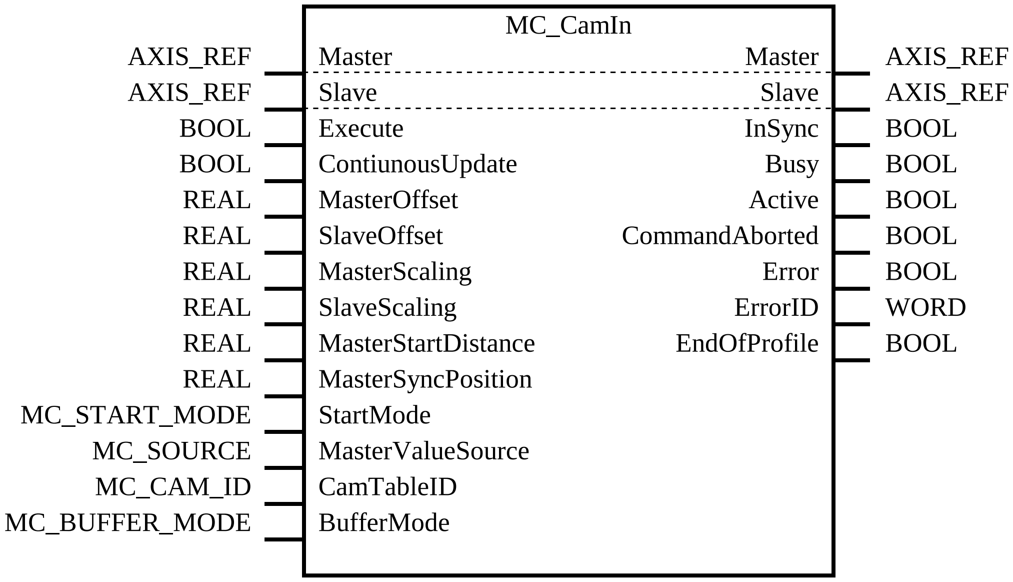
FB-Name		**MC_CamIn**	
This Function Block engages the CAM			
VAR_IN_OUT			
B	Master	AXIS_REF	Reference to the master axis
B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	BOOL	Start at rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate’
E	MasterOffset	REAL	Offset of the master shaft to cam.
E	SlaveOffset	REAL	Offset of slave table.
E	MasterScaling	REAL	Factor for the master profile (default = 1.0). From the slave point of view the master overall profile is multiplied by this factor
E	SlaveScaling	REAL	Factor for the slave profile (default = 1.0). The overall slave profile is multiplied by this factor.
E	MasterStartDistance	REAL	The master distance for the slave to start to synchronize to the master.
E	MasterSyncPosition	REAL	The position of the master in the CAM profile where the slave is in-sync with the master. (if the ‘MasterSyncPosition’ does not exist, at the first point of the CAM profile the master and slave are synchronized.) Note: the inputs acceleration, decelerations and jerk are not added here
E	StartMode	MC_START_MODE	Start mode: mcAbsolute, mcRelative, or mcRampIn
E	MasterValueSource	MC_SOURCE	Defines the source for synchronization: mcSetValue - Synchronization on master set value mcActualValue - Synchronization on master actual value
E	CamTableID	MC_CAM_ID	Identifier of CAM Table to be used, linked to output of MC_CamTableSelect
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	InSync	BOOL	Is TRUE if the set value = the commanded value.
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
E	EndOfProfile	BOOL	Pulsed output signaling the cyclic end of the CAM Profile It is displayed every time the end of the cam profile is reached. In reverse direction, the ‘EndOfProfile’ is displayed also at the end of the cam profile (in this case the first point of the cam profile)

Notes:

- It is not required that the master is stationary
- If the actual master and slave positions do not correspond to the offset values when MC_CamIn is executed, either an error occurs or the system deals with the difference automatically
- The Cam is placed either absolute or relative to the current master and slave positions.

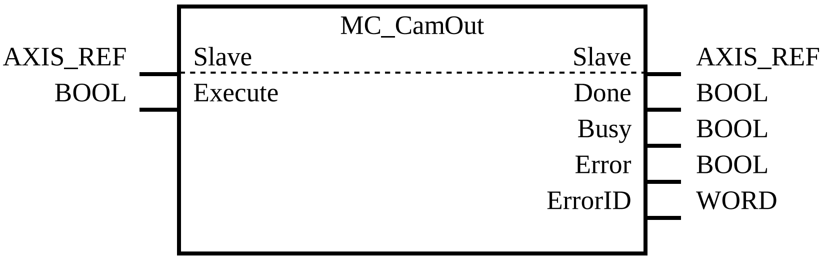
Absolute: the profile between master and slave is seen as an absolute relationship.
Relative: the relationship between master and slave is in a relative mode.

- Ramp-in is a supplier specific mode. It can be coupled to additional parameters, such as a master-distance parameter, acceleration parameter, or other supplier specific parameters where the slave to ramp-in into the cam profile (“ flying coupling”)
- This FB is not merged with the MC_CamTableSelect FB because this separation enables changes on the fly
- A mechanical analogy to a slave offset is a cam welded with additional constant layer thickness. Because of this the slave positions have a constant offset and the offset could be interpreted as axis offset of the master shaft, if linear guided slave tappets are assumed.



4.4. MC_CamOut

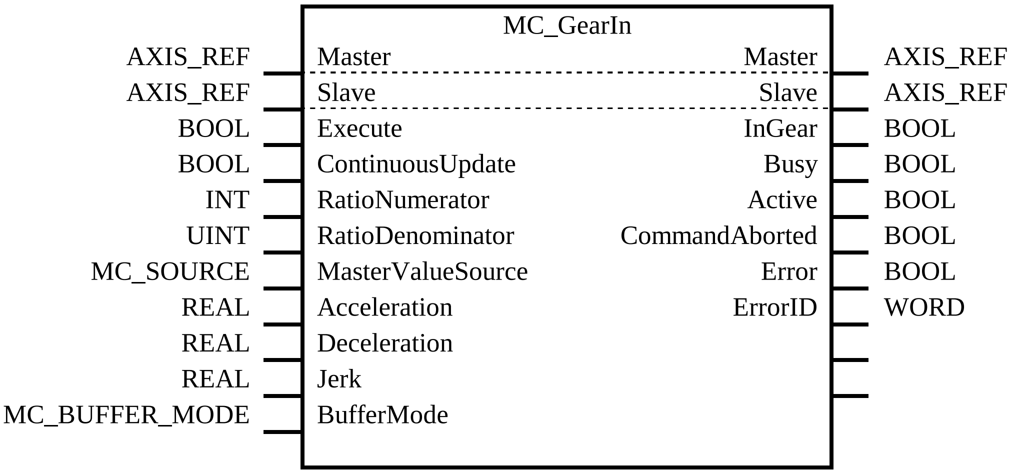
FB-Name		**MC_CamOut**	
This Function Block disengages the Slave axis from the Master axis immediately			
VAR_IN_OUT			
B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	BOOL	Start to disengage the slave from the master
VAR_OUTPUT			
B	Done	BOOL	Disengaging completed
E	Busy	BOOL	The FB is not finished and new output values are to be expected
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Notes: • It is assumed that this command is followed by another command, for instance MC_Stop, MC_GearIn, or any other command. If there is no new command, the default condition should be: maintain last velocity. • After issuing the FB there is no FB active on the slave axis till the next FB is issued (what can result in problems because no motion command is controlling the axis). Alternatively one can read the actual velocity via MC_ReadActualVelocity and issue MC_MoveVelocity on the slave axis with the actual velocity as input. The FB is here because of compatibility reasons			



4.5. MC_GearIn

FB-Name		**MC_GearIn**	
This Function Block commands a ratio between the VELOCITY of the slave and master axis.			
VAR_IN_OUT			
B	Master	AXIS_REF	Reference to the master axis
B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	BOOL	Start the gearing process at the rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input ‘ContinuousUpdate
B	RatioNumerator	INT	Gear ratio Numerator
B	RatioDenominator	UINT	Gear ratio Denominator
E	MasterValueSource	MC_SOURCE	Defines the source for synchronization: mcSetValue - Synchronization on master set value mcActualValue - Synchronization on master actual value
E	Acceleration	REAL	Acceleration for gearing in
E	Deceleration	REAL	Deceleration for gearing in
E	Jerk	REAL	Jerk of Gearing
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	InGear	BOOL	Is TRUE if the set value = the commanded value.
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	‘Command’ is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:
1. The slave ramps up to the ratio of the master velocity and locks in when this is reached. Any lost distance during synchronization is not caught up.
2. The gearing ratio can be changed while MC_GearIn is running, using a consecutive MC_GearIn command without the necessity to MC_GearOut first
3. After being ‘InGear’, a position locking or just a velocity locking is system specific.



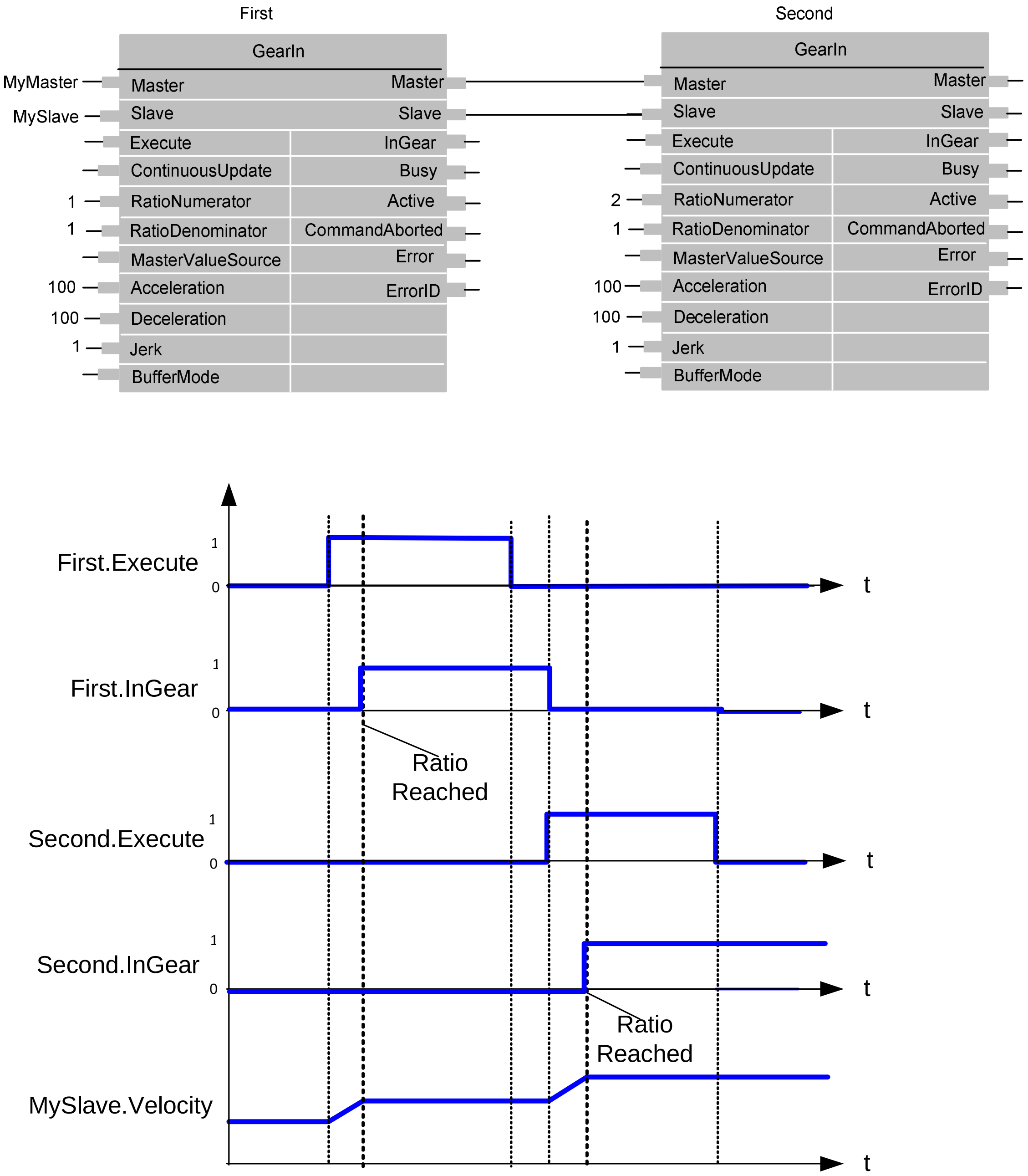
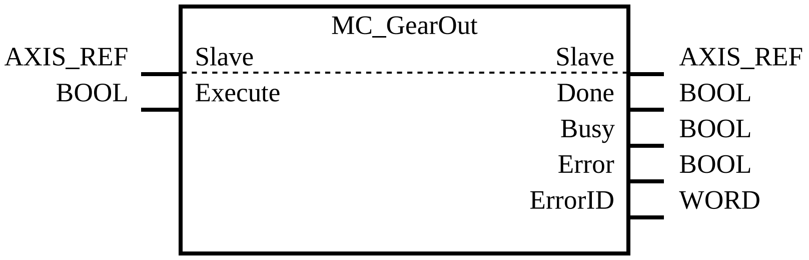


Figure 43: Gear timing diagram

4.6. MC_GearOut

FB-Name			**MC_GearOut**	
This Function Block disengages the Slave axis from the Master axis				
VAR_IN_OUT				
	B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT				
	B	Execute	BOOL	Start disengaging process at the rising edge
VAR_OUTPUT				
	B	Done	BOOL	Disengaging completed
	E	Busy	BOOL	The FB is not finished and new output values are to be expected
	B	Error	BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	WORD	Error identification
Notes: • It is assumed that this command is followed by another command, for instance MC_Stop, MC_GearIn, or any other command. If there is no new command, the default condition should be: maintain last velocity. • After issuing the FB there is no FB active on the slave axis till the next FB is issued (what can result in problems because no motion command is controlling the axis). Alternatively one can read the actual velocity via MC_ReadActualVelocity and issue MC_MoveVelocity on the slave axis with the actual velocity as input. The FB is here because of compatibility reasons				

