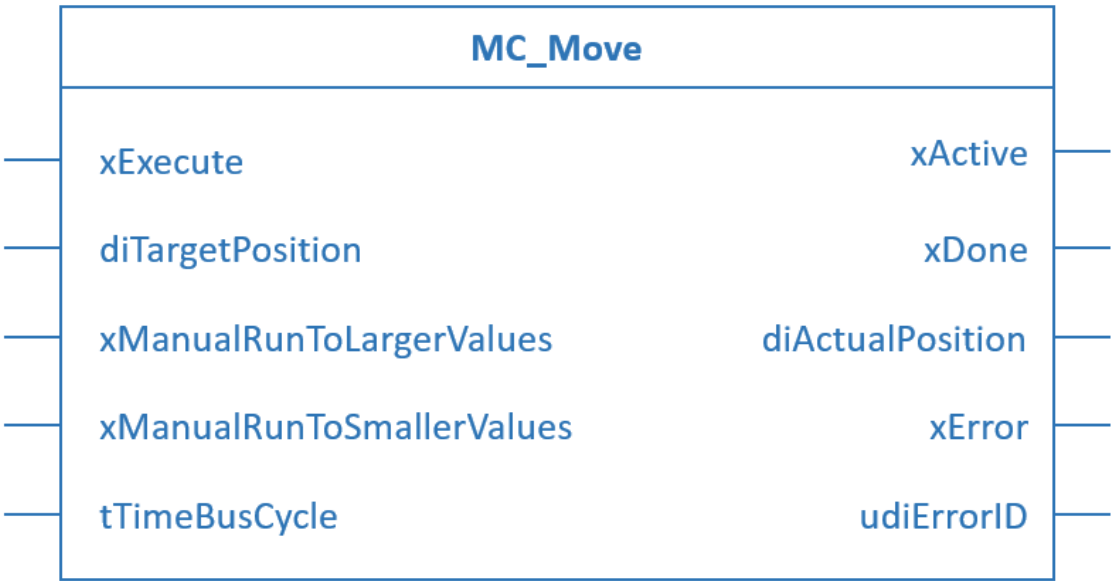


## Function Blocks PSx3xx for TwinCAT 3



## Purpose of instruction manual

This instruction manual describes the function blocks for the PSx3xx EC (with EtherCAT interface).

Improper use of these devices or failure to follow these instructions may cause injury or equipment damage. Every person who uses the devices must therefore read the instruction manual and has to understand the possible risks. The instruction manual and in particular the safety precautions, contained therein, must be followed carefully.

**Contact them manufacturer if you do not understand any part of this instruction manual.**

The manufacturer reserves the right to continue developing these function blocks without documenting such development in each individual case. The manufacturer will be happy to determine whether this manual is up-to-date.

The copyright to these operating instructions is with the manufacturer. It contains technical data, instructions and drawings for the function and handling of the software. It is forbidden to be reproduced or made accessible to third parties in whole or in part.

halstrup-walcher GmbH  
Stegener Straße 10  
79199 Kirchzarten

Tel. +49 (7661) 39 63-0  
[info@halstrup-walcher.de](mailto:info@halstrup-walcher.de)  
[www.halstrup-walcher.de](http://www.halstrup-walcher.de)

© 2024

**15.04.2024, TS & FS**

**7100.007014 Version 2.1.1 Instruction Manual Function Blocks PSx3xx EC**

## Table of contents

1	Safety precautions .....	5
1.1	Appropriate use .....	5
1.2	Symbols .....	5
2	User-specific data types.....	6
2.1	<ST_DriveData_PSx3xx> .....	6
2.2	<ST_FunctionBlocks_PSx3xx> .....	7
2.3	<ST_Variables_PSx3xx> .....	8
3	<GVL_Data_Store_PSx3xx>.....	8
4	Error description (<udiErrorID>) .....	9
5	Description of the function blocks.....	11
5.1	FB_MC_Move_PSx3xx.....	11
5.2	FB_MC_Error_PSx3xx .....	12
5.3	FUN_MC_Error_ID_PSx3xx .....	12
5.4	FB_MC_ReadParameter_PSx3xx .....	12
5.5	FB_MC_WriteParameter_PSx3xx.....	13
5.6	FB_MC_Parametrization_PSx3xx .....	13
5.7	FB_MC_PosParametrization_PSx3xx.....	14
6	Description of the parameters .....	15
6.1	<xActive> .....	15
6.2	<diActualPosition> .....	15
6.3	<xDeliveryState> .....	16
6.4	<uiDirection> .....	16
6.5	<xDone> .....	16
6.6	<stDrive> .....	17
6.7	<xError> .....	17
6.8	<sErrorDescription> .....	17
6.9	<udiErrorID> .....	18
6.10	<uiErrorParameter>.....	18
6.11	<xExecute> .....	19
6.12	<diLowerLimit>.....	19
6.13	<xManualRunToLargerValues> .....	19
6.14	<xManualRunToSmallerValues> .....	20
6.15	<sNetId> .....	20
6.16	<stParameter> .....	20
6.17	<uiParameterNumber> .....	21
6.18	<xSaveSettings>.....	21

6.19	<diSetPoint>.....	21
6.20	<uiSlaveAddr>.....	22
6.21	<uiStepsPerTurn>.....	22
6.22	<usiSubIndex> .....	22
6.23	<diTargetPosition> .....	23
6.24	<tTimeBusCycle> .....	23
6.25	<diUpperLimit> .....	24
6.26	<diValue> .....	24
6.27	<sValue> .....	24
7	Application of the function blocks .....	25
7.1	Project.....	25
7.2	Library .....	25
7.3	Interlock between the function blocks .....	26
7.4	Several PSx3xx in a project .....	27
8	Example programs.....	28
8.1	Example 1: positioning mode.....	28
8.2	Example 2: manual mode .....	29
8.3	Example 3: read actual value.....	29
8.4	Example 4: writing the delivery state .....	30
8.5	Example 5: Parameterize parameters.....	30
8.6	Example 6: Parameterize position parameters.....	31
8.7	Example 7: Error upper end limit reached.....	31
	List of Figures.....	32

# 1 Safety precautions




## 1.1 Appropriate use

The positioning systems PS-3xx EC are especially suitable for automatically setting tools, stops or spindles for wood-processing equipment, packing lines, printing equipment, filling units and other types of special machines.

**PSx3xx PN positioning systems are not stand-alone devices and may only be used, if coupled to another machine.**

## 1.2 Symbols

The symbols given below are used throughout this instruction manual to indicate instances when improper operation could result in the following hazards:

 <b>DANGER</b>	<b>DANGER!</b>  Indicates a situation of imminent danger, which will lead to a fatality or serious injuries if not prevented.
 <b>WARNING</b>	<b>WARNING!</b>  Indicates a potentially dangerous situation, which may lead to a fatality or serious injuries if not prevented.
 <b>CAUTION</b>	<b>CAUTION!</b>  Indicates a potentially dangerous situation, which may lead to minor/slight injuries if not prevented.
<b>NOTICE</b>	<b>NOTICE</b>  Indicates a potentially harmful situation, which may lead to material damage if not prevented.

