Technical Specification
PLCopen - Technical Committee 2 – Task Force

Function blocks for motion control

Version 1.0

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:

Ch. Huber	ACC Motion SA, Penthaz, Switzerland
Ph. Moeschler	ACC Motion SA, Penthaz, Switzerland
A. Thome	Beckhoff Industrie Elektronik, Verl, Germany
J. Papenfort	Beckhoff Industrie Elektronik, Verl, Germany
K. Manton	Control Techniques, Newtown, United Kingdom
B. Busher	Proface America Inc., Glendale Heights IL, USA
A. Moeltner	ELAU Elektronik Automations AG, Marktheidenfeld, Germany
D. Boden	Giddings & Lewis / Thyssen, Prescot Merseyside, United Kingdom
S. Partridge	Giddings & Lewis / Thyssen, Prescot Merseyside, United Kingdom
U. Gossmann	Indramat, Lohr am Main, Germany
W. Brendel	Infoteam Software GmbH, Bubenreuth, Germany
B. Seeberger	Infoteam Software GmbH, Bubenreuth, Germany
A. Orzelski	Klöpper & Wiege Software GmbH, Lemgo, Germany
M. Petig	Klöpper & Wiege Software GmbH, Lemgo, Germany
M. Schütte	Lenze, Hameln, Germany
M. Stöwer	Lenze, Hameln, Germany
R. O'Brien	Nyquist Industrial Control, Netherlands
E. van der Wal	PLCopen, Zaltbommel, Netherlands
R. Schmitt	Siemens AG, Nuremberg, Germany
H.-P. Otto	Siemens AG, Nuremberg, Germany
W. Gagsteiger	Siemens AG, Nuremberg, Germany
A. Oksas	Softing GmbH, Munich, Germany
R. Mittmann	Softing GmbH, Munich, Germany
I. Ulvros (chairman)	Tetra Pak Research & Development AB, Lund, Sweden

Change Status List:

Version number	Date	Change comment
V 0.1	August, 21 1997	Preliminary version
V 0.2	October, 31 1997	Modified structure after Munich's meeting (Oct.7.97)
V 0.3	December, 12 1997	Added substance to some elements (Bubenreuth's meeting Nov.28.97)
V 0.4	January 17 1998	Added Objectives, modification during execution commented
V 0.5	May 18, 1998	Incorporated changes meeting March 25, 1998, Frankfurt
V 0.51	December 8, 1998	Incorporated Data Type Axis and comments during meeting Dec. 8, 1998
V 0.52	February 2, 1999	Incorporated comments General Motors PT and results meeting Dec. 8 General lay out reconfiguration. Embedded Excel to Tables
V 0.53	February 4 & 5, 1999	Inclusion of comments and reworking input/output variables of FB
V 0.54	May 5 & 6, 1999	Inclusion of comments during meeting
V 0.6	May 10, 1999	Inclusion of comments meeting May 5&6. Released version for comments, cf. MoM
V 0.7	March 23, 2000	Merge both parts together and modifications according to comments of meeting of February 9&10 2000.
V 0.8	June 23, 2000	Meeting at Lenze, April 18 & 19, 2000. Clarification of AXIS_REF.
V 0.81	July 13, 14 & 20, 2000	Changes according to meeting July 13 & 14 (see minutes of meeting)
V 0.91	October 15, 2000	Prepared for the Nürnberg meeting 26/27. Oct. 2000
V0.92	October 22, 2000	Camming, readError, readParameter, writeParameter
V 0.93	October 27, 2000	Changes according to meeting Oct. 26 & 27 (see minutes of meeting)
V 0.99	November 13, 2000	Published for feedback – incl. Request for Change form
V.099A	February 22, 2001	First merge with available feedback: doc. 31 and Doc 32
V.099B	May 8+9	Feedback work during meeting in Amsterdam

V.099C	July 18 + 19	Graphics changed to Viso graphics (as decided last meeting) Feedback work done during meeting Verl
V.099D	August 29	Items from meeting included. Drawings adopted
V.099E	Oct. 17-19 and till Oct. 29	Items from meeting and defined included. Last version before publication
V. 1.0	November 23, 2001	Included feedback from participants on version 0.99E

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 itself, (as it is done within 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 help 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.

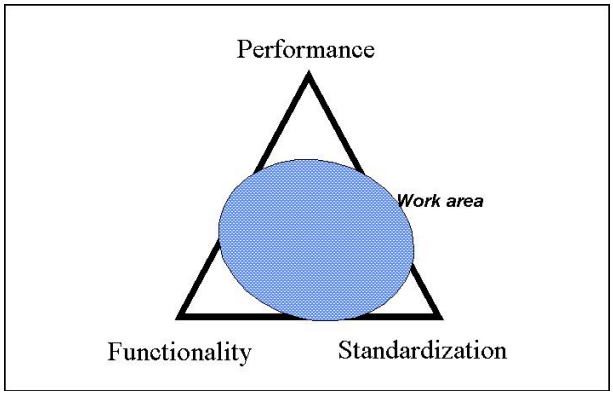


Figure 1: The triangle with user options

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 close to the hardware with dedicated functions, in order to get the highest performance possible as dictated by their environment. This limits the user in their options with respect to the target hardware and the reusability of the control software and raises the training investment.
The second user option allows for a very broad range of software functionality that can 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 get very close to the maximum functionality, meaning that the bottom of the triangle is very short.

1.1. Objectives

The Motion Control Function Blocks shall be 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
  - Data Types
  according to the IEC 61131-3.
- These Function Blocks and data types can be used in all IEC 61131-3 languages.
- The examples in this draft are just 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.
- Reusable applications composed from these Function Blocks and data types can be achieved by PLCopen Conformity Level and Reusability Level of IEC 61131-3 languages, and future PLCopen certification and 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 MC 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_MoveAbsolute	MC_CamIn
MC_ReadStatus		MC_MoveRelative	MC_CamOut
MC_ReadAxisError		MC_MoveAdditive	MC_GearIn
MC_ReadParameter		MC_MoveSuperimposed	MC_GearOut
MC_ReadBoolParameter		MC_MoveVelocity	MC_Phasing
MC_WriteParameter		MC_Home	
MC_WriteBoolParameter		MC_Stop	
MC_ReadActualPosition		MC_PositionProfile	
MC_Reset		MC_VelocityProfile	
		MC_AccelerationProfile	

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 of 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 A.

- **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 specific parameters to the specified Function Blocks.

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 they are consistent for the whole set of Function Blocks and parameters.
Implementation allows to extend data types as long as the basic data type is kept. For example: WORD may be changed to DWORD, but not to REAL.

2. Model

The following Function Blocks (FB) library is designed for the purpose of controlling axes via the language elements consistent with those as 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 datatype 'Axis', which offers flexibility, ease of use and re-usability.

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. It will not touch the algorithm part.

This leads to some consequences that are described in this chapter.

2.1. The State Diagram

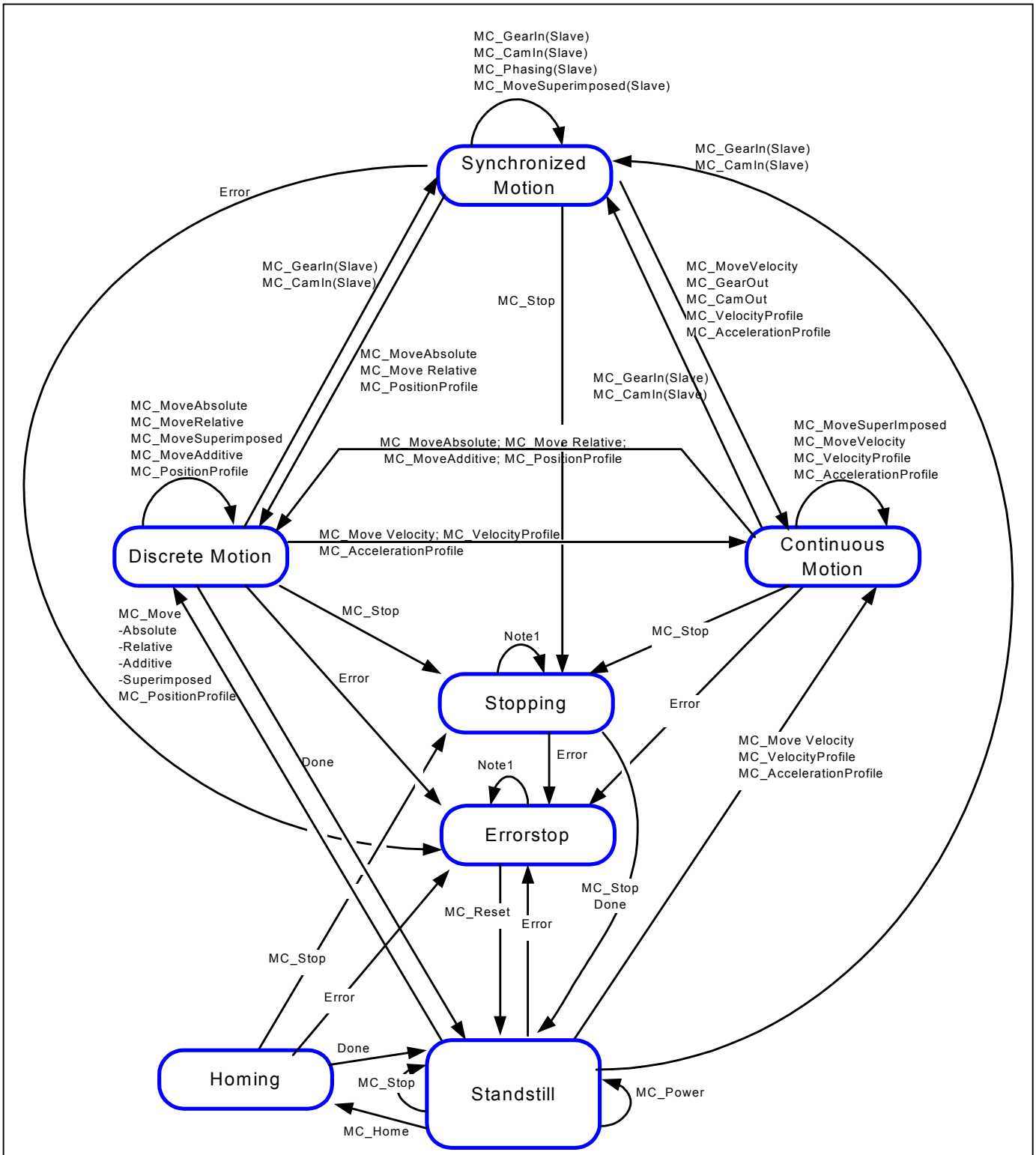
The following diagram normatively defines the behavior of the axis at a high level when multiple motion control Function Blocks are «simultaneously» activated. This combination of motion profiles is useful in building a more complicated profile or to treat exceptions within a program. (In real implementations 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 state (see diagram below). Any motion command is a transition that changes the state of the axis and, as a consequence, modifies the way the current motion is computed.

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 'Continuous Motion'. The corresponding slave is in the state ' Synchronized Motion'. Connecting a slave axis to a master axis has no influence on the master axis.

The following administrative Function Blocks do not have any effect on the State Diagram: MC_ReadStatus; MC_ReadAxisError; MC_ReadParameter; MC_ReadBoolParameter; MC_WriteParameter; MC_WriteBoolParameter; MC_ReadActualPosition and MC_CamTableSelect.



Note 1: All Function Blocks can be called, although they will not be executed, except MC_Reset and Error – they will generate the transition to StandStill or ErrorStop respectively.
Note 2: Issuing MC_Home in any other state than Standstill will go to ErrorStop, even Homing itself.
Note 3: The transition Error 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'.

Figure 2: FB state behavior

2.2. Error handling

All access to the drive/motion control is done via Function Blocks. Internally these Function Blocks provide basic error checking on the input data. How exactly 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, a basic error report 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 can be stored locally. In this case the FB will generate the error locally.
The error outputs of the relevant FB are reset with falling edge of Execute.

Both centralized and decentralized error handling are possible in combination with 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.

