Manual_PtP_Package_Festo

Codesys, TwinCAT and Sysmac Studio library for Festo motor controllers in point-to-point mode

Date
14.02.2023

Festo SE & Co. KG
Ruiter Straße 82
73734 Esslingen

www.festo.com/contact

Contents

1 Important information 4
1. 1 Version overview 4
1. 2 Copyright Notice 5
1. 3 Legal Notice 5
1. 4 Intended use 5
1. 5 Safety instructions 5
1. 6 Target group 6
1. 7 Service 6

2 Introduction 7
2. 1 Installing the library package 7
2. 1. 1 Codesys extension in Automation Suite 7
2. 1. 2 Codesys 3.5 (stand-alone) 7
2. 1. 3 Beckhoff TwinCAT 3 8
2. 1. 4 Omron Sysmac Studio 10
2. 1. 5 Lenze PLC Designer 12
2. 2 Correct use of acyclic communication (SDO access) 13

3 PLCopen function blocks 14
3. 1 General 14
3. 2 PLCopen diagram 15
3. 3 MC_Power_Festo 16
3. 4 MC_Home_Festo 17
3. 5 MC_Stop_Festo 18
3. 6 MC_Halt_Festo 19
3. 7 MC_MoveAbsolute_Festo 20
3. 8 MC_MoveRelative_Festo 21
3. 9 MC_MoveAdditive_Festo 22
3. 10 MC_MoveVelocity_Festo 23
3. 11 MC_TorqueControl_Festo 24
3. 12 MC_ReadParameter_Festo 25
3. 13 MC_WriteParameter_Festo 26
3. 14 MC_ReadStringParameter_Festo 27
3. 15 MC_WriteStringParameter_Festo 28
3. 16 MC_ReadActualPosition_Festo 29
3. 17 MC_ReadActualVelocity_Festo 30
3. 18 MC_ReadActualTorque_Festo 31
3. 19 MC_ReadStatus_Festo 32
3. 20 MC_ReadAxisError_Festo 33
3. 21 MC_Reset_Festo 34
3. 22 MC_Jog_Festo 35
3. 23 MC_RecordTable_Festo 36
3. 24 MC_ReadAxisInfo_Festo 37
3. 25 MC_DeviceService_Festo 38
3. 26 MC_ForceControl_Festo 39

4 Control via method call 40

5 Diagnostics 42

6 BufferMode 47

7 Direction 48

8 Visualisation of function blocks 49

9 Application examples 50
9. 1 Example of function blocks 50
9. 2 Example of a method call 51

1 Important information

1. 1 Version overview

Author	Date	Comments
chmm	28.05.2018	Manual created
chmm	23.10.2018	Manual extended (MC_ReadAxisInfo_Festo, AxisIsMoving, AxisWarning)
chmm	28.11.2018	CMMT-ST added
chmm	17.12.2018	Property and Error-IDs added
chmm	22.01.2019	Function blocks extended by input “ContinuousUpdate” and output “Active”, examples
chmm	28.01.2019	New error numbers, TwinCAT picture changed
chmm	29.04.2019	Description of Diagnostics extended, note for “MC_Home_Festo” extended
chmm	06.06.2019	Description of FB “MC_ReadStatus_Festo” and ErrorID 2069 extended
chmm	09.09.2019	Bugfix in section 4 “Control via method call”
chmm	12.09.2019	Error ID 0x800 added
chmm	17.09.2019	Datatype of FB outputs “ErrorID” changed into “WORD” Number range of Error ID adapted (+1000)
chmm	30.10.2019	Fb input “Direchtion” added, section Diagnostics extended
chmm	20.11.2019	Section “Omron Sysmac Studio” added
chmm	22.01.2020	Init error extended, bugfix in MC_Power_Festo, new input „WcState_In“ for TC3
chmm	13.02.2020	Init error extended, Figure 8 updated
chmm	17.02.2020	Legal form of the company changed (cover sheet)
chmm	27.02.2020	UpdateTime to function block "MC_ReadAxisInfo_Festo" added
chmm	05.03.2020	Function block "MC_DeviceService_Festo" added
chmm	11.03.2020	Name of subsections in section 1 improved
chmm	30.06.2020	Section 2.2 added
chmm	02.09.2020	Stroke limitation added and error list (section 5) updated
chmm	04.05.2021	Description of section 7 (Direction) extended, Bugfix “MC_Jog_Festo”, section “Copyright Notice” and “Legal Notice” added, example for method call extended
chmm	28.06.2021	Section “Lenze PLC Designer” added
chmm	21.12.2022	Section “MC_ForceControl_Festo” added
chmm	14.02.2023	Bugfixes

1. 2 Copyright Notice

This documentation is the intellectual property of Festo SE & Co. KG, which also has the exclusive copyright. Any modification of the content, duplication or reprinting of this documentation as well as distribution to third parties can only be made with the express consent of Festo SE & Co. KG. Festo SE & Co KG reserves the right to make modifications to this document in whole or in part. All brand and product names are trademarks or registered trademarks of their respective owners.

1. 3 Legal Notice

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with components recommended by Festo SE & Co. KG. Festo SE & Co. KG does not accept any liability for damages arising from the use of any incorrect or incomplete information contained in this documentation or any information missing therefrom. Defects resulting from the improper handling of devices and modules are excluded from the warranty. The data and information specified in this document should not be used for the implementation of safety functions relating to the protection of personnel and machinery. No liability is accepted for claims for damages arising from a failure or functional defect. In other respects, the regulations with regard to liability from the terms and conditions of delivery, payment and use of software of Festo SE & Co. KG, which can be found at www.festo.com and can be supplied on request, shall apply. All data contained in this document do not represent guaranteed specifications, particularly with regard to functionality, condition or quality, in the legal sense. The information in this document serves only as basic information for the implementation of a specific, hypothetical application and is in no way intended as a substitute for the operating instructions of the respective manufacturers and the design and testing of the respective application by the user. The operating instructions for Festo products can be found at www.festo.com/sp . Users of this document (application note) must verify that all functions described here also work correctly in the application. By reading this document and adhering to the specifications contained therein, users are also solely responsible for their own application.

1. 4 Intended use

The function blocks based on PLCopen or the direct method calls allow the varied functions of the motor controllers from Festo to be conveniently integrated into the PLC program. The function blocks are called cyclically using a separate instance for each motor controller (each axis) integrated into the user program. Simultaneous use of other function blocks for controlling the same device is not permitted. The library described is used to control and parameterise the following motor controllers:

- Servo drives CMMT-AS
- Servo drives CMMT-ST

Observe the "Safety instructions" and instructions on the intended use of the respective devices, components and modules. When connected with commercially available components, such as sensors and actuators, the specified limits for pressures, temperatures, electrical data, torques, etc. must be observed.

1. 5 Safety instructions

When commissioning and programming positioning systems, you must observe the safety regulations in the manuals and operating instructions for the components used. The user must ensure that nobody has access to the sphere of influence of the connected actuators. Access to the potential danger area must be prevented by suitable measures such as barriers and warning signs.

1. 6 Target group

This description is intended exclusively for technicians trained in control and automation technology, who have experience in installing, commissioning, programming and diagnosing positioning systems and the relevant fieldbuses.

1. 7 Service

If you have a technical problem, please contact your local Festo service partner or use the respective contact form at the following website.

www.festo.com/contact

2 Introduction

2. 1 Installing the library package

2. 1. 1 Codesys extension in Automation Suite

Separate installation of the point-to-point package is not necessary since this is automatically updated when the CMMT plug-in is installed. All that is necessary during configuration is to select which operation mode and which version of the package should be used. These settings can be found in the CMMT plug-in under "Operation Mode" in the "Fieldbus" category on the "Parameterisation" tab (see Figure 1).

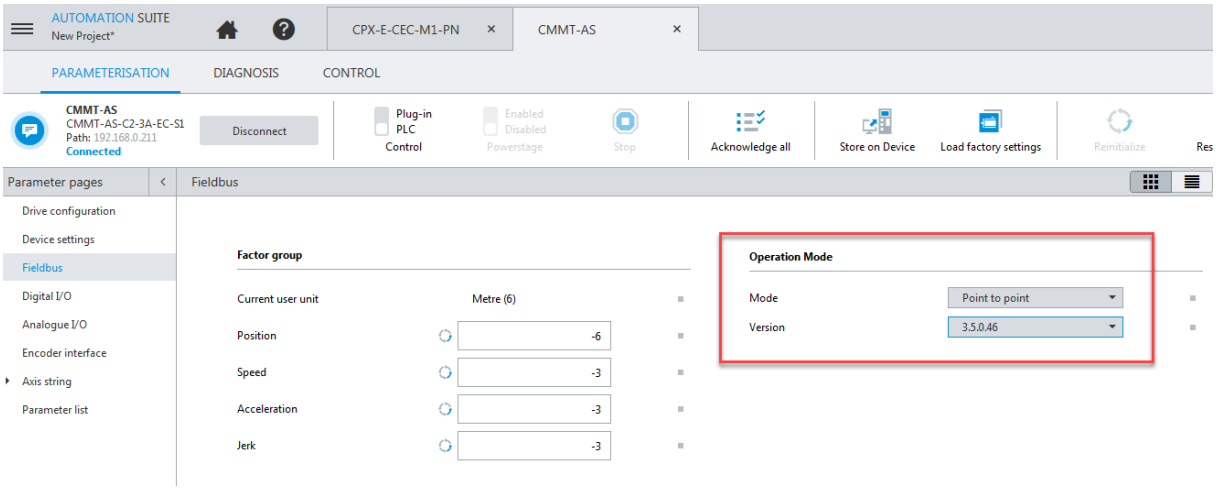


Figure 1: Automation Suite

2. 1. 2 Codesys 3.5 (stand-alone)

If Codesys provided by Festo (stand-alone) is used, the package must be installed by the Codesys Package Manager. Please contact the central service department to get the package. Double-clicking the package file starts the Package Manager. Once the installation has been successfully completed, Codesys must be restarted to initialise the data. The Festo motor controller can then be added to an EtherCAT® master (see Figure 2). The motor controller can then be used in the user program with immediate effect. Separate instancing or linking of the input and output data is not necessary. The device name is used as the axis reference (AXIS_REF_FESTO).

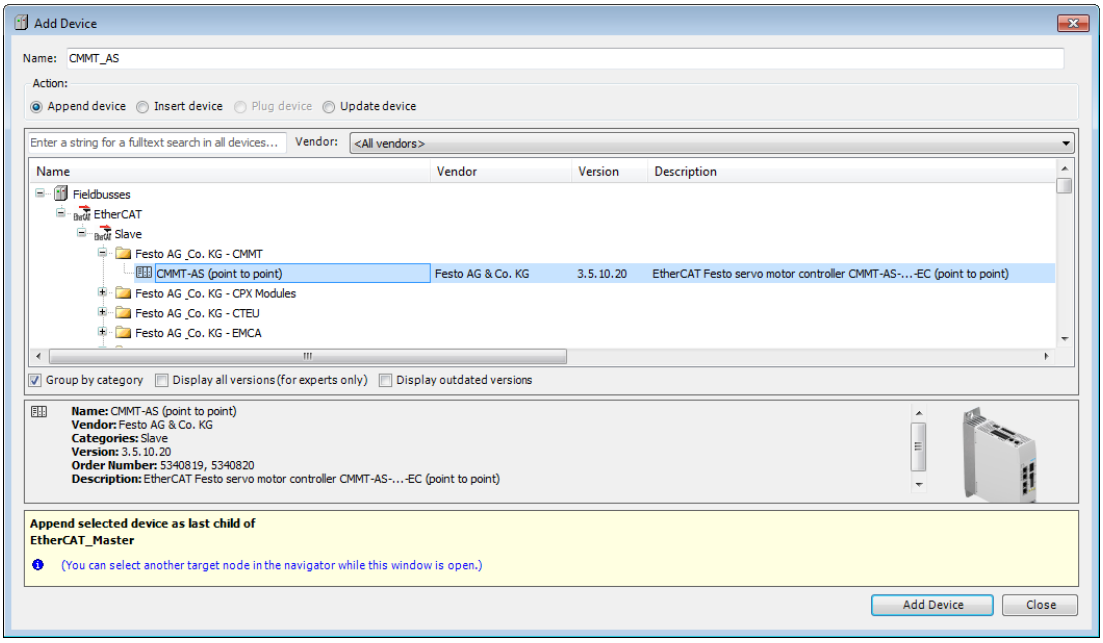


Figure 2: Codesys - adding a device

2. 1. 3 Beckhoff TwinCAT 3

The package described above cannot be used if the motor controller CMMT-AS from Festo is integrated into a TwinCAT 3 project from Beckhoff. In this case, the device description file and the libraries must be integrated manually. The EtherCAT® slave information (esi) must be copied into the TwinCAT installation path provided for this (see Figure 3).