## 2 User-specific data types

There are many user-specific data types. Only the three most important data types are documented below.

### 2.1 <ST\_DriveData\_PSx3xx>

This data structure is used to store some of the drive's data. For each drive, a global instance of this structure is required. This instance must be provided to each function block (FB) that operates on the corresponding drive. This is used for example, to prevent two FB from accessing the parameter interface of a single drive at the same time. Furthermore, the address of the EtherCAT master and the address of the EtherCAT slave (PSx3xx) must be stored in this data structure.

Parameter name	Data type	Written by	Description
<sAxisName>	String[16]	User (optional)	Name of axis
<sAxisDescription>	String[32]	User (optional)	Description (e. g. function, task of this axis)
<iActiveFunctionBlock>	Int	Function blocks	Active function block
<xCommunicationError>	Bool	Function blocks	Communication error to IO-Device
<sNetId>	T_AmsNetId	User User	Address of EtherCAT-Master
<uiSlaveAddr>	UInt	User	Address of EtherCAT-Slave
<stPrivate>	ST_Private	Function blocks	Data structure for internal use

The following illustrations show how the assigned addresses of the master and the slave can be checked in the TwinCAT automation software:

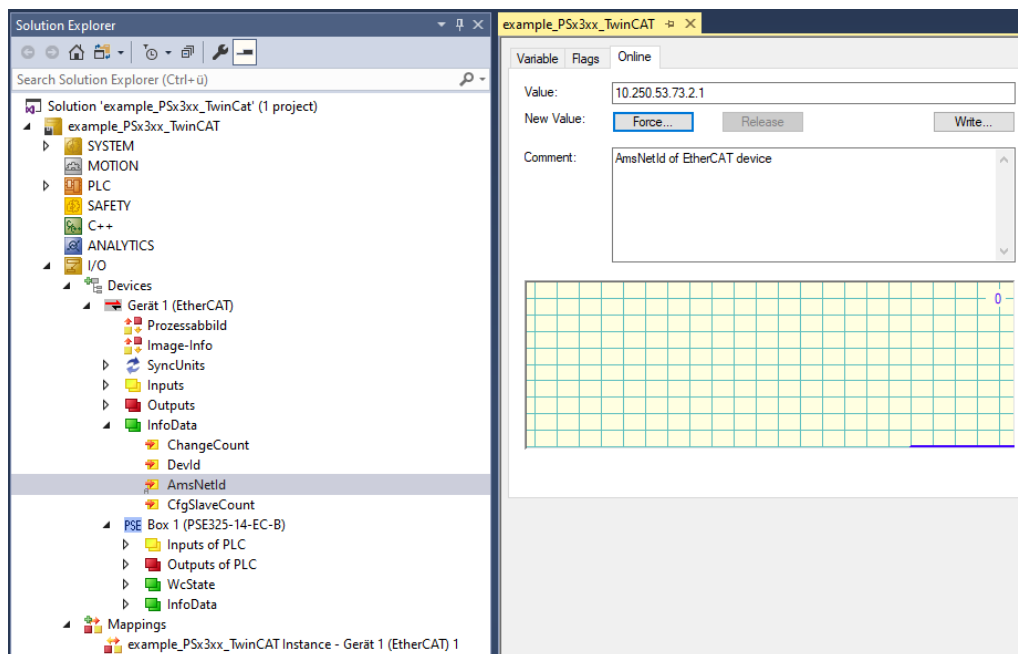


Figure 1: Address of EtherCAT-Master

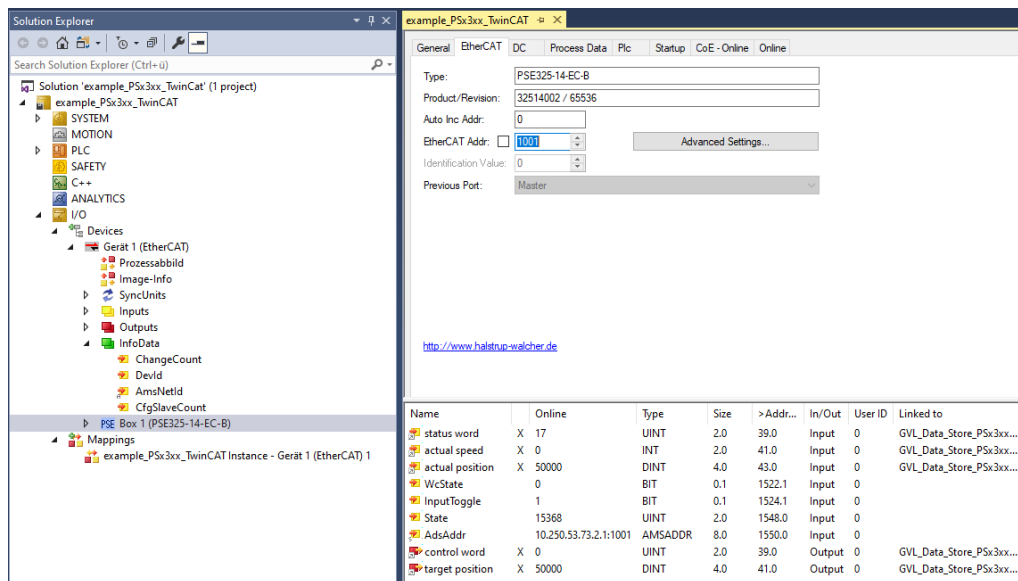


Figure 2: Address of EtherCAT-Slaves

## 2.2 <ST\_FunctionBlocks\_PSx3xx>

This structure contains all variables necessary to connect to the inputs and outputs of the function blocks, except for stDrive, which requires ST\_DriveData (described in 2.1). The structure has several substructures, which are not explained further in the documentation. It is recommended to create a separate global instance of this structure, for every drive in the project alternatively, the necessary variables can be created individually. If function blocks are not used, it is recommended to adapt the data structure so that memory is not unnecessarily occupied in the PLC.