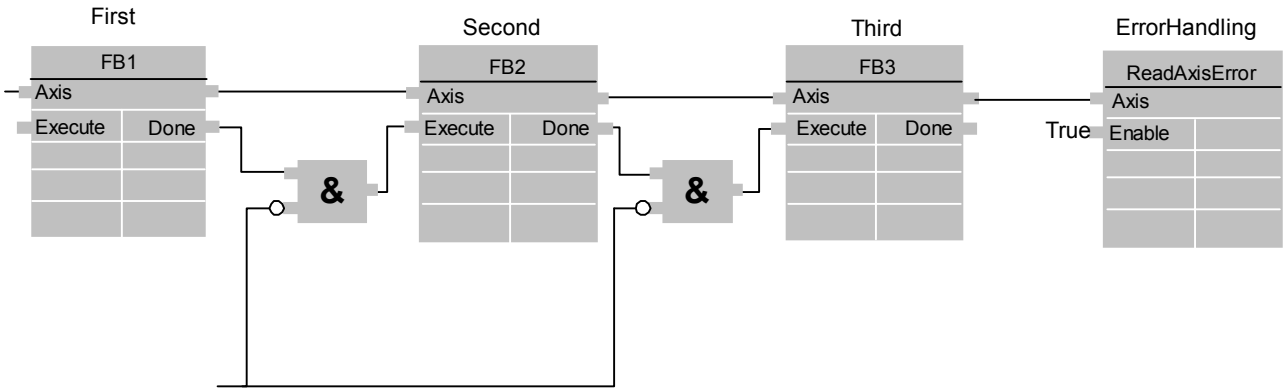


Figure 3: Function Blocks with centralized error handling

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

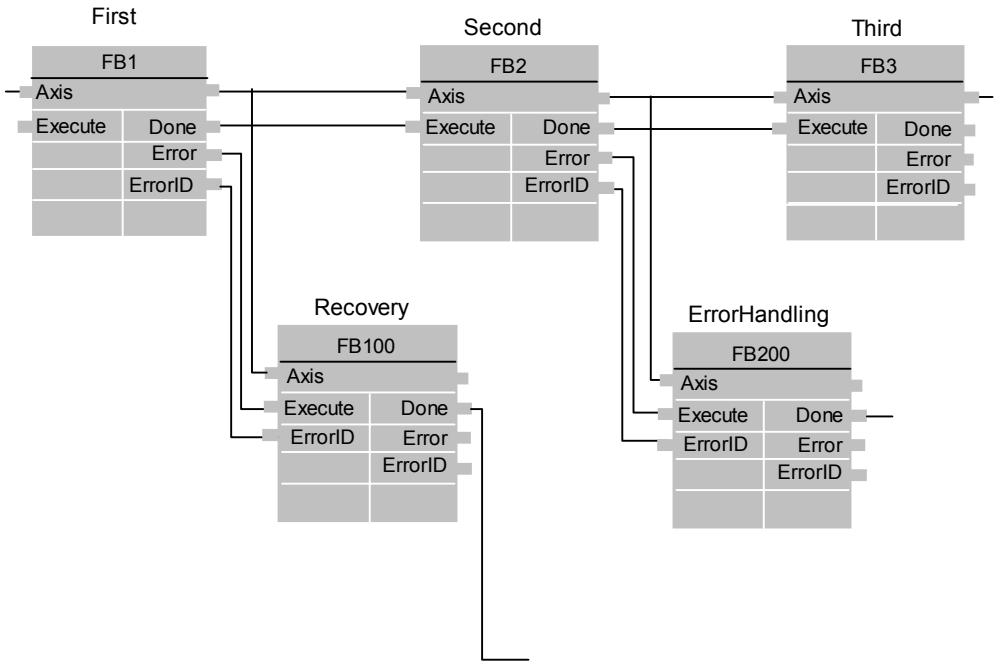


Figure 4: Function blocks with decentralized error handling

2.3. FB interface

2.3.1. General rules

Output status	The Done, InGear, InSync, InVelocity, Error, ErrorID and CommandAborted outputs are reset with the falling edge of execute. In must be guaranteed that they are set for at least one cycle if the corresponding situation occurs, even if execute was reset before. Done and Error outputs are mutually exclusive (cannot be true at the same time). 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.
Input parameters	The parameters are used with the rising edge of the execute input. To modify any parameter it is necessary to put the correct set of values and to trigger the motion again.
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.
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 Velocity, Acceleration, Deceleration and Jerk are always positive values. Position and Distance can be both positive and negative.
Error Handling Behavior	All blocks have two outputs which are dealing with errors that can occur while executing a 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 number Done, InVelocity, InGear, and InSync mean successful completion so these signals are logically exclusive to Error. Types of errors: • Function blocks (e.g. parameters outside range, state machine) • Communication • Drive Instance errors are not always resulting in an axis error (bringing the axis to standstill)
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.
Behavior of Done output	The Done output (as well as InGear, InSync, ..) 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 CommandAborted output	CommandAborted is set, when a commanded motion is interrupted by another motion command or MC_Stop. The reset-behavior of CommandAborted is like Done. When CommandAborted occurs, the other output-signals like InVelocity are reset.

Table 2: General Rules

2.3.2. 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 it but this is outside of the scope of this document. The refresh rate of this structure is also implementation dependent.

AXIS_REF data type declaration:

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

Example:

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

2.3.3. 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.

2.3.4. Why the command input is edge sensitive

The “execute” input for the different Function Blocks described in this document is always triggering the function with its rising edge. The reason for this is that with edge triggered “execute” input you may command new input values 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 axes 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 5 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”.

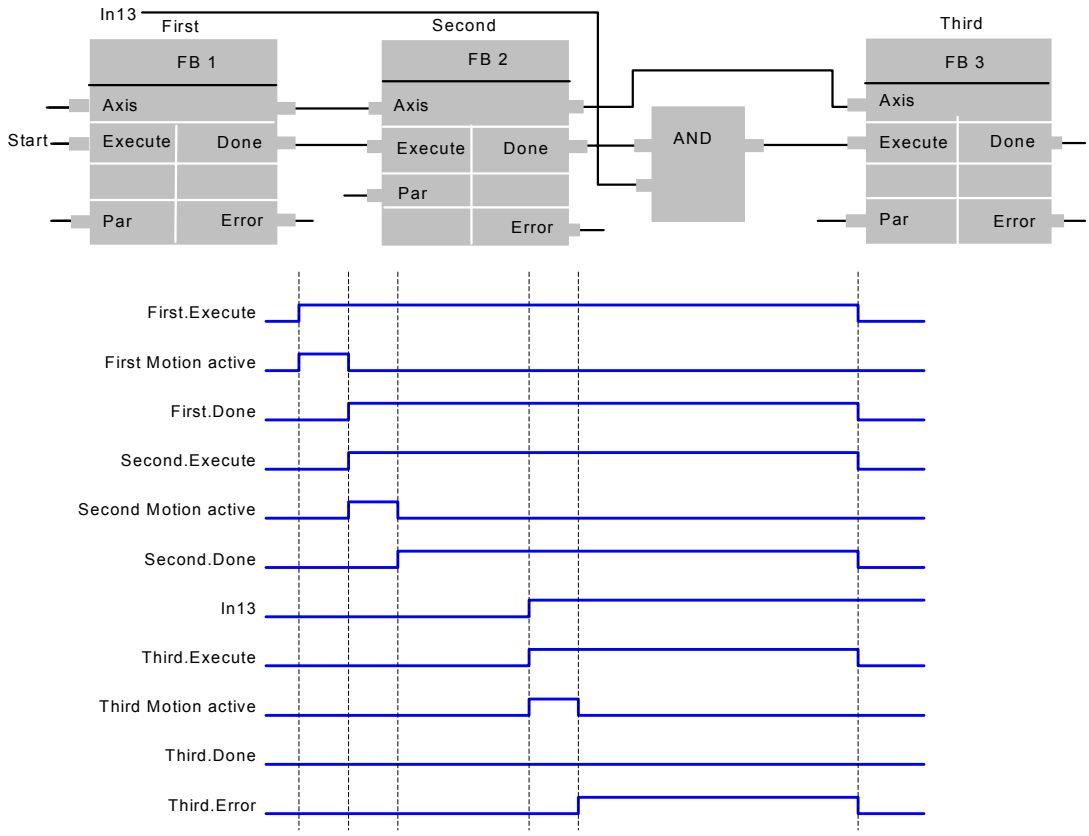


Figure 5: Function blocks to perform a complex movement

2.4. Example 1: the same Function block instance controls different motions of an axis

Figure 6 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.

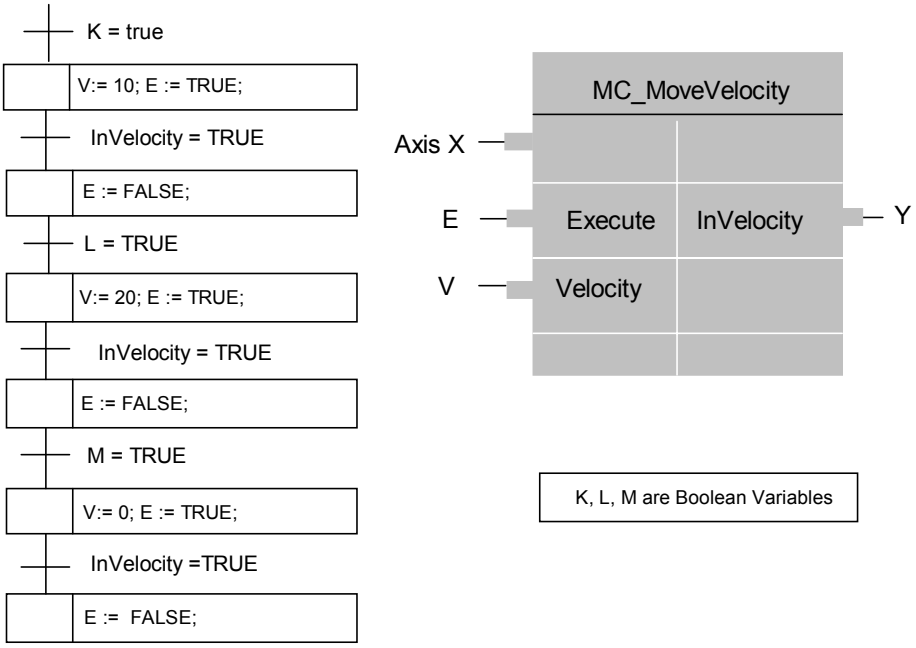


Figure 6: Single FB usage with a SFC

The following timing diagram explains how it will work.

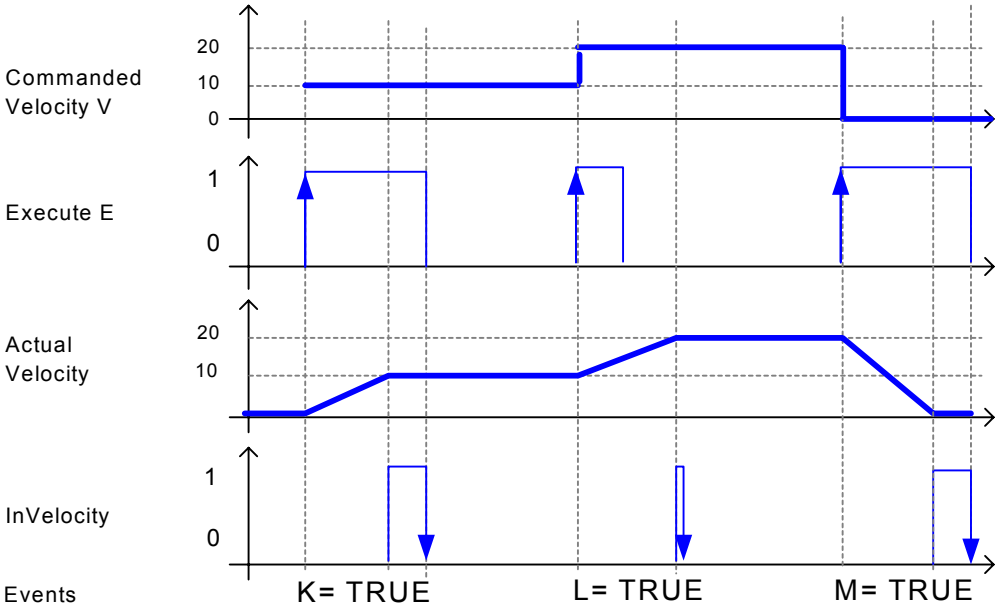


Figure 7: Timing diagram for a usage of a single FB

Note: The second InVelocity is set for only one cycle because the Execute has gone low before the Actual Velocity equals Commanded Velocity.

2.5. 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.

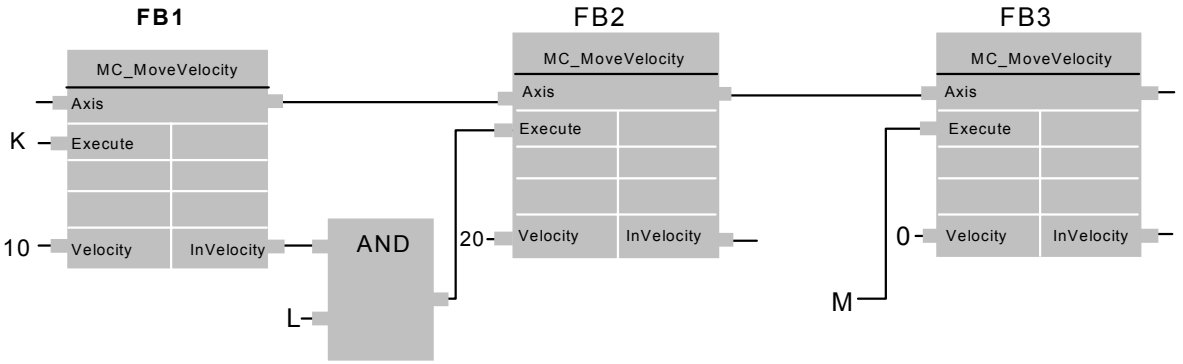


Figure 8: Cascaded Function Blocks

The timing diagram:

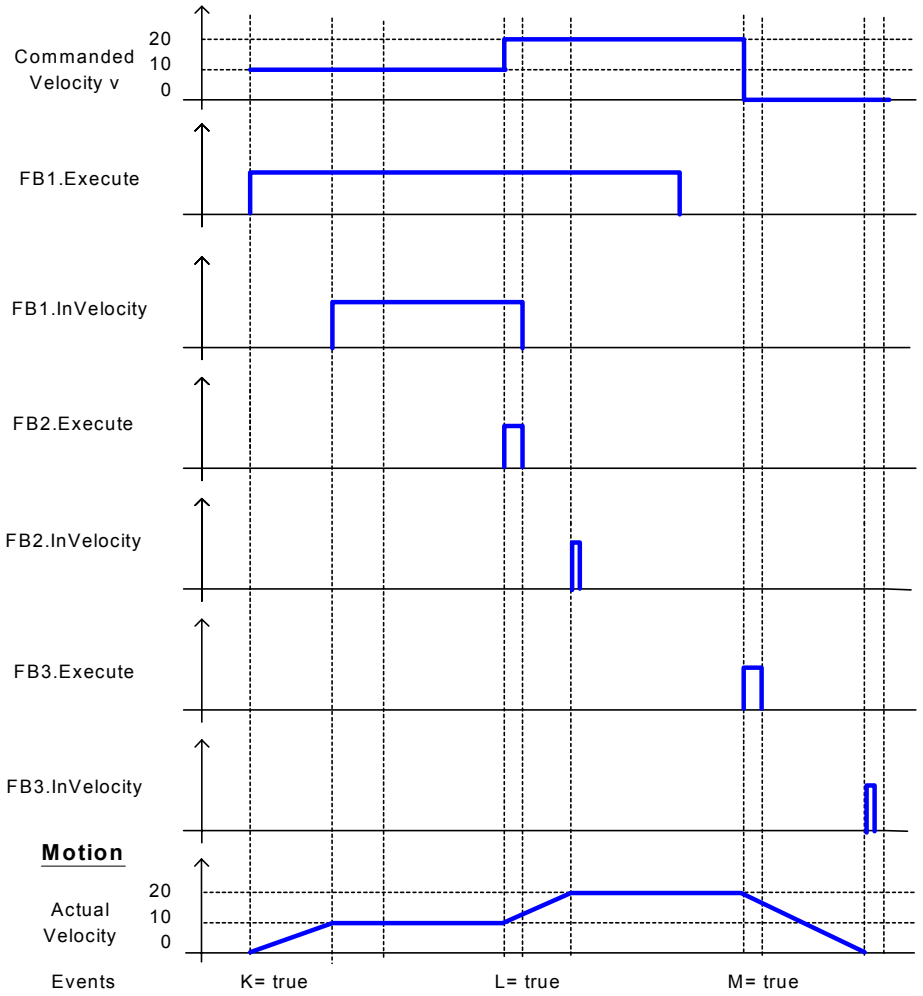


Figure 9: Cascaded Function Blocks timing diagram

A corresponding solution written in LD looks like:

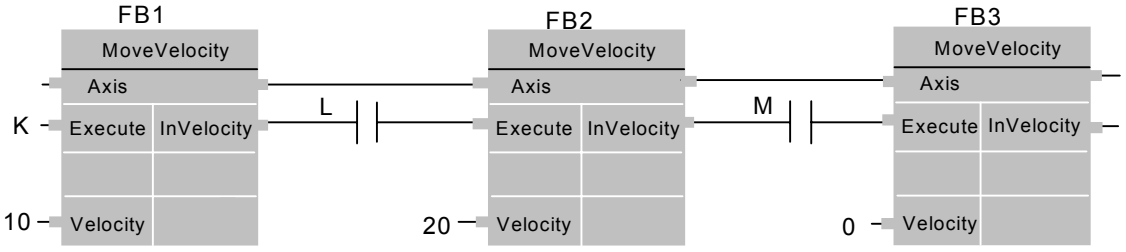
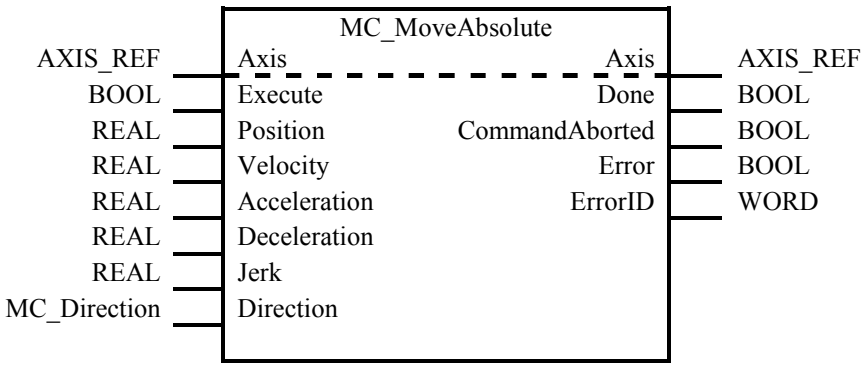


Figure 10: Cascaded Function Blocks with LD

3. Single-Axis Function Blocks

3.1. MoveAbsolute

FB-Name			**MC_MoveAbsolute**	
This Function Block commands a controlled motion at a specified absolute position.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
VAR_INPUT				
	B	Execute	BOOL	Start the motion at rising edge
	B	Position	REAL	Target position for the motion (in technical unit [u]) (negative or positive)
	E	Velocity	REAL	Value of the maximum velocity (always positive) (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)
	E	Direction	MC_Direction	Enum type (1-of-4 values: positive direction, shortest way, negative direction, current direction)
VAR_OUTPUT				
	B	Done	BOOL	Commanded position reached
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
Notes: -				



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

1. The left part of 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 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 moves directly to the position 10000 although the position of 6000 is not yet reached.

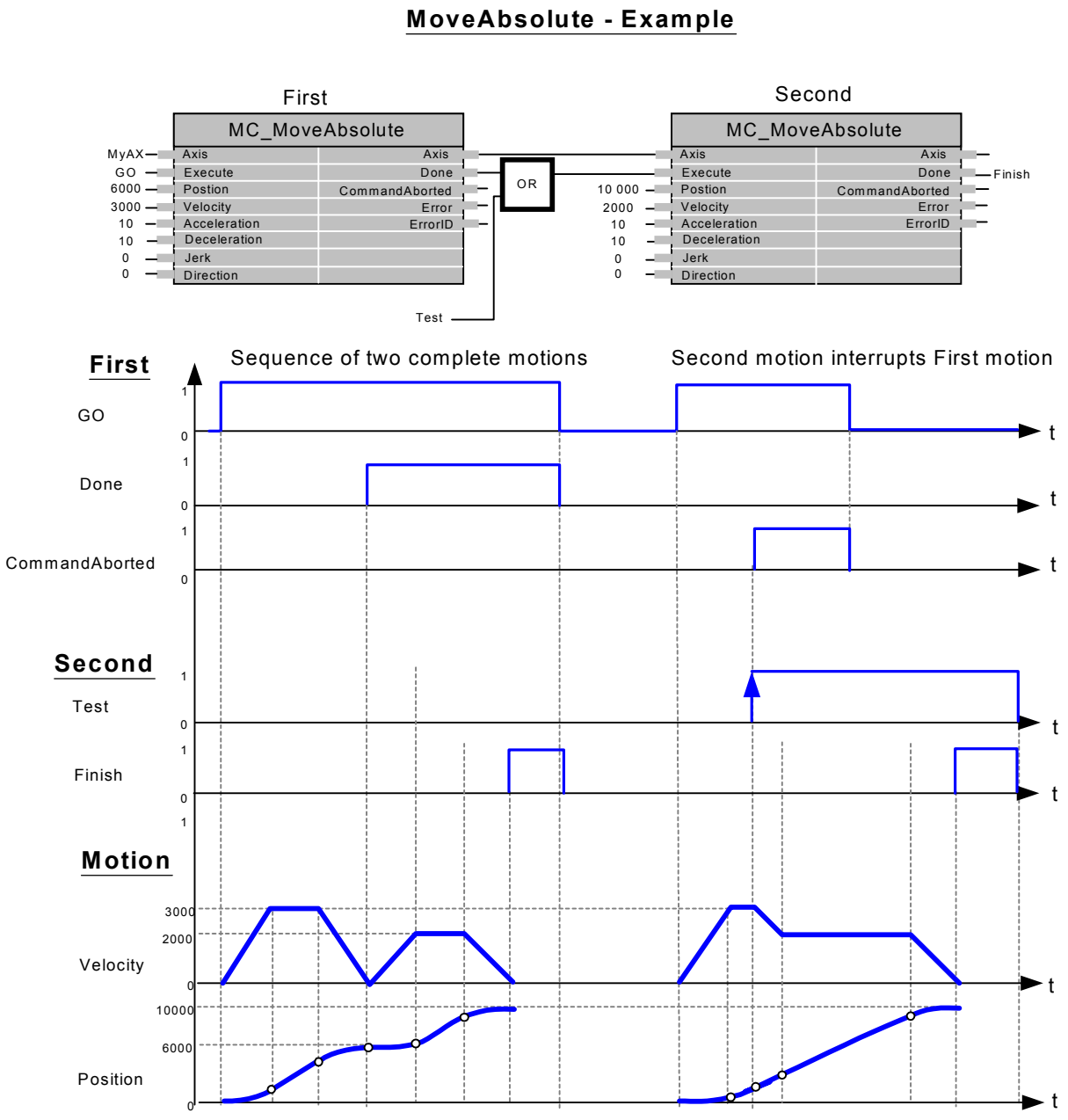
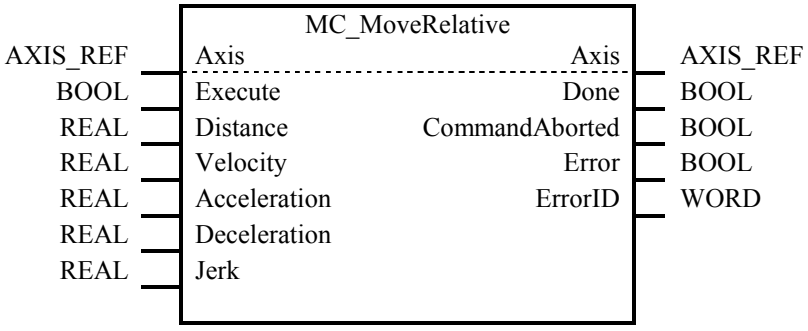


Figure 11: Timing diagram for MC_MoveAbsolute

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

3.2. Move Relative

FB-Name		**MC_MoveRelative**		
This Function Block commands a controlled motion of a specified distance relative to the actual position at the time of the execution.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
VAR_INPUT				
	B	Execute	BOOL	Start the motion at rising edge
	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]
VAR_OUTPUT				
	B	Done	BOOL	Commanded distance reached
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
Notes: -				



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

1. The left part of 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 to the distance 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.

MoveRelative - Example

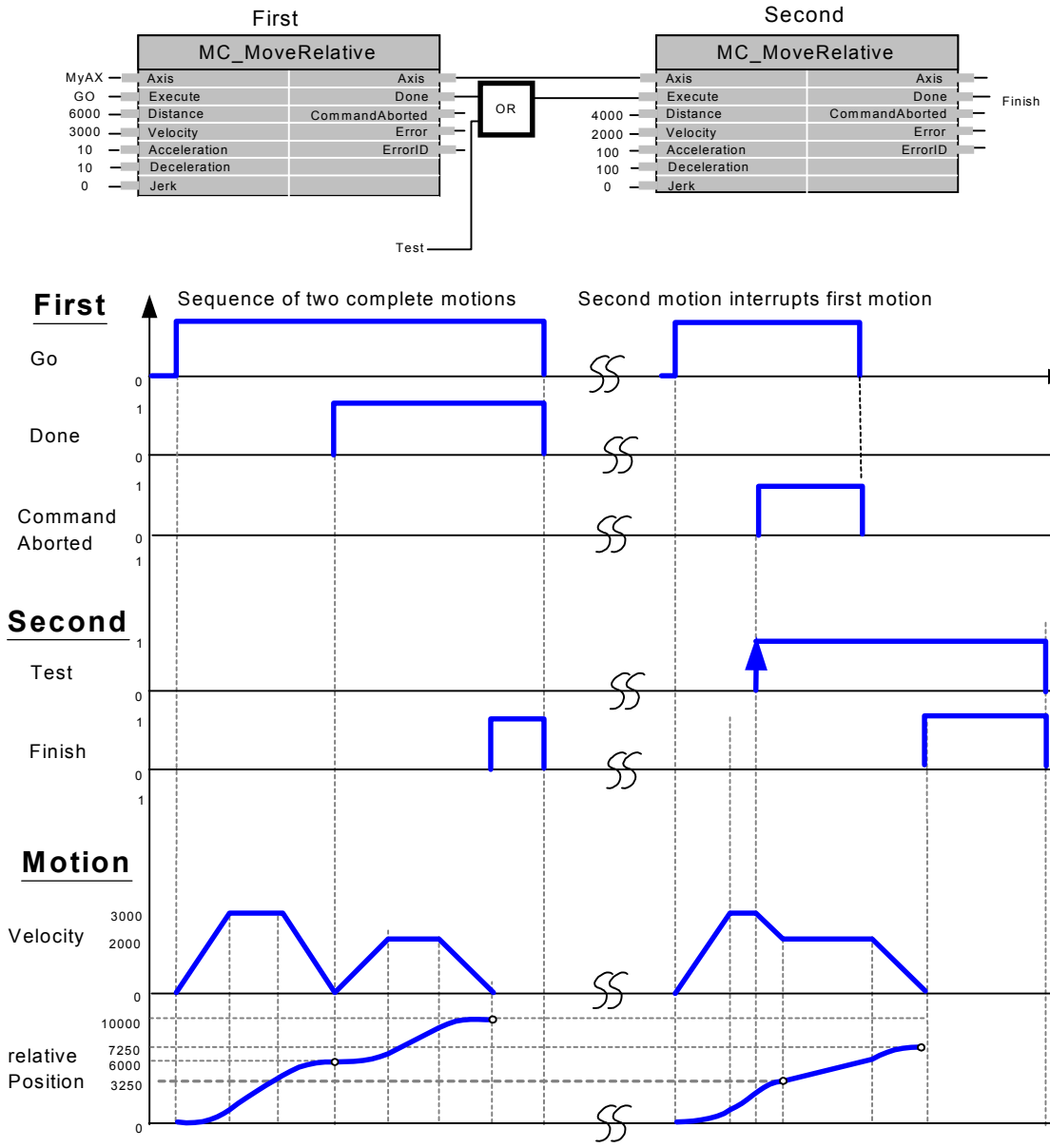
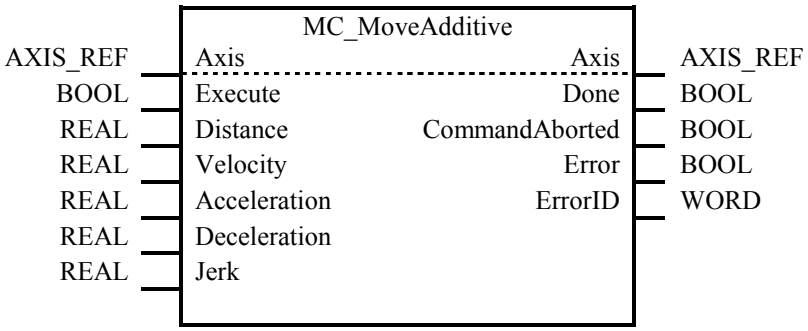


Figure 12: Timing diagram for MC_MoveRelative

3.3. Move Additive

FB-Name		**MC_MoveAdditive**	
This Function Block commands a controlled motion of a specified relative distance additional to the original commanded position in the discrete motion state. If the FB is activated in the Continuous Mode the specified relative distance is added to the actual position at the time of the execution.			
VAR_IN_OUT			
B	Axis	AXIS_REF	
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
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]
VAR_OUTPUT			
B	Done	BOOL	Commanded distance reached
E	CommandAborted	BOOL	Command is aborted by another command
B	Error	BOOL	Signals that error has occurred within 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 Discrete Motion state:

1. The left part of 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 to the distance 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.