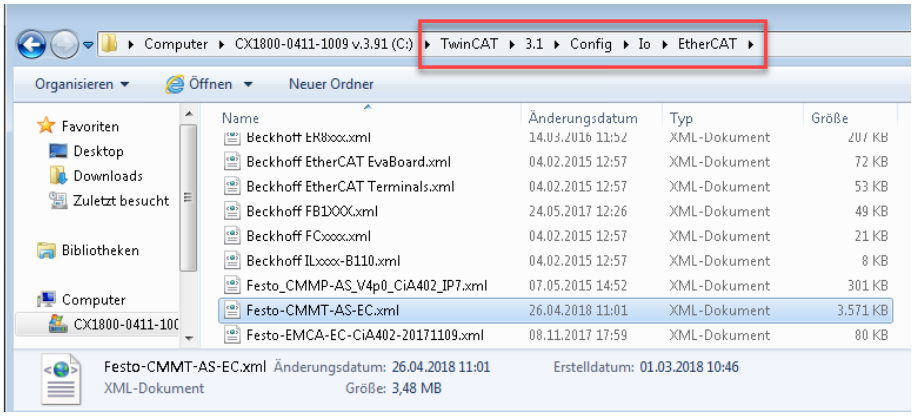


Figure 3: Esi directory in TwinCAT

Note
The specified libraries will not be supported by TwinCAT XAE engineering enviroments older than version 3.1.4020.0. (Reason: Compiler version is not compatible).

The programmer can stat TwinCAT once the device description file has been added. The libraries for the drive must then be installed in the Library Repository. If, for example, the motor controller CMMT-AS is being used, the user must install five libraries (see Figure 4).

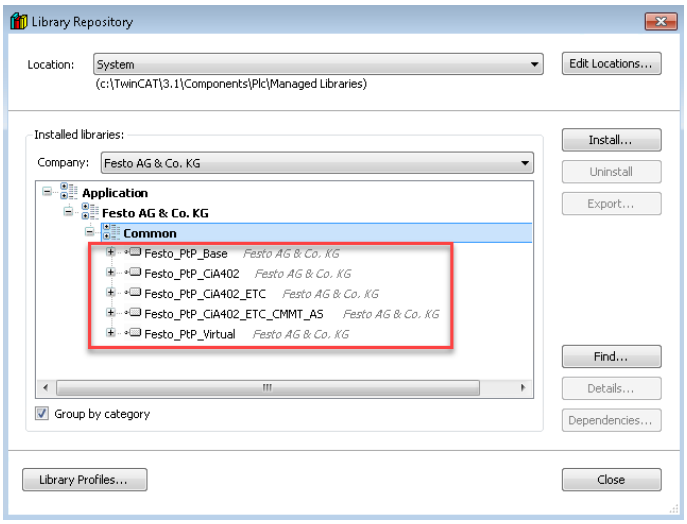


Figure 4: TwinCAT Library Repository

In the next step, the user must add the installed libraries to the project (see Figure 5 – purple frame). The drive library must then be instanced and called cyclically (Figure 5 – red frames). Once the user project has been successfully compiled, the library variables can be linked with the drive's project data objects (PDOs). This is done by linking the library variables with the motor controller variables of the same name (see Figure 5 – orange and green frames). The created block instance can then be used as an axis reference in the user project. On the one hand, it can be used as an axis variable of the type "AXIS_REF_FESTO" for the blocks described below. On the other hand, methods can be started directly from this instance. A more detailed description of the method call can be found in section 4.

Note

The instance variables “WcState_In”, “SlaveState_In” and “AdsAddr_In” must be linked to the drive . Only this linking enables SDO communication (Service-Data-Object communication) between PLC and CMMT-AS or CMMT-ST.

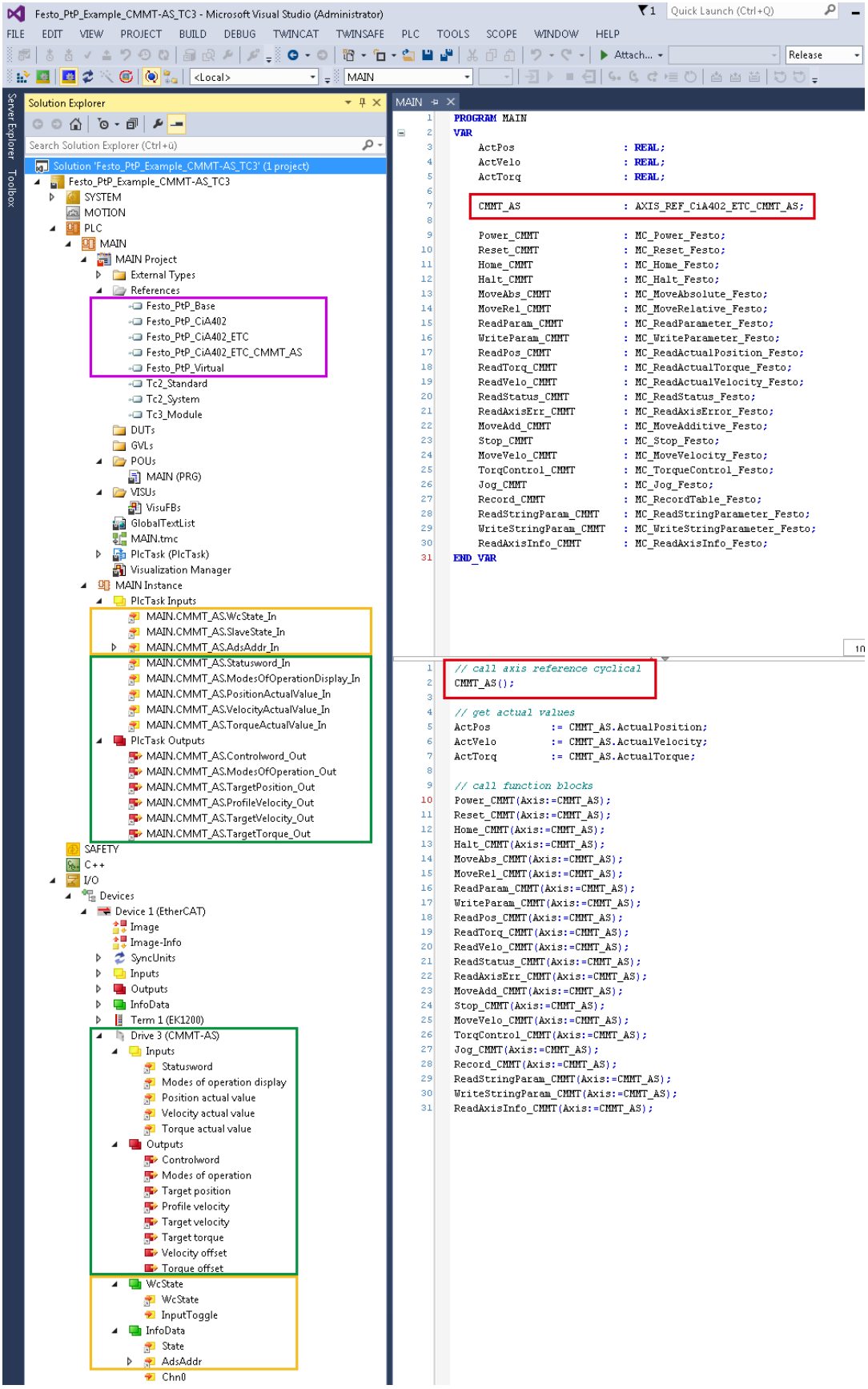


Figure 5: Device integration in TwinCAT

2. 1. 4 **Omron Sysmac Studio**

In order to be able to use the PtP library in the Sysmac Studio from Omron, it's advisable to copy it into the default folder provided for this purpose (C:\OMRON\Data\Lib). Afterwards, the library can be integrated into the project under "Project – Library – Show references".

After successful configuration of the network, the servo drive must be linked to the library's axis structure (PTP_AXIS_REF_FESTO). The axis structure and the logic variables for the drive's input and output data (PDOs) must be instantiated to this end. This can be accomplished, for example, in the global variables list (see Figure 6: Global variables list Sysmac Studio).

Note
A separate type "PTP_AXIS_REF_FESTO" axis structure must be instantiated in the PLC program for each servo drive.

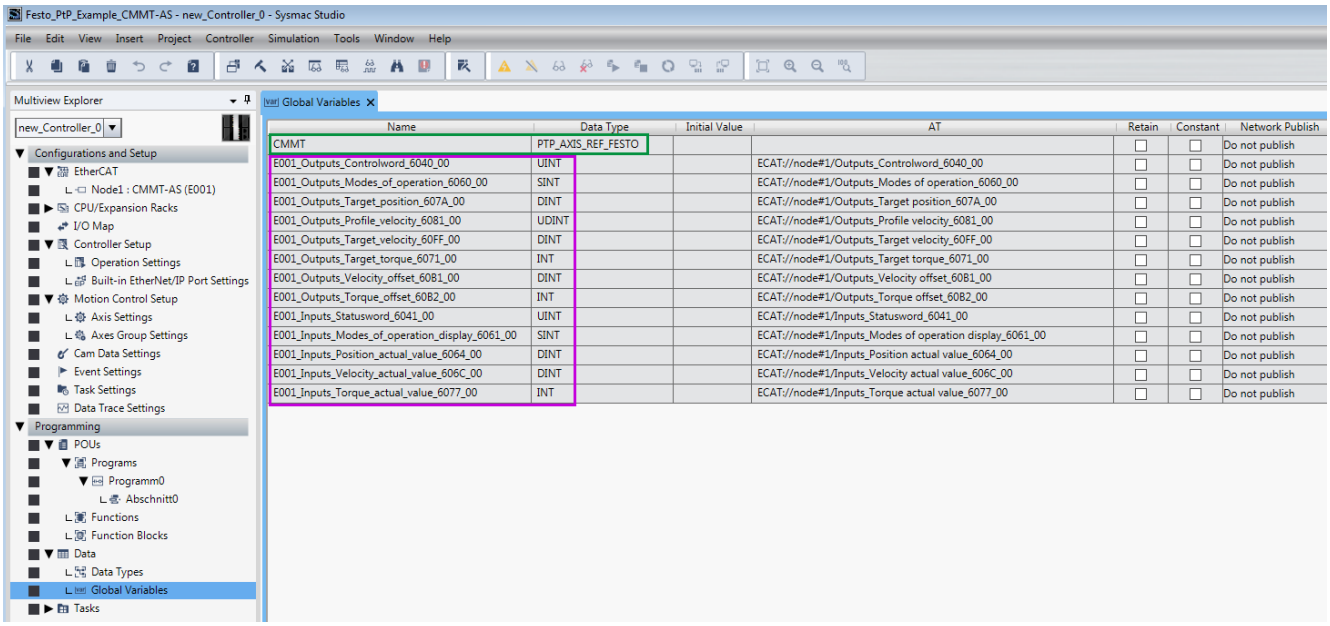


Figure 6: Global variables list Sysmac Studio

After successful instantiation, the motor controller's cyclical process data must be linked to the available variables (PDO mapping) (see Figure 7: PDO mapping Sysmac Studio).