Parameter name	Data type	Written by	Description
<stMove_PSx3xx>	ST_Move_PSx3xx	Function blocks	variable for FB_MC_Move_PSx3xx
<stError_PSx3xx>	ST_Error_PSx3xx	Function blocks	variable for FB_MC_Error_PSx3xx
<stReadParameter_PSx3xx>	ST_ReadParameter_PSx3xx	Function blocks	variable for FB_MC_ReadParameter_PSx3xx
<stWriteParameter_PSx3xx>	ST_WriteParameter_PSx3xx	Function blocks	variable for FB_MC_WriteParameter_PSx3xx
<stParametrization_PSx3xx>	ST_Parametrization_PSx3xx	Function blocks	variable for FB_MC_Parametrization_PSx3xx
<stPosParametrization_PSx3xx>	ST_PosParametrization_PSx3xx	Function blocks	variable for FB_MC_PosParametrization_PSx3xx

## 2.3 <ST\_Variables\_PSx3xx>

This structure combines <ST\_FunctionBlocks\_PSx3xx> and <ST\_DriveData\_PSx3xx>. It is recommended to create an instance for every drive in <GVL\_Data\_Store\_PSx3xx>.

## 3 <GVL\_Data\_Store\_PSx3xx>

The GVL (global variable list) <GVL\_Data\_Store\_PSx3xx> provides the necessary variables to assign all inputs and outputs of all available function blocks.

Each variable is currently only available once. If there are several PSx3xx in a project, the number of variables must be adjusted accordingly.

### Example

```

19 //fbMC_Move_PSx3xx_1 --> instance of FB_MC_Move_PSx3xx (1. PSx3xx)
20 fbMC_Move_PSx3xx_1(
21     xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xExecute,
22     diTargetPosition:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.diTargetPosition,
23     xManualRunToLargerValues:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xManualRunToLargerValues,
24     xManualRunToSmallerValues:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xManualRunToSmallerValues,
25     tTimeBusCycle:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.tTimeBusCycle,
26     xActive:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xActive,
27     xDone:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xDone,
28     diActualPosition:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.diActualPosition,
29     xError:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xError,
30     udiErrorID:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.udiErrorID,
31     stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx);

```

Figure 3: Allocation of the variables from <GVL\_Data\_Store\_PSx3xx> to FB\_MC\_Move\_PSx3xx

If the GVL is transferred to the project, it is recommended to delete the unused variables (so that memory is not unnecessarily occupied in the PLC) and to adapt the block to the number of drives actually present.

example_PSx3xx_TwinCAT.example_PSx3xx_TwinCAT.GVL_Data_Store_PSx3xx					
Expression	Type	Value	Prepared value	Address	Comment
PSE_1	ST_Variables_PSx3xx				1. PSx3xx (here: PSE325_14_EC_B)
stDriveData_PSx3xx	ST_DriveData_PSx3xx				variables for the data of PSx3xx
sAxisName	STRING(16)				name of the axis (optional)
sAxisDescription	STRING(32)				description for example function/task (optional)
uiState	UINT	0			actual state (interlock)
xCommunicationError	BOOL	FALSE			communication error with IO-device
sNetId	T_AmsNetId	10.250.53.73.2.1			AmsNetId of the EtherCAT master device
uiSlaveAddr	UINT	1001			address of the slave device
stPrivate	ST_Private				data structure for internal use
stFunctionBlocks_PSx3xx	ST_FunctionBlocks_...				variables for all function blocks
stMove_PSx3xx	ST_Move_PSx3xx				variables for FB_MC_Move_PSx3xx
stError_PSx3xx	ST_Error_PSx3xx				variables for FB_MC_Error_PSx3xx
stReadParameter_PSx3xx	ST_ReadParameter_...				variables for FB_MC_ReadParameter_PSx3xx
stWriteParameter_PSx3xx	ST_WriteParameter_...				variables for FB_MC_WriteParameter_PSx3xx
stParametrization_PSx3xx	ST_Parametrization_...				variables for FB_MC_Parametrization_PSx3xx
stPosParametrization_PSx3xx	ST_PosParametrizati...				variables for FB_MC_PosParametrization_PSx3xx

Figure 4: <GVL\_Data\_Store\_PSx3xx> with the variables for a drive



## 4 Error description (<udiErrorID>)

The error codes output by the function blocks are described below:

<b>&lt;udiErrorID&gt; (hex)</b>	<b>Description</b>
<b>16#F0000 (Mask)</b>	<b>Function block</b>
16#0xxxx	No error code
16#1xxxx	Error in FB_MC_Move_PSx3xx
16#2xxxx	Error in FB_MC_Error_PSx3xx
16#3xxxx	Error in FB_MC_ReadParameter_PSx3xx
16#4xxxx	Error in FB_MC_WriteParameter_PSx3xx
16#5xxxx	Error in FB_MC_Parametrization_PSx3xx
16#6xxxx	Error in FB_MC_PosParametrization_PSx3xx
<b>16#0F000 (Mask)</b>	<b>Internal errors in the function blocks and process data errors</b>
16#x0xxx	No internal error in the function blocks and process data error
16#x1xxx	Error in the state machine (interlock)
16#x2xxx	Invalid process data input address
16#x3xxx	Invalid process data output address
16#x4xxx	Error while reading process data
16#x5xxx	Error while writing process data
16#x6xxx	Invalid write command
<b>16#00F00 (Mask)</b>	<b>Error in parameter</b>
16#xx0xx	No parameter errors
16#xx1xx	Parameter: communication timeout (1000 ms)
16#xx2xx	Parameter: invalid parameter number
16#xx3xx	Parameter: parameter value is read only
16#xx4xx	Parameter: below lower limit or upper limit exceeded
16#xx5xx	Parameter: invalid subindex
16#xx6xx	Parameter: no array
16#xx7xx	Parameter: incorrect data type
16#xx8xx	Parameter: setting not allowed (resetting only)
16#xx9xx	Parameter: request cannot be processed due to operating state
16#xxAxx	Other error
<b>16#000F0 (Mask)</b>	<b>Drive errors</b>
16#xxx0x	No drive errors
16#xxx1x	Drag error
16#xxx2x	Under- or overvoltage motor supply (STO-enabling inactive*)
16#xxx3x	Positioning run aborted
16#xxx4x	Temperature exceeded
16#xxx5x	Measuring system error (Measuring system error or STO hardware error*)
16#xxx6x	Positioning error (block)
16#xxx7x	Manual displacement
16#xxx8x	Incorrect target value
16#xxx9x	Under- or overvoltage during run / failure voltage control
16#xxxAx	Below lower limit
16#xxxBx	Upper limit exceeded

\* If the PSx3xx is a device with STO, then the error description in brackets must be taken into account! (<udiErrorID>: 16#xxx2x and 16#xxx5x)

The errors "Drive Errors" are a copy of the error bits in the status word of the PSx3xx.

#### **Examples**

- Run command (**MC\_Move\_PSx3xx**) with incorrect target value → <udiErrorID> = 16#10080
- Writing a parameter (**MC\_WriteParameter\_PSx3xx**) with invalid parameter number → <udiErrorID> = 16#40200

## 5 Description of the function blocks

Communication with the drive takes place via the process data of the drive. This process data is directly linked to the variables in the global variable list <GVL\_Data\_Store\_PSx3xx>.