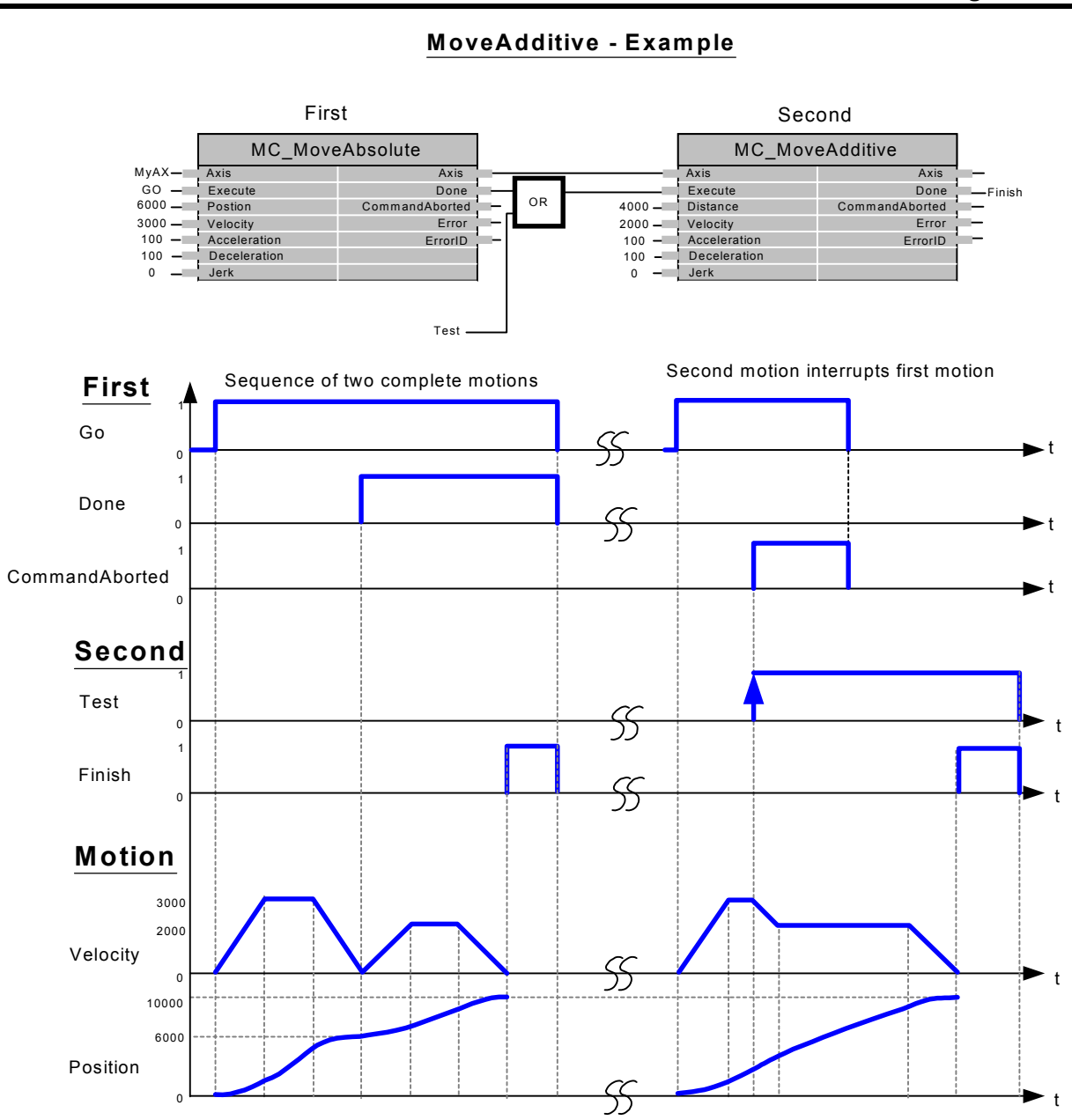
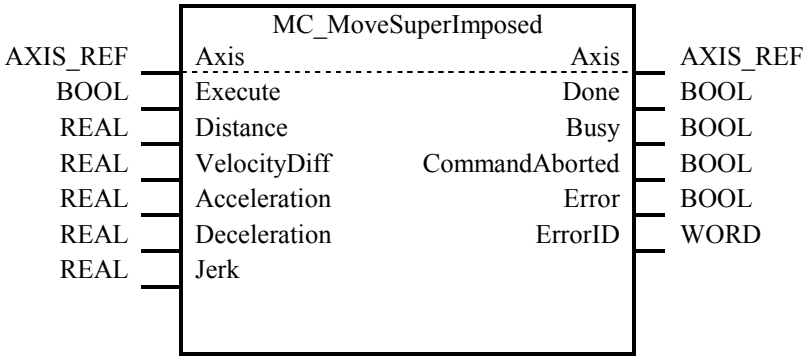


Figure 13: Timing diagram for MC_MoveAdditive

3.4. 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	
VAR_INPUT				
	B	Execute	BOOL	Start the motion at rising edge
	B	Distance	REAL	Additional Distance that is superimposed (in technical unit [u])
	E	VelocityDiff	REAL	Value of the maximum velocity difference to the ongoing 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
	B	Busy	BOOL	Superimposed motion currently processed
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification

Note:
- In the state discrete motion the FB MC_MoveSuperimposed causes a change of the velocity and the commanded position of an ongoing motion.
- In the state standstill the FB MC_MoveSuperimposed acts like MC_MoveRelative
- MoveSuperimposed does not interrupt the active command



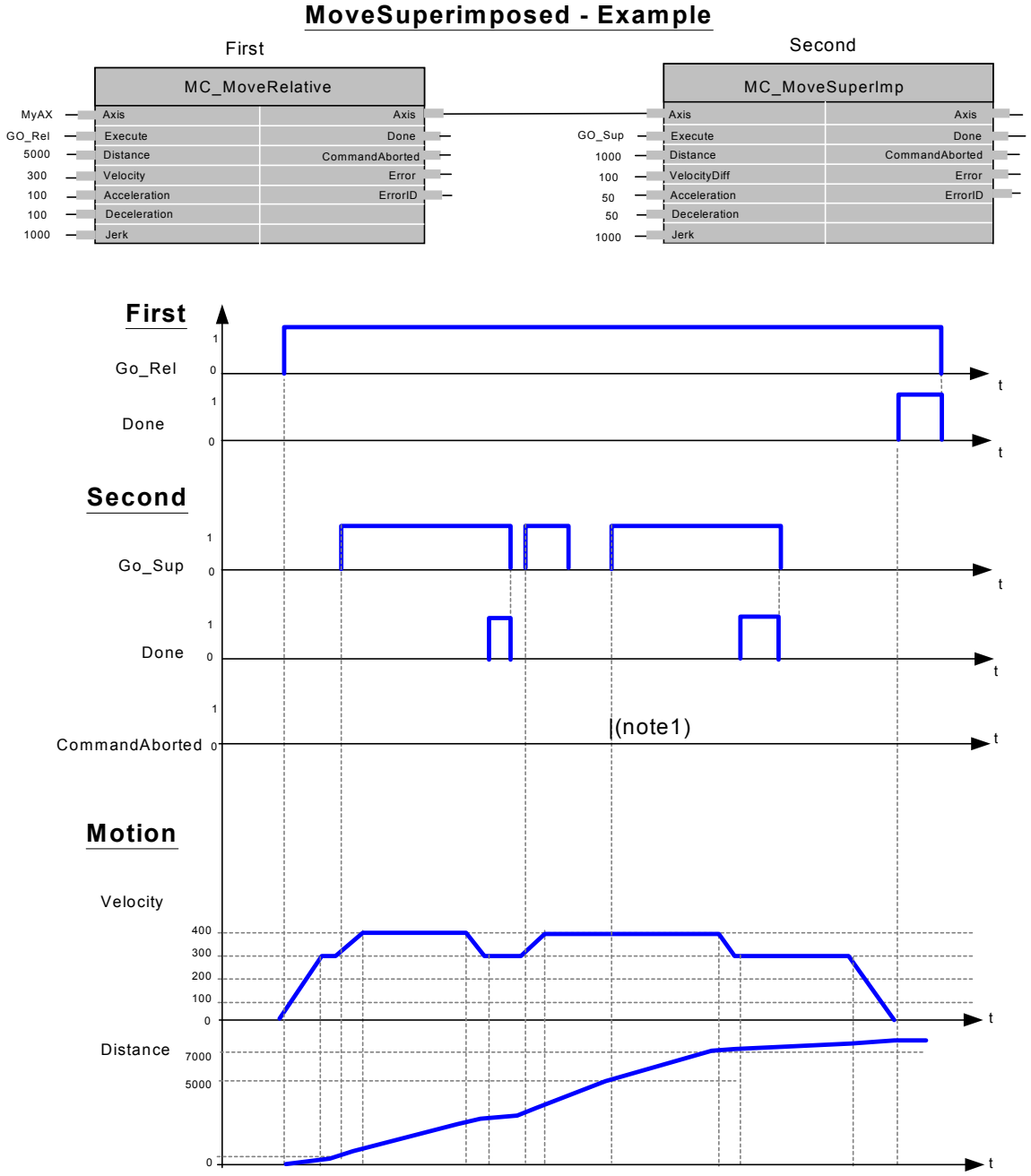


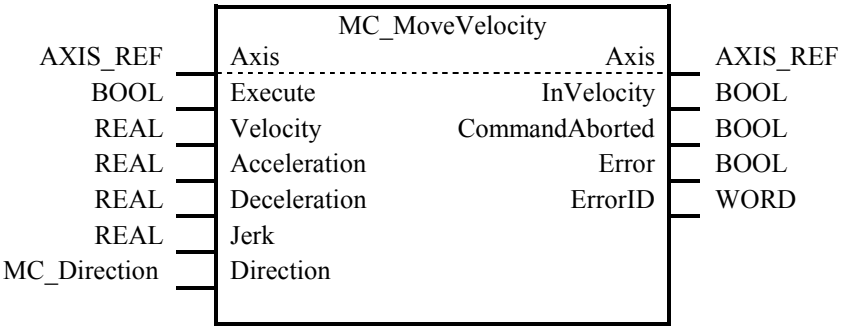
Figure 14: Timing diagram for MC_MoveSuperimposed

Note 1: the CommandAborted is not visible here, because the new command works on the same instance (see general rules 2.3.1)
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

3.5. 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	
VAR_INPUT				
	B	Execute	BOOL	Start the motion at rising edge
	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	Direction	MC_Direction	Enum type (1-of-4 values: positive direction, negative direction, and current direction. Note: shortest way not applicable)
VAR_OUTPUT				
	B	InVelocity	BOOL	Commanded velocity reached (first time reached)
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within 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 or at the falling edge of Execute.
- In combination with MC_MoveSuperimposed, the output InVelocity stays TRUE once the velocity setpoint of the axis equaled the commanded velocity.



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

1. The left part of 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.
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 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.

MoveVelocity - Example

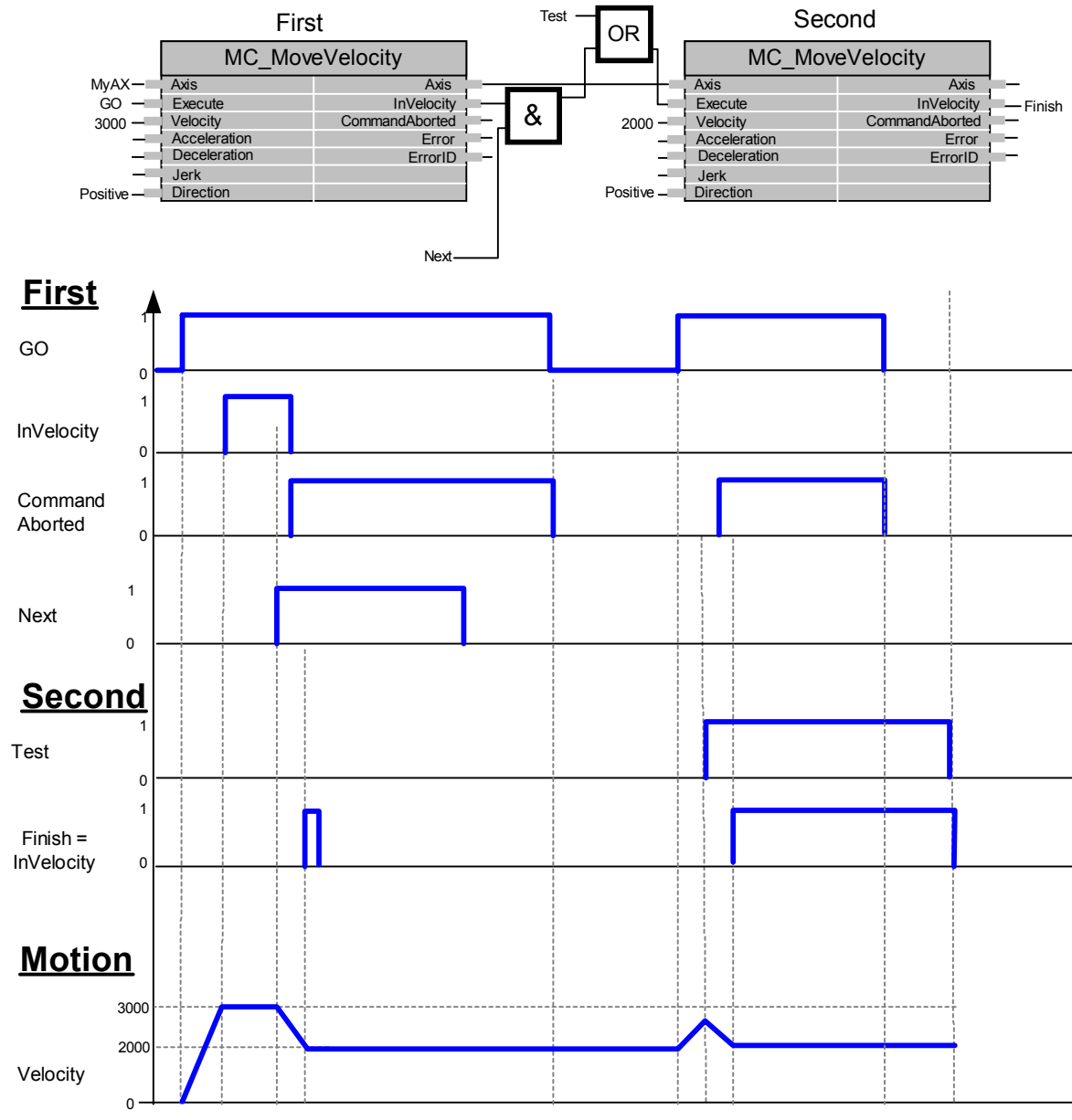
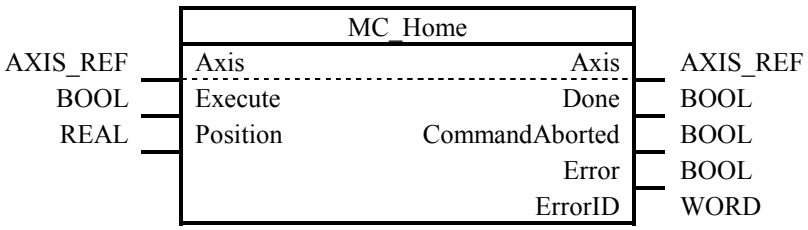


Figure 15: MC_MoveVelocity timing diagram

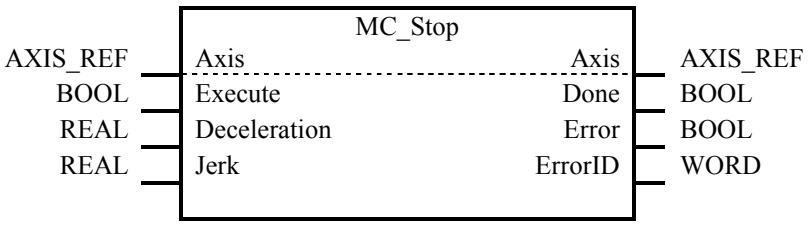
3.6. Home

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 axis' parameters. The position input is used to set the absolute position when reference signal is detected. It completes at standstill.			
VAR_IN_OUT			
B	Axis	AXIS_REF	
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
B	Position	REAL	Absolute position when the reference signal is detected [u]
VAR_OUTPUT			
B	Done	BOOL	Standstill is reached
E	CommandAborted	BOOL	Command is aborted by another command
B	Error	BOOL	Signals that error has occurred within Function block
E	ErrorID	WORD	Error identification



3.7. 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. With the Done output set, the state is transferred to Standstill. While the axis is in state Stopping, no other FB can perform any motion on the same axis.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
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 and execute in not True
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
Note: As long as Execute is high, the axis remains in the state 'Stopping' and may not be executing any other command.				