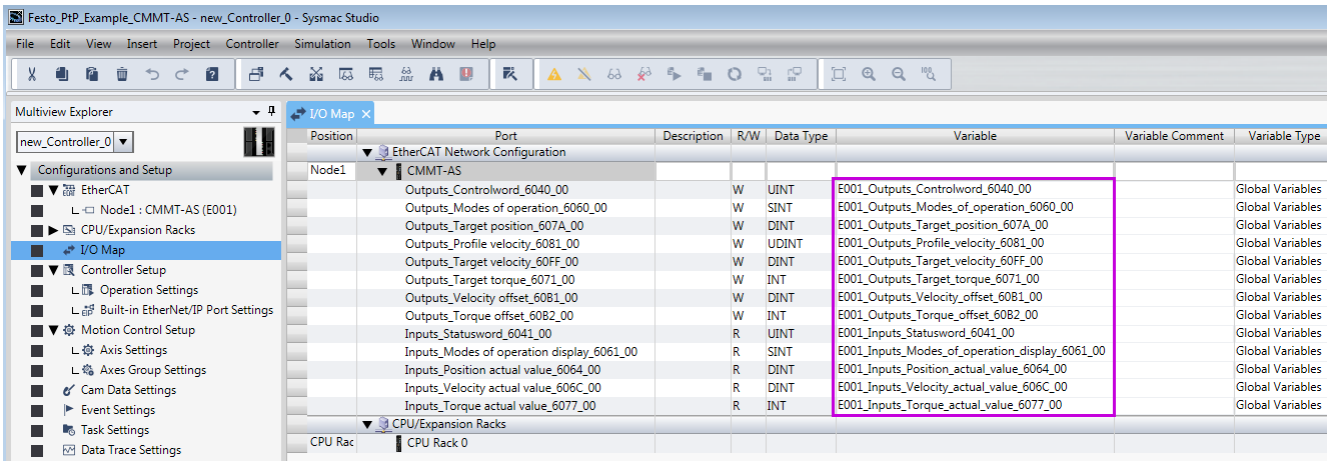


Figure 7: PDO mapping Sysmac Studio

The axis module must then be instantiated in the PLC program and cyclically invoked (red frames). The previously set up axis structure (green frames) and the logic variables (purple frames) of the drive's inputs and outputs serve as input and output variables, (see **Error! Reference source not found.**). The input variable „EC_PDActive" of the axis m odule (PTP_CIA402_CMMT_AS) must be active for processing. It can be assigned, for example, with the "_EC_PDActive" system variable (yellow frame). In order to ensure mandatory SDO communication (service data object communication) between the PLC and the drive, the "SlaveNodeInfo" must also be transferred to the axis module (blue arrow).

Note
The type "PTP_CIA402_CMMT_FESTO" axis module must always be cyclically invoked.

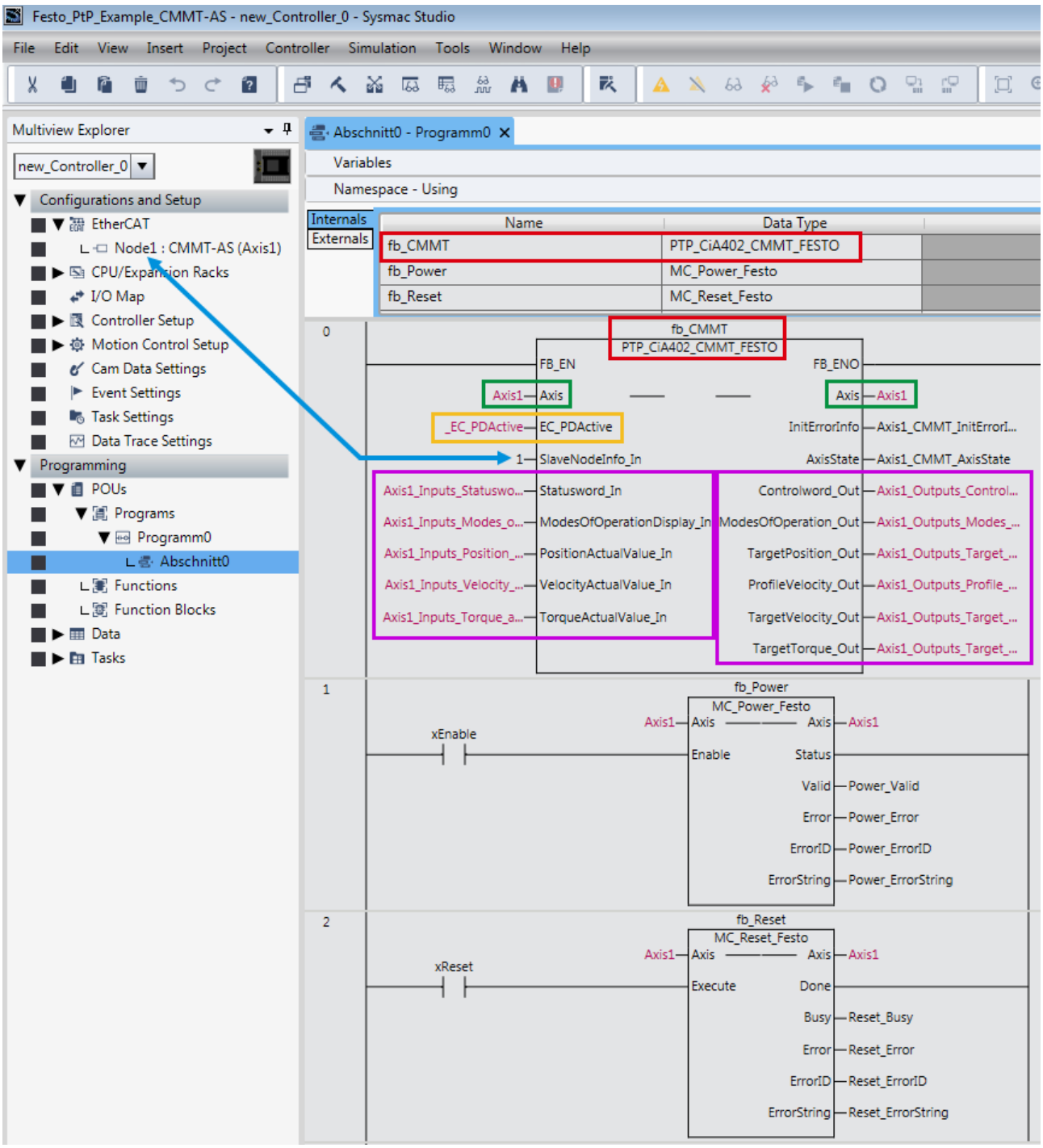


Figure 8: Cyclical invocation of the axis module

2. 1. 5 Lenze PLC Designer

First, the respective package of the motor controller for Lenze PLCs must be installed under "Tools" with the "CODESYS Package Manager". Then the libraries and the drive must be added to the PLC Designer project. (see Figure 9: Adding libraries and drive in Lenze PLC Designer).

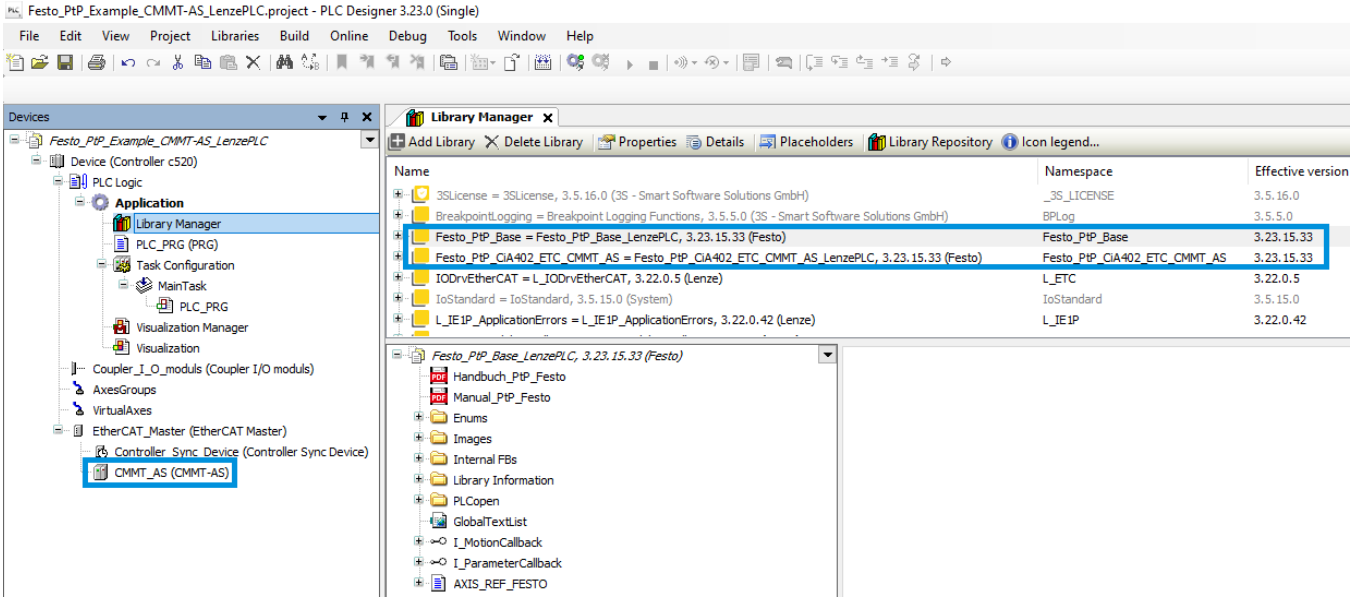


Figure 9: Adding libraries and drive in Lenze PLC Designer

In the next step, the axis driver function block (AXIS_REF_CiA402_ETC_CMMT_xx) can be declared and called cyclically. Here the instance of the EtherCAT master and the EtherCAT slave address of the motor controller must be specified. (see Figure 10: Cyclic call of the axis driver for Lenze PLCs).

Hinweis

Der Achsbaustein vom Typ „AXIS_REF_CiA402_ETC_CMMT_xx“ muss immer zyklisch aufgerufen werden.

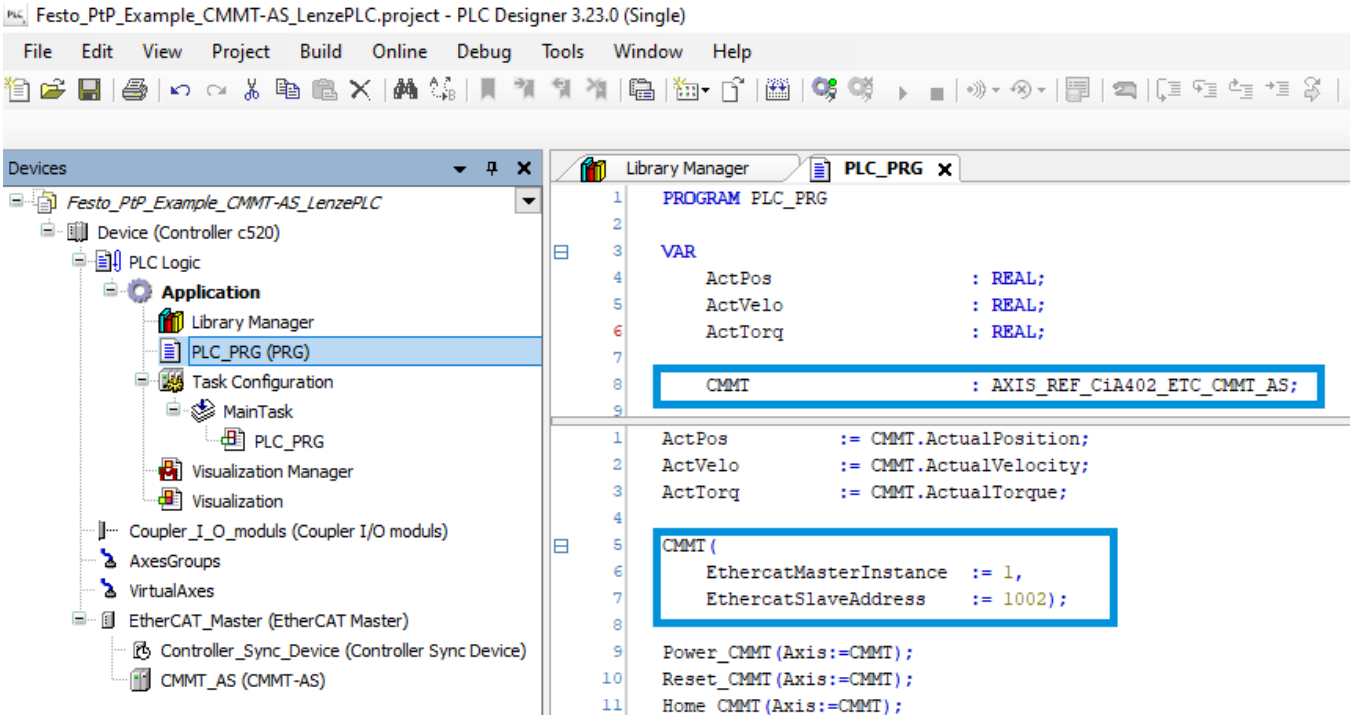


Figure 10: Cyclic call of the axis driver for Lenze PLCs

Finally, the process data objects (PDOs) must be linked to the axis driver function block. (see Figure 11: Linking PDOs with axis driver structure in Lenze PLC Designer**Error! Reference source not found.**).