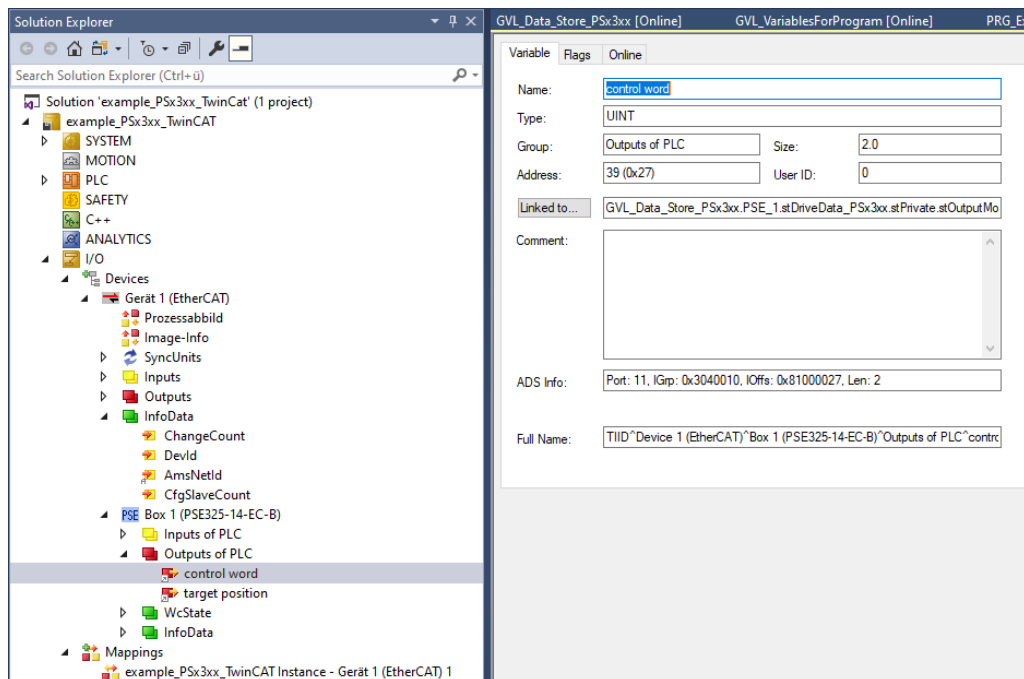


Figure 5: Allocation of *control word* with a variable from <GVL\_Data\_Store\_PSx3xx>.

### 5.1 FB\_MC\_Move\_PSx3xx

This function block is used to move the drive.

```

19 //fbMC_Move_PSx3xx_1 --> instance of FB_MC_Move_PSx3xx (1. PSx3xx)
20 fbMC_Move_PSx3xx_1 {
21     xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xExecute,
22     diTargetPosition:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.diTargetPosition,
23     xManualRunToLargerValues:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xManualRunToLargerValues,
24     xManualRunToSmallerValues:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xManualRunToSmallerValues,
25     tTimeBusCycle:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.tTimeBusCycle,
26     xActive=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xActive,
27     xDone=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xDone,
28     diActualPosition:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.diActualPosition,
29     xError=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xError,
30     udiErrorID=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.udiErrorID,
31     stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx;
32 }

```

Figure 6: Allocation of variables from DB <Data\_Store> to FB **MC\_Move\_PSx3xx**

If the drive reports multiple errors, the <udiErrorID> with the highest priority is shown. This applies to all FBs. This priority corresponds to the order in the following table (highest priority has 16#x1xxx):

<udiErrorID>	Description
16#x1xxx	Error in the state machine (interlock)
16#x2xxx	Invalid process data input address
16#x3xxx	Invalid process data output address
16#x4xxx	Error while reading process data
16#x5xxx	Error while writing process data
16#xxx2x	Under- or overvoltage motor supply (STO-enabling inactive*)
16#xxx4x	Temperature exceeded
16#xxx5x	Measuring system error (Measuring system error or STO hardware error*)
16#xxx8x	Incorrect target value
16#xxx9x	Under- or overvoltage during run / failure voltage control
16#xxx6x	Positioning error (block)
16#xxx7x	Manual displacement
16#xxxAx	Below lower limit
16#xxxBx	Upper limit exceeded
16#xxx3x	Positioning run aborted
16#xxx1x	Drag error

\*If the PSx3xx is a device with STO, then the error description in brackets must be taken into account! (<udiErrorID >: 16#xxx2x und 16#xxx5x)

## 5.2 FB\_MC\_Error\_PSx3xx

This FB reports the state of the drive and the FB as error bit, error code (<udiErrorID >) and as text output.

```

58 //FBMC_Error_PSx3xx_1 --> instance of FB_MC_Error_PSx3xx (1. PSx3xx)
59 fbMC_Error_PSx3xx_1 {
60   xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stInput_Error_PSx3xx.xExecute,
61   xError=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stOutput_Error_PSx3xx.xError,
62   udiErrorID=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stOutput_Error_PSx3xx.udiErrorID,
63   sErrorDescription=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stOutput_Error_PSx3xx.sErrorDescription,
64   stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx;
65 }

```

Figure 7: Allocation of variables from DB <GVL\_Data\_Store\_PSx3xx> to FB\_MC\_Error\_PSx3xx

## 5.3 FUN\_MC\_Error\_ID\_PSx3xx

This function is used internally as a sub function of the FBs **FB\_MC\_Move\_PSx3xx**, **FB\_MC\_ReadParameter\_PSx3xx**, **FB\_MC\_WriteParameter\_PSx3xx** and **FB\_MC\_Error\_PSx3xx**. It just has to be copied from the library into the application program. The setting of the inputs is done completely internally.

## 5.4 FB\_MC\_ReadParameter\_PSx3xx

This FB can be used to read out parameter values from the drive.

```

33 //FBMC_ReadParameter_PSx3xx_1 --> instance of FB_MC_ReadParameter_PSx3xx (1. PSx3xx)
34 fbMC_ReadParameter_PSx3xx_1 {
35   xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stInput_ReadParameter_PSx3xx.xExecute,
36   uiParameterNumber:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stInput_ReadParameter_PSx3xx.uiParameterNumber,
37   usiSubindex:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stInput_ReadParameter_PSx3xx.usiSubindex,
38   xActive=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.xActive,
39   xDone=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.xDone,
40   xError=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.xError,
41   udiErrorID=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.udiErrorID,
42   diValue:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.diValue,
43   sValue:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.sValue,
44   stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx;
45 }

```

Figure 8: Allocation of variables from <GVL\_Data\_Store\_PSx3xx> to FB\_MC\_ReadParameter\_PSx3xx