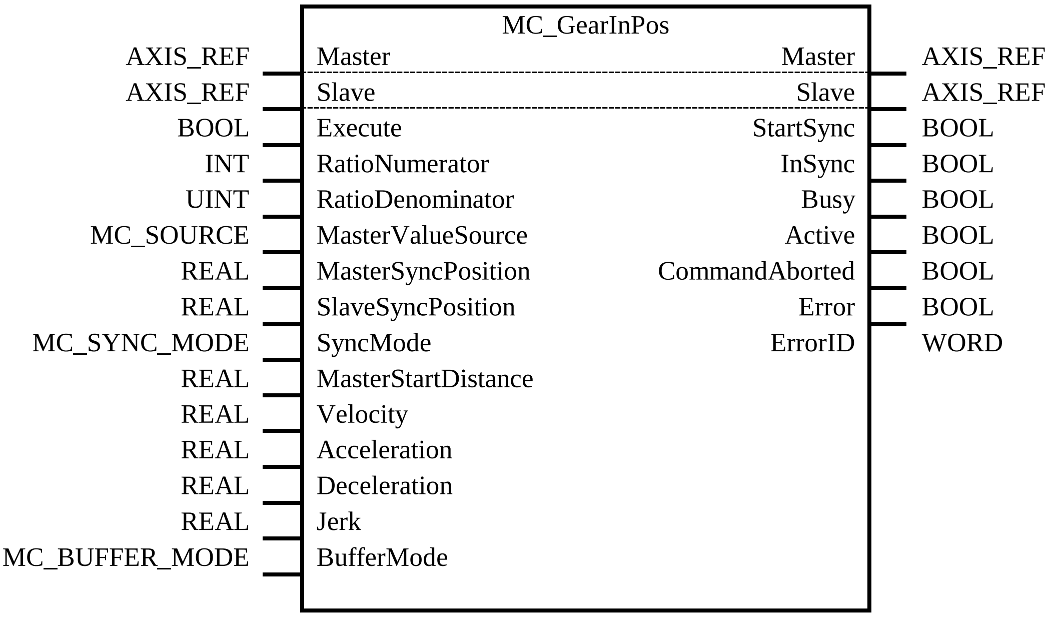


4.7. MC_GearInPos

FB-Name		**MC_GearInPos**	
This Function Block commands a gear ratio between the position of the slave and master axes from the synchronization point onwards.			
VAR_IN_OUT			
B	Master	AXIS_REF	Reference to the master axis
B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	BOOL	Start the gearing process at the rising edge
B	RatioNumerator	INT	Gear ratio Numerator
B	RatioDenominator	UINT	Gear ratio Denominator
E	MasterValueSource	MC_SOURCE	Defines the source for synchronization: mcSetValue - Synchronization on master set value mcActualValue - Synchronization on master actual value
B	MasterSyncPosition	REAL	The position of the master in the CAM profile where the slave is in-sync with the master. (if the 'MasterSyncPosition' does not exist, at the first point of the CAM profile the master and slave are synchronized.) Note: the inputs acceleration, decelerations and jerk are not added here
B	SlaveSyncPosition	REAL	Slave Position at which the axes are running in sync
E	SyncMode	MC_SYNC_MODE	Defines the way to synchronize (like 'mcShortest'; 'mcCatchUp'; 'mcSlowDown'). Vendor specific
E	MasterStartDistance	REAL	Master Distance for gear in procedure (when the Slave axis is started to get into synchronization)
E	Velocity	REAL	Maximum Velocity during the time difference 'StartSync' and 'InSync'
E	Acceleration	REAL	Maximum Acceleration during the time difference 'StartSync' and 'InSync'
E	Deceleration	REAL	Maximum Deceleration during the time difference 'StartSync' and 'InSync'
E	Jerk	REAL	Maximum Jerk during the time difference 'StartSync' and 'InSync'
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
E	StartSync	BOOL	Commanded gearing starts
B	InSync	BOOL	Is TRUE if the set value = the commanded value (is calculated set of values derived of master position and gear ratio.)
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification

Notes:

1. If ‘MasterStartDistance’ is implemented, any previous motion is continued until master crosses ‘MasterSyncPosition’ – ‘MasterStartDistance’ in the correct direction (according to the sign of ‘MasterStartDistance’). At that point in time the output ‘StartSync’ is set. When a ‘Stop’ command is executed on the ‘Slave’ axis before the synchronization has happened, it inhibits the synchronization and the function block issues ‘CommandAborted’
2. If the ‘MasterStartDistance’ is not specified, the system itself could calculate the set point for ‘StartSync’ based on the other relevant inputs.
3. The difference between the ‘SyncModes’ ‘CatchUp’ and ‘SlowDown’ is in the energy needed to synchronize. ‘SlowDown’ costs the lowest energy vs. ‘CatchUp’.



MC_GearInPos

AXIS_REF	Master	Master	AXIS_REF
AXIS_REF	Slave	Slave	AXIS_REF
BOOL	Execute	StartSync	BOOL
INT	RatioNumerator	InSync	BOOL
UINT	RatioDenominator	Busy	BOOL
MC_SOURCE	MasterValueSource	Active	BOOL
REAL	MasterSyncPosition	CommandAborted	BOOL
REAL	SlaveSyncPosition	Error	BOOL
MC_SYNC_MODE	SyncMode	ErrorID	WORD
REAL	MasterStartDistance		
REAL	Velocity		
REAL	Acceleration		
REAL	Deceleration		
REAL	Jerk		
MC_BUFFER_MODE	BufferMode		

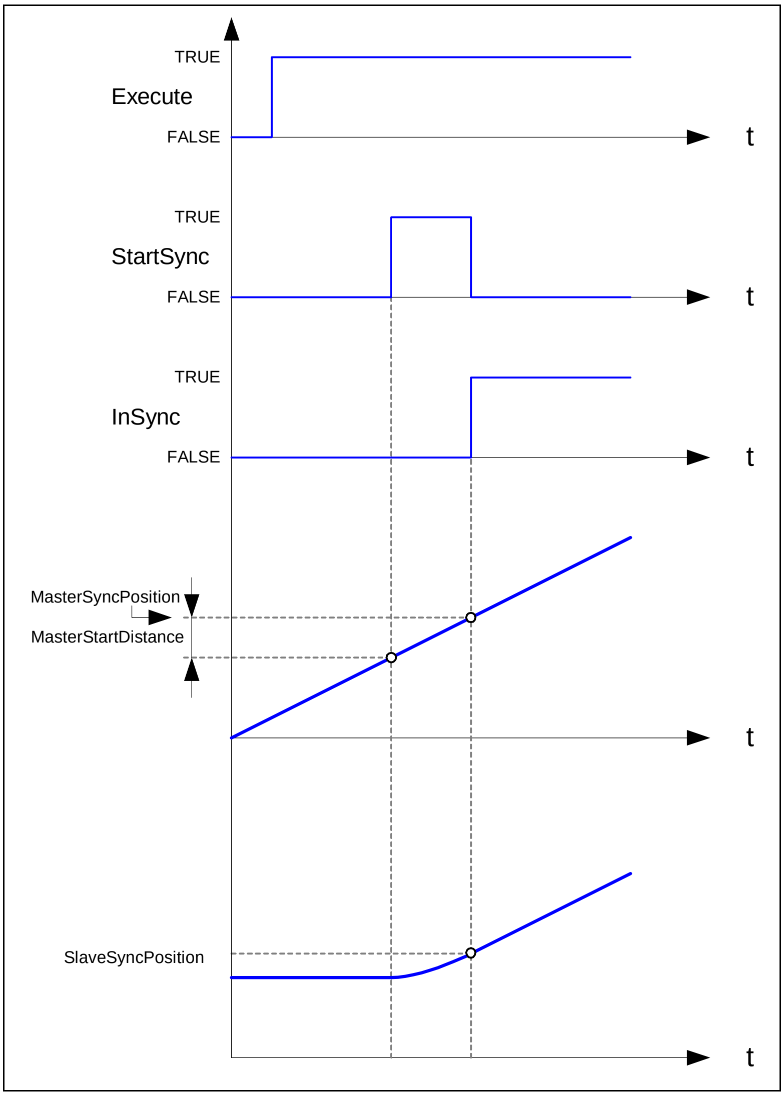


Figure 44: Timing Diagram of MC_GearInPos

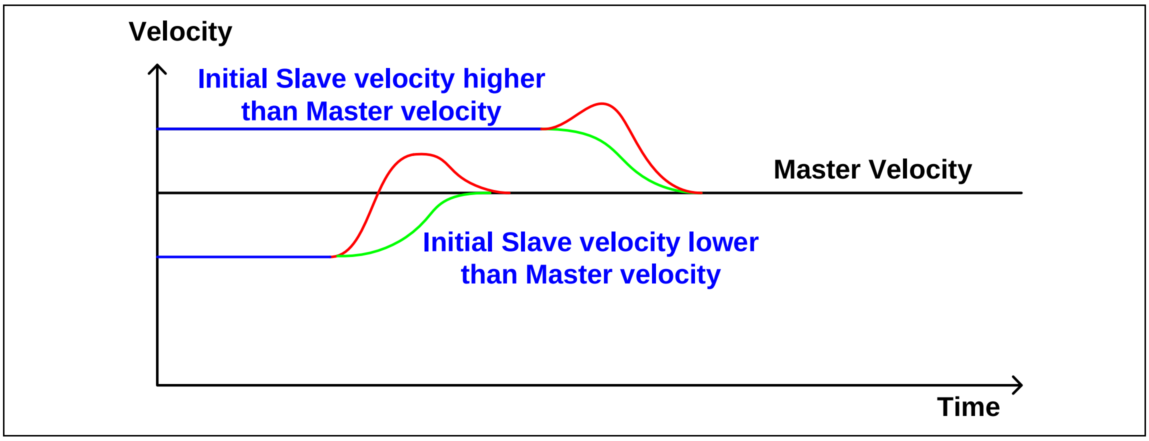


Figure 45: Example of the difference between 'SyncModes' 'SlowDown' (green) and 'CatchUp' (red) with different initial velocities of the slave

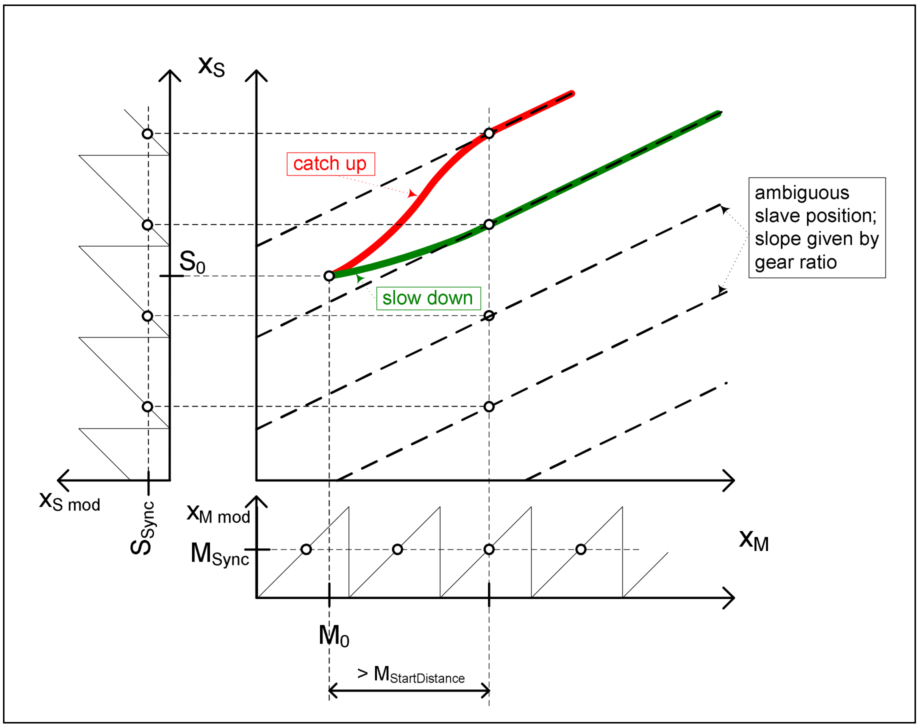


Figure 46: Example of MC_GearInPos where the initial velocity of the slave is in the same direction of the master

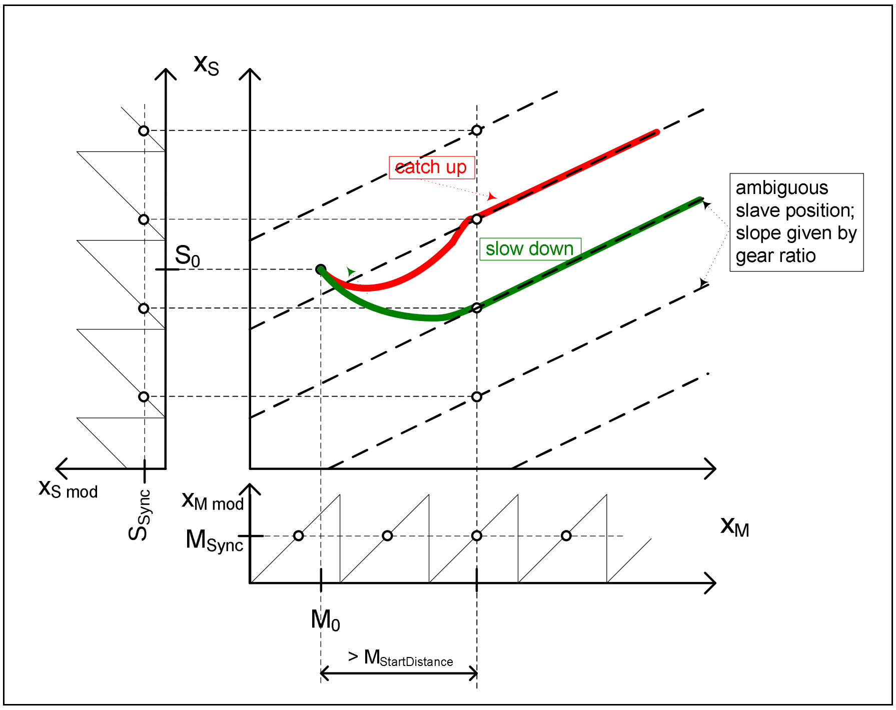
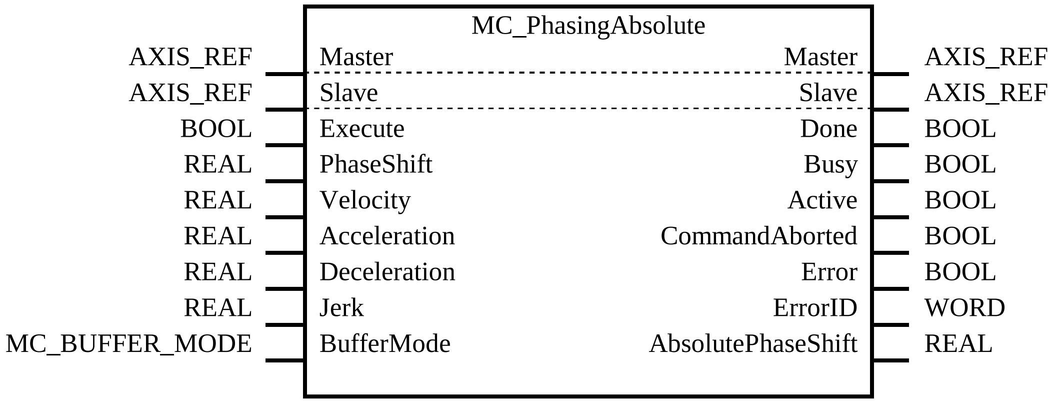


Figure 47: Example of MC_GearInPos where the initial velocity of the slave is in the inverse direction of the master