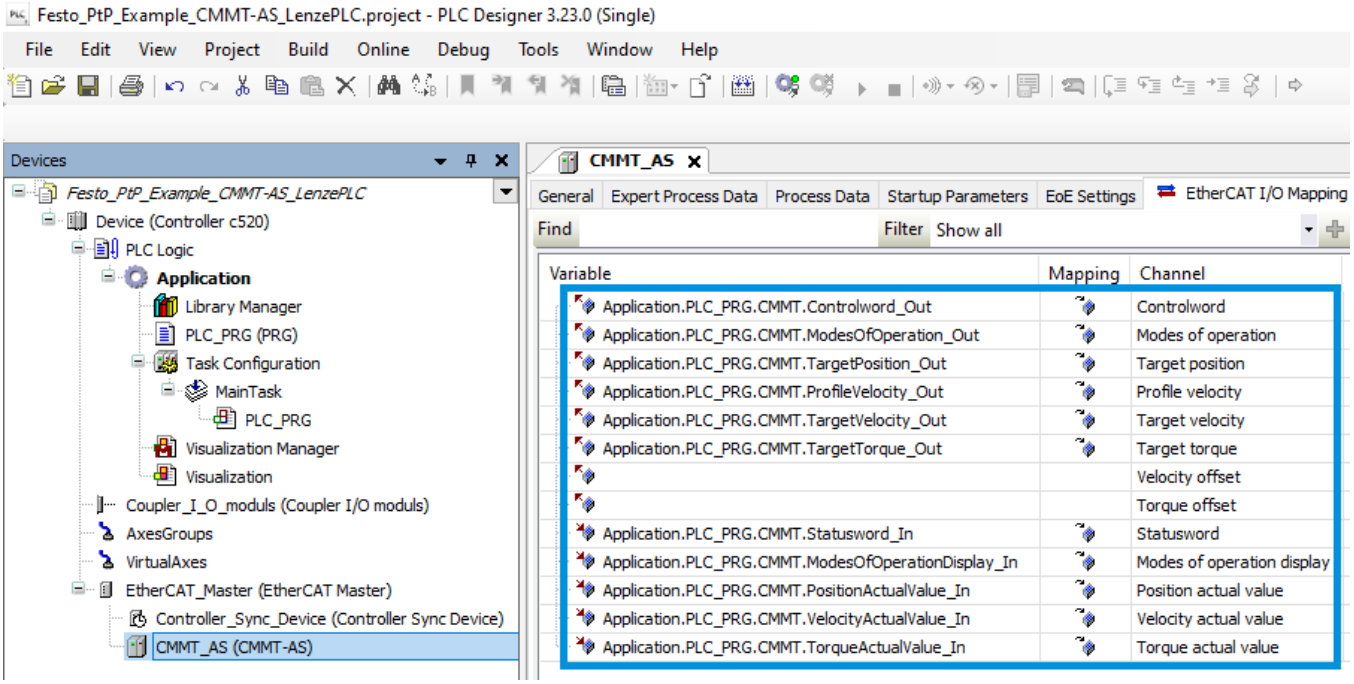


Figure 11: Linking PDOs with axis driver structure in Lenze PLC Designer

2. 2 Correct use of acyclic communication (SDO access)

Via the service data object (SDO) write or read command, the PLC can access the object directory of the drive acyclically. As a result, the data of a parameter are written or read. After the SDO command has been processed, the drive sends an acknowledgement to the PLC. This communication generates acyclic bus load. Depending on the number of drives and the number of SDO commands sent at the same time, the PLC can reach the limit of the maximum number of SDO communication commands permitted at the same time. It is therefore recommended to keep the number of acyclic read and write accesses to a minimum.

The function blocks of the PtP library from Festo also implicitly generate SDO commands if input data at the function block change compared to the previous call (e.g. "Acceleration", "Decceleration", "Jerk", "Direction", "BufferMode", "HomingMethod", etc.). If the values remain unchanged from the previous call, the respective block generates one SDO write command per input parameter only with the very first call. This first write command can also be prevented if the input "Acceleration", "Deceleration", "Jerk", "Direction" or "BufferMode" is = 0 at the time of the start of movement. In this case, the value set in the motor controller is used for this parameter. No acyclic write command is generated.

In addition, the PtP library generates several SDO commands per drive during the initialization phase in order to read out set parameters such as the factor group. These acyclic read commands cannot be deactivated.

Note

Omron controllers can execute a maximum of 32 instructions simultaneously. If the number of 32 is exceeded, the error "Communication Resource Overflow" is displayed. In this case it is recommended to implement the above instructions to minimize acyclic instructions.

The following Omron blocks generate an instruction: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj, and IOL_WriteObj.

3 PLCopen function blocks

3. 1 General

The blocks described here were developed using the technical PLCopen specification "Function blocks for motion control - Version 2.0". Most of the single-axis blocks have been implemented. In order to extend the function range for motor controllers from Festo, further blocks have been additionally implemented with the same design and the same behaviour. Further information such as time charts can be found in the PLCopen specification. This library contains the following elements:

- MC_Power_Festo
- MC_Home_Festo
- MC_Stop_Festo
- MC_Halt_Festo
- MC_MoveAbsolute_Festo
- MC_MoveRelative_Festo
- MC_MoveAdditive_Festo
- MC_MoveVelocity_Festo
- MC_TorqueControl_Festo
- MC_ReadParameter_Festo
- MC_WriteParameter_Festo
- MC_ReadStringParameter_Festo
- MC_WriteStringParameter_Festo
- MC_ReadActualPosition_Festo
- MC_ReadActualVelocity_Festo
- MC_ReadActualTorque_Festo
- MC_ReadStatus_Festo
- MC_ReadAxisError_Festo
- MC_Reset_Festo
- MC_Jog_Festo
- MC_RecordTable_Festo
- MC_ReadAxisInfo_Festo
- MC_DeviceService_Festo
- MC_ForceControl_Festo

Note

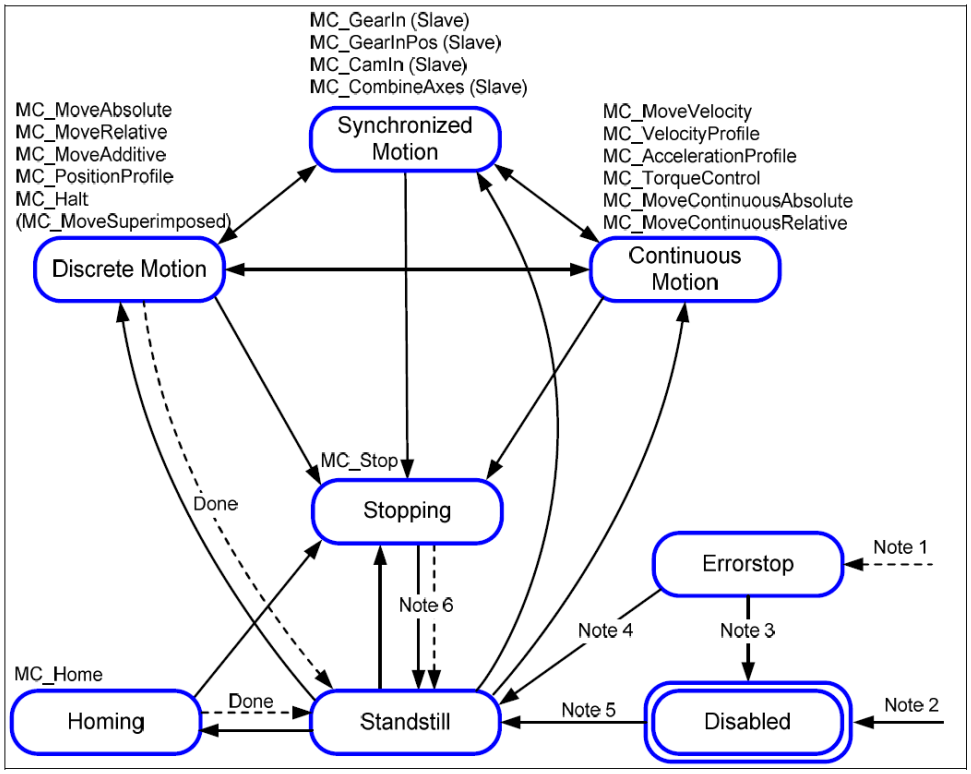
Function block inputs have to set in user units partly (e.g. position, distance, velocity und dynamic values). This is always the user unit parameterized in the drive in the "Fieldbus" tab (e.g. rounds, degree, inch, etc.).

Peculiarity:

If the user unit "Metric" was set in the drive, the function block inputs must be specified in mm, mm/s, mm/s² oder mm/s³. The function block outputs, for example the current position, are also specified in mm in this case.

3. 2 PLCopen diagram

The diagram below shows the individual states of an axis and the transitions possible in each case. The current status of the axis can be queried using the variable "DeviceName.AxisState". A detailed description of the individual states is not provided here since can this also be found in the PLCopen specification.



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 12: PLCopen diagram (source: Technical Specification PLCopen - FB for motion control)

3. 3 MC_Power_Festo

This function block controls the power output stage (enable or block).

Figure 13: MC_Power_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Enable drive • TRUE = Enable power output stage • FALSE = Block power output stage
VAR_OUTPUT		
Status	BOOL	Status of the output stage • TRUE = Enabled • FALSE = Blocked
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 4 MC_Home_Festo

This function block starts homing. If the reference signal is detected, the value of the "Position" input is set as the absolute position. The homing method can be selected via the "HomingMethod" input.

Figure 14: MC_Home_Festo

Note
If the "HomingMethod" input = 0, the homing method and axis zero point parameterised in the device are used. The axis zero point specified in the block ("Position" input) is ignored.
If the "HomingMethod" input ‹› 0, the homing method and axis zero point parameterised in the device are overwritten.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start homing • FALSE --› True = Start homing
Position	REAL	Absolute position if reference switch was detected (axis zero point in user unit).
HomingMethod	DINT	Homing method to CiA402 specification. The homing methods supported by the device can be found in the manual for the device.
VAR_OUTPUT		
Done	BOOL	Homing successfully completed
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Homing was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 5 MC_Stop_Festo

This function block stops an axis with the stop ramp parameterised in the Automation Suite and sets the axis to the status "Stopping". All other motion commands are aborted. No other block can move this axis as long as it has the status "Stopping". The output "Done" is set once the current axis velocity is zero. The axis remains in the status "Stopping" as long as "Execute" is active (TRUE) or the current velocity is not equal to zero. The axis status changes to "Standstill" as soon as "Done" has been set and "Execute" is inactive (FALSE).

Figure 15: MC_Stop_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Stop axis motion • FALSE --> True = Stop axis motion
VAR_OUTPUT		
Done	BOOL	Current axis velocity is zero (axis stopped)
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
CommandAborted	BOOL	Job was aborted due to power output stage being switched off (only abort option)
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 6 MC_Halt_Festo

This function block pauses an axis with the set braking ramp for the current motion and sets it to the status "DiscreteMotion". The output "Done" is set and the axis assumes the status "Standstill" once the current axis velocity is zero.

Figure 16: MC_Halt_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Pause axis motion • FALSE --> True = Pause axis motion
VAR_OUTPUT		
Done	BOOL	Current axis velocity is zero (axis stopped)
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Pausing the axis was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 7 MC_MoveAbsolute_Festo

This function block starts a motion to an absolute position on the basis of the set dynamic parameters.

Figure 17: MC_MoveAbsolute_Festo

Note
If one of the inputs "Acceleration", "Deceleration" or "Jerk" = 0 when the motion starts, the dynamic value set in the motor controller is used for this motion parameter.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start positioning task • FALSE --> True = Start positioning task
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
Position	REAL	Absolute target position (in user unit)
Velocity	REAL	Maximum velocity (in user unit)
Acceleration	REAL	Acceleration (in user unit)
Deceleration	REAL	Deceleration (in user unit)
Jerk	REAL	Jerk (in user unit)
Direction	MC_DIRECTION	Defines the positioning direction if "modulo" is activated (see section 7 – Direction)
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Positioning task successfully completed
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 8 MC_MoveRelative_Festo

This function block starts a motion by a distance relative to the axis set position at the time of block enable on the basis of the set dynamic parameters.