## 5.5 FB\_MC\_WriteParameter\_PSx3xx

Parameter values can be written to the drive with this FB.

```

46 //fbMC_WriteParameter_PSx3xx_1 --> instance of FB_MC_WriteParameter_PSx3xx (1. PSx3xx)
47 fbMC_WriteParameter_PSx3xx_1(
48   xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.xExecute,
49   uiParameterNumber:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.uiParameterNumber,
50   usiSubindex:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.usiSubindex,
51   diValue:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.diValue,
52   xActive:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.xActive,
53   xDone:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.xDone,
54   xError:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.xError,
55   udiErrorID:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.udiErrorID,
56   stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx);

```

Figure 9: Allocation of variables from <GVL\_Data\_Store\_PSx3xx> to FB\_MC\_WriteParameter\_PSx3xx

## 5.6 FB\_MC\_Parametrization\_PSx3xx

With this FB all parameters of the drive can be written at once. This is especially useful for the initialisation. A clear structure was chosen, as all parameters are transferred as a block with the data type <ST\_Parameter\_PSx3xx>.

```

66 //fbMC_Parametrization_PSx3xx_1 --> instance of FB_MC_Parametrization_PSx3xx (1. PSx3xx)
67 fbMC_Parametrization_PSx3xx_1(
68   xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.xExecute,
69   xDeliveryState:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.xDeliveryState,
70   stParameter:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.stParameter,
71   xSaveSettings:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.xSaveSettings,
72   xActive:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.xActive,
73   xDone:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.xDone,
74   xError:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.xError,
75   udiErrorID:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.udiErrorID,
76   uiErrorParameter:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.uiErrorParameter,
77   stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx);

```

Figure 10: Allocation of variables from <GVL\_Data\_Store\_PSx3xx> to FB\_MC\_Parametrization\_PSx3xx

**The following must be observed when using the FB:**

For each parameter, there is a variable <xEnable> and a variable <diValue> (depending on the data type of the parameter).

### Example

```

124 GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.stParameter.
125 stLoopLength_201F.xEnable := TRUE;
126 GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.stParameter.
127 stLoopLength_201F.diValue := 0;

```

Figure 11: Parameter „loop length“ with the value of 0

- By setting <xEnable> = FALSE, the respective parameter is not written.
- Optionally, a delivery state can be requested before setting individual parameters. To do this, the input <xDeliveryState> must be set to TRUE. This sets the values of all parameters to the delivery state (initially without saving).
- Optionally, the written values can also be saved at the end. To do this, the input <xSaveSettings> must be set to TRUE.
- The order of the write accesses is as follows:  
<xDeliveryState>, parameter 16#202C, 16#2010, 16#2011, 16#2003...and <xSaveSettings>.
- In the event of a written error, the following parameters are no longer written and the values are not saved if the input <xSaveSettings> is set.

## 5.7 FB\_MC\_PosParametrization\_PSx3xx

This FB can be used to parameterize the position parameters (parameters that affect the displayed actual position). This is especially useful for the initialisation.

```

79 //FB_MC_PosParametrization_PSx3xx_1 --> instance of FB_MC_PosParametrization_PSx3xx (1. PSx3xx)
80 fbMC_PosParametrization_PSx3xx_1(
81   xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.xExecute,
82   uiDirection:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.uiDirection,
83   uiStepsPerTurn:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.uiStepsPerTurn,
84   diLowerLimit:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.diLowerLimit,
85   diUpperLimit:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.diUpperLimit,
86   diSetPoint:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.diSetPoint,
87   xSaveSettings:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.xSaveSettings,
88   xActive:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.xActive,
89   xDone:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.xDone,
90   xError:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.xError,
91   udiErrorID:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.udiErrorID,
92   uiErrorParameter:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.uiErrorParameter,
93   stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx);
94

```

Figure 12: Allocation of variables from <GVL\_Data\_Store\_PSx3xx> to FB\_MC\_PosParametrization\_PSx3xx

The following must be observed when using the FB:

- All values must be set and the values must be in a meaningful context to each other. All values are processed, after which the following parameters are written in the specified order:
  1. Direction of rotation (Parameter 16#202C) → <uiDirection>
  2. Position scaling numerator (Parameter 16#2010) → 400
  3. Position scaling denominator (Parameter 16#2011) → <uiStepsPerTurn>
  4. Actual value (Parameter 16#2003) → <diSetPoint>
  5. If (<diSetPoint> > <diUpperLimit>):
    - Upper mapping end (Parameter 16#2028) = <diSetPoint> + (3 x <uiStepsPerTurn>)
    - Else:
      - Upper mapping end (parameter 16#2028) = <diUpperLimit> + (3 x <uiStepsPerTurn>)
  6. Upper limit (parameter 16#2016) → <diUpperLimit>
  7. Lower limit (parameter 16#2017) → <diLowerLimit>

Subsequently the conditions and error codes are listed below, which are displayed, if a condition is not satisfied.

Condition	<udiErrorID>	<uiErrorParameter>
<uiStepsPerTurn> ≥ 1	16#61400	16#2011
<uiStepsPerTurn> ≤ 10000	16#61400	16#2011
<diLowerLimit> ≤ <diUpperLimit>	16#61400	16#2016
(<diUpperLimit> – <diLowerLimit>) / <uiStepsPerTurn> ≤ 250	16#61400	16#2017
If <diSetPoint> < <diLowerLimit>: (<diUpperLimit> – <diSetPoint>) / <uiStepsPerTurn> ≤ 250	16#61400	16#2003
If <diSetPoint> > <diUpperLimit>: (<diSetPoint> – <diLowerLimit>) / <uiStepsPerTurn> ≤ 250	16#61400	16#2003

- Optionally at the end, the written values might be saved permanently. To do this, the input <xSaveSettings> has to be set to TRUE before the execution of the FB.
- In case of an error while writing a parameter, the subsequent parameters are not written anymore. In addition, no saving of the values is carried out, if the input <xSaveSettings> is set.

## 6 Description of the parameters

### 6.1 <xActive>

Block	FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx	
FBs is active or movement order or rather movement is active		
Data type	Bool	
Default	-	
Type	Output	
Description		
<b><u>FB MC Move PSx3xx</u></b>		
This output is set when		
<ul style="list-style-type: none"><li>the release for moving &lt;xExecute&gt; is set from FALSE to TRUE</li><li>the release for moving &lt;xExecute&gt; is already available and the target position changes, the bit “Drive is running” is set in the status word of the drive (e.g. when readjusting the drive)</li></ul>		
This output is reset when		
<ul style="list-style-type: none"><li>At the end of a run, the bit “Drive is running” is no longer set in the status of the drive and the time of the bus cycle parameter (also see 6.24 &lt;tTimeBusCycle&gt;) has expired</li><li>a communication error occurs</li></ul>		
<b><u>All other blocks</u></b>		
The bit is set as long as the respective function block is running. As soon as the process has been completed or an error has occurred, the bit is reset.		