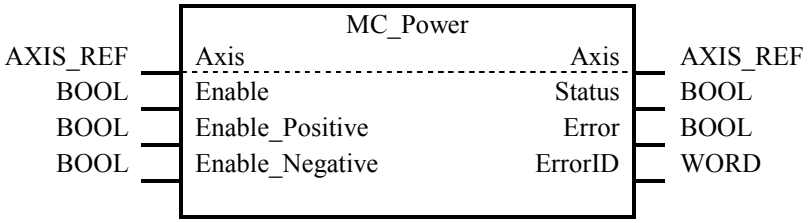


3.8. Power

FB-Name		**MC_Power**		
This Function Block controls the power stage (on or off).				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
VAR_INPUT				
	B	Enable	BOOL	As long as ‘ Enable’ is true, power is on.
	E	Enable_Positive	BOOL	As long as ‘ Enable’ is true, permits motion in positive direction only
	E	Enable_Negative	BOOL	As long as ‘ Enable’ is true, permits motion in negative direction only (_Pos & _Neg can be switched on both)
VAR_OUTPUT				
	B	Status	BOOL	Effective state of the power stage
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification

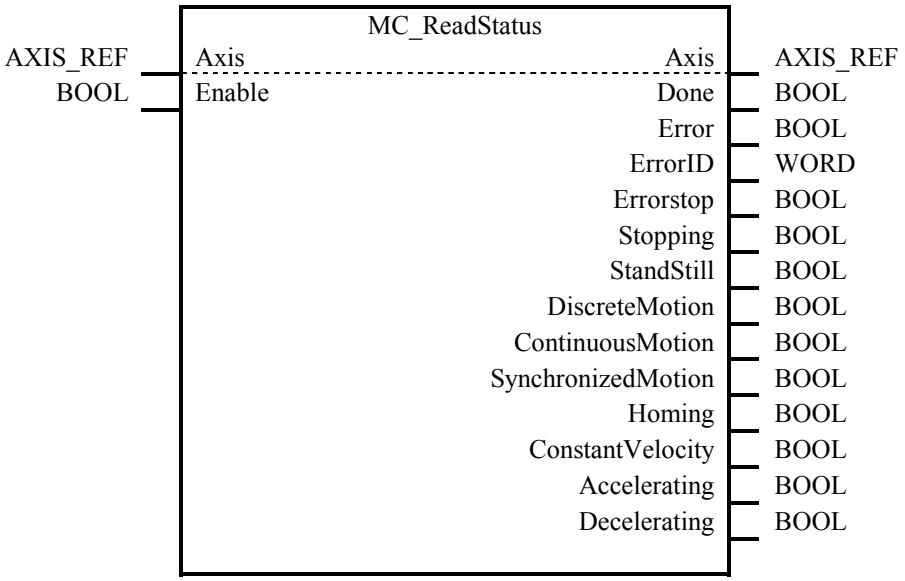
Notes:

- 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
- Enable_Positive and Enable_Negative are both level triggered



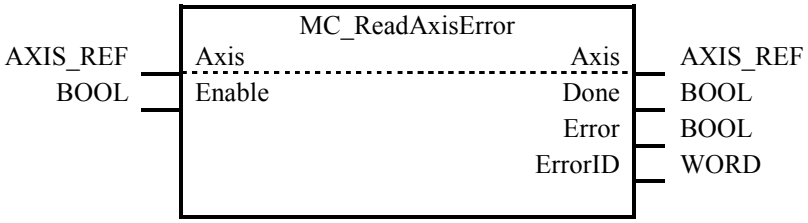
3.9. ReadStatus

FB-Name		**MC_ReadStatus**		
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	
VAR_INPUT				
	B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Done	BOOL	True if valid outputs available
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
	B	Errorstop	BOOL	See state diagram
	B	Stopping	BOOL	See state diagram
	B	StandStill	BOOL	See state diagram
	B	DiscreteMotion	BOOL	See state diagram
	B	ContinuousMotion	BOOL	See state diagram
	E	SynchronizedMotion	BOOL	See State Diagram
	E	Homing	BOOL	See state diagram
	E	ConstantVelocity	BOOL	Motor moves with constant velocity
	E	Accelerating	BOOL	Increasing energy of the motor
	E	Decelerating	BOOL	Decreasing energy of the motor



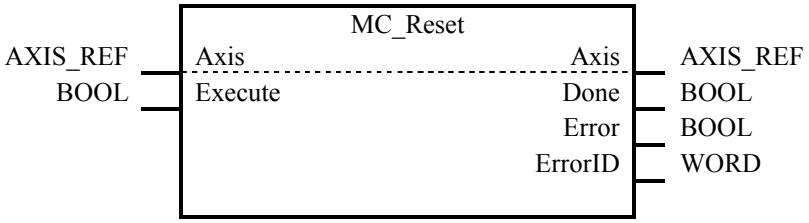
3.10. ReadAxisError

FB-Name		**MC_ReadAxisError**		
This Function Block indicates general axis errors not relating to the Function Blocks.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
VAR_INPUT				
		Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Done	BOOL	Value is available
	B	Error	BOOL	Error flag
	B	ErrorID	WORD	Indicates the kind of errors
Notes: • this is equivalent to the MC_ReadParameter FB error parameter. • error codes are vendor specific.				



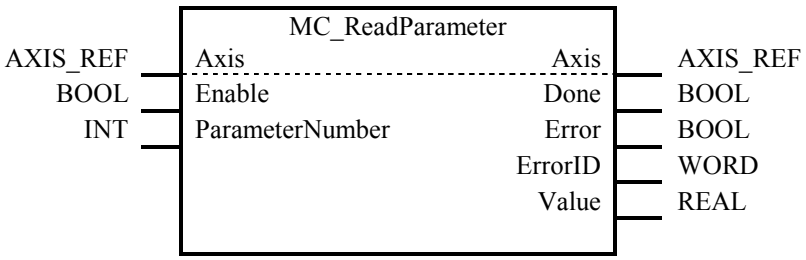
3.11. Reset

FB-Name		**MC_Reset**		
This Function Block makes the transition from the state ErrorStop to StandStill by resetting all internal axis-related errors – it does not effect the output of the FB instances.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
VAR_INPUT				
	B	Execute	BOOL	Resets the axis at the rising edge
VAR_OUTPUT				
	B	Done	BOOL	Standstill state is reached
	B	Error	BOOL	Error flag
	B	ErrorID	WORD	Indicates the kind of errors
Note: the application of MC_RESET in other states then the state ErrorStop is vendor specific				

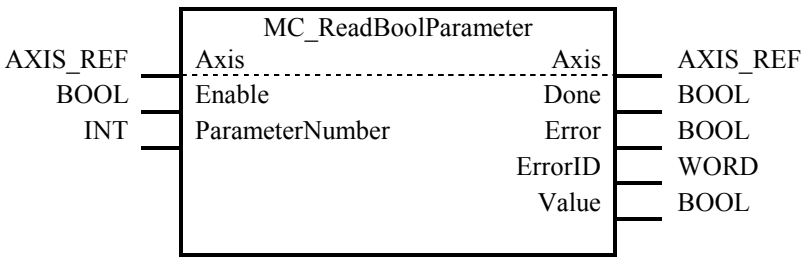


3.12. ReadParameter & ReadBoolParameter

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 supplier dependent FB for it				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
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	Done	BOOL	Parameter available
	B	Error	BOOL	Signals that error has occurred within 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	
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	Done	BOOL	Parameter vailable
	B	Error	BOOL	Signals that error has occurred within 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 a vendor is 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)

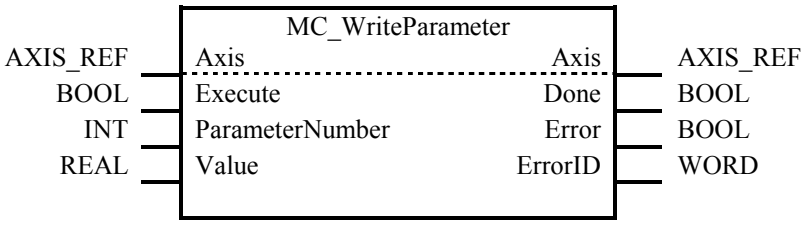
PN	Name	Datatype	B/E	R/W	Comments
1	CommandedPosition	REAL	B	R	Commanded position
2	SWLimitPos	REAL	E	R/W	Positive Software limit switch position
3	SWLimitNeg	REAL	E	R/W	Negative Software limit switch position
4	EnableLimitPos	BOOL	E	R/W	Enable positive software limit switch
5	EnableLimitNeg	BOOL	E	R/W	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
15	MaxDecelerationAppl	REAL	E	R/W	Maximal allowed deceleration of the axis
16	MaxJerk	REAL	E	R/W	Maximal allowed jerk of the axis

Table 3: Parameters for MC_ReadParameter and MC_WriteParameter

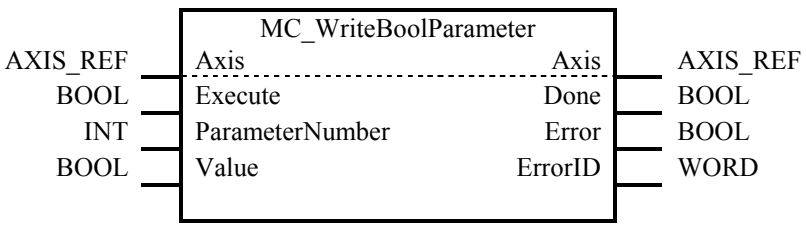
Extensions by any supplier or user are also allowed at the end of the list, although this can affect portability between different platforms. Parameter-numbers from 0 to 999 are reserved for the standard. Numbers greater than 999 indicate supplier-specific parameters.

3.13. WriteParameter & WriteBoolParameter

FB-Name		**MC_WriteParameter**		
This Function Block modifies the value of a vendor specific parameter.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
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 to be specified later)
	B	Value	REAL	New value of the specified parameter
VAR_OUTPUT				
	B	Done	BOOL	Parameter successfully written
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
Notes: The parameters are defined in the table above (under MC_ReadParameter, writing allowed)				

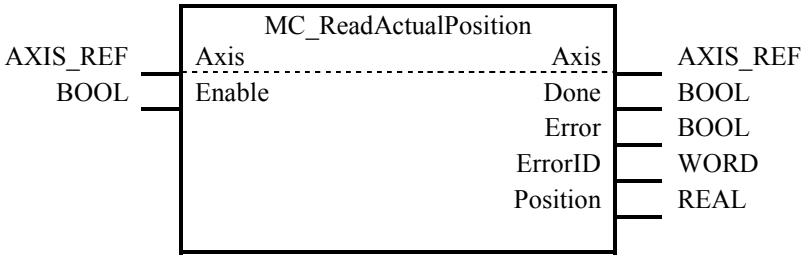


FB-Name		**MC_WriteBoolParameter**		
This Function Block modifies the value of a vendor specific parameter of type BOOL.				
VAR_IN_OUT				
	B	Axis	AXIS_REF	
VAR_INPUT				
	B	Execute	BOOL	Write the value of the parameter at rising edge
	B	ParameterNumber	INT	Number of the parameter (correspondence between number and parameter is to be specified later)
	B	Value	BOOL	New value of the specified parameter
VAR_OUTPUT				
	B	Done	BOOL	Parameter successfully written
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
Notes: The parameters are defined in the table above (under MC_ReadParameter, writing allowed)				



3.14. ReadActualPosition

FB-Name		**MC_ReadActualPosition**	
This Function Block returns the actual position.			
VAR_IN_OUT			
B	Axis	AXIS_REF	
VAR_INPUT			
B	Enable	BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT			
B	Done	BOOL	Value is available
B	Error	BOOL	Signals that error has occurred within Function block
E	ErrorID	WORD	Error identification
B	Position	REAL	New absolute position (in axis' unit [u])



3.15. Position Profile

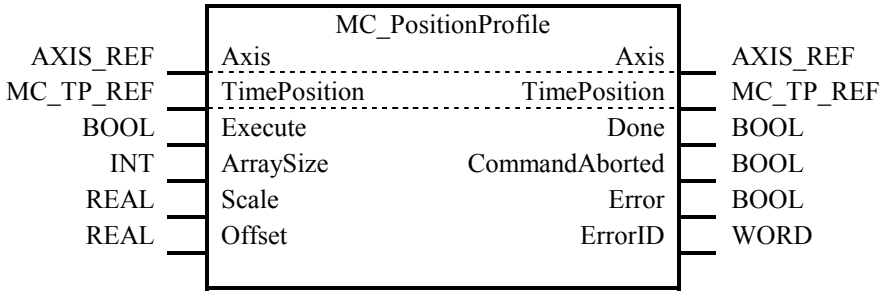
FB-Name		**MC_PositionProfile**	
This Function Block commands a time-position locked motion profile			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to axis
B	TimePosition	MC_TP_REF	Reference to Time / Position. Description - see note below
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
B	ArraySize	INT	Dimension of array
E	Scale	REAL	Overall scaling factor of the profile
E	Offset	REAL	Overall offset for profile [u]
VAR_OUTPUT			
B	Done	BOOL	Profile completed
E	CommandAborted	BOOL	Command is aborted by another command
B	Error	BOOL	Signals that error has occurred within Function block
E	ErrorID	WORD	Error identification