Figure 18: MC_MoveRelative_Festo

Note
If one of the inputs "Acceleration", "Deceleration" or "Jerk" = 0 when the motion starts, the dynamic value set in the motor controller is used for this motion parameter.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start positioning task • FALSE --> True = Start positioning task
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
Distance	REAL	Relative distance of the motion (in user unit)
Velocity	REAL	Maximum velocity (in user unit)
Acceleration	REAL	Acceleration (in user unit)
Deceleration	REAL	Deceleration (in user unit)
Jerk	REAL	Jerk (in user unit)
Direction	MC_DIRECTION	Defines the positioning direction if "modulo" is activated (see section 7 – Direction)
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Positioning task successfully completed
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 9 MC_MoveAdditive_Festo

This function block starts a motion by a distance relative to the currently planned target position in the axis status "DiscreteMotion" on the basis of the set dynamic parameters. This target position can be a result of a previous block of the same type which has been aborted. If the function block is started when the axis status is "ContinuousMotion", it executes a motion by a distance relative to the axis position at the time of block enable.

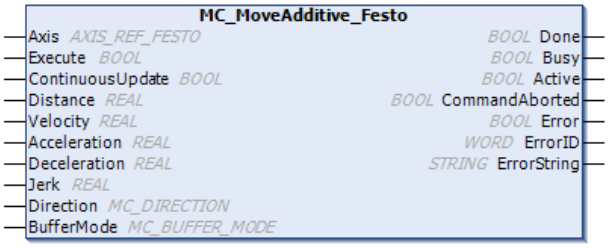


Figure 19: MC_MoveAdditive_Festo

Note
If one of the inputs "Acceleration", "Deceleration" or "Jerk" = 0 when the motion starts, the dynamic value set in the motor controller is used for this motion parameter.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start positioning task • FALSE --> True = Start positioning task
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
Distance	REAL	Relative distance to the target position (in user unit)
Velocity	REAL	Maximum velocity (in user unit)
Acceleration	REAL	Acceleration (in user unit)
Deceleration	REAL	Deceleration (in user unit)
Jerk	REAL	Jerk (in user unit)
Direction	MC_DIRECTION	Defines the positioning direction if "modulo" is activated (see section 7 – Direction)
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Positioning task successfully completed
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 10 MC_MoveVelocity_Festo

This function block starts a controlled motion with a defined velocity. This is velocity mode (ProfileVelocityMode).

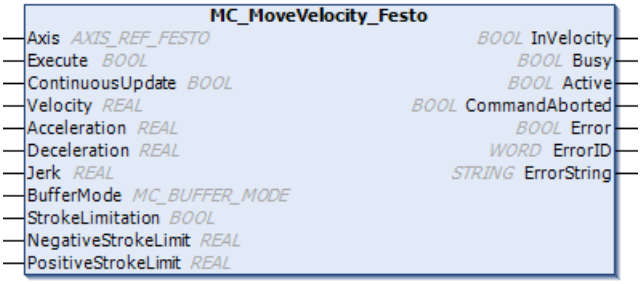


Figure 20: MC_MoveVelocity_Festo

Note
If one of the inputs "Acceleration", "Deceleration", "Jerk", “NegativeStrokeLimit“ or “PositiveStrokeLimit“ = 0 , when the motion starts, the dynamic value or limit value set in the motor controller is used for this motion parameter.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start velocity mode • FALSE --› True = Start velocity mode (with set dynamic parameters)
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
Velocity	REAL	Maximum velocity (in user unit)
Acceleration	REAL	Acceleration (in user unit)
Deceleration	REAL	Deceleration (in user unit)
Jerk	REAL	Jerk (in user unit)
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
StrokeLimitation	BOOL	Activation of stroke limitation • TRUE = Stroke limitation is active for this motion job
NegativeStrokeLimit	REAL	Negative stroke limitation (in user unit)
PositiveStrokeLimit	REAL	Positive stroke limitation (in user unit)
VAR_OUTPUT		
InVelocity	BOOL	Target velocity reached
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 11 MC_TorqueControl_Festo

This function block starts a motion with a continuous torque measured at the gear shaft. The specified velocity is not exceeded. This is force mode (ProfileTorqueMode).

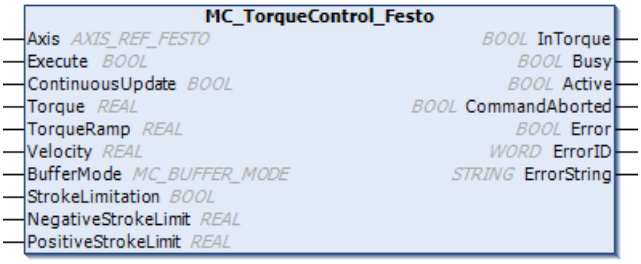


Figure 21: MC_TorqueControl_Festo

Note
If one of the inputs “NegativeStrokeLimit“ or “PositiveStrokeLimit“ = 0 , when the motion starts, the limit value set in the motor controller is used for this motion parameter.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start force mode • FALSE --> True = Start force mode
Torque	REAL	Setpoint torque at the gear shaft (in Nm)
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
TorqueRamp	REAL	Torque ramp (in Nm/s)
Velocity	REAL	Maximum velocity (in user unit)
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
StrokeLimitation	BOOL	Activation of stroke limitation • TRUE = Stroke limitation is active for this motion job
NegativeStrokeLimit	REAL	Negative stroke limitation (in user unit)
PositiveStrokeLimit	REAL	Positive stroke limitation (in user unit)
VAR_OUTPUT		
InTorque	BOOL	Target torque reached
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 12 MC_ReadParameter_Festo

This function block reads out the value of a parameter via service data object access (SDO access).

Figure 22: MC_ReadParameter_Festo

Note
This function block causes a steady SDO-communication between drive and PLC. Permanent reading leads to a heavy load of the fieldbus. Please use this function block situational.

Note
This block is not suitable for reading parameters of the data type "STRING". The block "MC_ReadStringParameter_Festo" must be used for this.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read SDO value • TRUE = Read SDO continuously as long as TRUE • FALSE = Disable reading
ParameterNumber	UINT	Number of the parameter to be read
SubindexNumber	USINT	Subindex of the parameter to be read
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Value	LREAL	Value of the read parameter
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 13 MC_WriteParameter_Festo

This function block modifies the value of a parameter via service data object access (SDO access).

Figure 23: MC_WriteParameter_Festo

Note
This block is not suitable for writing parameters of the data type "STRING". The block "MC_WriteStringParameter_Festo" must be used for this.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Modify SDO value • FALSE --> TRUE = Write new SDO value
ParameterNumber	UINT	Number of the parameter to be written
SubindexNumber	USINT	Subindex of the parameter to be written
Value	LREAL	New value of the parameter to be written
VAR_OUTPUT		
Done	BOOL	Parameter successfully written
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 14 MC_ReadStringParameter_Festo

This function block reads out the value of a service data object (SDO) of the data type "STRING". A character string with a maximum length of 35 characters is read out.

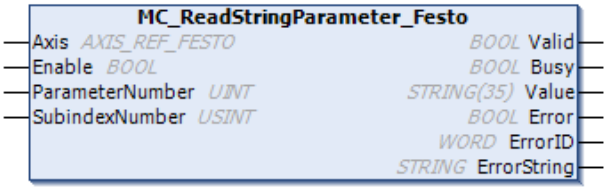


Figure 24: MC_ReadStringParameter_Festo

Note
This function block causes a steady SDO-communication between drive and PLC. Permanent reading leads to a heavy load of the fieldbus. Please use this function block situational.

Note
This block is only suitable for reading parameters of the data type "STRING". The block "MC_ReadParameter_Festo" must be used for parameters with other data types.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read SDO value • TRUE = Read SDO continuously as long as TRUE • FALSE = Disable reading
ParameterNumber	UINT	Number of the parameter to be read
SubindexNumber	USINT	Subindex of the parameter to be read
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Value	STRING(35)	Character string of the read parameter
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 15 MC_WriteStringParameter_Festo

This function block modifies the character string of a service data object (SDO) of the data type "STRING". A character string with a maximum length of 35 characters is written.

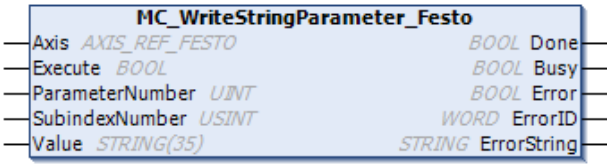


Figure 25: MC_WriteStringParameter_Festo

Note
This block is only suitable for writing parameters of the data type "STRING". The block "MC_WriteParameter_Festo" must be used for parameters with other data types.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Modify SDO value • FALSE --> TRUE = Write new SDO value
ParameterNumber	UINT	Number of the parameter to be written
SubindexNumber	USINT	Subindex of the parameter to be written
Value	STRING(35)	New character string of the parameter to be written
VAR_OUTPUT		
Done	BOOL	Parameter successfully written
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 16 MC_ReadActualPosition_Festo

This function block reads the current axis position.

Figure 26: MC_ReadActualPosition_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read current axis position (ActPos) • TRUE = Read ActPos continuously as long as TRUE • FALSE = Disable reading
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Position	REAL	Current axis position (in user unit)
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 17 MC_ReadActualVelocity_Festo

This function block reads the current axis velocity.

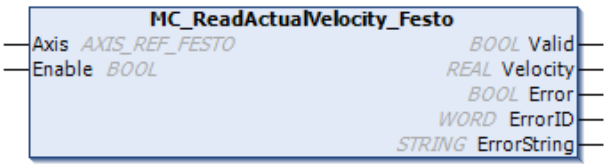


Figure 27: MC_ReadActualVelocity_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read current axis velocity (ActVelo) • TRUE = Read ActVelo continuously as long as TRUE • FALSE = Disable reading
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Velocity	REAL	Current axis velocity (in user unit)
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 18 MC_ReadActualTorque_Festo

This function block reads the current axis torque at gear shaft.

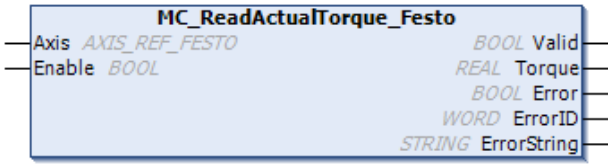


Figure 28: MC_ReadActualTorque_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read current axis torque (ActTorq) • TRUE = Read ActTorq continuously as long as TRUE • FALSE = Disable reading
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Torque	REAL	Read current axis torque (in user unit)
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 19 MC_ReadStatus_Festo

This function block reads the current status of the motor controller according to the "PLCopen state diagram" (see Section 3. 2).

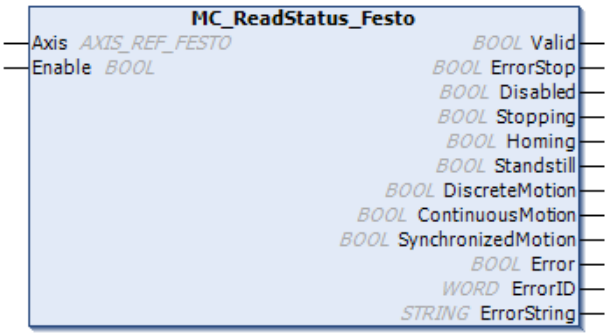


Figure 29: MC_ReadStatus_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read current axis status • TRUE = Read status continuously as long as TRUE • FALSE = Disable reading
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
ErrorStop	BOOL	Drive is in status “ErrorStop” (see the section 3. 2 – PLCopen diagram)
Disabled	BOOL	Drive is in status “Disabled” (see the section 3. 2 – PLCopen diagram)
Stopping	BOOL	Drive is in status “Stopping” (see the section 3. 2 – PLCopen diagram)
Homing	BOOL	Drive is in status “Homing” (see the section 3. 2 – PLCopen diagram)
Standstill	BOOL	Drive is in status “Standstill” (see the section 3. 2 – PLCopen diagram)
DiscreteMotion	BOOL	Drive is in status “DiscreteMotion” (see the section 3. 2 – PLCopen diagram)
ContinuousMotion	BOOL	Drive is in status “ContinuousMotion” (see the section 3. 2 – PLCopen diagram)
SynchronizedMotion	BOOL	Drive is in status “SynchronizedMotion” (see the section 3. 2 – PLCopen diagram))
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 20 MC_ReadAxisError_Festo

This function block continuously reads the current motor controller error.