### 6.2 <diActualPosition>

Block	FB_MC_Move_PSx3xx	
Actual value		
Data type	DInt	
Default	-	
Type	Output	
Description		
This value is an image of the actual position. If a communication error occurs, the value is set to 0.		

### 6.3 <xDeliveryState>

<b>Block</b>	<b>FB_MC_Parametrization_PSx3xx</b>
<b>Loading the factory settings (without saving)</b>	
<b>Data type</b>	Bool
<b>Default</b>	FALSE
<b>Type</b>	Input
<b>Description</b>	
However, the station name and IP address remain unaffected.	

### 6.4 <uiDirection>

<b>Block</b>	<b>FB_MC_PosParametrization_PSx3xx</b>
<b>Direction of rotation</b>	
<b>Data type</b>	UInt
<b>Default</b>	0
<b>Type</b>	Input
<b>Description</b>	
<p>This parameter corresponds to parameter number 16#202C "direction of rotation".  Direction in which the drive should rotate for larger values (when looking at the output shaft):  (0 → clockwise; 1 → counterclockwise)</p>	

### 6.5 <xDone>

<b>Block</b>	<b>FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx</b>
<b>Target position reached or FBs is done</b>	
<b>Data type</b>	Bool
<b>Default</b>	-
<b>Type</b>	Output
<b>Description</b>	
<p><b><u>FB MC Move PSx3xx</u></b>  This output is an image of the status bit "target position reached". In addition, the output is set to 0 for the duration of the parameter "bus cycle" (also see 6.24 &lt;timeBusCycle&gt;) after the start by &lt;xExecute&gt;. If a communication error occurs, it will be reset.</p>	
<p><b><u>All other blocks</u></b>  The bit is set as soon as the process has been successfully completed. It is reset at the start of the respective process.</p>	



## 6.6 <stDrive>

<b>Block</b>	FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx
<b>Data structure for communication</b>	
<b>Data type</b>	ST_DriveData_PSx3xx
<b>Type</b>	Input & Output
<b>Description</b>	
A global instance of this structure is required for each drive. This instance is transferred to each function block (FB) that acts on the operated drive.	

## 6.7 <xError>

<b>Baustein</b>	FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx
<b>FB execution error or drive error</b>	
<b>Data type</b>	Bool
<b>Default</b>	FALSE
<b>Type</b>	Output
<b>Description</b>	
<p><b><u>FB MC Move PSx3xx</u></b> The error bit is updated every cycle and is set, if an error occurs during execution of the FB. It can also be set, while the drive is active (e.g. drag error).</p> <p><b><u>FB MC Error PSx3xx</u></b> The output &lt;xError&gt; is updated every cycle as long as &lt;xExecute&gt; is set. If &lt;xExecute&gt; is reset, this output assumes the specified default value.</p> <p><b><u>All other blocks</u></b> The error bit is updated every cycle and is set, if an error occurred during an execution of the FB.</p>	

## 6.8 <sErrorDescription>

<b>Block</b>	FB_MC_Error_PSx3xx
<b>Error description as text output</b>	
<b>Data type</b>	String[254]
<b>Default</b>	""
<b>Type</b>	Output
<b>Description</b>	
An error description as text output	

## 6.9 <udiErrorID>

<b>Block</b>	FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx
<b>Error identifier (also see chapter 4 Error description (&lt;udiErrorID&gt;))</b>	
<b>Data type</b>	UDint
<b>Default</b>	0
<b>Type</b>	Output
<b>Description</b>	
The error identifier can also be set, while the drive is moving. (e.g. drag error)	
<b><u>FB MC Move PSx3xx</u></b>	
The error identifier is updated every cycle and is set, if an error occurs during execution of the FB. It can also be set, while the drive is active (e.g. drag error).	
<b><u>FB MC Error PSx3xx</u></b>	
The output <udiErrorID> is updated every cycle as long as <xExecute> is set. If <xExecute> is reset, this output assumes the specified default value.	
<b><u>All other blocks</u></b>	
The error identifier bit is updated every cycle and is set, if an error occurred during execution of the FB.	



### CAUTION

The outputs <xError> and <udiErrorID> of the function blocks are always updated - even if the <xExecute> input is not set, except for **MC\_Error\_PSx3xx**.

## 6.10 <uiErrorParameter>

<b>Block</b>	FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx
<b>Parameter number at which the error occurred</b>	
<b>Data type</b>	UInt
<b>Default</b>	0
<b>Type</b>	Output
<b>Description</b>	
If there was no error, the value 0 is returned.	

## 6.11 <xExecute>

Block	FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx	
Release of the drive		
Data type	Bool	
Default	FALSE	
Type	Input	
Description		
<b><u>FB MC Move PSx3xx</u></b> A setpoint is only approached when this input is set. This input acts directly on the enable bit (bit 4) in the control word. If the input remains set and the readjustment is active in the drive, the drive readjusts automatically. If the input is set and the setpoint is changed, the drive moves to it immediately. An edge is not required. If the input is reset during a movement, the drive stops.		
<b><u>All other blocks</u></b> With a rising edge, the process of the respective function block is executed. A rising edge must be generated again for a new process. If the bit is reset, the outputs assume the specified default values.		

## 6.12 <diLowerLimit>

Block	FB_MC_PosParametrization_PSx3xx	
Lower limit		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to parameter 16#2017 “lower limit”.		

## 6.13 <xManualRunToLargerValues>

Block	FB_MC_Move_PSx3xx	
Manual run to larger values		
Data type	Bool	
Default	FALSE	
Type	Input	
Description		
Manual run is used to increase the values up to the upper limit. The input <xExecute> must also be set.		



### CAUTION

When resetting the input <xManualRunToLargerValues> <xExecute> must also be reset, otherwise the drive will approach the target position <diTargetPosition>.

#### 6.14 <xManualRunToSmallerValues>

<b>Block</b>	<b>FB_MC_Move_PSx3xx</b>
<b>Manual run to smaller values</b>	
<b>Data type</b>	Bool
<b>Default</b>	FALSE
<b>Type</b>	Input
<b>Description</b>	
Manual run is used to decrease the values down to the lower limit. The input <xExecute> must also be set.	



### CAUTION

When resetting the input <xManualRunToSmallerValues>, <xExecute> must also be reset, otherwise the drive will approach the target position <diTargetPosition>.

#### 6.15 <sNetId>

<b>Block</b>	<b>FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx</b>
<b>Address of the EtherCAT master</b>	
<b>Data type</b>	T_AmsNetID
<b>Default</b>	10.250.53.73.2.1
<b>Type</b>	-
<b>Description</b>	
The address of the EtherCAT master is accessed internally in <b>FB_MC_ReadParameter_PSx3xx</b> and <b>FB_MC_WriteParameter_PSx3xx</b> .	