4.8. MC_PhasingAbsolute

FB-Name		**MC_PhasingAbsolute**	
This Function Block creates an absolute phase shift in the master position of a slave axis. The master position is shifted in relation to the real physical position. This is analogous to opening a coupling on the master shaft for a moment, and is used to delay or advance an axis to its master. The phase shift is seen from the slave. The master does not know that there is a phase shift experienced by the slave (MasterPos as seen from the SlaveAxis = PhysicalMasterPos + PhaseShiftValueSlaveAxis, the phase shift value has the character of a position offset) The phase shift remains until another 'Phasing' command changes it again.			
VAR_IN_OUT			
B	Master	AXIS_REF	Reference to the master axis
B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	BOOL	Start the phasing process at the rising edge
B	PhaseShift	REAL	Absolut phase difference in master position of the slave axis [u]
E	Velocity	REAL	Maximum Velocity to reach phase difference [u/s]
E	Acceleration	REAL	Maximum Acceleration to reach phase difference [u/s^2]
E	Deceleration	REAL	Maximum Deceleration to reach phase difference [u/s^2]
E	Jerk	REAL	Maximum Jerk to reach phase difference [u/s^3]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	Done	BOOL	Commanded phasing reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
E	AbsolutePhaseShift	REAL	Displays continuously the absolute phase shift [u] while Busy is set.
Note: • 'Phase', 'Velocity', 'Acceleration', 'Deceleration' and 'Jerk' of a phase shift are controlled by the FB. • For comparison: MC_MoveSuperimposed could also be used to act on the slave axis. MC_Phasing acts on the master side, as seen from the slave			



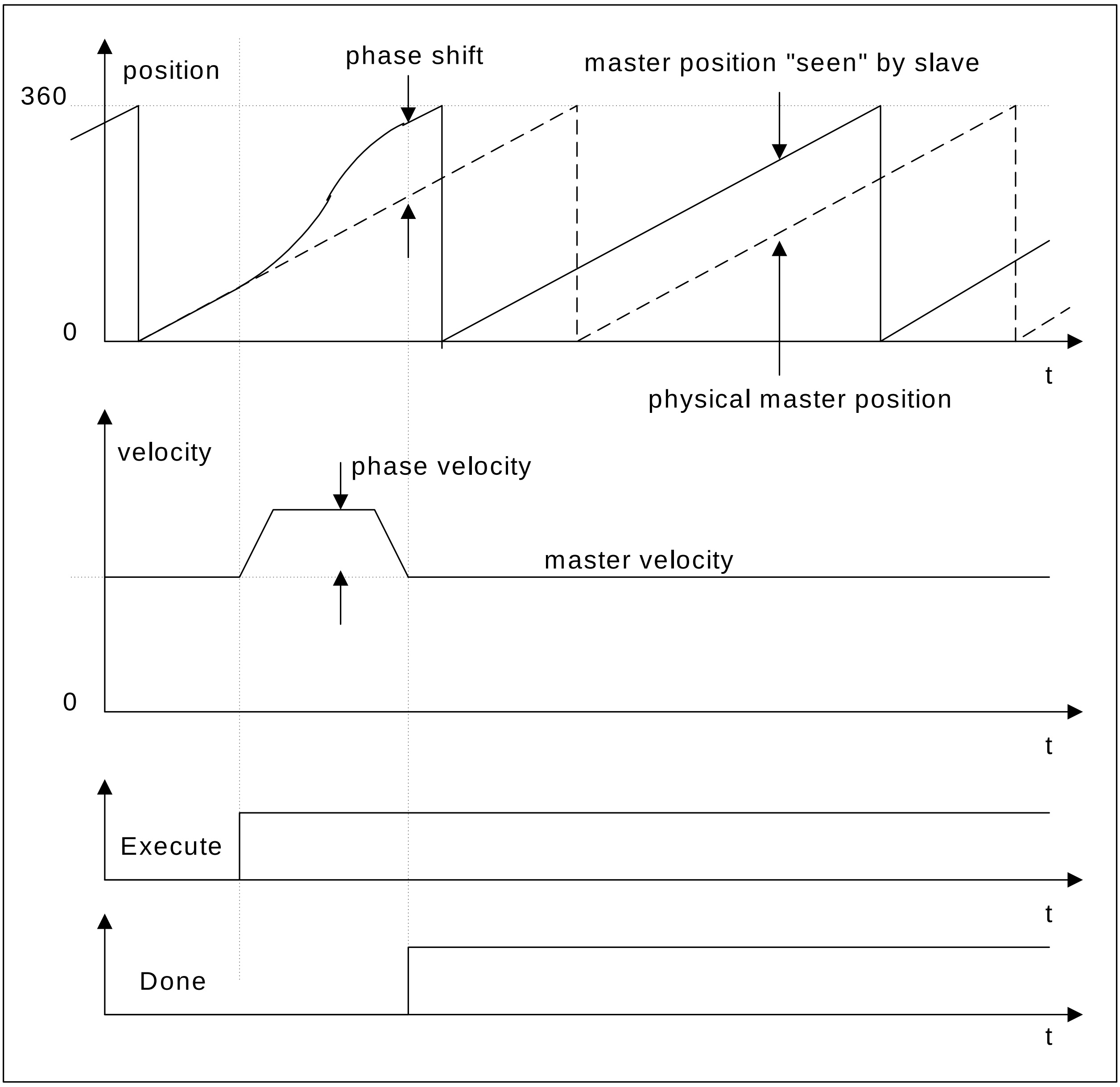


Figure 48: Timing example of MC_Phasing – both for absolute and relative

In this example the slave axis follows the master axis (in red – periodically) with a sine cam profile. Both the slave positions (green) and the slave velocity (blue) are shown. The effect of phasing is shown on the slave axis.

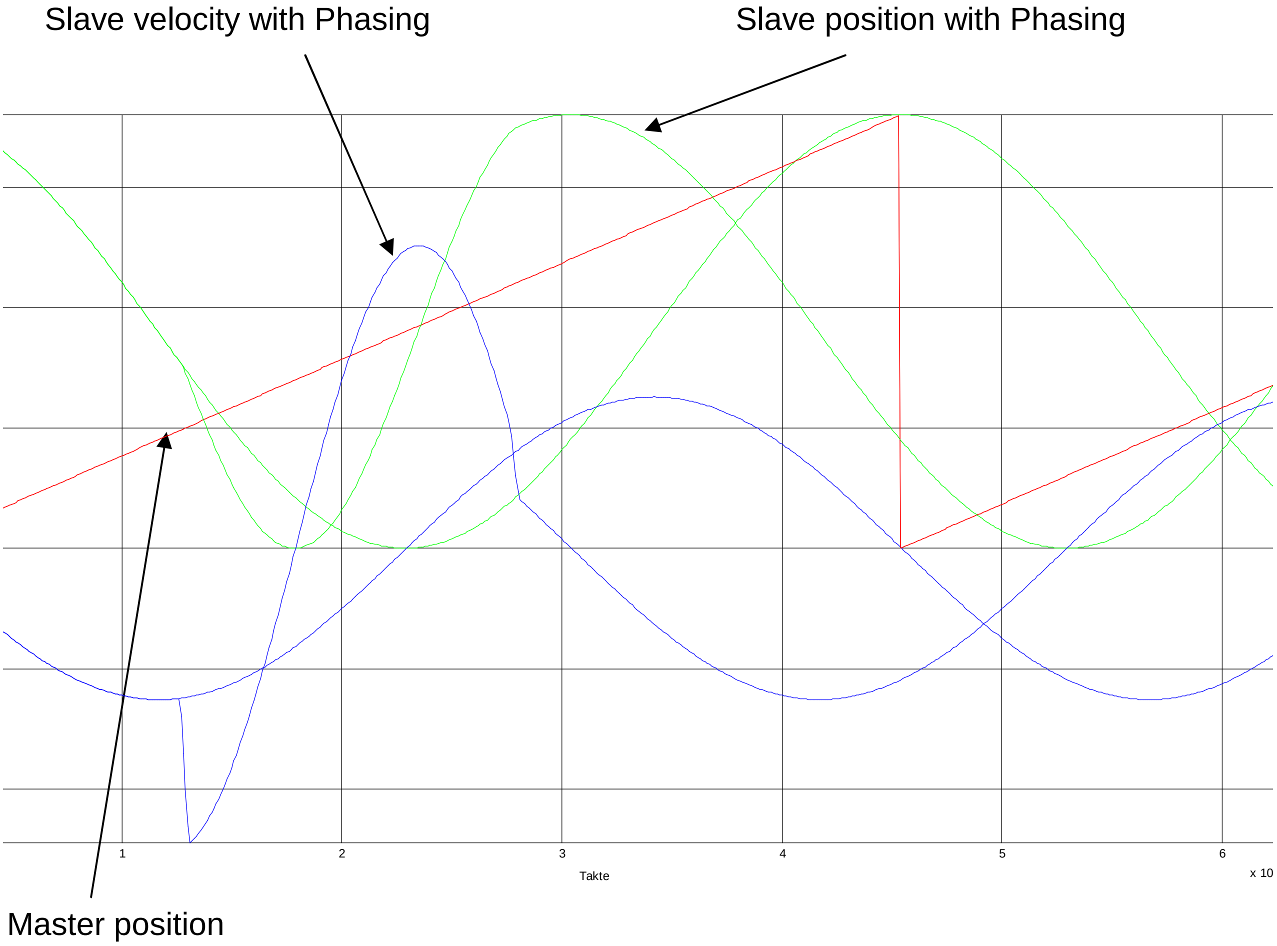
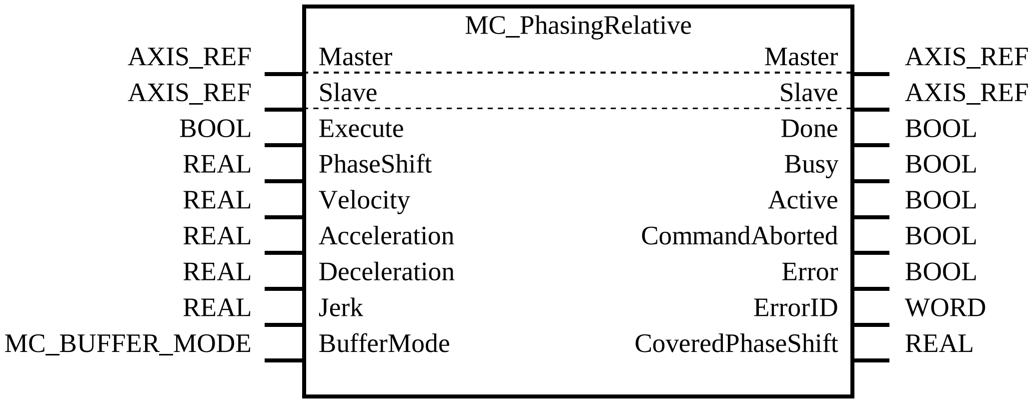


Figure 49: Example of MC_Phasing – both for absolute and relative

4.9. MC_PhasingRelative

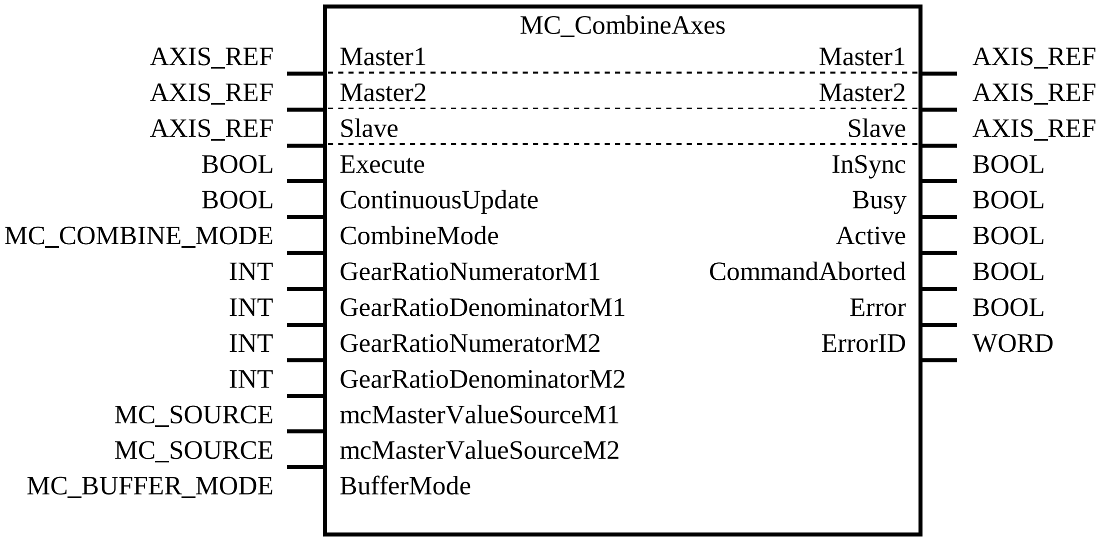
FB-Name		**MC_PhasingRelative**	
This Function Block creates a phase shift in the master position of a slave axis relative to the existing phase shift. The master position is shifted in relation to the real physical position. This is analogous to opening a coupling on the master shaft for a moment, and is used to delay or advance an axis to its master. The phase shift is seen from the slave. The master does not know that there is a phase shift experienced by the slave. (MasterPos as seen from SlaveAxis = PhysicalMasterPos + PhaseShiftValueSlaveAxis, the phase shift value has the character of a position offset) The phase shift remains until another 'Phasing' command changes it again. Relative phase shifts are added to each other for the applicable phase shift			
VAR_IN_OUT			
B	Master	AXIS_REF	Reference to the master axis
B	Slave	AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	BOOL	Start the phasing process at the rising edge
B	PhaseShift	REAL	Additional phase difference in master position of the slave axis [u]
E	Velocity	REAL	Maximum Velocity to reach phase difference [u/s]
E	Acceleration	REAL	Maximum Acceleration to reach phase difference [u/s^2]
E	Deceleration	REAL	Maximum Deceleration to reach phase difference [u/s^2]
E	Jerk	REAL	Maximum Jerk to reach phase difference [u/s^3]
E	BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the FB. See 2.4.2 Aborting versus Buffered modes
VAR_OUTPUT			
B	Done	BOOL	Commanded phasing reached
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
E	CoveredPhaseShift	REAL	Displays continuously the covered phase shift since it was started
Note: • 'Phase', 'Velocity', 'Acceleration', 'Deceleration' and 'Jerk' of a phase shift are controlled by the FB. • For comparison: MC_MoveSuperimposed could also be used to act on the slave axis. MC_Phasing acts on the master side, as seen from the slave			



For examples, see at MC_PhasingAbsolute in the previous paragraph.