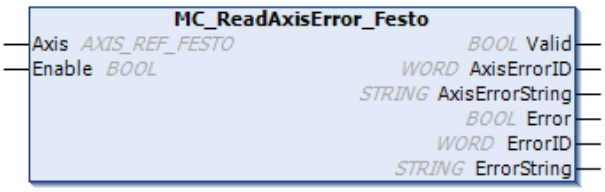


Figure 30: MC_ReadAxisError_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read current axis error • TRUE = Read error continuously as long as TRUE • FALSE = Disable reading
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
AxisErrorID	WORD	Current axis error number
AxisErrorString	STRING	Current axis error description
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 21 MC_Reset_Festo

This function block triggers a reset of the motor controller errors using the control word from the drive profile. In addition, the errors of the EtherCAT® slave in the PLC program are reset by the "ClearEmergency" method.

Figure 31: MC_Reset_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Reset errors • FALSE --> TRUE = Reset axis errors
VAR_OUTPUT		
Done	BOOL	Errors successfully reset (no axis errors)
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 22 MC_Jog_Festo

This function block starts a motor controller motion in ProfileJogMode.

Figure 32: MC_Jog_Festo

Note
The axis remains stationary if the "JogPositive" and "JogNegative" inputs have the same status (both TRUE or both FALSE).

Note
The "JogMode" input has three permissible input values (modes). If the input value 0 is specified, the drive first jogs at velocity 1. After a parameterised time, the drive accelerates/decelerates to velocity 2. The respective velocity and the time can be set in the motor controller (further information can be found in the manual for the device).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
JogPositive	BOOL	Jog in positive direction • TRUE = Axis jogs in positive direction
JogNegative	BOOL	Jog in negative direction • TRUE = Axis jogs in negative direction
JogMode	MC_JOG_MODE	Jog mode • 0 (mcDefault) = Velocity 1 and 2 • 1 (mcOnlyVelocity1) = Only velocity 1 • 2 (mcOnlyVelocity2) = Only velocity 2
VAR_OUTPUT		
Velocity1	BOOL	Drive jogs at velocity 1
Velocity2	BOOL	Drive jogs at velocity 2
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 23 MC_RecordTable_Festo

This function block starts a position record parameterised in the motor controller. This is RecordTableMode.

Figure 33: MC_RecordTableMode_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start position record • FALSE --> True = Start position record
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
RecordNumber	DINT	Number of the position record to be started
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Position record successfully executed
ActualRecordNumber	DINT	Active position record number
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 24 MC_ReadAxisInfo_Festo

This function block reads information concerning the axis, like home switch and end switches.

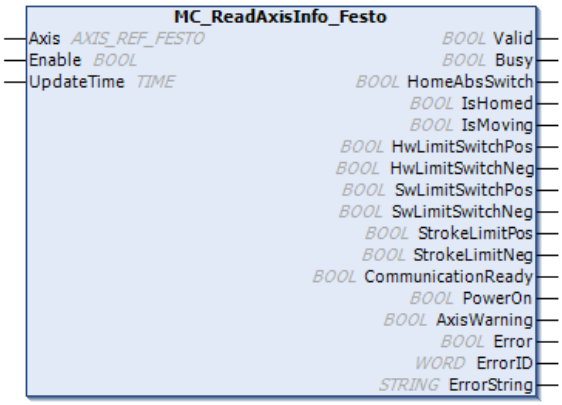


Figure 34: MC_ReadAxisInfo_Festo

Note
This function block causes a steady SDO-communication between drive and PLC. Permanent reading leads to a heavy load of the fieldbus. Please use this function block situational. A lot of information also available via the „Properties“ (see section 4 – Control via method call). This aren´t load the fieldbus additionally.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Enable	BOOL	Read axis information continuously • TRUE = Read axis info continuously as long as TRUE • FALSE = Disable reading
UpdateTime	TIME	Delay time between request of two SDO read commands (default = T#1S = 1 second)
VAR_OUTPUT		
Valid	BOOL	Status of the block outputs • TRUE = Output status valid • FALSE = Output status invalid
Busy	BOOL	Job is not yet complete • TRUE = Job active • FALSE = Job completed or not started
HomeAbsSwitch	BOOL	Digital home switch is active
IsHomed	BOOL	Absolute reference position is knwon for the axis (axis is homed)
IsMoving	BOOL	Achse bewegt sich
HwLimitSwitchPos	BOOL	Positive hardware end switch is active
HwLimitSwitchNeg	BOOL	Negative hardware end switch is active
SwLimitSwitchPos	BOOL	Positive software end switch is active
SwLimitSwitchNeg	BOOL	Negative software end switch is active
StrokeLimitPos	BOOL	Positve stroke limit reached
StrokeLimitNeg	BOOL	Negative stroke limit reached
CommunicationReady	BOOL	Network is initialized and ready for communication
PowerOn	BOOL	Power stage is switched ON
AxisWarning	BOOL	Warning(s) on the axis is present
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 25 MC_DeviceService_Festo

This function block requests a selected device service of the servo drive.

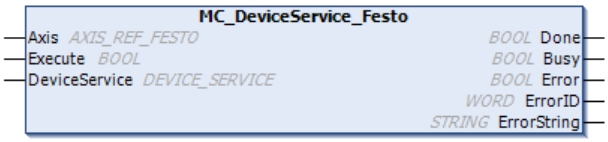


Figure 35: MC_DeviceService_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Request of device service FALSE --> True = request device service
DeviceService	DEVICE_SERVICE	Selection of the device service • 0 (none) = no selection • 1 (SaveZeroPointOffset) = save zero point offset to encoder • 2 (SaveParameterset) = save parameterset • 3 (ReinitDrive) = reinitialization of drive • 4 (ResetDrive) = reset drive • 5 (OpenHoldingBrake) = open holding brake • 6 (CloseHoldingBrake) = close holding brake
VAR_OUTPUT		
Done	BOOL	Device service successfully executed
Busy	BOOL	Device service will be requested • TRUE = Job started but not completed • FALSE = Job completed or not started
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

3. 26 MC_ForceControl_Festo

This function block starts a motion with a continuous force measured at the gear shaft. The setpoint force is given as a percentage of the nominal force. The specified velocity is not exceeded. This is force mode (ProfileTorqueMode).

Figure 36: MC_ForceControl_Festo

Note
If one of the inputs “NegativeStrokeLimit“ or “PositiveStrokeLimit“ = 0 , when the motion starts, the limit value set in the motor controller is used for this motion parameter.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Reference structure for the axis
VAR_INPUT		
Execute	BOOL	Start force mode • FALSE --› True = Start force mode
Force	REAL	Setpoint force in percent of the nominal force (100 = 100 % = nominal force)
ContinuousUpdate	BOOL	Continuous motion update • If input is TRUE during rising edge of Execute, changes at input variables will start a new movement immediately
ForceRamp	REAL	Force ramp in percent of nominal force per second (in %/s)
Velocity	REAL	Maximum velocity (in user unit)
BufferMode	MC_BUFFER_MODE	Defines the chronological sequence of the block (see section 6 – BufferMode)
StrokeLimitation	BOOL	Activation of stroke limitation • TRUE = Stroke limitation is active for this motion job
NegativeStrokeLimit	REAL	Negative stroke limitation (in user unit)
PositiveStrokeLimit	REAL	Positive stroke limitation (in user unit)
VAR_OUTPUT		
InForce	BOOL	Target force reached
Busy	BOOL	Job is not yet complete • TRUE = Job started but not completed • FALSE = Job completed or not started
Active	BOOL	Indicates that FB has control on the axis • TRUE = FB has control on the axis • FALSE = No control on the axis by this FB
CommandAborted	BOOL	Positioning task was aborted by another job during processing
Error	BOOL	An error occurred during processing • TRUE = Error active (see output ErrorID) • FALSE = No error
ErrorID	WORD	Error number (see the section 5 – Diagnostics)
ErrorString	STRING	Error report using the error number

4 Control via method call

In addition to calling the function blocks cyclically, a function can alternatively also be executed by calling a method. In contrast to executing a function by means of a block, instancing is not required with a method call. If the device name is typed in followed by a full stop, the available methods appear in a drop-down box (blue "M"). A method signals the successful execution with a Boolean return value. This is set in the same PLC cycle in which the method is called. In addition, each method has the outputs (ErrorID and ErrorString). These can be used to output any errors that may occur (see the section 5 – Diagnostics). It must be ensured that a method is only **called once**, not cyclically. Further information about how to use methods can be found in the section "Example of a method call" or in the online help for Codesys. The table below shows the available methods together with the respective transfer parameters. A detailed explanation of the individual transfer parameters (data types and meaning) can be found in the description of the respective function block (see above).

Method(*transfer parameter*)	Description
DisableDrive()	Disable output stage (OFF)
EnableDrive()	Enable output stage (ON)
Halt()	Pause active positioning task (pause ramp)
ResetHalt()	Continue active positioning task
Home(*Position, HomingMethod*)	Start homing
Jog(*JogPositive, JogNegative, JogMode*)	Start motion in ProfileJogMode starten (jogging)
MoveAbsolute(*Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode*)	Start absolute positioning
MoveAdditive(*Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode*)	Start positioning relative to current target position
MoveRelative(*Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode*)	Start positioning relative to current position
MoveVelocity(*Velocity, Acceleration, Deceleration, Jerk, BufferMode*)	Start motion with constant velocity (ProfileVelocityMode)
RecordTable(*RecordNumber, BufferMode*)	Start parameterised position record
Reset()	Reset motor controller errors
Stop()	Stop active positioning task (stop ramp)
ResetStop()	Transition of state from "Stopping to "Standstill" (deactivation of the stop command)
TorqueControl(*Torque, TorqueRamp, Velocity, BufferMode*)	Start motion with constant force (ProfileTorqueMode)
SaveZeroPointOffset()	Save zero point offset to encoder
SaveParameterset()	Save parameterset
ReinitDrive()	Reinitialization of drive
ResetDrive()	Reset drive
OpenHoldingBrake()	Open holding brake
CloseHoldingBrake ()	Close holding brake

In the case of the blocks described above, a successfully executed is action is usually signaled with the output variable "Done". In contrast, the return value of a method only means that it could be started successfully. Motion monitoring (e.g. "Target reached") must take place in the user program. Properties are available to the user for this. These can also be seen in the drop-down box by typing in the device name followed by a full stop (red "E"). Properties are updated before every call. Further general information about properties can be found in the online help for Codesys. An example of how to use a property can be found in the section Example of a method call. The table below shows the properties available for motor controllers from Festo.

Property	Data type	Description
ActualPosition	REAL	Current axis position (in user units)
ActualTorque	REAL	Current axis torque at the shaft (in Nm)
ActualVelocity	REAL	Current axis velocity (in user units)
HomingValid	BOOL	Axis has been homed
TargetReached	BOOL	Current positioning task successfully completed or no job active
AxisError	BOOL	Error has occurred in the motor controller
AxisErrorID	WORD	Error number
AxisErrorString	STRING	Error description using the error number
DeviceControl	DEVICE_CONTROL	Instance with master control (fieldbus or parameterisation software)
Enabled	BOOL	Output stage is energised (ON)
SDOsInitialized	BOOL	Drive has been successfully initialised
SlaveConnectorFlags	DWORD	Connection parameters of the EtherCAT® slave
AxisWarning	BOOL	Warning(s) on the axis is present
AxisIsMoving	BOOL	Axis is moving
SetpointAcknowledge	BOOL	Shows if commanded movement is acknowledged by the servo drive

5 Diagnostics

A distinction is made between three error categories: initialization error, motor controller errors and function block errors. The error category can be identified by means of the prefix in the "ErrorString" output variable of a function block:

- **“Init:” prefix (ErrorID ‹10)** = Error during initialization of EtherCAT slave or library (Diagnostics listed below)
- **"Axis:" prefix (ErrorID ›10 and ‹ 1000)** = Controller error (Diagnostics and fault clearance must be looked up in the user manual)
- **"FB:" prefix (ErrorID › 1000)** = Function block error (Diagnostics and fault clearance are listed below)