#### 6.16 <stParameter>

<b>Block</b>	<b>FB_MC_Parametrization_PSx3xx</b>
<b>Transfer of the parameter set</b>	
<b>Data type</b>	ST_Parameter_PSx3xx
<b>Default</b>	-
<b>Type</b>	Input
<b>Description</b>	
The parameter number is specified after the parameter name. The data type, a description and the range of values can be found in the instruction manual PSx-3xx-EC.	

#### 6.17 <uiParameterNumber>

Block	FB_MC_WriteParameter_PSx3xx, FB_MC_ReadParameter_PSx3xx	
Parameter number of the parameter to be read or written		
Data type	UInt	
Default	0	
Type	Input	
Description		
The number of the parameter to be written or read (e.g. 16#2003 for “actual value”).		

#### 6.18 <xSaveSettings>

Block	FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx	
Saving the parameter settings		
Data type	Bool	
Default	FALSE	
Type	Input	
Description		
This parameter corresponds to parameter 16#204F „delivery state. (function <Writing “1”>)		

#### 6.19 <diSetPoint>

Block	FB_MC_PosParametrization_PSx3xx	
Setpoint of the measuring system		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to parameter 16#2003 "actual value".		

## 6.20 <uiSlaveAddr>

<b>Block</b>	<b>FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx</b>
<b>Address of the EtherCAT-Slave (PSx3xx)</b>	
<b>Data type</b>	UInt
<b>Default</b>	1001
<b>Type</b>	-
<b>Description</b>	
The address of the EtherCAT slave is accessed internally in <b>FB_MC_ReadParameter_PSx3xx</b> and <b>FB_MC_WriteParameter_PSx3xx</b> .	

## 6.21 <uiStepsPerTurn>

<b>Block</b>	<b>FB_MC_PosParametrization_PSx3xx</b>
<b>Steps per rotation on the output shaft (resolution)</b>	
<b>Data type</b>	Int
<b>Default</b>	0
<b>Type</b>	Input
<b>Description</b>	
The number of steps per rotation <uiStepsPerTurn> directly results in the value of the parameter "position scaling denominator" (Parameter 16#2011). It is assumed that the value of "position scaling nominator" (Parameter 16#2010) is in the delivery state (400).	

## 6.22 <usiSubIndex>

<b>Block</b>	<b>FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx</b>
<b>Subindex of the parameters</b>	
<b>Data type</b>	USInt
<b>Default</b>	0
<b>Type</b>	Input
<b>Description</b>	
The value of the parameter <usiSubIndex> can be left at the default value. Other values are currently not planned.	

## 6.23 <diTargetPosition>

<b>Block</b>	<b>FB_MC_Move_PSx3xx</b>
<b>Target position to be approached</b>	
<b>Data type</b>	DInt
<b>Default</b>	0
<b>Type</b>	Input
<b>Description</b>	
If a new target position is set during a run, it is approached immediately. If <xExecute> is still set after the end of the run and the target position is changed, the drive moves to it immediately.	

### NOTICE

In order to approach the same target position, e.g. after a blocking error, the release of the drive <xExecute> must be reset and set again.

## 6.24 <tTimeBusCycle>

<b>Block</b>	<b>FB_MC_Move_PSx3xx</b>
<b>Cycle time of the Profinet bus cycle</b>	
<b>Data type</b>	Time
<b>Default</b>	40ms
<b>Type</b>	Input
<b>Description</b>	
<p>With long cycle times on the EtherCAT-Bus, it can happen that the PLC does not receive a change in the bit "drive running". This is the case when the time for the run command is shorter than the bus cycle.</p> <p>To counteract this, the parameter &lt;tTimeBusCycle&gt; has been implemented. 40 ms are set as the initial value for a bus cycle. For this duration, the output &lt;xActive&gt; is always set and the output &lt;xDone&gt; is reset after a movement order. After that, these outputs assume the corresponding state depending on the feedback of the status word (bit "drive running" and bit "target position reached").</p> <p>In case of longer cycle times of the bus cycle, it is advisable to set the actual duration of the cycle multiplied by 2 instead of the initial value.</p> <p>If a shorter bus cycle time can be kept for sure, the time for positioning is very short and the higher-level sequence is to be continued quickly after positioning is complete, the value for &lt;tTimeBusCycle&gt; can also be reduced.</p>	

#### 6.25 <diUpperLimit>

<b>Block</b>	<b>FB_MC_PosParametrization_PSx3xx</b>
<b>Upper limit</b>	
<b>Data type</b>	DInt
<b>Default</b>	0
<b>Type</b>	Input
<b>Description</b>	
This parameter corresponds to parameter 16#2016 "upper limit".	

#### 6.26 <diValue>

<b>Block</b>	<b>FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx</b>
<b>value to be written or read value of a parameter</b>	
<b>Data type</b>	DInt
<b>Default</b>	0
<b>Type</b>	Input for <b>FB_MC_WriteParameter_PSx3xx</b> Output for <b>FB_MC_ReadParameter_PSx3xx</b>
<b>Description</b>	
<b>FB_MC_ReadParameter:</b> If xDone is set this value represents the value of a numeric parameter.	
<b>FB_MC_WriteParameter:</b> A rising edge on xExecute will cause this value to be written to the device.	

#### 6.27 <sValue>

<b>Block</b>	<b>FB_MC_ReadParameter_PSx3xx</b>
<b>Value of a parameter to be read</b>	
<b>Data type</b>	String[31]
<b>Default</b>	0
<b>Type</b>	Output
<b>Description</b>	
If xDone is set this value represents the value of the read string parameter (currently only Parameters "Manufacturer Software Version" (16#100A) and "Device Model" (16#204D)).	



## 7 Application of the function blocks

The function blocks are programmed in TwinCAT version v3.1.4024. Upgrades to higher versions can be done without any problems.

There are two ways of using the function blocks:

- As Project or
- As embedded library

### 7.1 Project

Open and upgrade the project if necessary.

### 7.2 Library

First, the library must be installed in the library repository.

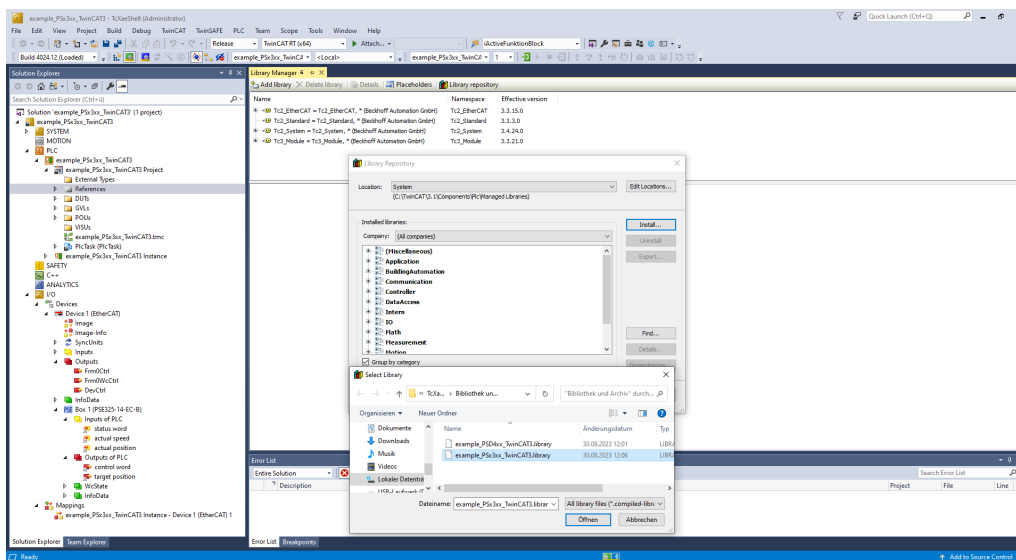


Figure 13: Installing the library in the library repository

After that, the library can be selected and added.