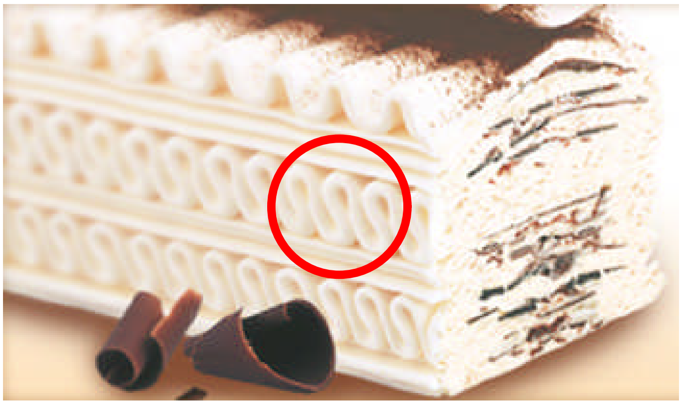
4.10. MC_CombineAxes

FB-Name		**MC_CombineAxes**	
This Function Block combines the motion of 2 axes into a third axis with selectable combination method. Basically it is a calculation of a new position setpoint based on the 2 position setpoints of the input axes. This FB is reflected in the state diagram like a synchronized motion type. As application example one can work with a separate profile synchronized to an object on a moving belt, or a rotating knife with flexible covered distance to be cut.			
VAR_IN_OUT			
B	Master1	AXIS_REF	Reference to the first master axis
B	Master2	AXIS_REF	Reference to the second master axis
B	Slave	AXIS_REF	Reference to the resulting combined axis. Can be a virtual axis or linked directly to a real axis
VAR_INPUT			
B	Execute	BOOL	Start the combination process at the rising edge
E	ContinuousUpdate	BOOL	See 2.4.6 The input 'ContinuousUpdate'
E	CombineMode	MC_COMBINE_MODE	Defines the type of combination applied to AxisOut : mcAddAxes : Addition of the 2 input axes positions mcSubAxes : Substraction of the 2 input axes positions
E	GearRatioNumeratorM1	INT	Numerator for the gear factor for master axis 1 towards the slave
E	GearRatioDenominatorM1	INT	Corresponding denominator for master axis 1
E	GearRatioNumeratorM2	INT	Numerator for the gear factor for master axis 2 towards the slave
E	GearRatioDenominatorM2	INT	Corresponding denominator for master axis 2
E	MasterValueSourceM1	MC_SOURCE	Defines the source for synchronization for master axis 1: mcSetValue - Synchronization on master set value mcActualValue - Synchronization on master actual value
E	MasterValueSourceM2	MC_SOURCE	Defines the source for synchronization for master axis 2: mcSetValue - Synchronization on master set value mcActualValue - Synchronization on master actual value
E	BufferMode	MC_BUFFER_MODE	Defines the behavior of the axis: modes are 'Aborting', 'Buffered', 'Blending'
VAR_OUTPUT			
B	InSync	BOOL	Is TRUE if the set value = the commanded value.
E	Busy	BOOL	The FB is not finished and new output values are to be expected
E	Active	BOOL	Indicates that the FB has control on the combined axis
E	CommandAborted	BOOL	'Command' is aborted by another command
B	Error	BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	WORD	Error identification
Note: To stop the motion, the FB has to be interrupted by another FB issuing a new command			


MC_CombineAxes
AXIS_REF Master1 — Master1 AXIS_REF
AXIS_REF Master2 — Master2 AXIS_REF
AXIS_REF Slave — Slave AXIS_REF
BOOL Execute — InSync BOOL
BOOL ContinuousUpdate — Busy BOOL
MC_COMBINE_MODE CombineMode — Active BOOL
INT GearRatioNumeratorM1 — CommandAborted BOOL
INT GearRatioDenominatorM1 — Error BOOL
INT GearRatioNumeratorM2 — ErrorID WORD
INT GearRatioDenominatorM2
MC_SOURCE mcMasterValueSourceM1
MC_SOURCE mcMasterValueSourceM2
MC_BUFFER_MODE BufferMode

MC_CombineAxes can generate special synchronized movements that are not possible or complex to generate in other ways. In the following example, a CAM FB and the result of a Gear FB are both synchronized to a conveyor master, are added to generate a virtual master for a MC_GearInPos function of the final axis that will execute the movement.

The particular application of this example could be a machine to deposit the icecream waving layers on top of the icecream base travelling through the freezer line in icecream factory. The dosing axis has to synchronize with a waving manner to the conveyor carrying the icecream base block. And it has to do this in a particular starting position and wave phase to achieve the expected result (therefore the GearInPos). With the CAM FB one can define different wave patterns easily (like the one longer in the top of icecream).



Another case application can be chocolate bars with decoration (individual bars in mouldings). The dosificator makes the wave synchronized with conveyor and returns for the next

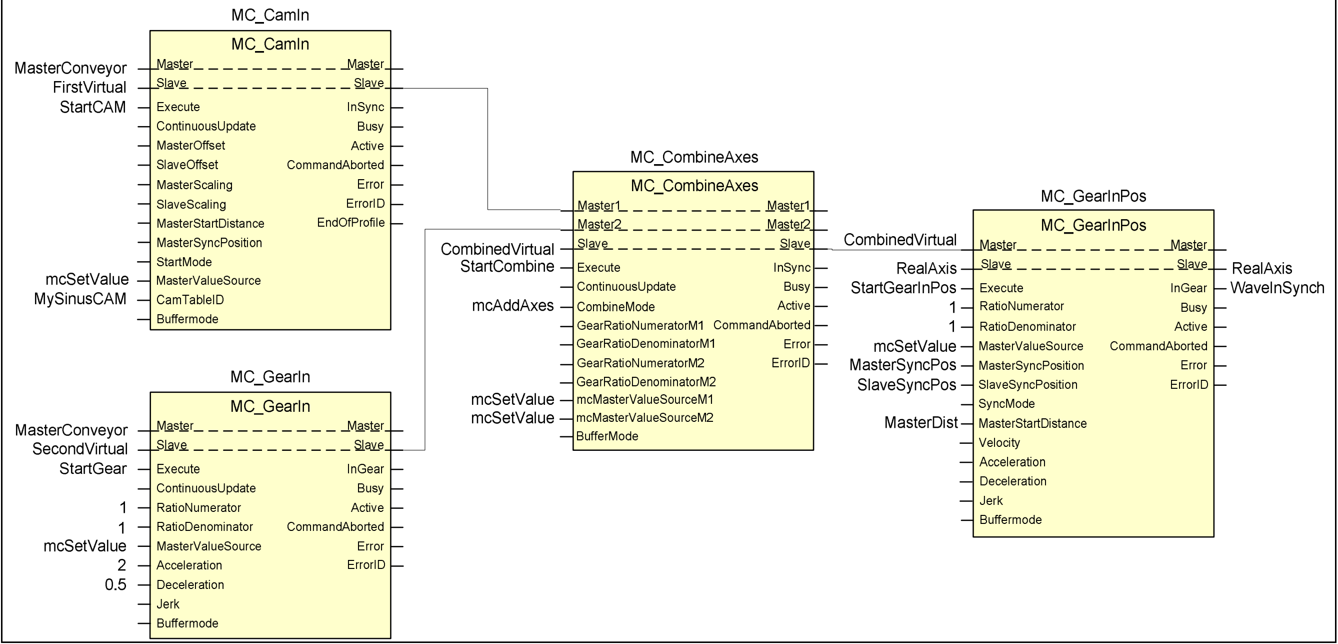


Figure 50: Application example of MC_CombineAxes

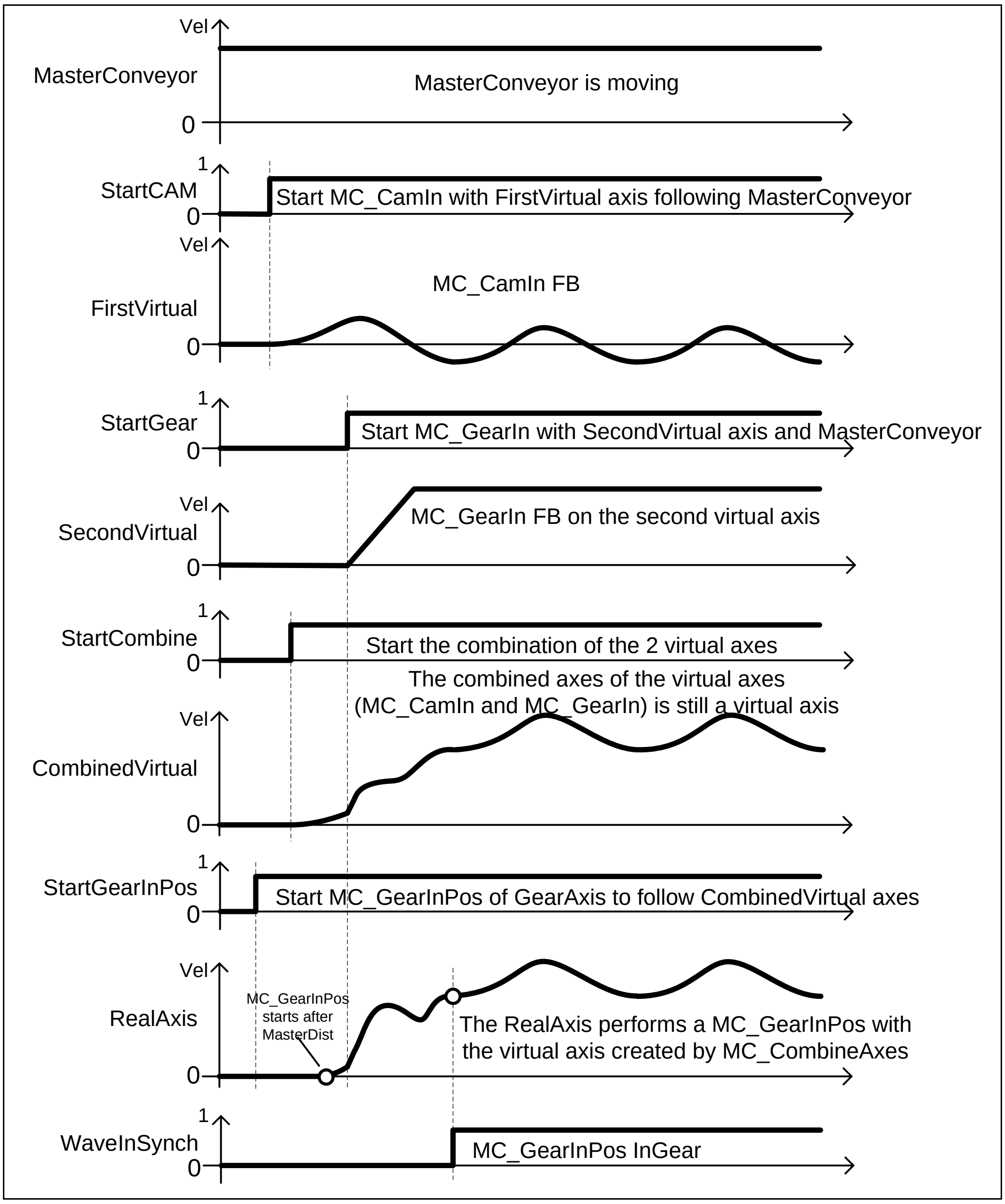

Vel
MasterConveyor
MasterConveyor is moving
0
1
StartCAM
0
Start MC_CamIn with FirstVirtual axis following MasterConveyor
Vel
MC_CamIn FB
FirstVirtual
0
1
StartGear
0
Start MC_GearIn with SecondVirtual axis and MasterConveyor
Vel
SecondVirtual
MC_GearIn FB on the second virtual axis
0
1
StartCombine
0
Start the combination of the 2 virtual axes
The combined axes of the virtual axes
(MC_CamIn and MC_GearIn) is still a virtual axis
Vel
CombinedVirtual
0
1
StartGearInPos
0
Start MC_GearInPos of GearAxis to follow CombinedVirtual axes
Vel
RealAxis
MC_GearInPos
starts after
MasterDist
The RealAxis performs a MC_GearInPos with
the virtual axis created by MC_CombineAxes
0
1
WaveInSynch
0
MC_GearInPos InGear

Figure 51: The corresponding timing diagram for MC_CombineAxes example

5. Application of MC FB – A Drilling Example with ‘Aborting’ versus ‘Blending’

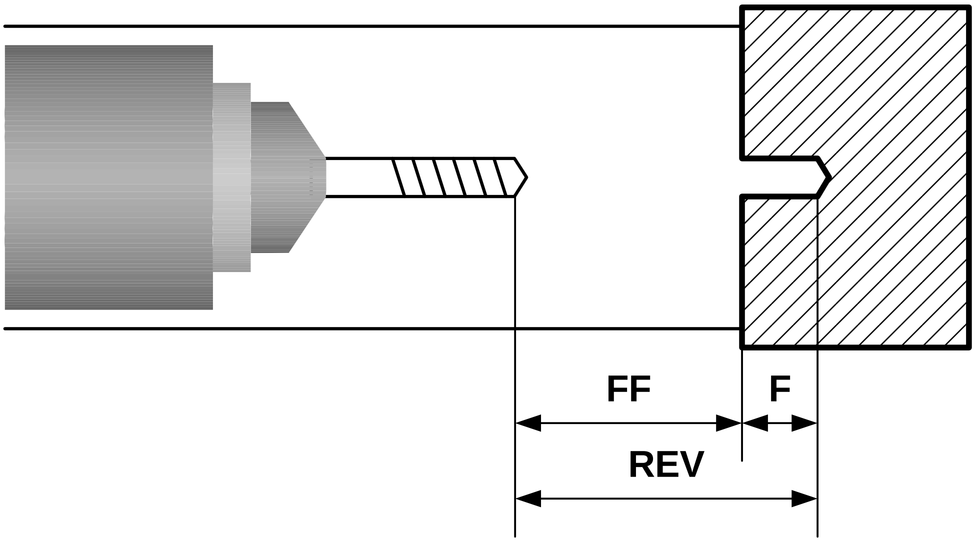


Figure 52: Example of a simple drilling unit

This simple example of drilling a hole shows the difference between two modes.
In order to drill the hole, the following steps have to be done:
Step 1: Initialization, for instance at power up.
Step 2: Move forward to drilling position and start the drill turning. In this way it will be fully operational before the position is reached and then check if both actions are completed.
Step 3: Drill the hole.
Step 4: After drilling the hole we have to wait for the step-chain sequence to finish dwelling to free the hole of any debris, which might have been stuck in the hole.
Step 5: Move drill back to starting position and shut the spindle off. Combining the completion of moving backwards and stopping the spindle we signal the step-chain to start over.