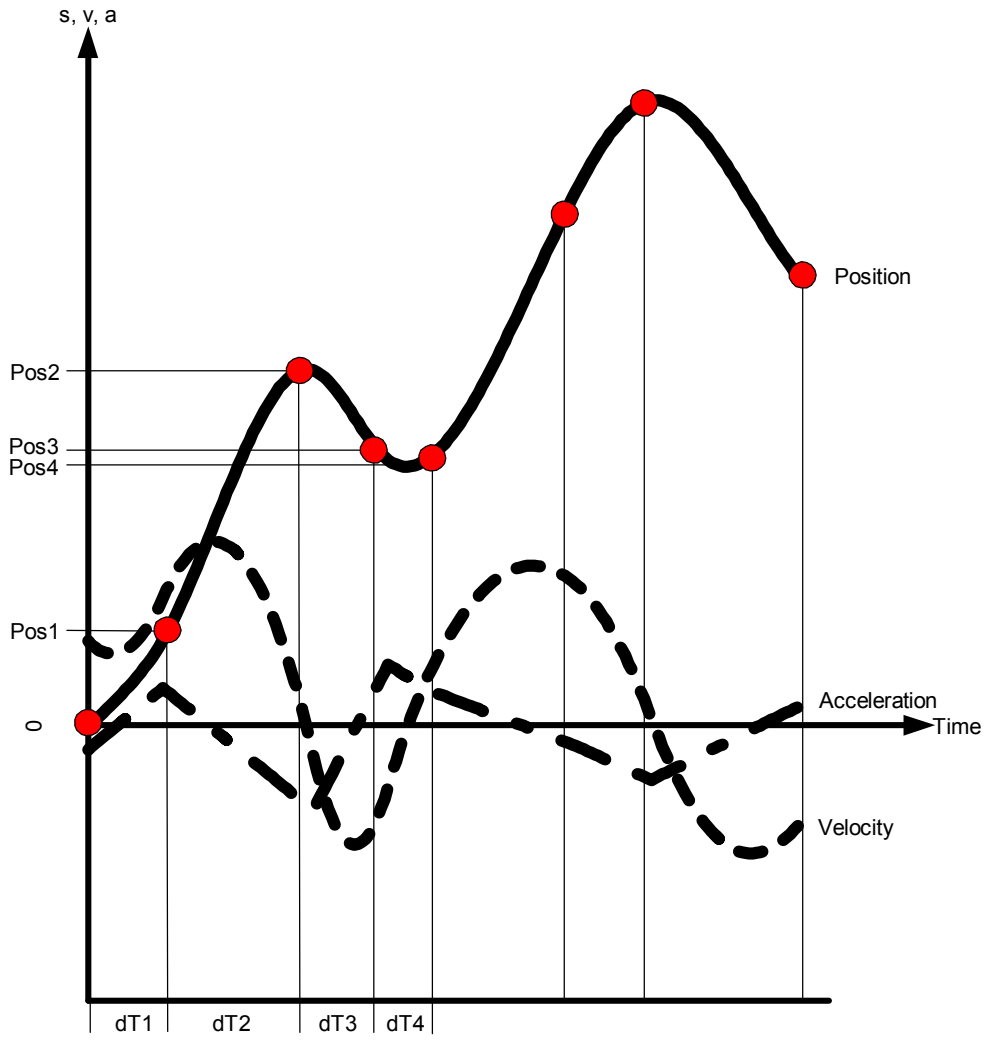
Notes:

- MC_TP_REF is a supplier specific datatype. An example for this datatype is given here below:
  - The content of 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
    delta_time : TIME
    position : REAL
    END_STRUCT
    END_TYPE

  - TYPE MC_TP_TABLE STRUCT
    Number_of_pairs : INT
    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 shift between different profiles
- Alternatively to this FB, the CAM FB coupled to a virtual master can be used





deltaTime	absPos
dT1	Pos1
dT2	Pos2
dT3	Pos3
dT4	Pos4

Figure 16: Example of Time / Position Profile

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

3.16. Velocity Profile

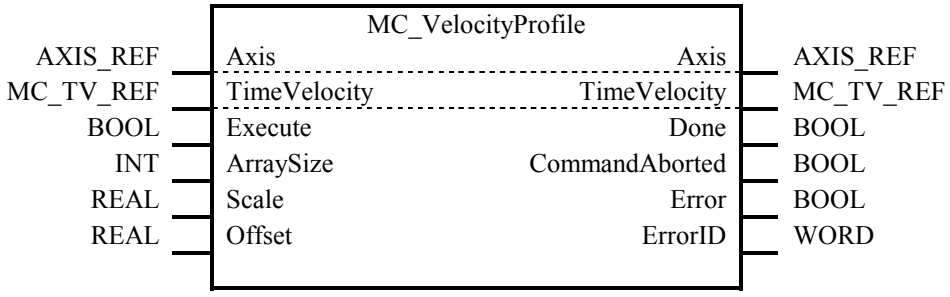
FB-Name		**MC_VelocityProfile**	
This Function Block commands a time-velocity locked motion profile			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to axis
B	TimeVelocity	MC_TV_REF	Reference to Time / Velocity. Description - see note below
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
B	ArraySize	INT	Dimension of array
E	Scale	REAL	Overall scaling factor of the profile
E	Offset	REAL	Overall offset for profile [u/s]
VAR_OUTPUT			
B	Done	BOOL	Profile completed
E	CommandAborted	BOOL	Command is aborted by another command
B	Error	BOOL	Signals that error has occurred within 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.
  - TYPE MC_TV STRUCT
    delta_time : TIME
    velocity : REAL
    END_STRUCT
    END_TYPE

  - TYPE MC_TV_TABLE STRUCT
    Number_of_pairs : INT
    IsAbsolute : BOOL
    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 shift between different profiles
- Alternatively to this FB, the CAM FB coupled to a virtual master can be used



3.17. Acceleration Profile

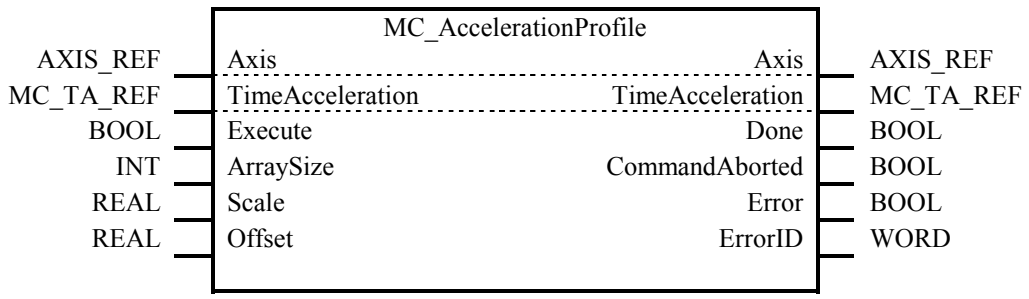
FB-Name		**MC_AccelerationProfile**	
This Function Block commands a time-acceleration locked motion profile			
VAR_IN_OUT			
B	Axis	AXIS_REF	Reference to axis
B	TimeAcceleration	MC_TA_REF	Reference to Time / Acceleration. Description – see note below
VAR_INPUT			
B	Execute	BOOL	Start the motion at rising edge
B	ArraySize	INTEGER	Dimension of array
E	Scale	REAL	Scale factor for acceleration amplitude
E	Offset	REAL	Overall offset for profile [u/s^2]
VAR_OUTPUT			
B	Done	BOOL	Profile completed
E	CommandAborted	BOOL	Command is aborted by another command
B	Error	BOOL	Signals that error has occurred within 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
    delta_time : TIME
    acceleration : REAL
    END_STRUCT
    END_TYPE

  - TYPE MC_TA_TABLE STRUCT
    Number_of_pairs : INT
    IsAbsolute : BOOL
    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

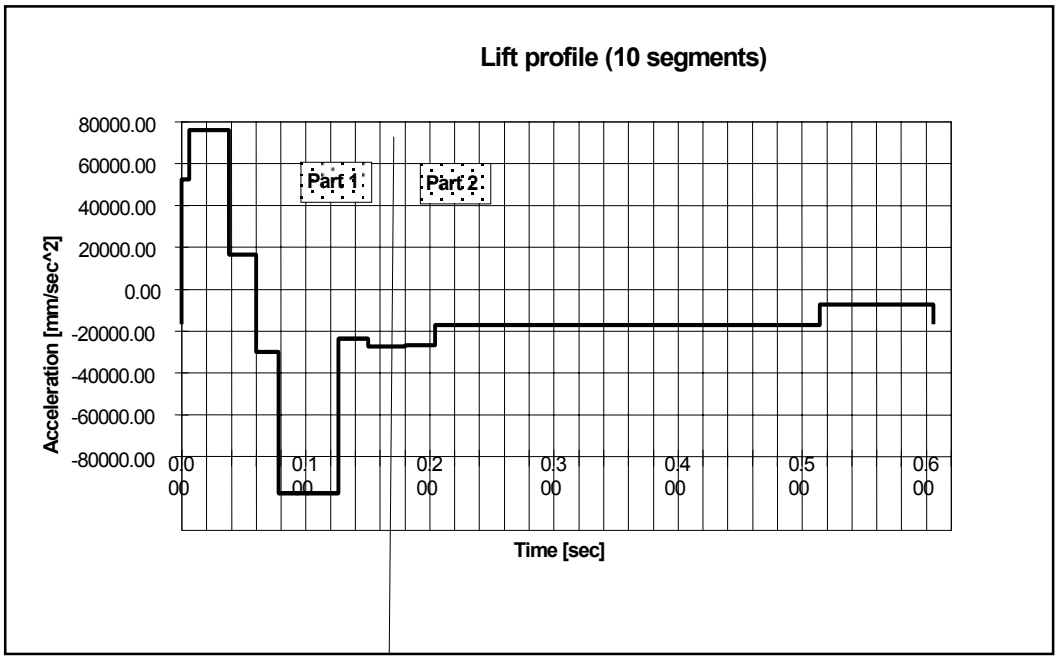
Advantages:

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

Disadvantages

- Higher programming abstraction level with existing tools

Acceleration Profile, 10 segments only



Resulting Position Profile

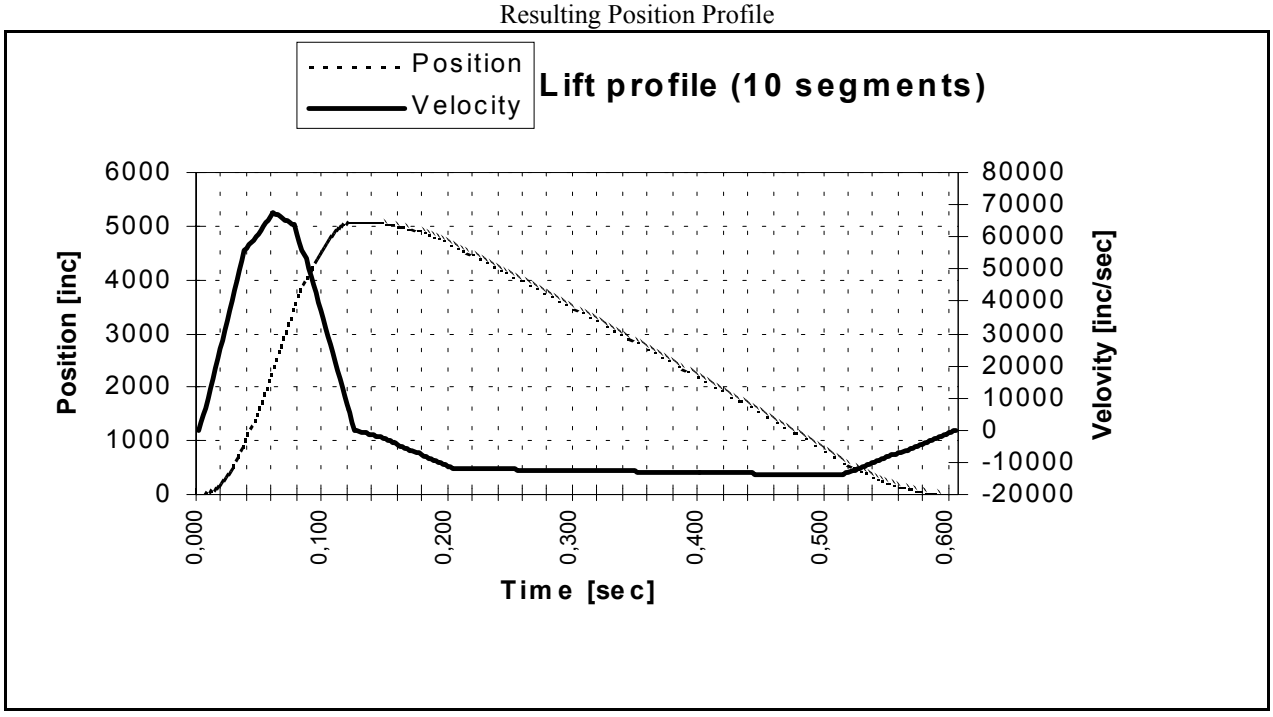


Figure 17: Example of Time / Acceleration Profile

4. Multi-Axes Function Blocks

With Multi-Axes 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-axes 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. Introduction into Camming

Two types of Camming:
1. Periodic: Once a curve is executed the camming immediately starts again at the beginning of the curve.
2. Non periodic: After a curve is executed the execution stops

Camming may be done with several combined cam tables, which are executed sequentially. Between the different cam curves may be a gap (wait for trigger) in the execution.

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 done 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.