ID (dec)	ID (hex)	Description
General errors		
0	0x00	No error
Initialization errors		
1	0x1	Error in profile library (for example CiA402) – one or more PDO pointer to NULL
2	0x2	Error in device library (for example CMMT-AS – right EtherCAT master not found
3	0x3	Error in device library (for example CMMT-AS – right EtherCAT slave not found
4	0x4	Error in fielbus library (EtherCAT – error while reading SDO List of EtherCAT slave
5	0x5	Error in fielbus library (EtherCAT – error while reading SDO List Length of EtherCAT slave (only Beckhoff PLCs)
6	0x6	Error fieldbus – EtherCAT Slave not in operational
7	0x7	Error fieldbus – communication data of EtherCAT Slave not valid
8	0x8	Input „EC_PDActive“ of axis function block not active (systemvariable „_EC_PDActive“)
EtherCAT library errors		
1001	0x03E9	Service Data Object (SDO) command failed • Write access error: Check whether Service Data Object (SDO) can be written
1002	0x03EA	Other EtherCAT library error
1003	0x03EB	Data overflow in EtherCAT library
1004	0x03EC	Timeout in EtherCAT library • Check device state of drive • Check communication between PLC and drive
Errors in communication between PLC and drive		
1050	0x041A	Communication error between PLC and drive – communication is not active • Check state of master and slave

1051	0x041B	Communication error between PLC and drive – bus error • Check whether connecting cable is plugged in
1052	0x041C	Communication error between PLC and drive – general error
1053	0x041D	Communication error between PLC and drive – extended error
1054	0x041E	Communication error between PLC and drive – drive does not have "Operational" communication state
PtP library errors		
1100	0x044C	Axis is in incorrect state for the requested motion • Check the current axis state using the "AxisState" variables • To enable the output stage, the axis should be in state 1 (Disabled) • For motion, the axis should be in state 2 (Standstill)
1101	0x044D	Axis has an error • Eliminate the cause of the error and acknowledge the error using the "MC_Reset_Festo" function block
1102	0x044E	Timeout • Requested function could not be implemented within the specified time • Timeout for "MC_Power_Festo" = Check whether reinitialisation of the drive is necessary or check the axis state • Timeout for "MC_Reset_Festo" = Axis error could not be acknowledged
1103	0x044F	EtherCAT slave has no connection to master • Check communication connection
1104	0x0450	PLC (fieldbus) has no master control • Check whether configuration tool has master control • Check communication connection and state of drive
1105	0x0451	Library initialisation error
1106	0x0452	Requested action is not supported • Function is not supported by device • Library referencing incorrect
1107	0x0453	Action cannot be executed due to incorrect start conditions
1108	0x0454	Requested homing method is not supported • Drive does not support the selected homing method
1109	0x0455	Output stage is no longer energised • Requested function could not be executed or was interrupted because the drive is no longer energised
1110	0x0456	An error occurred during the homing process/the homing process was terminated • Check diagnostic message of drive
1111	0x0457	Error during deletion of a motion command • Active motion command or motion command in the queue could not be deleted
1112	0x0458	An additional "MC_Halt_Festo" function block has adopted master control • At least two different instances of "MC_Reset_Festo" are accessing the drive at the same time

1113	0x0459	Too many motion commands active at the same time • Too many buffered motion commands were entered
1114	0x045A	No device service is selected • Select a existing device service (see the section 3. 25 – MC_DeviceService_Festo)
1115	0x045B	Unknown device service is selected • Select a existing device service (see the section 3. 25 – MC_DeviceService_Festo)
1116	0x045C	No brake control possible • Axis is enabled, therefore no master control over holding brake possible by user program
1117	0x045D	Error could not be acknowledged • Error could not be acknowledged, please eliminate the cause of the error.
Parameter access errors (SDO access)		
1120	0x0460	Dynamic values for motion could not be written • Acceleration (object 0x6083), deceleration (object 0x6084) or jerk (object 0x60A4) could not be written
1121	0x0461	Acceleration (object 0x6083) could not be written
1122	0x0462	Deceleration (object 0x6084) could not be written
1123	0x0463	Jerk (object 0x60A4) could not be written
1124	0x0464	Option code for positioning (object 0x60F2) could not be written
1125	0x0465	Force ramp in force mode (object 0x6087) could not be written
1126	0x0466	Velocity limit in force mode (object 0x2060) could not be written
1127	0x0467	Invalid parameter number • Parameter number 0 cannot be read/written
1128	0x0468	Error during read access to the "Homing Method" (object 0x6098)
1129	0x0469	Error during write access to the "Homing Method" (object 0x6098)
1130	0x046A	Error during read access to the "Home Offset" (object 0x607C)
1131	0x046B	Error during write access to the "Home Offset" (object 0x607C)
1132	0x046C	Error during write access to the "Activation of stroke limitation" (object 0x216F.15)
1133	0x046D	Error during write access to the "Negative stroke limit" (object 0x216F.11)
1134	0x046E	Error during write access to the "Positive stroke limit" (object 0x216F.10)

Internal parameter handler errors		
1200	0x04B0	Internal queue for SDO commands is full • Too many read/write commands were requested
1201	0x04B1	Data type of the SDO is not identifiable • Drive could not report/find a data type
1202	0x04B2	Requested parameter number not available • Requested parameter number is not in the drive's object directory
1203	0x04B3	Requested subindex not available • Requested subindex is not in the drive's object directory (requested subindex is too high)
1204	0x04B4	Error identifying the SDO • Data type could not be identified by the drive (simple)
1205	0x04B5	Requested subindex not available • Requested subindex is not in the drive's object directory (requested subindex does not exist)
1206	0x04B6	Error identifying the SDO • Data type could not be identified by the drive (extended)
1207	0x04B7	Data type of the SDO is not identifiable • Unknown data type was reported by drive
1208	0x04B8	Data type of parameter is not a "STRING" • The reported data type is not a "STRING"
Errors in record mode		
1300	0x0514	Next position set number could not be written (object 0x216F.20)
Errors reading the parameter file		
1400	0x0578	Error opening the parameter file
1401	0x0579	Error reading the parameter file
1402	0x057A	Read parameter file entry is invalid (SDO description invalid)
1403	0x057B	Read parameter file entry is invalid (parameter number = 0)
1404	0x057C	Read parameter file entry is invalid (unknown data type)

Hardware errors		
2048	0x0800	Axis error ID is currently being read. An axis error occurred and the description is currently being read. Note: It is an acyclic service data object (SDO command). The processing is typically longer than with the cyclic process data (PDOs) and can take up to 300 ms. Maybe at the time when the error occurred (Fault bit in the status word) there was no corresponding error description available and has to be read.
2069	0x0815	General error – drive is not ready • The drive has an error but there is no "Emergency Message" • Eliminate the cause of the error and acknowledge the error Note: The "Emergency Message" is an acyclic service data object (SDO command). The processing is typically longer than with the cyclic process data (PDOs) and can take up to 300 ms. Maybe at the time when the error occurred (Fault bit in the status word) there was no corresponding "Emergency Message" available. Read out the current ErrorID and the ErrorString again via the "MC_ReadAxisError_Festo" block. If this function block also outputs the same error, there is no "Emergency Message".

6 BufferMode

A number of modules have the input variable "BufferMode". This variable can be used to select whether a block should be started with or without buffering. In non-buffered mode, the block is started immediately. If another block is active, it is aborted. By default, each block is started in non-buffered mode. In buffered mode, the started block does not interrupt an active block; it waits until this block has been completed and then starts. This option can be used, for example, to add a motion to an active motion. The following modes are available:

Value	MC_BUFFER_MODE	Description
0	mcAborting	Block is started immediately (default setting)
1	mcBuffered	Block is started once the current motion has been completed

7 Direction

Modulo mode simplifies the implementation of intermittent endless movements, e.g. the operation of rotary indexing tables and conveyor belts. The modulo range is described by a minimum value and a maximum value. If a setpoint value is specified that lies outside the defined modulo range, only the remaining distance resulting from the modulo division is moved. For activation via the drive profile, the following parameters must be written (both unequal 0):

- Upper limit value modulo
- Lower limit value modulo

To deactivate modulo mode via the drive profile, both parameters must be written with the value 0. When modulo mode is active, you can switch between the "Position Mode PP" profile and the "Velocity Mode PV" profile without deactivating modulo mode. If modulo mode is activated in the drive, the positioning direction can be forced by the input “Direction”. This input is of type “MC_DIRECTION” and can be accept all the values of followintg table. The input is in the function blocks „MC_MoveAbsolute_Festo“, „MC_MoveRelative_Festo“ and „MC_MoveAdditive_Festo“ available.

Note
More information about the modulo mode can be found in the respective device manual.

Value	MC_DIRECTION	Description
0	mcNormal	Normal positioning according to linear axis (default settings)
64	mcNegativeDirection	Positioning only in negative direction
128	mcPositiveDirection	Positioning only in positive direction
192	mcShortestWay	The drive moves the shortest path to the target position resulting from the modulo division. The modulo limits are not considered.

8 Visualisation of function blocks

A visualisation element is available for each function block. Its main purpose is to help with the initial commissioning of the motor controller. The visualisation element must be linked to the respective cyclically called block instance. The diagram below shows this by way of example using "MC_Power_Festo".

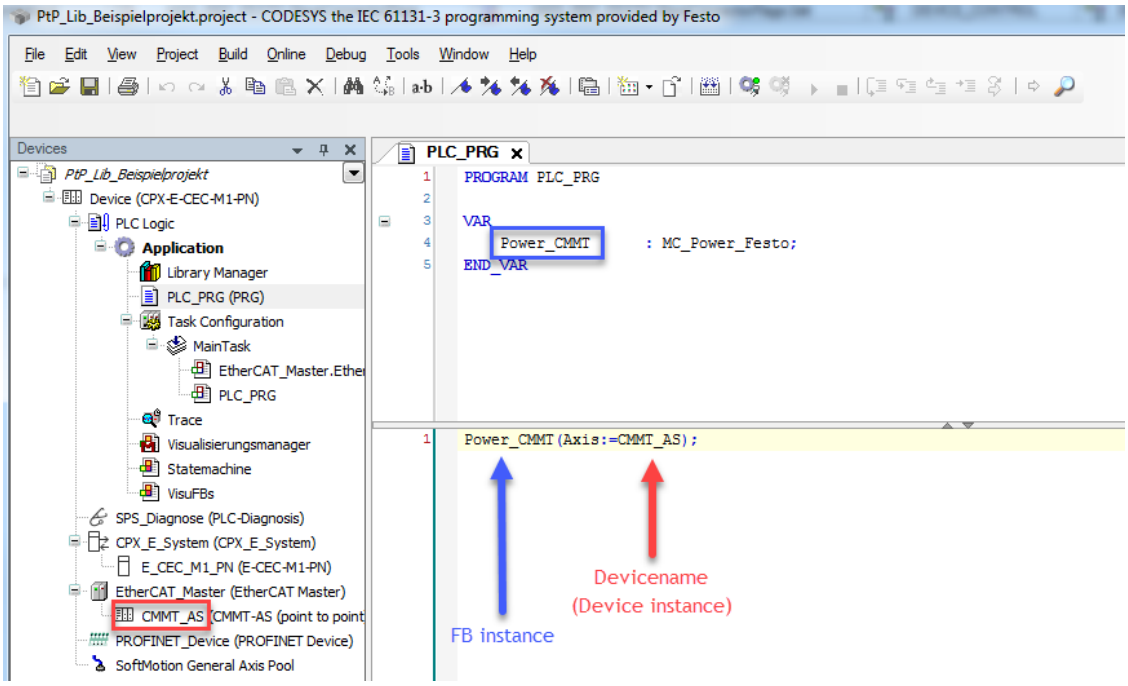


Figure 367: Cyclical block call

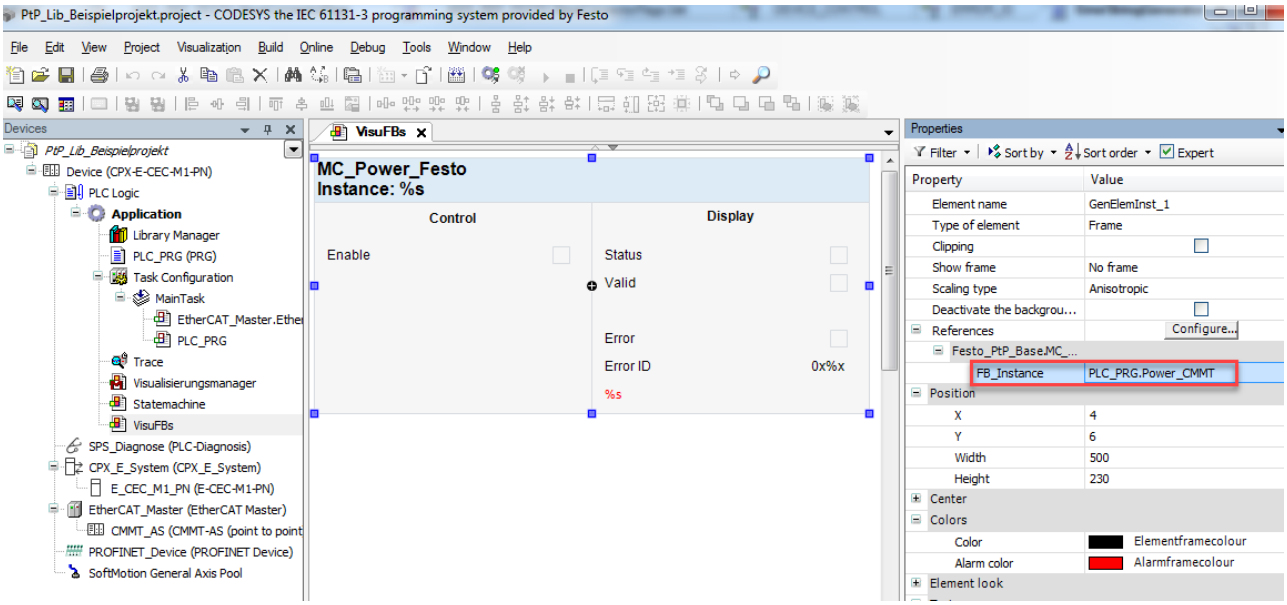


Figure 378: Linking a visualisation element

9 Application examples

As already described, there are two ways of executing a function in the user program. The sections below show both ways of executing a function to enable a drive and then execute a relative motion of 100 mm. The drive must be homed for this.