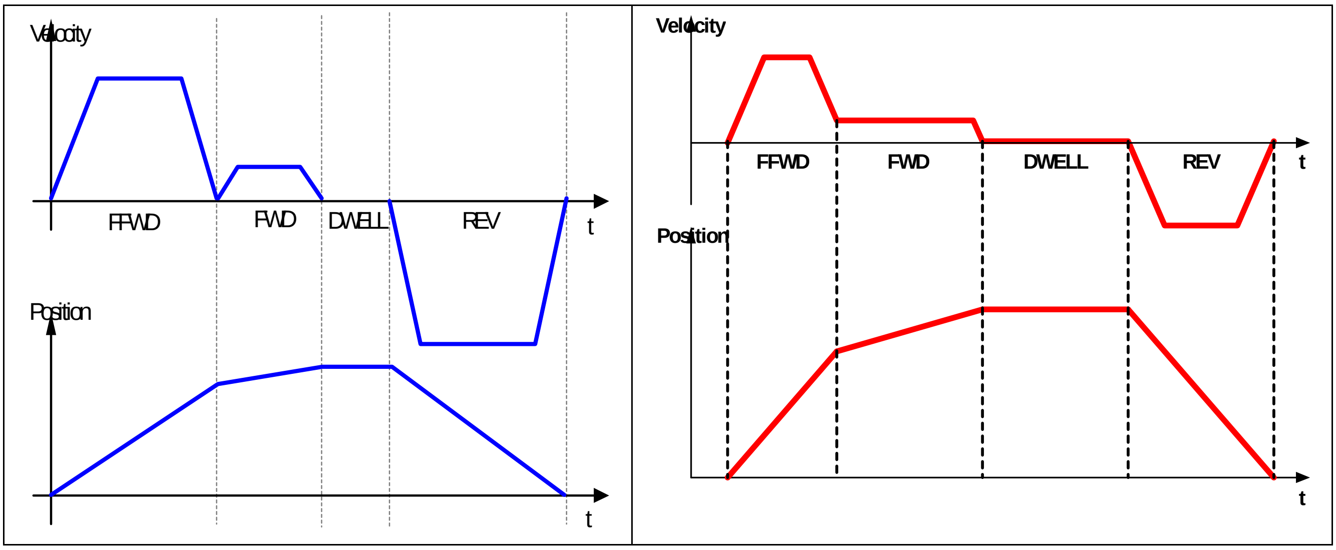


Figure 53: Timing diagrams for drilling. Left side no blending, right side with blending

5.1. Solution with Function Block diagram

Both examples can be described with the same program in FBD. The difference is in the input of the ‘BufferMode’ at the second FB, the MC_MoveRelative. The modes shown in this example are ‘Aborting’ or ‘BlendingLow’.

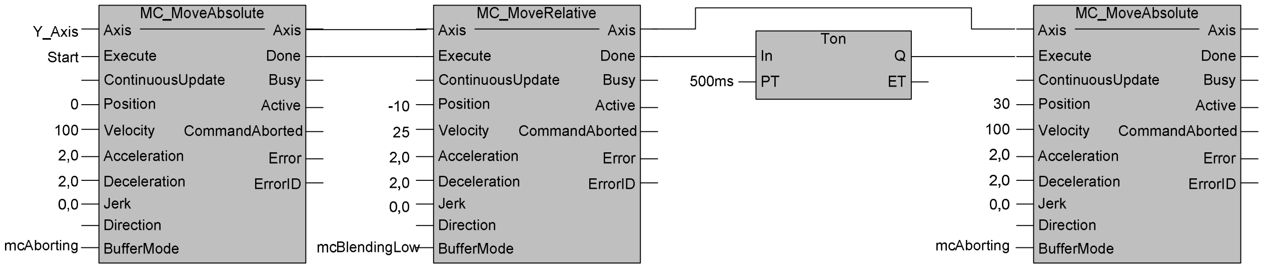


Figure 54: Solution with Function Block diagram

5.2. Solution with Sequential Function Chart

This is the classical approach using Sequential Function Charts for the specification of sequencing steps.
The SFC implements the timing diagram given in the example above.

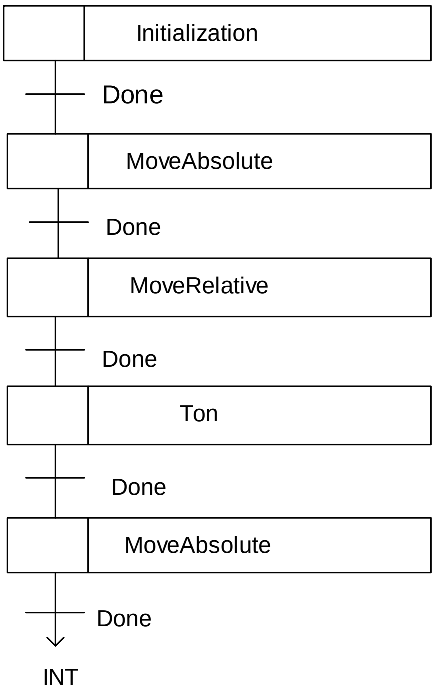


Figure 55: Straight forward step-transition chain for drilling example in SFC

Appendix A. Examples of the different buffer modes

Example 1: Standard behaviour of 2 following absolute movements

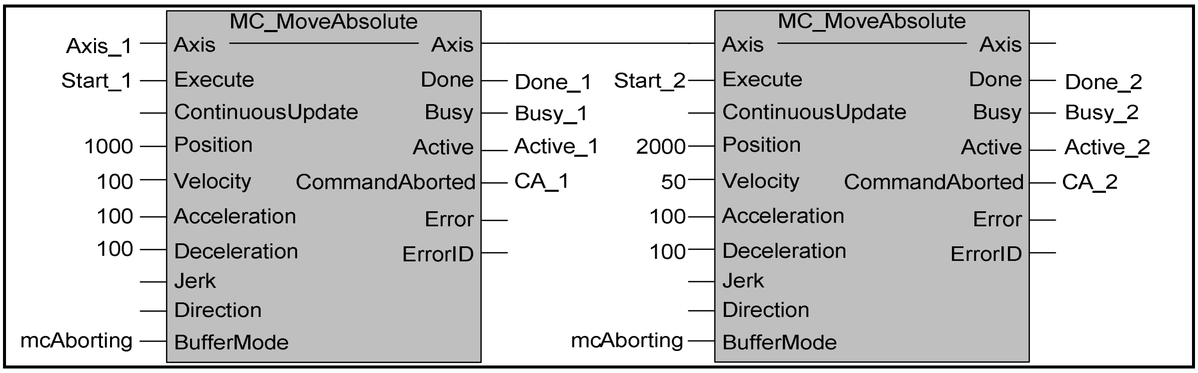


Figure 56: Basic example with two MC_MoveAbsolute on same axis

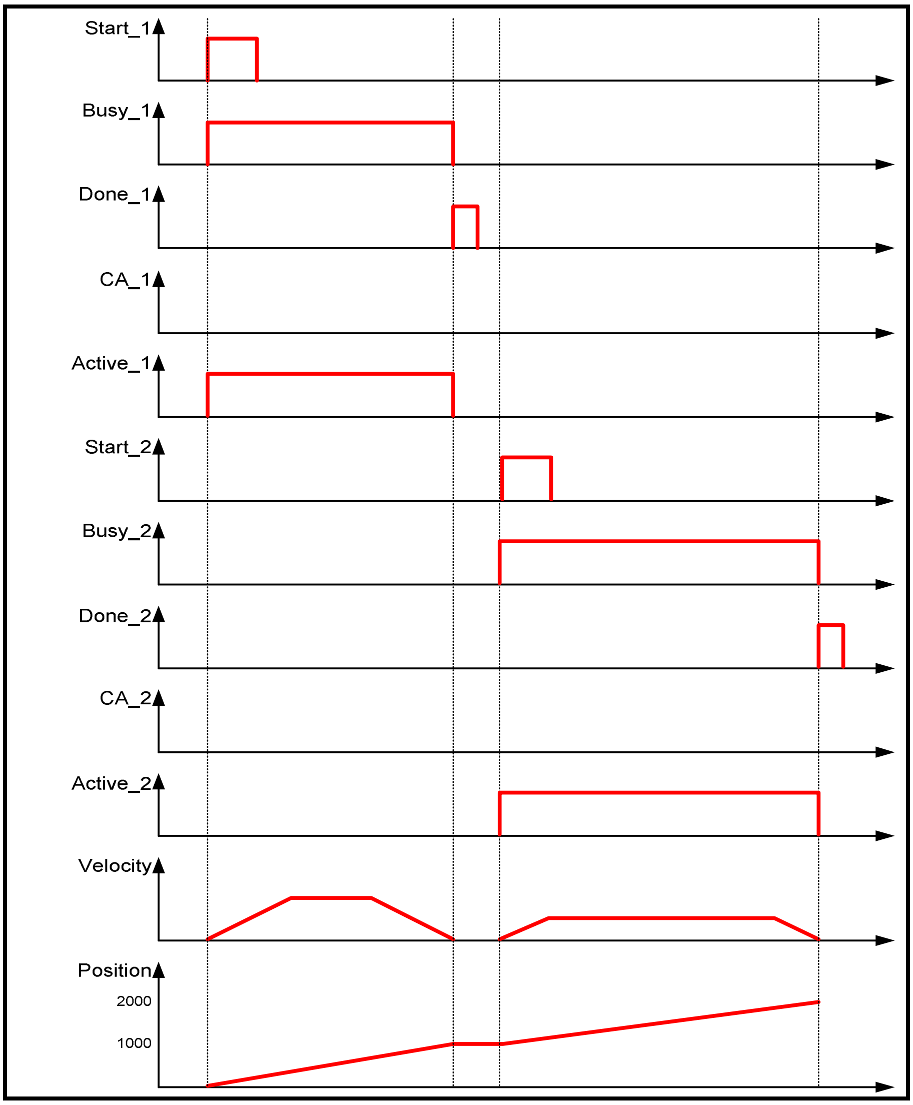
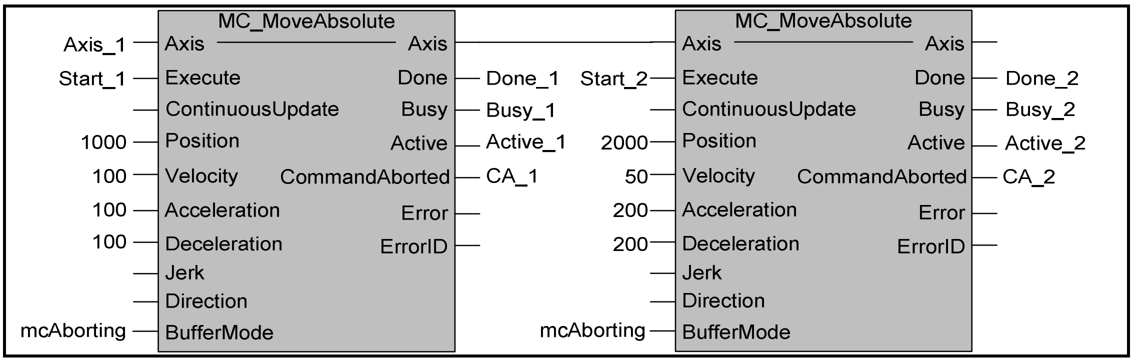


Figure 57: Timing diagram for example above without interference between FB1 and FB2 ('Aborting' Mode)

Example 2: 'Aborting' motion



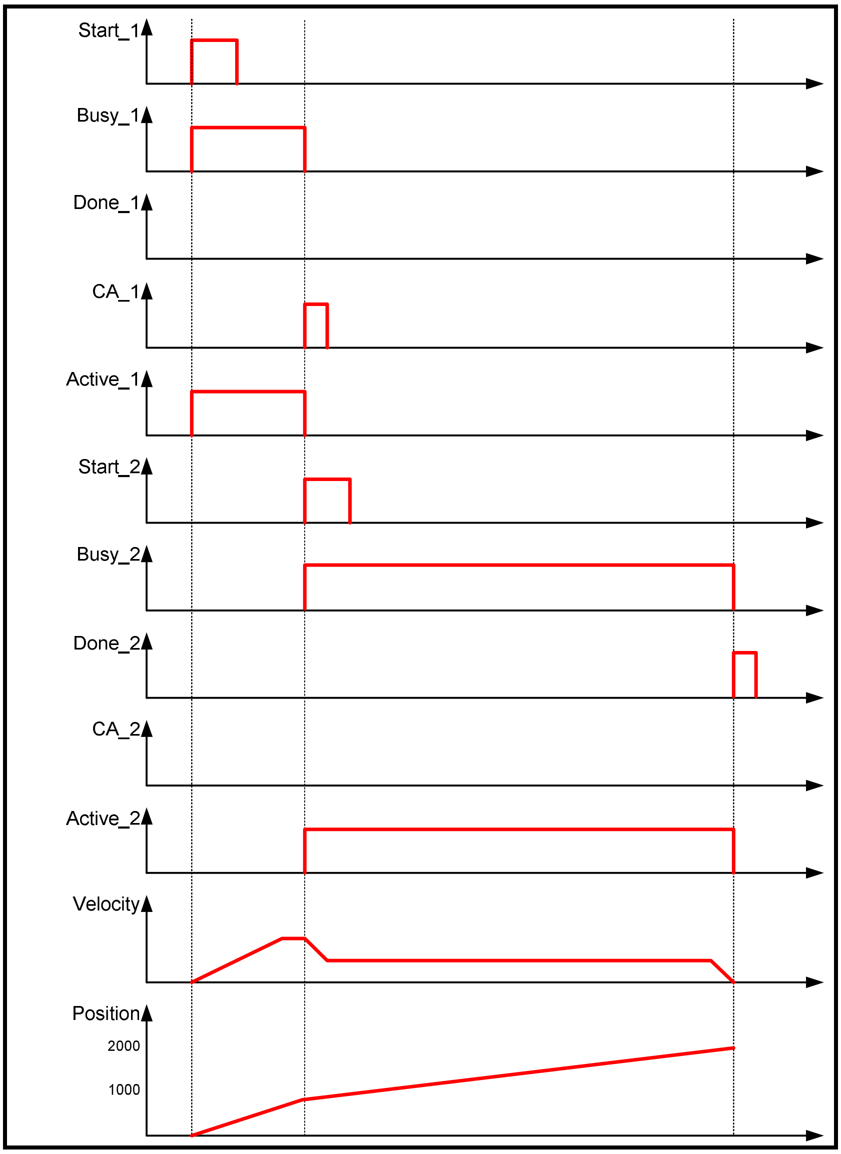
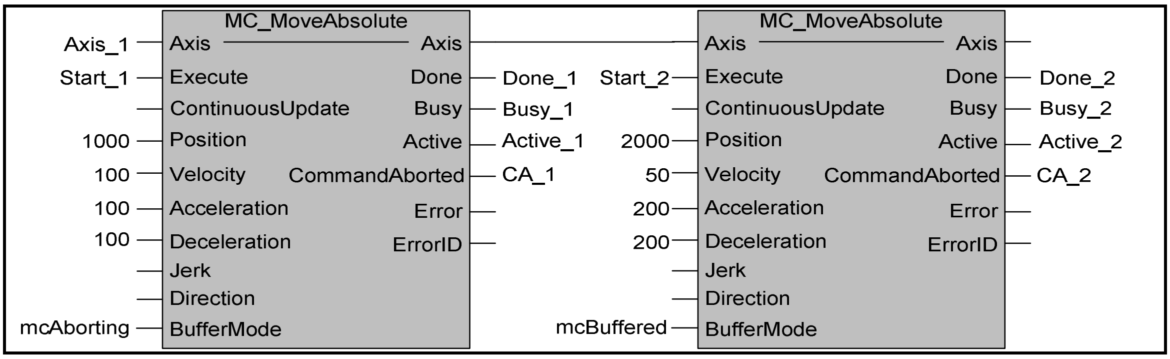


Figure 58: Timing diagram for example above with FB2 interrupting FB1 ('Aborting' Mode)