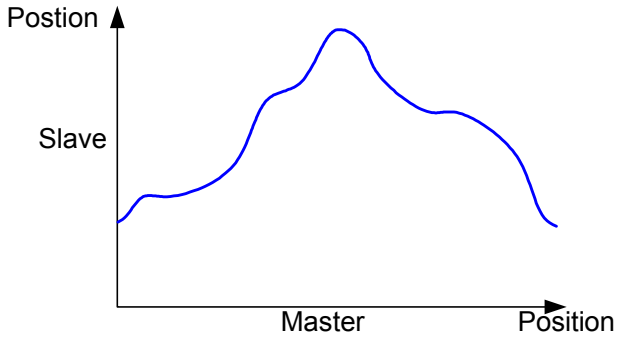


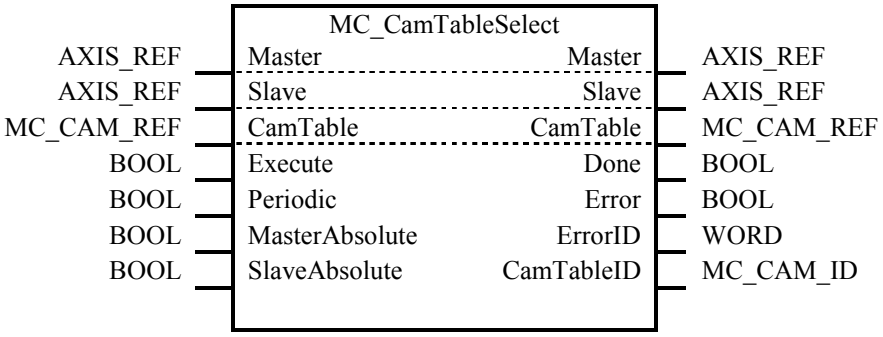
Figure 18: CAM profile illustration

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. CamTableSelect

FB-Name		**MC_CamTableSelect**	
This Function Block selects the CAM tables by setting the pointers to the relevant tables			
VAR_IN_OUT			
B	Master	AXIS_REF	Reference to master axis
B	Slave	AXIS_REF	Reference to 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
E	MasterAbsolute	BOOL	1 = absolute; 0 = relative coordinates
E	SlaveAbsolute	BOOL	1 = absolute; 0 = relative coordinates
VAR_OUTPUT			
B	Done	BOOL	Pre-selection done
B	Error	BOOL	Signals that error has occurred within 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 datatype
- MC_CAM_ID is a supplier specific datatype

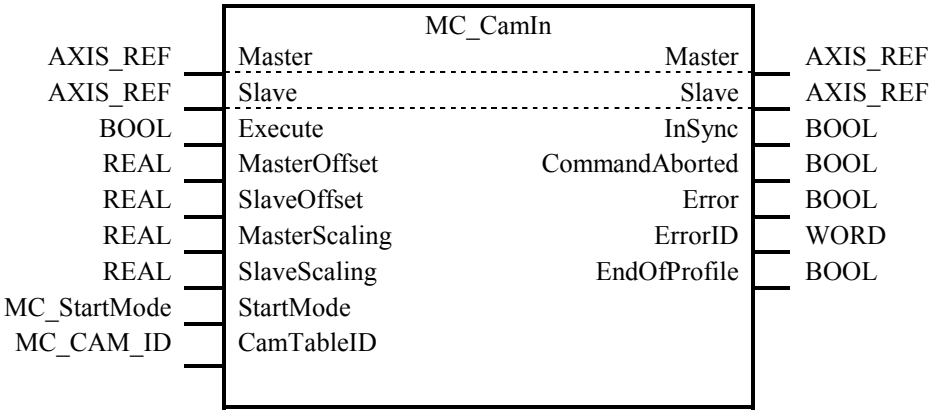


4.3. CamIn

FB-Name		**MC_CamIn**		
This Function Block engages the CAM				
VAR_IN_OUT				
	B	Master	AXIS_REF	Reference to master axis
	B	Slave	AXIS_REF	Reference to slave axis
VAR_INPUT				
	B	Execute	BOOL	Start at rising edge
	E	MasterOffset	REAL	Offset of master table
	E	SlaveOffset	REAL	Offset of slave table
	E	MasterScaling	REAL	Factor for the master profile (default = 1.0)
	E	SlaveScaling	REAL	Factor for the slave profile (default = 1.0)
	E	StartMode	MC_StartMode	Start mode: absolute, relative, or ramp-in
	E	CamTableID	MC_CAM_ID	Identifier of CAM Table to be used, linked to output of MC_CamTableSelect
VAR_OUTPUT				
	B	InSync	BOOL	Cam is engaged for the first time.
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
	E	EndOfProfile	BOOL	Pulsed output signaling the cyclic end of the CAM Profile

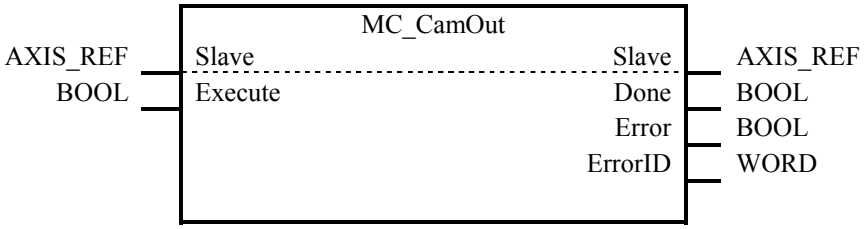
Notes:

- It is not required that the master is at standstill
- If the actual master and slave positions are not corresponding to the offset values when MC_CamIn is executed, either an error occurs or the system deals with it automatically
- Ramp-in is the master-distance parameter where the slave to ramp in into the cam profile (" flying coupling") (Supplier specific)
- This FB is not merged with the MC_CamTableSelect FB because this separation enables changes on the fly



4.4. CamOut

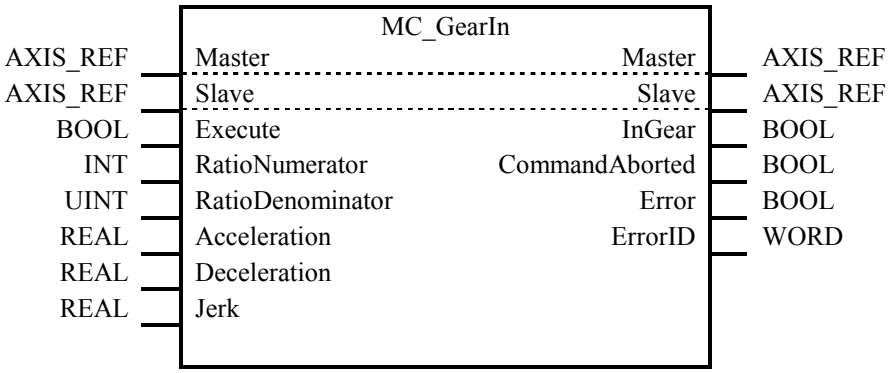
FB-Name			**MC_CamOut**	
This Function Block disengage the Slave from the Master axis immediately				
VAR_IN_OUT				
	B	Slave	AXIS_REF	Slave Axis reference
VAR_INPUT				
	B	Execute	BOOL	Start to disengage the slave from the master
VAR_OUTPUT				
	B	Done	BOOL	Disengaging completed
	B	Error	BOOL	Signals that error has occurred within 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: keep last velocity.				



4.5. 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 master axis
	B	Slave	AXIS_REF	Slave Axis reference
VAR_INPUT				
	B	Execute	BOOL	Start the gearing process at the rising edge
	B	RatioNumerator	INT	Gear ratio Numerator
	B	RatioDenominator	INT	Gear ratio Denominator
	E	Acceleration	REAL	Acceleration for gearing in
	E	Deceleration	REAL	Deceleration for gearing in
	E	Jerk	REAL	Jerk of Gearing
VAR_OUTPUT				
	B	InGear	BOOL	Commanded gearing completed
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification

Notes:
1. The gearing ratio can be changed while MC_GearIn is running, using a consecutive MC_GearIn command without the necessity to MC_GearOut first
2. InGear is set the first time the ratio is reached.



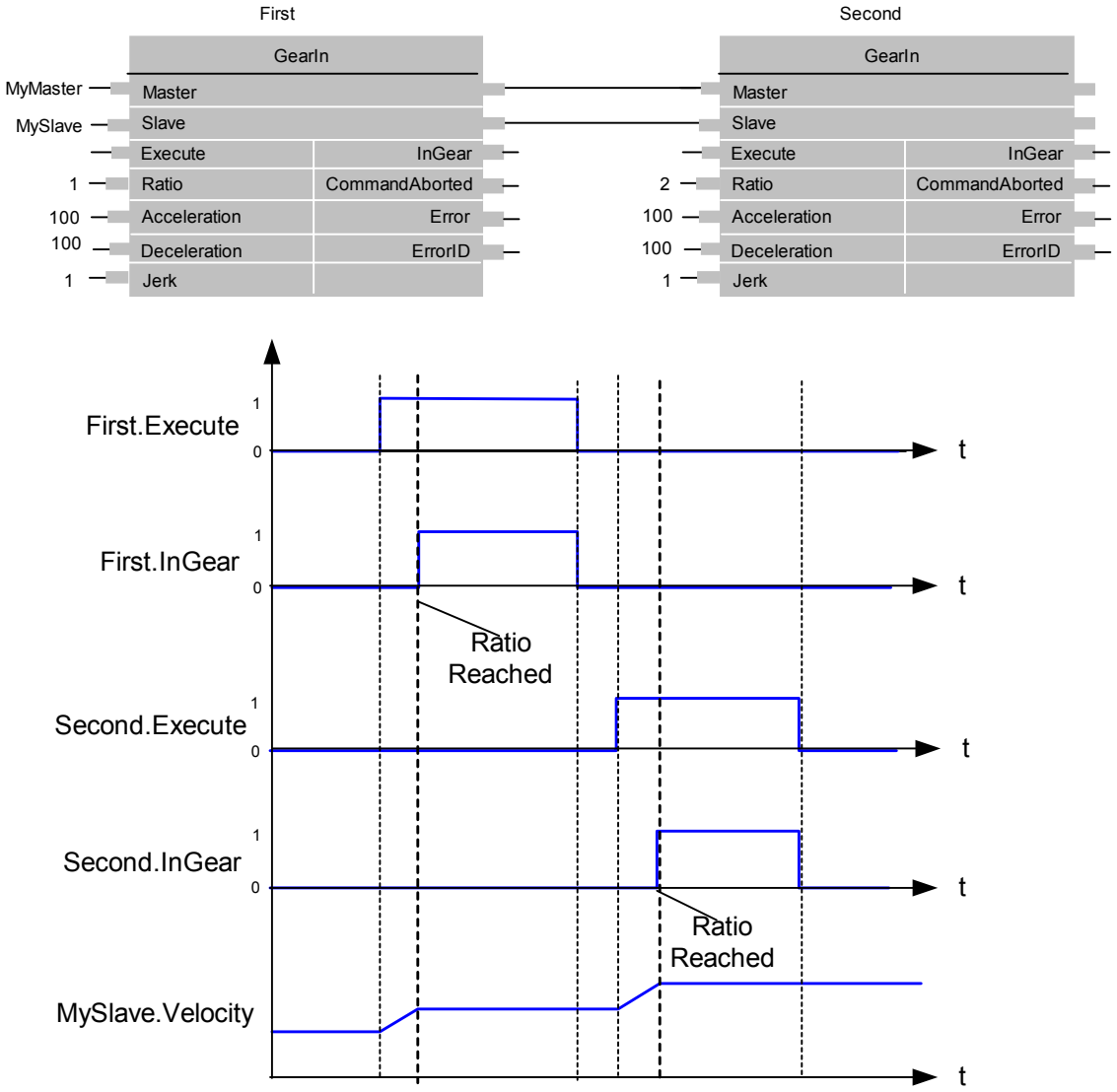
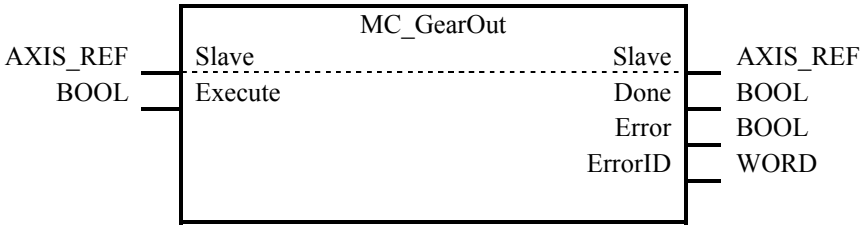


Figure 19: Gear timing diagram

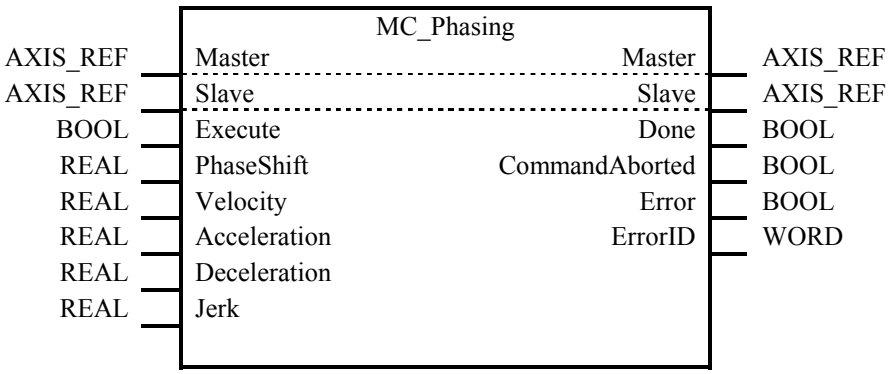
4.6. GearOut

FB-Name			**MC_GearOut**	
This Function Block disengages the Slave from the Master axis				
VAR_IN_OUT				
	B	Slave	AXIS_REF	Slave Axis reference
VAR_INPUT				
	B	Execute	BOOL	Start disengaging process at the rising edge
VAR_OUTPUT				
	B	Done	BOOL	Disengaging completed
	B	Error	BOOL	Signals that error has occurred within 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: keep last velocity.				



4.7. Phasing

FB-Name		**MC_Phasing**		
This Function Block creates a phase shift in the master position of a slave axis. The master position is shifted in relation to the real physical position. It is like opening a coupling on the master shaft for a moment. It is used to delay or advance an axis to its master. The phase shift is seen from the slave. The master itself does not know that there is a phase shift. The phase shift remains until another 'Phasing' command changes it again.				
VAR_IN_OUT				
	B	Master	AXIS_REF	Reference to master axis
	B	Slave	AXIS_REF	Slave Axis reference
VAR_INPUT				
	B	Execute	BOOL	Start the phasing process at the rising edge
	B	PhaseShift	REAL	Phase difference in master [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]
VAR_OUTPUT				
	B	Done	BOOL	Commanded phasing reached
	E	CommandAborted	BOOL	Command is aborted by another command
	B	Error	BOOL	Signals that error has occurred within Function block
	E	ErrorID	WORD	Error identification
Note: Phase, Velocity, Acceleration, Deceleration and Jerk of a phase shift are controlled by the FB.				