9. 1 Example of function blocks

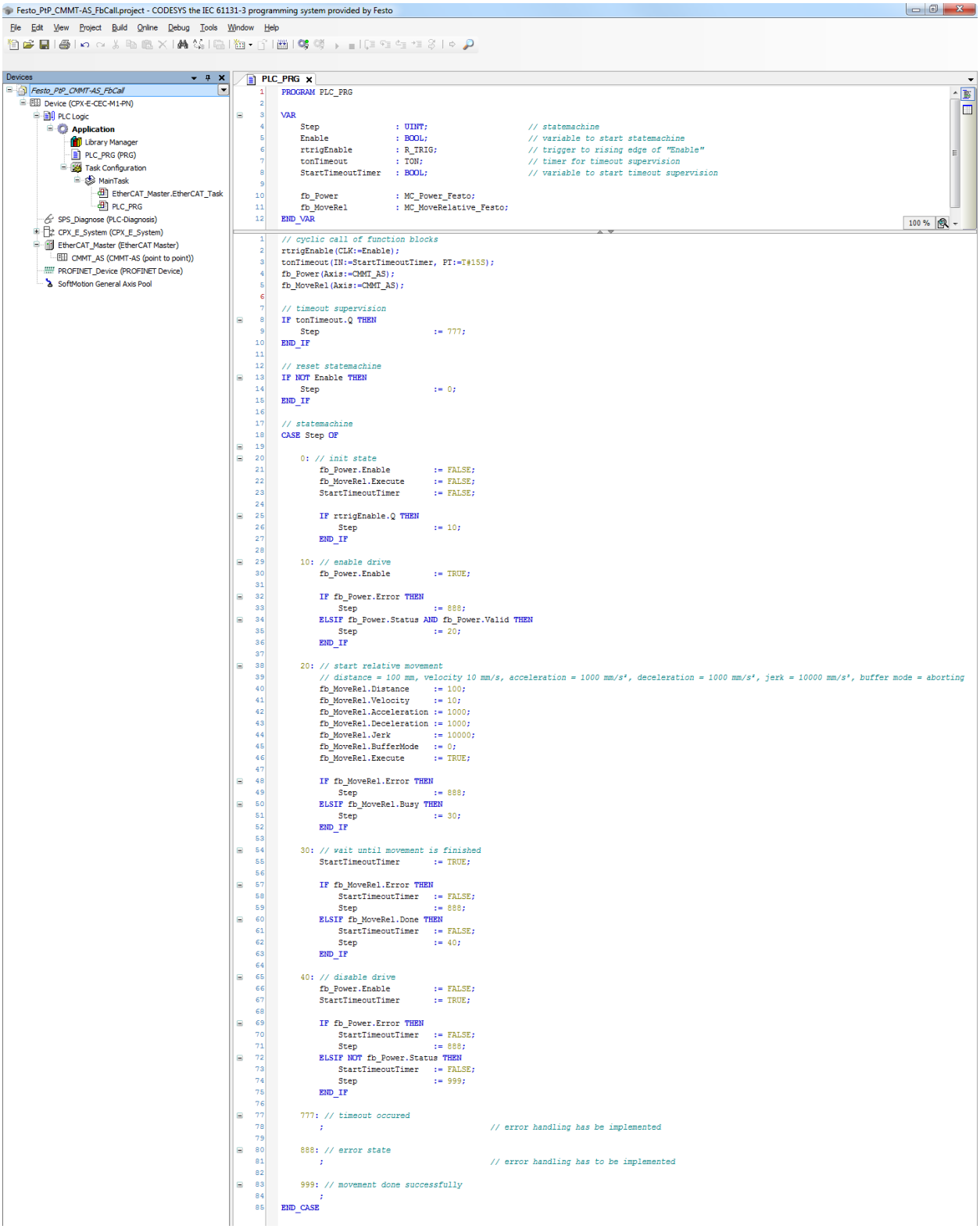


Figure 389: Example of a block call

9. 2 Example of a method call

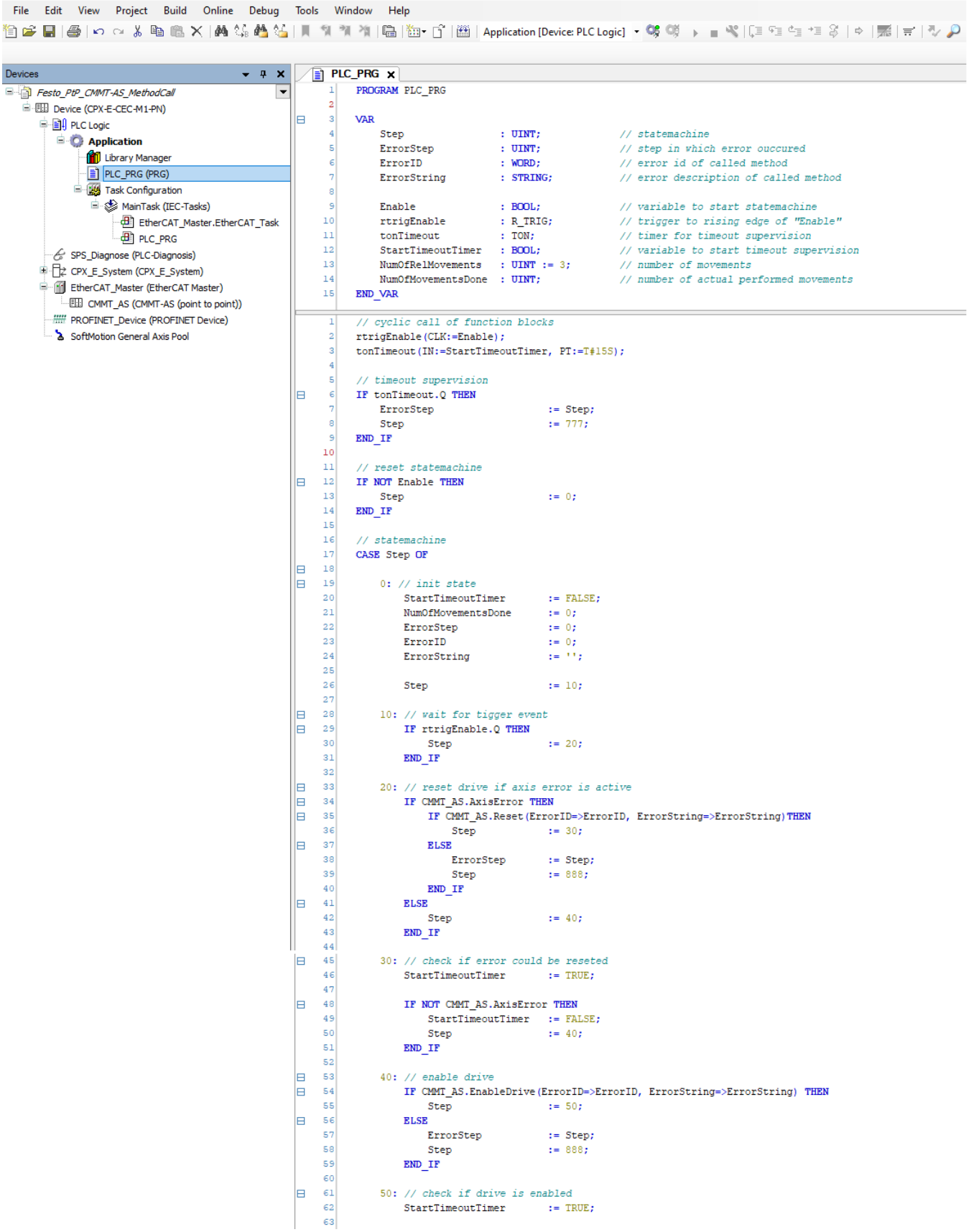


Figure 40: Example of a method call part 1

```
64          IF CMMT_AS.Enabled AND NOT CMMT_AS.HomingValid THEN
65              StartTimeoutTimer   := FALSE;
66              Step                := 60;
67          ELSIF CMMT_AS.Enabled AND CMMT_AS.HomingValid THEN
68              StartTimeoutTimer   := FALSE;
69              Step                := 80;
70          END_IF
71
72      60: // start homing as configured in Festo Automation Suite
73          IF CMMT_AS.Home(0, 0, ErrorID=>ErrorID, ErrorString=>ErrorString) THEN
74              Step                := 70;
75          ELSE
76              ErrorStep           := Step;
77              Step                := 888;
78          END_IF
79
80      70: // check if homing is valid
81          StartTimeoutTimer       := TRUE;
82
83          IF CMMT_AS.HomingValid THEN
84              StartTimeoutTimer   := FALSE;
85              Step                := 80;
86          END_IF
87
88      80: // start relative movement
89          // distance = 10 mm, velocity 100 mm/s, acceleration = 10000 mm/s², deceleration = 1000 mm/s², jerk = 10000 mm/s³, buffer mode = aborting
90          IF CMMT_AS.MoveRelative(10, 100, 1000, 1000, 10000, 0, 0, ErrorID=>ErrorID, ErrorString=>ErrorString) THEN
91              Step                := 90;
92          ELSE
93              ErrorStep           := Step;
94              Step                := 888;
95          END_IF
96
97      90: // wait until start of movement is acknowledged from drive
98          StartTimeoutTimer       := TRUE;
99
100         IF CMMT_AS.SetpointAcknowledge AND NOT CMMT_AS.TargetReached THEN
101             NumOfMovementsDone  := NumOfMovementsDone + 1;
102             StartTimeoutTimer   := FALSE;
103             Step                := 100;
104         END_IF
105
106     100: // wait until movement is finished
107         StartTimeoutTimer       := TRUE;
108
109         IF CMMT_AS.TargetReached THEN
110             StartTimeoutTimer   := FALSE;
111             Step                := 110;
112         END_IF
113
114     110: // check number of movements
115         IF NumOfMovementsDone >= NumOfRelMovements THEN
116             Step                := 120;
117         ELSE
118             Step                := 80;
119         END_IF
120
121     120: // disable drive
122         IF CMMT_AS.DisableDrive(ErrorID=>ErrorID, ErrorString=>ErrorString) THEN
123             Step                := 130;
124         ELSE
125             ErrorStep           := Step;
126             Step                := 888;
127         END_IF
128
129     130: // wait until drive is disabled
130         StartTimeoutTimer       := TRUE;
131
132         IF NOT CMMT_AS.Enabled THEN
133             StartTimeoutTimer   := FALSE;
134             Step                := 999;
135         END_IF
136
137     777: // timeout occured
138         ;                                   // error handling has be implemented
139
140     888: // error state
141         ;                                   // error handling has to be implemented
142
143     999: // movements done successfully
144         ;
145 END_CASE
```

Figure 41: Example of a method call part 2