Example 3: 'Buffered' motion



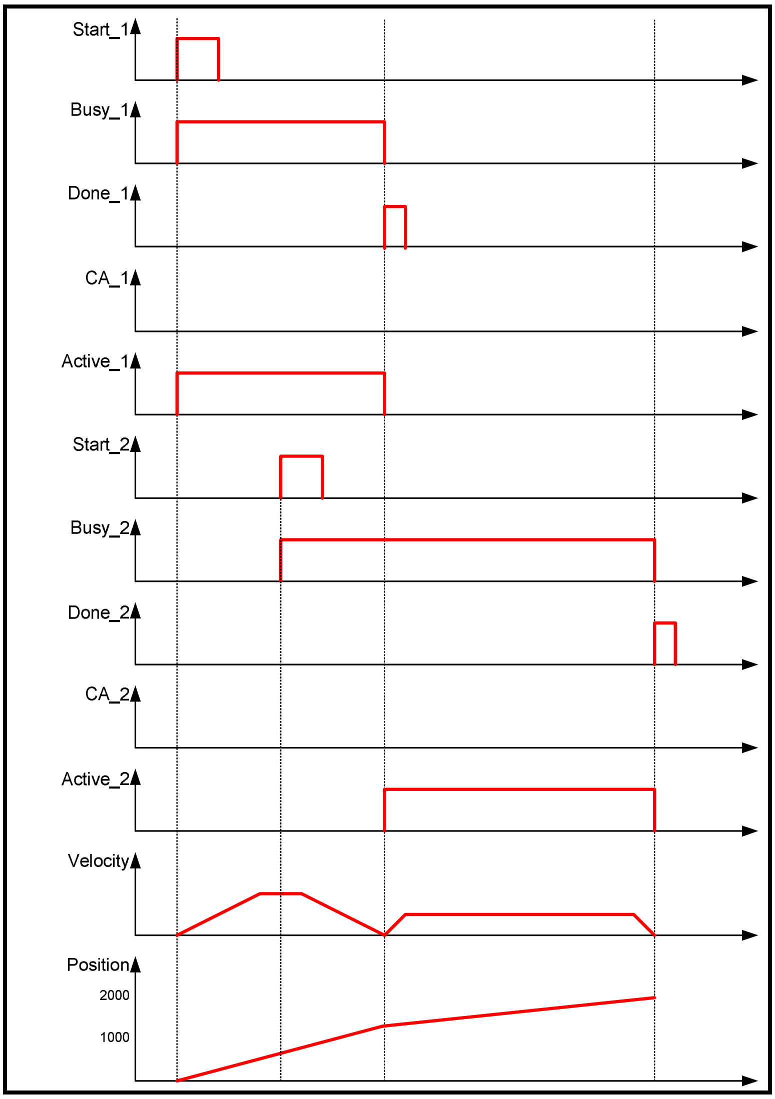
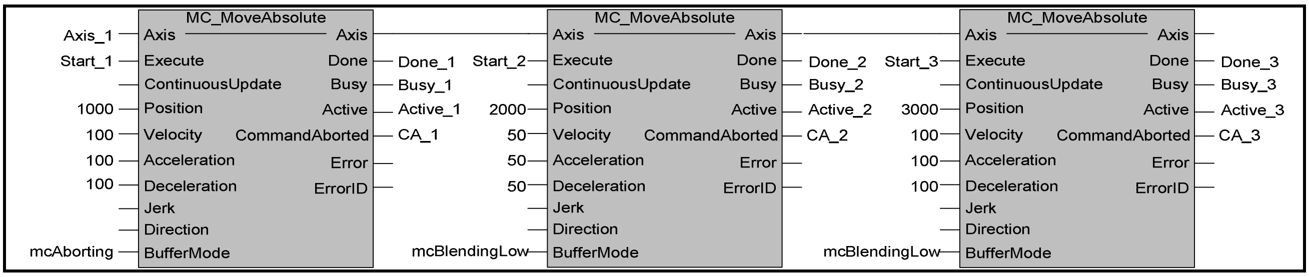


Figure 59: Timing diagram for example above in 'Buffered' Mode

(Stopping to velocity 0 and starting FB2 at that point without delay)

Example 4: 'BlendingLow' motion



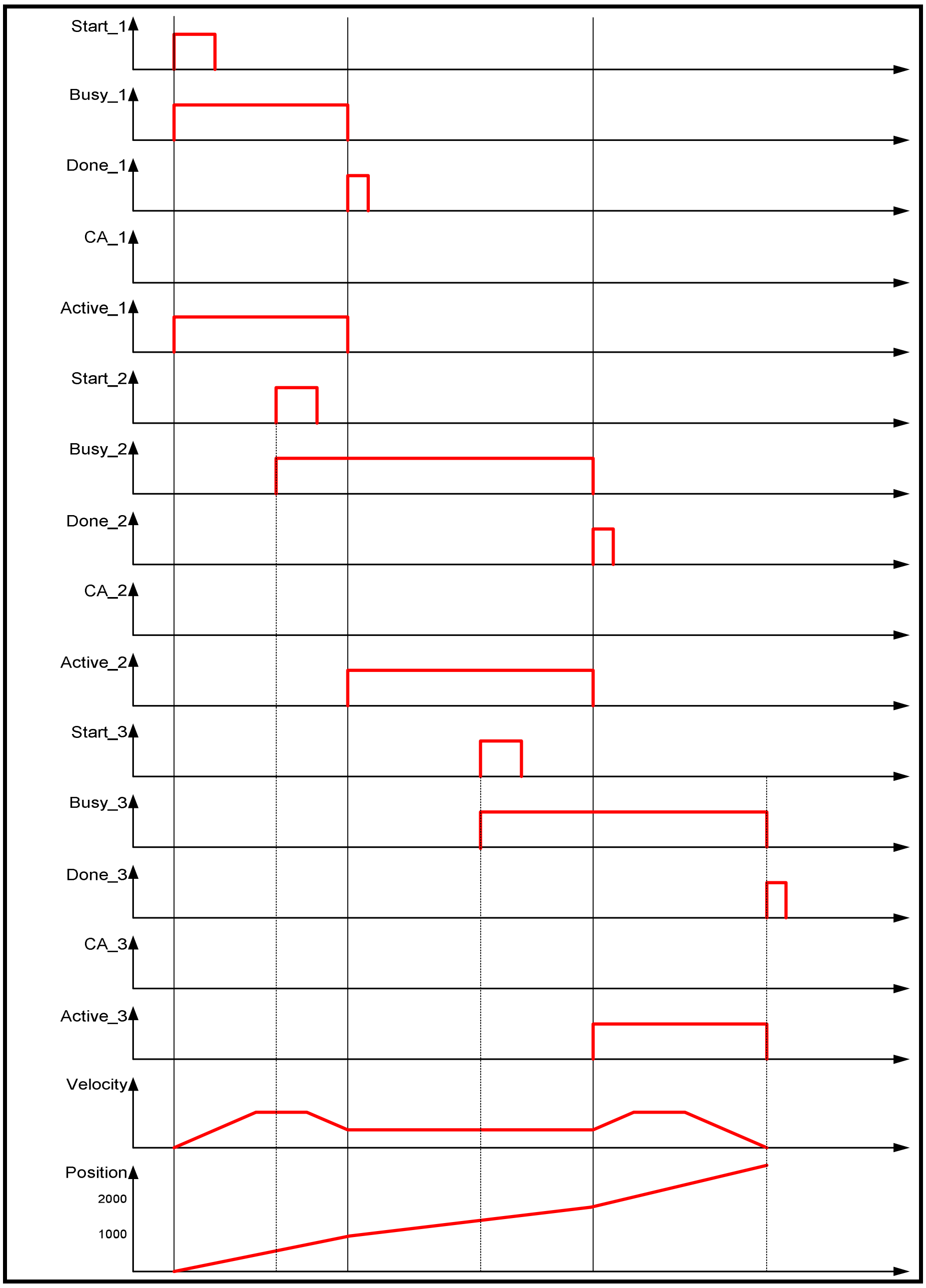
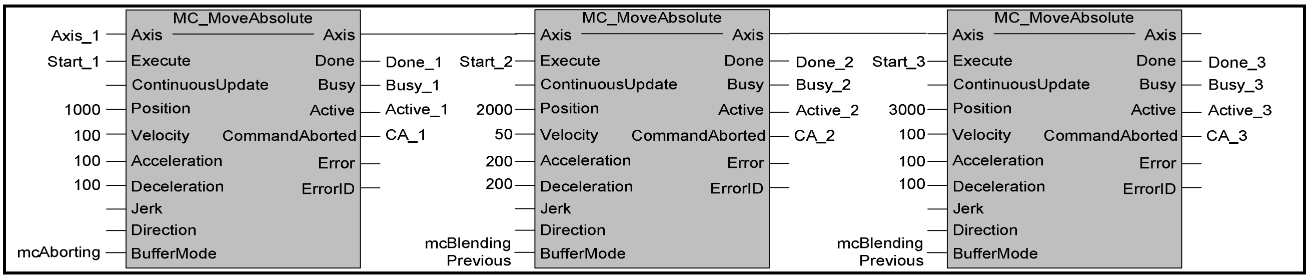


Figure 60: Timing diagram for example above with mode 'BlendingLow'

(Using lowest velocity (=velocity 2) from final position of FB1 until final position of FB2)

With the blending (and other FBs working on the same axis at the same time (like MC_MoveAdditive)), the system has to combine the different values working on the axis before giving the positions to the relevant axis.

Example 5: 'BlendingPrevious' motion


MC_MoveAbsolute
Axis_1 — Axis — Axis
Start_1 — Execute — Done — Done_1
ContinuousUpdate — Busy — Busy_1
1000 — Position — Active — Active_1
100 — Velocity — CommandAborted — CA_1
100 — Acceleration — Error
100 — Deceleration — ErrorID
Jerk
Direction
mcAborting — BufferMode
MC_MoveAbsolute
Axis — Axis
Start_2 — Execute — Done — Done_2
ContinuousUpdate — Busy — Busy_2
2000 — Position — Active — Active_2
50 — Velocity — CommandAborted — CA_2
200 — Acceleration — Error
200 — Deceleration — ErrorID
Jerk
Direction
mcBlending Previous — BufferMode
MC_MoveAbsolute
Axis — Axis
Start_3 — Execute — Done — Done_3
ContinuousUpdate — Busy — Busy_3
3000 — Position — Active — Active_3
100 — Velocity — CommandAborted — CA_3
100 — Acceleration — Error
100 — Deceleration — ErrorID
Jerk
Direction
mcBlending Previous — BufferMode

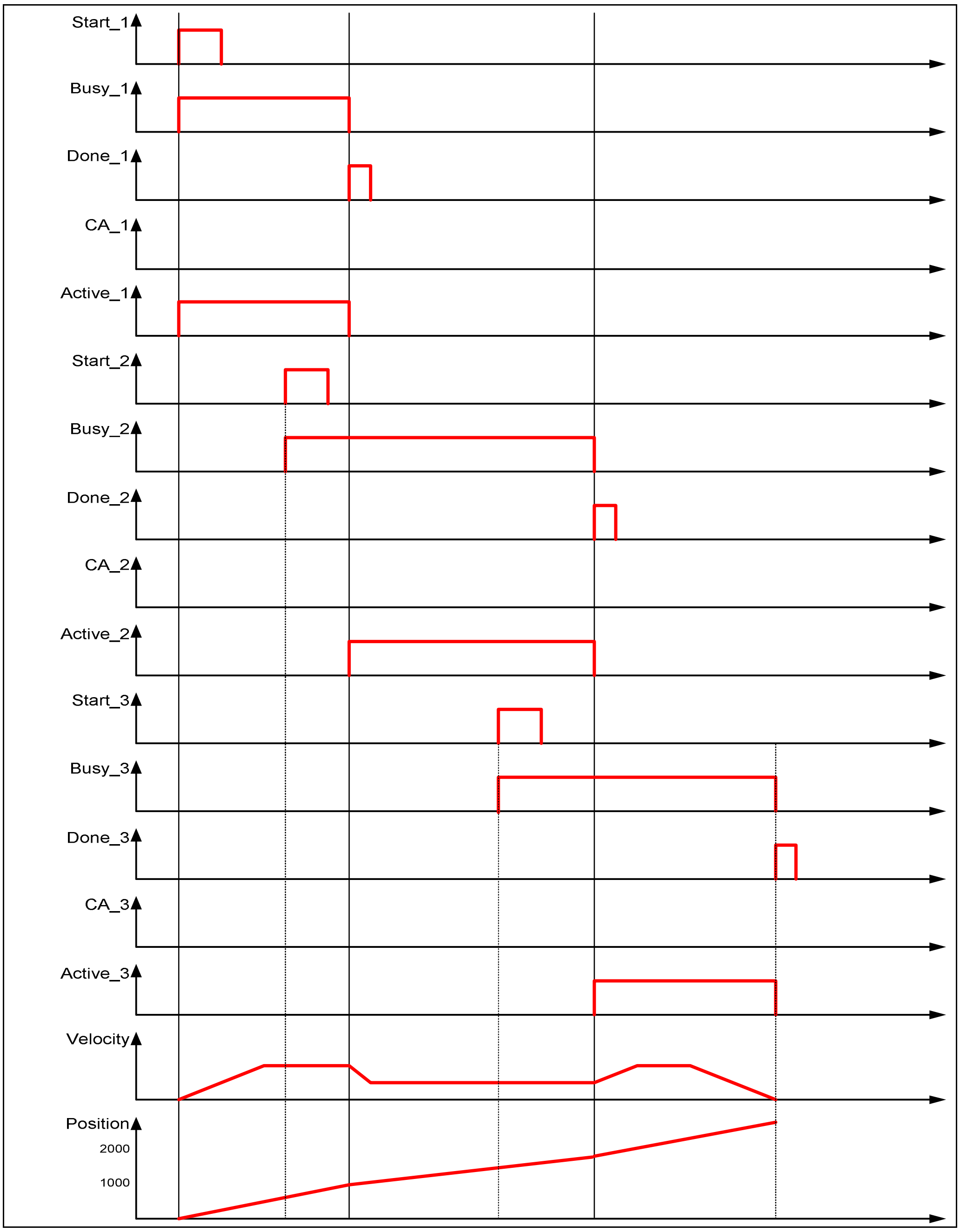
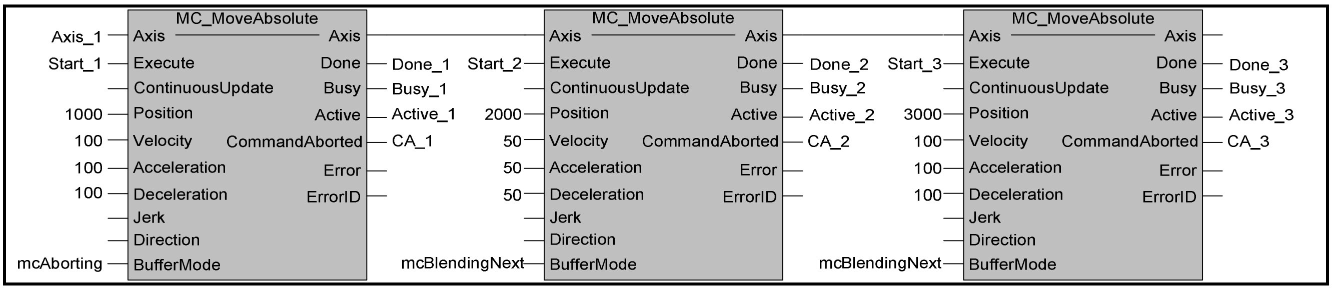