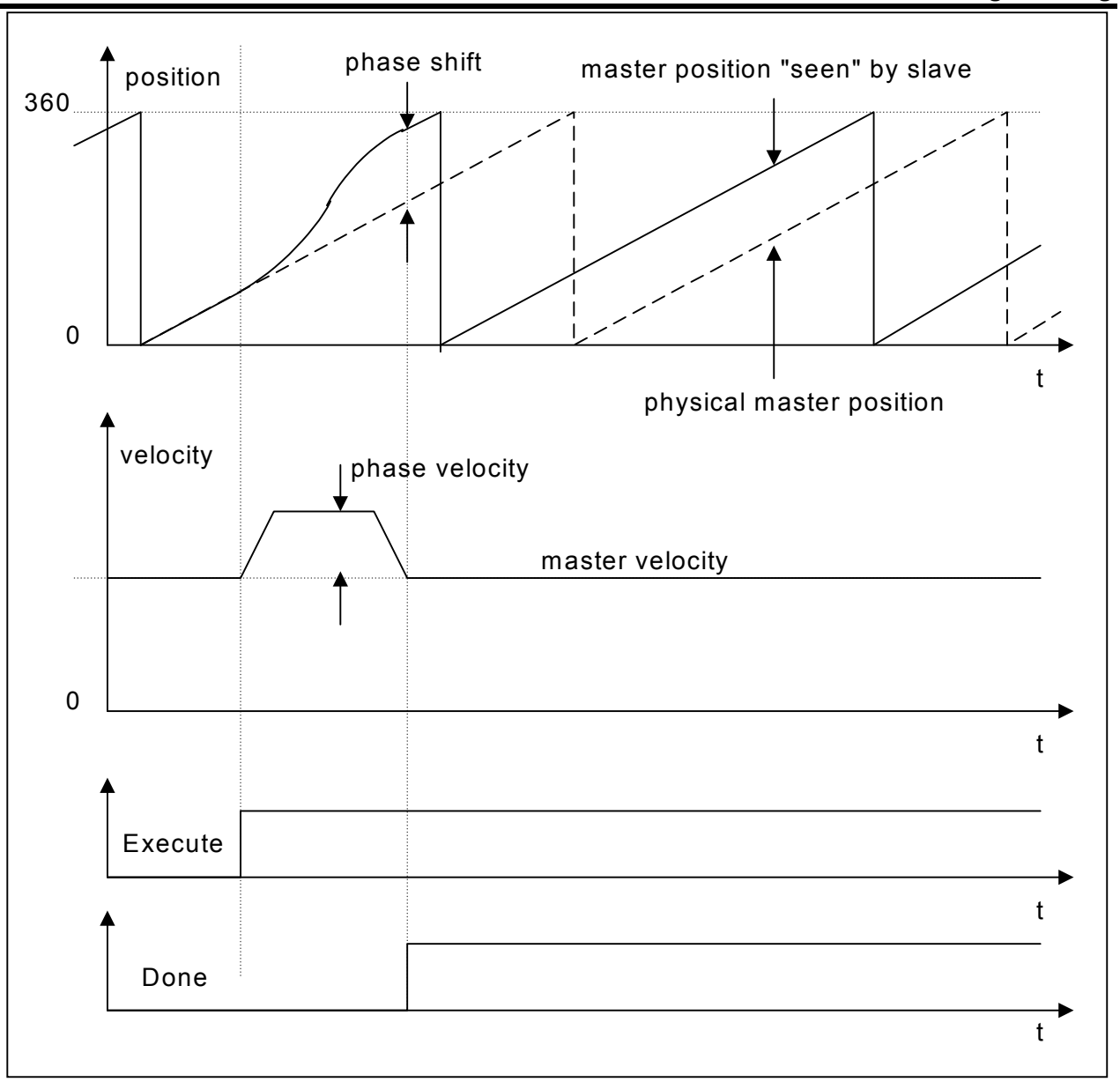


Figure 20: Timing example of MC_Phasing

5. Application of MC FB – Drilling Example

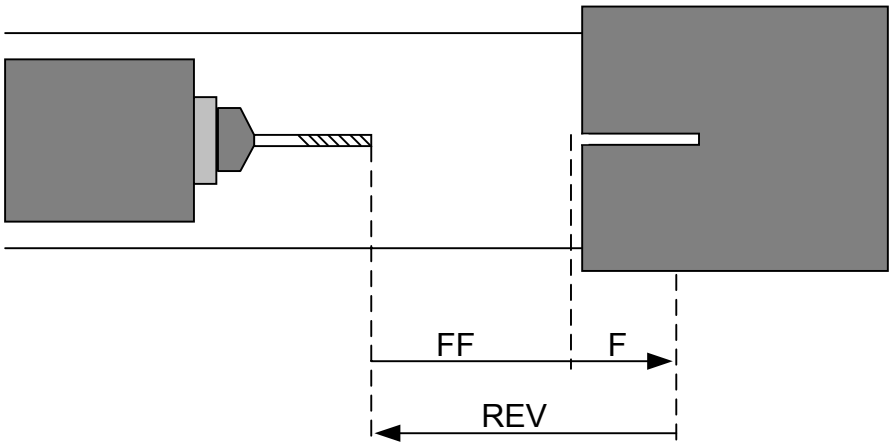


Figure 21: Example of a simple drilling unit

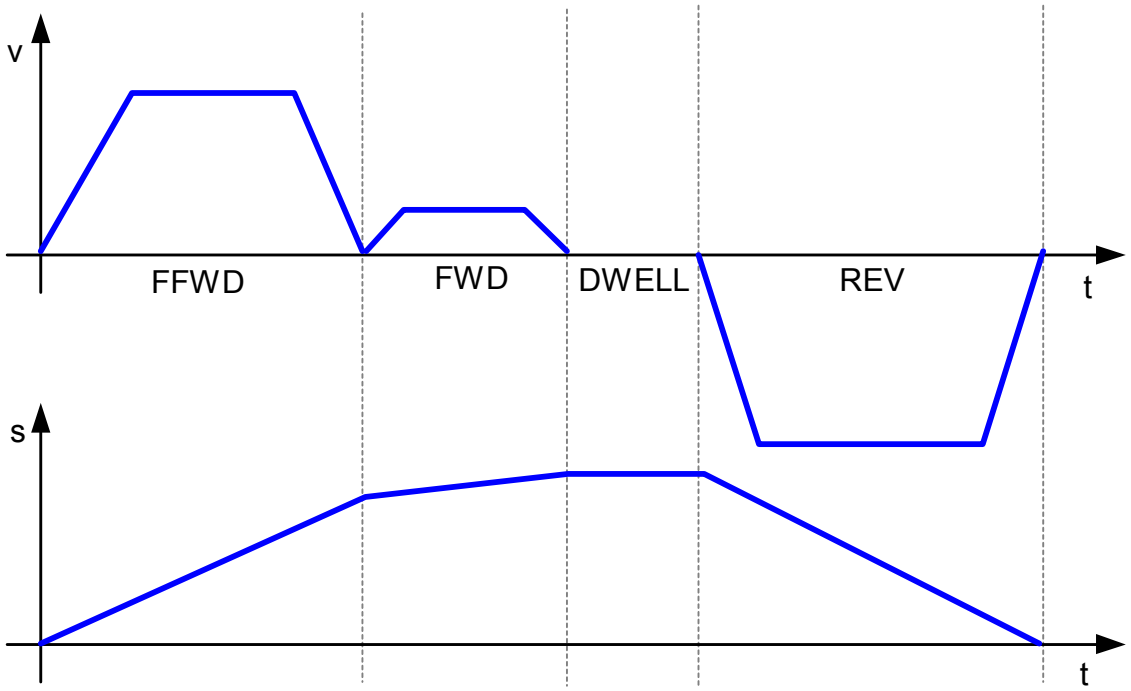


Figure 22: Timing diagram for drilling

5.1. Solution with Function Block diagram

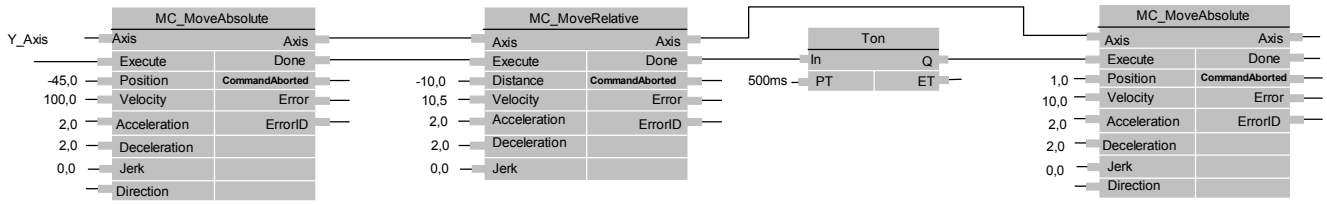


Figure 23: Solution with Function Block diagram

5.2. Sequential Function Chart

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

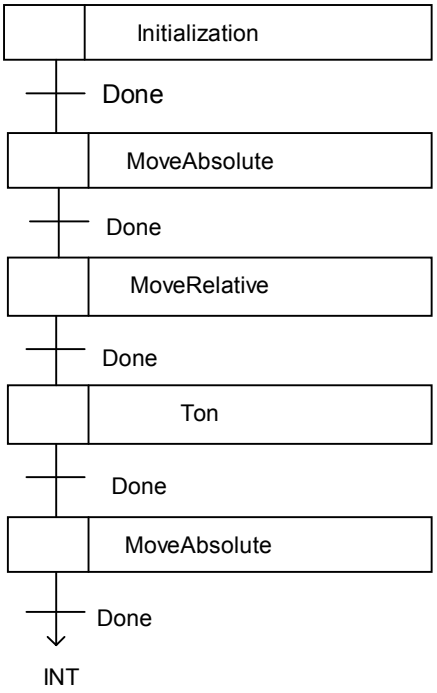


Figure 24: Straight forward step-transition chain for drilling example in SFC

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.

Appendix A. 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 datatypes (see Appendix A 2 Supported Datatypes) and supported Function Blocks, in combination with the applicable inputs and outputs (see Appendix A 3 Overview of the Function Blocks and its paragraphs). The supplier has to fill out the tables for the used datatypes 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 web-site, www.plcopen.org , as well as a shortform overview, as specified in Appendix A 2 Supported Datatypes and Appendix A 3 Overview of the Function Blocks here below.

In addition to this approval, the supplier gets access and usage rights of the PLCopen Motion Control logo, as described in chapter Appendix A 4 The PLCopen Motion Control Logo and Its Usage.

Datatypes

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 to extend 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, 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**"

Appendix A 1. Statement of Supplier

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

I herewith 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 A 2. Supported Datatypes

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

Table 4: Supported datatypes

Within the specification the following derived datatypes are defined. Which structure is used in this system:

Derived datatypes:	Where used	Supported	Which structure
Axis_Ref	Nearly all FBs		
MC_Direction (extended)	MC_MoveAbsolute MC_MoveVelocity		
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_StartMode (extended)	MC_CamIn		

Table 5: Supported derived datatypes

Appendix A 3. Overview of the Function Blocks

Single Axis Function Blocks	**Supported Yes / No**	**Comments (<= 48 char.)**
MC_MoveAbsolute		
MC_MoveRelative		
MC_MoveAdditive		
MC_MoveSuperimposed		
MC_MoveVelocity		
MC_Home		
MC_Stop		
MC_Power		
MC_ReadStatus		
MC_ReadAxisError		
MC_Reset		
MC_ReadParameter		
MC_ReadBoolParameter		
MC_WriteParameter		
MC_WriteBoolParameter		
MC_ReadActualPosition		
MC_PositionProfile		
MC_VelocityProfile		
MC_AccelerationProfile		
Multi-Axis Function Blocks	**Supported Yes / No**	**Comments (<= 48 char.)**
MC_CamTableSelect		
MC_CamIn		
MC_CamOut		
MC_GearIn		
MC_GearOut		
MC_Phasing		

Table 6: Short overview of the Function Blocks

Appendix A 6.1 MoveAbsolute

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

Appendix A 6.2 MoveRelative

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

Appendix A 6.3 MoveAdditive

If Supported	**MC_MoveAdditive**	Suported Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Execute		
B	Distance		
E	Velocity		
E	Acceleration		
E	Deceleration		
E	Jerk		
VAR_OUTPUT			
B	Done		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix A 6.4 MoveSuperimposed

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

Appendix A 6.5 MoveVelocity

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

Appendix A 6.6 Home

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

Appendix A 6.7 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		
B	Error		
E	ErrorID		

Appendix A 6.8 Power

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

Appendix A 6.9 ReadStatus

If Supported	**MC_ReadStatus**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
VAR_OUTPUT			
B	Done		
B	Error		
E	ErrorID		
B	Errorstop		
B	Stopping		
B	StandStill		
B	DiscreteMotion		
B	ContinuousMotion		
E	SynchronizedMotion		
E	Homing		
E	ConstantVelocity		
E	Accelerating		
E	Decelerating		

Appendix A 6.10 ReadAxisError

If Supported	**MC_ReadAxisError**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
	Enable		
VAR_OUTPUT			
B	Done		
B	Error		
B	ErrorID		

Appendix A 6.11 Reset

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

Appendix A 6.12 ReadParameter

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

Appendix A 6.13 ReadBoolParameter

If Supported	**MC_ReadBoolParameter**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
B	ParameterNumber		
VAR_OUTPUT			
B	Done		
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		
MaxJerk	E	R/W		

Table 7: Parameters for ReadParameter and WriteParameter

Appendix A 6.14 WriteParameter

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

Appendix A 6.15 WriteBoolParameter

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

Appendix A 6.16 ReadActualPosition

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

Appendix A 6.17 PositionProfile

If Supported	**MC_PositionProfile**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	TimePosition		
VAR_INPUT			
B	Execute		
B	ArraySize		
E	Scale		
E	Offset		
VAR_OUTPUT			
B	Done		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix A 6.18 VelocityProfile

If Supported	**MC_VelocityProfile**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	MC_TimeVelocity		
VAR_INPUT			
B	Execute		
B	ArraySize		
E	Scale		
E	Offset		
VAR_OUTPUT			
B	Done		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix A 6.19 AccelerationProfile

If Supported	**MC_AccelerationProfile**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Axis		
B	MC_TimeAcceleration		
VAR_INPUT			
B	Execute		
B	ArraySize		
E	Scale		
E	Offset		
VAR_OUTPUT			
B	Done		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix A 6.20 CamTableSelect

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

Appendix A 6.21 CamIn

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

Appendix A 6.22 CamOut

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

Appendix A 6.23 GearIn

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

Appendix A 6.24 GearOut

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

Appendix A 6.25 Phasing

If Supported	**MC_Phasing**	Sup. Y/N	Comments
VAR_IN_OUT			
B	Master		
B	Slave		
VAR_INPUT			
B	Execute		
B	Phase		
E	Acceleration		
E	Deceleration		
E	Jerk		
VAR_OUTPUT			
B	Done		
E	CommandAborted		
B	Error		
E	ErrorID		

Appendix A 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 25: 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 to 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 is done 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 statement on this from PLCopen in written form. 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 as long as the original scale and color setting is kept.
6. the logo has to be used in the context of Motion Control.