Figure 61: Timing diagram for example above with mode 'Merging1'
(Uses velocity FB1 at final position FB1)

Example 6: 'BlendingNext' motion


MC_MoveAbsolute
Axis_1 — Axis — Axis
Start_1 — Execute — Done — Done_1
ContinuousUpdate — Busy — Busy_1
1000 — Position — Active — Active_1
100 — Velocity — CommandAborted — CA_1
100 — Acceleration — Error
100 — Deceleration — ErrorID
Jerk
Direction
mcAborting — BufferMode
MC_MoveAbsolute
Axis — Axis
Start_2 — Execute — Done — Done_2
ContinuousUpdate — Busy — Busy_2
2000 — Position — Active — Active_2
50 — Velocity — CommandAborted — CA_2
50 — Acceleration — Error
50 — Deceleration — ErrorID
Jerk
Direction
mcBlendingNext — BufferMode
MC_MoveAbsolute
Axis — Axis
Start_3 — Execute — Done — Done_3
ContinuousUpdate — Busy — Busy_3
3000 — Position — Active — Active_3
100 — Velocity — CommandAborted — CA_3
100 — Acceleration — Error
100 — Deceleration — ErrorID
Jerk
Direction
mcBlendingNext — BufferMode

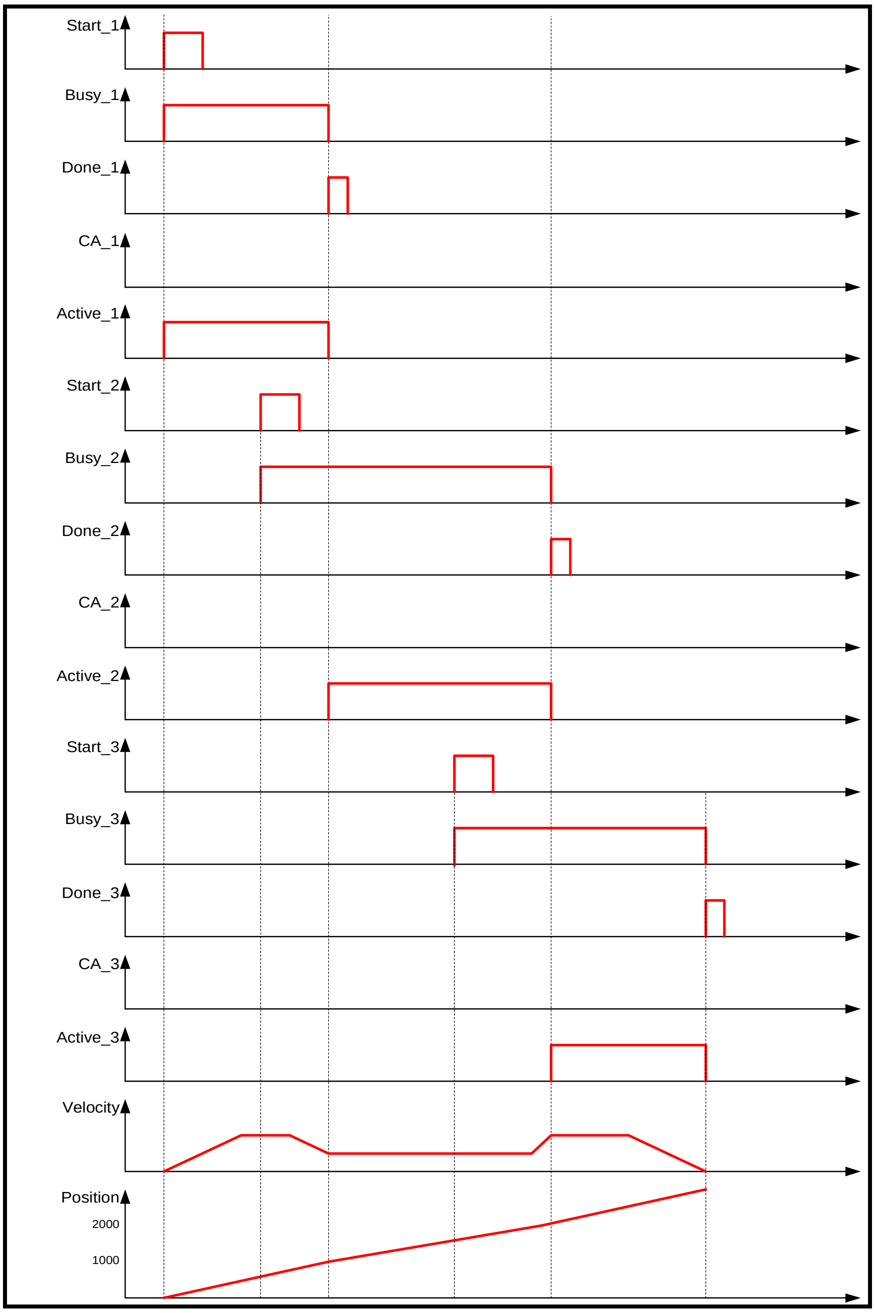
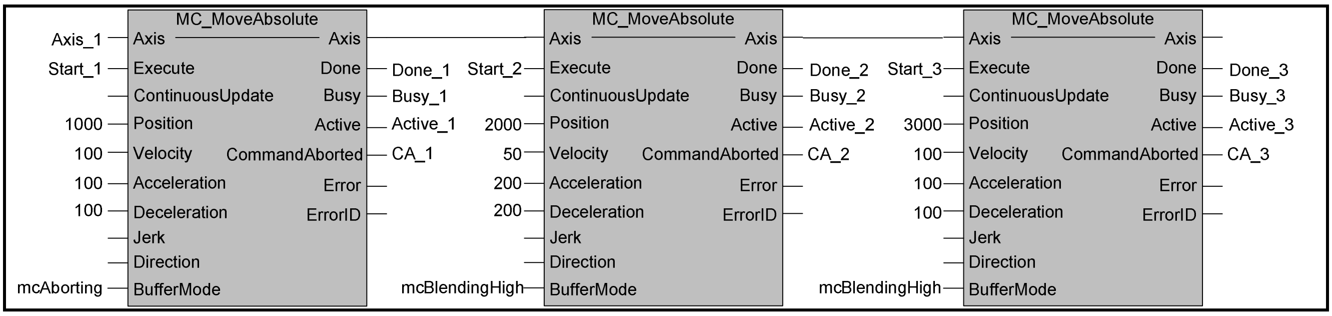


Figure 62: Timing diagram for example above with mode 'BlendingNext' motion

With a 2nd FB following MC_MoveVelocity all blending modes should work like blending previous or create an error.

Example 7: 'BlendingHigh' motion


MC_MoveAbsolute
Axis_1 — Axis Axis
Start_1 — Execute Done — Done_1
— ContinuousUpdate Busy — Busy_1
1000 — Position Active — Active_1
100 — Velocity CommandAborted — CA_1
100 — Acceleration Error —
100 — Deceleration ErrorID —
— Jerk
— Direction
mcAborting — BufferMode
MC_MoveAbsolute
Start_2 — Execute Done — Done_2
2000 — Position Active — Active_2
50 — Velocity CommandAborted — CA_2
200 — Acceleration
200 — Deceleration
mcBlendingHigh — BufferMode
Busy — Busy_2
MC_MoveAbsolute
Start_3 — Execute Done — Done_3
3000 — Position Active — Active_3
100 — Velocity CommandAborted — CA_3
100 — Acceleration
100 — Deceleration
mcBlendingHigh — BufferMode
Busy — Busy_3

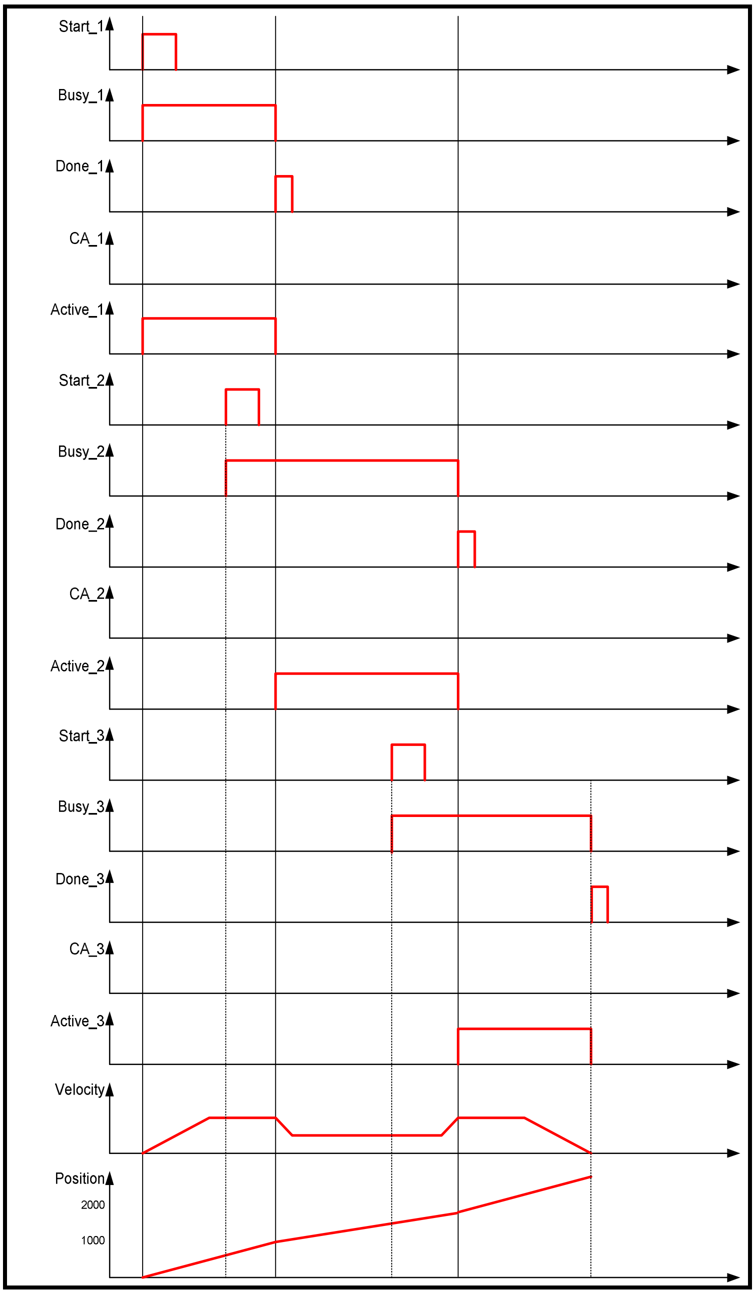


Figure 63: Timing diagram for example above with mode 'BlendingHigh' motion

Appendix B. Compliance Procedure and Compliance List

Listed in this Appendix are the requirements for the compliance statement from the supplier of the Motion Control Function Blocks. The compliance statement consists of two main groups: supported data types and supported Function Blocks, in combination with the applicable inputs and outputs. The supplier is required to fill out the tables for the used data types and Function Blocks, according to their product, committing their support to the specification.

By submitting these tables to PLCopen, and after approval by PLCopen, the list will be published on the PLCopen website, www.plcopen.org as well as a shortform overview, as specified in Appendix B 2 Supported Data types and Appendix B 3 Overview of the Function Blocks as below.

In addition to this approval, the supplier is granted access and usage rights of the PLCopen Motion Control logo, as described in Appendix B 4:

The PLCopen Motion Control Logo and Its Usage..

PLCopen
motion
control

Data types
The data type REAL listed in the Function Blocks and parameters (e.g. for velocity, acceleration, distance, etc.) may be exchanged to SINT, INT, DINT or LREAL without to be seen as incompliant to this standard, as long as they are consistent for the whole set of Function Blocks and parameters.
Implementation allows the extension of data types as long as the basic data type is kept. For example: WORD may be changed to DWORD, but not to REAL.

Function Blocks and Inputs and Outputs
An implementation which claims compliance with this PLCopen specification shall offer a set of Function Blocks for motion control, meaning one or more Function Blocks, with at least the **basic** input and output variables, marked as "**B**" in the tables. These inputs and outputs have to be supported to be compliant.
For higher-level systems and future extensions any subset of the **extended** input and output variables, marked as "**E**" in the tables can be implemented.
Vendor specific additions are marked with "**V**", and can be listed as such in the supplier documentation.

- **Basic** input/output variables are mandatory	Marked in the tables with the letter "**B**"
- **Extended** input /output variables are optional	Marked in the tables with the letter "**E**"
- **Vendor Specific** additions	Marked in the vendor's compliance documentation with "**V**"

All the vendor specific items will not be listed in the comparison table on the PLCopen website, but in the detailed vendor specific list, which also is published.
All vendor specific in- and outputs of all FBs must be listed in the certification list of the supplier. With this, the certification listing from a supplier describes all the I/Os of the relevant FBs, including vendor-specific extensions, and thus showing the complete FBs as used by the supplier.

Appendix B 1. Statement of Supplier

Supplier name	
Supplier address	
City	
Country	
Telephone	
Fax	
Email address	
Product Name	
Product version	
Release date	

I hereby state that the following tables as filled out and submitted do match our product as well as the accompanying user manual, as stated above.

Name of representation (person):

Date of signature (dd/mm/yyyy):

Signature:

Appendix B 2. Supported Data types

Defined datatypes with MC library:	Supported	If not supported, which datatype used
BOOL		
INT		
WORD		
REAL		
ENUM		
UINT		

Table 6: Supported datatypes

Within the specification the following derived datatypes are defined. Define which of these structures are used in this system:

Derived datatypes:	Where used	Supported	Which structure
AXIS_REF	Nearly all FBs		
MC_DIRECTION (extended)	MC_MoveAbsolute MC_MoveVelocity MC_TorqueControl MC_MoveContinuousAbsolute		
MC_TP_REF	MC_PositionProfile		
MC_TV_REF	MC_VelocityProfile		
MC_TA_REF	MC_AccelerationProfile		
MC_CAM_REF	MC_CamTableSelect		
MC_CAM_ID (extended)	MC_CamTableSelect MC_CamIn		
MC_START_MODE (extended)	MC_CamIn MC_CamTableSelect		
MC_BUFFER_MODE	Buffered FBs		
MC_EXECUTION_MODE	MC_SetPosition MC_WriteParameter MC_WriteBoolParameter MC_WriteDigitalOutput MC_CamTableSelect		
MC_SOURCE	MC_ReadMotionState MC_CamIn MC_GearIn MC_GearInPos MC_CombineAxes MC_DigitalCamSwitch		
MC_SYNC_MODE	MC_GearInPos		
MC_COMBINE_MODE	MC_CombineAxes		
MC_TRIGGER_REF	MC_TouchProbe MC_AbortTrigger		
MC_INPUT_REF	MC_ReadDigitalInput		
MC_OUTPUT_REF	MC_DigitalCamSwitch MC_ReadDigitalOutput MC_WriteDigitalOutput		
MC_CAMSWITCH_REF	MC_DigitalCamSwitch		
MC_TRACK_REF	MC_DigitalCamSwitch		

Table 7: Supported derived datatypes

Appendix B 3. Overview of the Function Blocks

Single Axis Function Blocks	**Supported as V1.0/ V1.1/ V2.0 or Not**	**Comments (<= 48 char.)**
MC_Power		
MC_Home		
MC_Stop		
MC_Halt		
MC_MoveAbsolute		
MC_MoveRelative		
MC_MoveAdditive		
MC_MoveSuperimposed		
MC_HaltSuperimposed		
MC_MoveVelocity		
MC_MoveContinuousAbsolute		
MC_MoveContinuousRelative		
MC_TorqueControl		
MC_PositionProfile		
MC_VelocityProfile		
MC_AccelerationProfile		
MC_SetPosition		
MC_SetOverride		
MC_ReadParameter & MC_ReadBoolParameter		
MC_WriteParameter & MC_WriteBoolParameter		
MC_ReadDigitalInput		
MC_ReadDigitalOutput		
MC_WriteDigitalOutput		
MC_ReadActualPosition		
MC_ReadActualVelocity		
MC_ReadActualTorque		
MC_ReadStatus		
MC_ReadMotionState		
MC_ReadAxisInfo		
MC_ReadAxisError		
MC_Reset		
MC_DigitalCamSwitch		
MC_TouchProbe		
MC_AbortTrigger		
Multi-Axis Function Blocks	**Supported as V1.0/ V1.1/ V2.0 or Not**	**Comments (<= 48 char.)**
MC_CamTableSelect		
MC_CamIn		
MC_CamOut		
MC_GearIn		
MC_GearOut		
MC_GearInPos		
MC_PhasingAbsolute		
MC_PhasingRelative		
MC_CombineAxes		

Table 8: Short overview of the Function Blocks

Appendix B 3.1 MC_Power

If Supported	**MC_Power**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
E	EnablePositive		
E	EnableNegative		
VAR_OUTPUT			
B	Status		
E	Valid		
B	Error		
E	ErrorID		

Appendix B 3.2 MC_Home

If Supported	**MC_Home**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
B	Position		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.3 MC_Stop

If Supported	**MC_Stop**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	Deceleration		
E	Jerk		
VAR_OUTPUT			
B	Done		
E	Busy		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.4 MC_Halt

If Supported	**MC_Halt**	Sup. Y/N	
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.5 MC_MoveAbsolute

If Supported	**MC_MoveAbsolute**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Position		
B	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
B	Direction		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.6 MC_MoveRelative

If Supported	**MC_MoveRelative**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Distance		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.7 MC_MoveAdditive

If Supported	**MC_MoveAdditive**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Distance		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.8 MC_MoveSuperimposed

If Supported	**MC_MoveSuperimposed**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Distance		
E	VelocityDiff		
E	Acceleration		
E	Deceleration		
E	Jerk		
VAR_OUTPUT			
B	Done		
E	Busy		
E	CommandAborted		
B	Error		
E	ErrorID		
E	CoveredDistance		

Appendix B 3.9 MC_HaltSuperimposed

If Supported	**MC_HaltSuperimposed**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	Deceleration		
E	Jerk		
VAR_OUTPUT			
B	Done		
E	Busy		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.10 MC_MoveVelocity

If Supported	**MC_MoveVelocity**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	Direction		
E	BufferMode		
VAR_OUTPUT			
B	InVelocity		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.11 MC_MoveContinuousAbsolute

If Supported	**MC_MoveContinuousAbsolute**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Position		
B	EndVelocity		
B	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	Direction		
E	BufferMode		
VAR_OUTPUT			
B	InEndVelocity		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.12 MC_MoveContinuousRelative

If Supported	**MC_MoveContinuousRelative**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Distance		
B	EndVelocity		
B	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	InEndVelocity		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.13 MC_TorqueControl

If Supported	**MC_TorqueControl**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	Torque		
E	TorqueRamp		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	Direction		
E	BufferMode		
VAR_OUTPUT			
B	InTorque		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.14 MC_PositionProfile

If Supported	**MC_PositionProfile**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	TimePosition		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
E	TimeScale		
E	PositionScale		
E	Offset		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.15 MC_VelocityProfile

If Supported	**MC_VelocityProfile**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	TimeVelocity		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
E	TimeScale		
E	VelocityScale		
E	Offset		
E	BufferMode		
VAR_OUTPUT			
B	ProfileCompleted		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.16 MC_AccelerationProfile

If Supported	**MC_AccelerationProfile**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	TimeAcceleration		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
E	TimeScale		
E	AccelerationScale		
E	Offset		
E	BufferMode		
VAR_OUTPUT			
B	ProfileCompleted		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.17 MC_SetPosition

If Supported	**MC_SetPosition**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
B	Position		
E	Relative		
E	ExecutionMode		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.18 MC_SetOverride

If Supported	**MC_SetOverride**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
B	VelFactor		
E	AccFactor		
E	JerkFactor		
VAR_OUTPUT			
B	Enabled		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.19 MC_ReadParameter & MC_ReadBoolParameter

If Supported	**MC_ReadParameter**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
B	ParameterNumber		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Value		

If Supported	**MC_ReadBoolParameter**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
B	ParameterNumber		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Value		

Name	**B/E**	**R/W**	**Supp. Y/N**	**Comments**
CommandedPosition	B	R		
SWLimitPos	E	R/W		
SWLimitNeg	E	R/W		
EnableLimitPos	E	R/W		
EnableLimitNeg	E	R/W		
EnablePosLagMonitoring	E	R/W		
MaxPositionLag	E	R/W		
MaxVelocitySystem	E	R		
MaxVelocityAppl	B	R/W		
ActualVelocity	B	R		
CommandedVelocity	B	R		
MaxAccelerationSystem	E	R		
MaxAccelerationAppl	E	R/W		
MaxDecelerationSystem	E	R		
MaxDecelerationAppl	E	R/W		
MaxJerkSystem	E	R		
MarkJerkAppl	E	R/W		

Table 9: Parameters for MC_Read(Bool)Parameter and MC_Write(Bool)Parameter

Appendix B 3.20 MC_WriteParameter & MC_WriteBoolParameter

If Supported	**MC_WriteParameter**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
B	ParameterNumber		
B	Value		
E	ExecutionMode		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

If Supported	**MC_WriteBoolParameter**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
B	ParameterNumber		
B	Value		
E	ExecutionMode		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.21 MC_ReadDigitalInput

If Supported	**MC_ReadDigitalInput**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Input		
VAR_INPUT			
B	Enable		
E	InputNumber		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Value		

Appendix B 3.22 MC_ReadDigitalOutput

If Supported	**MC_ReadDigitalOutput**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Output		
VAR_INPUT			
B	Enable		
E	OutputNumber		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Value		

Appendix B 3.23 MC_WriteDigitalOutput

If Supported	**MC_WriteDigitalOutput**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Output		
VAR_INPUT			
B	Execute		
E	OutputNumber		
B	Value		
E	ExecutionMode		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.24 MC_ReadActualPosition

If Supported	**MC_ReadActualPosition**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Position		

Appendix B 3.25 MC_ReadActualVelocity

If Supported	**MC_ReadActualVelocity**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Velocity		

Appendix B 3.26 MC_ReadActualTorque

If Supported	**MC_ReadActualTorque**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	Torque		

Appendix B 3.27 MC_ReadStatus

If Supported	**MC_ReadStatus**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
B	ErrorStop		
B	Disabled		
B	Stopping		
E	Homing		
B	Standstill		
E	DiscreteMotion		
E	ContinuousMotion		
E	SynchronizedMotion		

Appendix B 3.28 MC_ReadMotionState

If Supported	**MC_ReadMotionState**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
E	Source		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
E	ConstantVelocity		
E	Accelerating		
E	Decelerating		
E	DirectionPositive		
E	DirectionNegative		

Appendix B 3.29 MC_ReadAxisInfo

If Supported	**MC_ReadAxisInfo**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
E	ErrorID		
E	HomeAbsSwitch		
E	LimitSwitchPos		
E	LimitSwitchNeg		
E	Simulation		
E	CommunicationReady		
E	ReadyForPowerOn		
E	PowerOn		
E	IsHomed		
E	AxisWarning		

Appendix B 3.30 MC_ReadAxisError

If Supported	**MC_ReadAxisError**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Valid		
E	Busy		
B	Error		
B	ErrorID		
E	AxisErrorID		

Appendix B 3.31 MC_Reset

If Supported	**MC_Reset**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.32 MC_DigitalCamSwitch

If Supported	**MC_DigitalCamSwitch**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	Switches		
E	Outputs		
E	TrackOptions		
VAR_INPUT			
B	Enable		
E	EnableMask		
E	ValueSource		
VAR_OUTPUT			
B	InOperation		
E	Busy		
B	Error		
E	ErrorID		

Basic elements within the array structure of MC_CAMSWITCH_REF

B/E	**Parameter**	Sup.Y/N	Comments
B	TrackNumber		
B	FirstOnPosition [u]		
B	LastOnPosition [u]		
E	AxisDirection		
E	CamSwitchMode		
E	Duration		

Basic elements within the array structure of MC_TRACK_REF

B/E	**Parameter**	Sup.Y/N	Comments
E	OnCompensation		
E	OffCompensation		
E	Hysteresis [u]		

Appendix B 3.33 MC_TouchProbe

If Supported	**MC_TouchProbe**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
E	TriggerInput		
VAR_INPUT			
B	Execute		
E	WindowOnly		
E	FirstPosition		
E	LastPosition		
VAR_OUTPUT			
B	Done		
E	Busy		
E	CommandAborted		
B	Error		
E	ErrorID		
B	RecordedPosition		

Appendix B 3.34 MC_AbortTrigger

If Supported	**MC_AbortTrigger**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Axis		
E	TriggerInput		
VAR_INPUT			
B	Execute		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.35 MC_CamTableSelect

If Supported	**MC_CamTableSelect**	Sup. Y/N	Comments
VAR_IN_OUT			
E	Master		
E	Slave		
B	CamTable		
VAR_INPUT			
B	Execute		
E	Periodic		
E	MasterAbsolute		
E	SlaveAbsolute		
E	ExecutionMode		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		
E	CamTableID		

Appendix B 3.36 MC_CamIn

If Supported	**MC_CamIn**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Master		
B	Slave		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
E	MasterOffset		
E	SlaveOffset		
E	MasterScaling		
E	SlaveScaling		
E	MasterStartDistance		
E	MasterSyncPosition		
E	StartMode		
E	MasterValueSource		
E	CamTableID		
E	BufferMode		
VAR_OUTPUT			
B	InSync		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		
E	EndOfProfile		

Appendix B 3.37 MC_CamOut

If Supported	**MC_CamOut**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Slave		
VAR_INPUT			
B	Execute		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.38 MC_GearIn

If Supported	**MC_GearIn**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Master		
B	Slave		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
B	RatioNumerator		
B	RatioDenominator		
E	MasterValueSource		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	InGear		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.39 MC_GearOut

If Supported	**MC_GearOut**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Slave		
VAR_INPUT			
B	Execute		
VAR_OUTPUT			
B	Done		
E	Busy		
B	Error		
E	ErrorID		

Appendix B 3.40 MC_GearInPos

If Supported	**MC_GearInPos**	Sup.Y/N	Comments
VAR_IN_OUT			
B	Master		
B	Slave		
VAR_INPUT			
B	Execute		
B	RatioNumerator		
B	RatioDenominator		
E	MasterValueSource		
B	MasterSyncPosition		
B	SlaveSyncPosition		
E	SyncMode		
E	MasterStartDistance		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
E	StartSync		
B	InSync		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 3.41 MC_PhasingAbsolute

If Supported	**MC_PhasingAbsolute**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Master		
B	Slave		
VAR_INPUT			
B	Execute		
B	PhaseShift		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		
E	AbsolutePhaseShift		

Appendix B 3.42 MC_PhasingRelative

If Supported	**MC_PhasingRelative**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Master		
B	Slave		
VAR_INPUT			
B	Execute		
B	PhaseShift		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
E	BufferMode		
VAR_OUTPUT			
B	Done		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		
E	CoveredPhaseShift		

Appendix B 3.43 CombineAxes

If Supported	**MC_CombineAxes**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Master1		
B	Master2		
B	Slave		
VAR_INPUT			
B	Execute		
E	ContinuousUpdate		
E	CombineMode		
E	GearRationNumeratorM1		
E	GearRatioDenominatorM1		
E	GearRatioNumeratorM2		
E	GearRatioDenominatorM2		
E	MasterValueSourceM1		
E	MasterValueSourceM2		
E	BufferMode		
VAR_OUTPUT			
B	InSync		
E	Busy		
E	Active		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix B 4. The PLCopen Motion Control Logo and Its Usage

For quick identification of compliant products, PLCopen has developed a logo for the Motion Control Function Blocks:

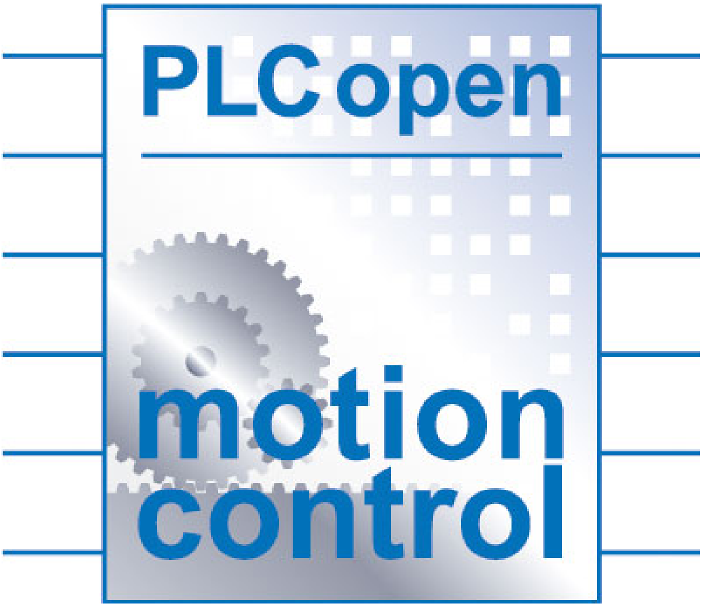


Figure 64: The PLCopen Motion Control Logo

This motion control logo is owned and trademarked by PLCopen.

In order to use this logo free-of-charge, the relevant company has to fulfill all the following requirements:

1. the company has to be a voting member of PLCopen;
2. the company has to comply with the existing specification, as specified by the PLCopen Task Force Motion Control, and as published by PLCopen, and of which this statement is a part;
3. this compliance application is provided in written form by the company to PLCopen, clearly stating the applicable software package and the supporting elements of all the specified tables, as specified in the document itself;
4. in case of non-fulfillment, which has to be decided by PLCopen, the company will receive a written statement concerning this from PLCopen. The company will have a one month period to either adopt their software package in such a way that it complies, represented by the issuing of a new compliance statement, or remove all reference to the specification, including the use of the logo, from all their specification, be it technical or promotional material;
5. the logo has to be used as is - meaning the full logo. It may be altered in size providing the original scale and color setting is kept.
6. the logo has to be used in the context of Motion Control.