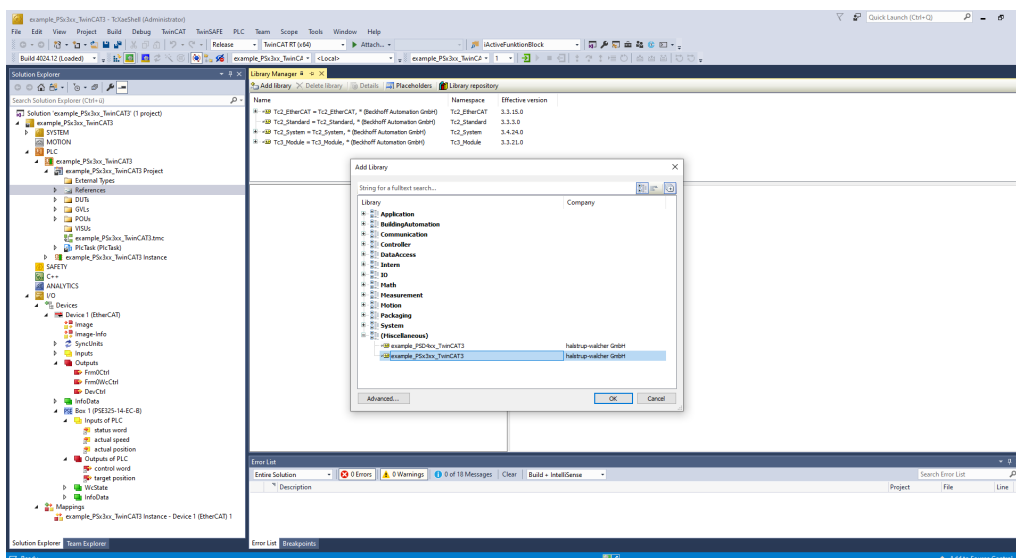


Figure 14: Inserting the library

Finally, the function blocks and sample programs can be selected.

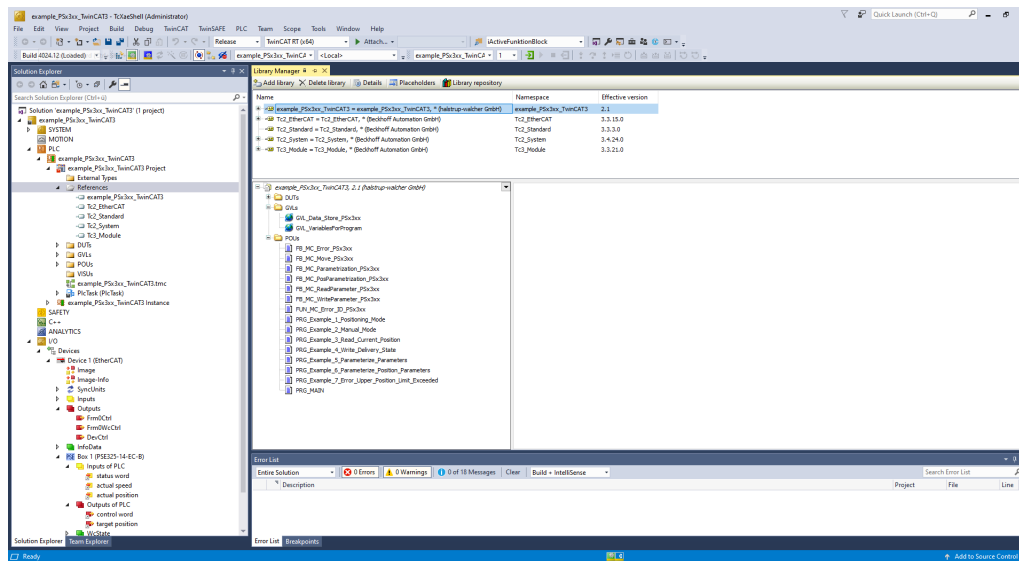


Figure 15: Files included in the library

The following function from the library must always be copied into the user's project (irrespective of the **FB\_MC\_...\_PSx3xx** function blocks actually used and the number of drives):

- **FB\_MC\_Error\_ID\_PSi3xx**

This function is used for error detection with the drive.

In addition, the required function blocks **FB\_MC\_...\_PSx3xx** must be copied to the user project. All or a selection of the following blocks can be used:

- **FB\_MC\_Move\_PSi3xx**
- **FB\_MC\_Error\_PSi3xx**
- **FB\_MC\_ReadParameter\_PSi3xx**
- **FB\_MC\_WriteParameter\_PSi3xx**
- **FB\_MC\_Parametrization\_PSi3xx**
- **FB\_MC\_PosParametrization\_PSi3xx**

Additionally, the global variable list <GVL\_Data\_Store\_PSi3xx> can be adopted as a template for the connection to the function blocks. However, this requires all data types (DUTs\_halstrup\_walcher)

For the inclusion of the sample programs the global variable list <GVL\_VariablesForProgram> should be adopted.

### 7.3 Interlock between the function blocks

The blocks are partially locked against each other. This ensures that two accesses out of different function blocks can't be executed at the same time.

The restrictions are:

- If the input <xExecute> of **FB\_MC\_Move\_PSi3xx** is set, the function blocks **FB\_MC\_Parametrization\_PSi3xx** and **FB\_MC\_PosParametrization\_PSi3xx** cannot be activated (<udiErrorID>: 16#x1000).
- On the other hand, it is possible to activate **FB\_MC\_ReadParameter\_PSi3xx** or **FB\_MC\_WriteParameter\_PSi3xx** during a movement (e.g. to read out the current torque or to change the target speed during the actual movement).
- If the input <xExecute> of **FB\_MC\_Parametrization\_PSi3xx** or **FB\_MC\_PosParametrization\_PSi3xx** is set, the function block **FB\_MC\_Move\_PSi3xx** cannot be activated (<udiErrorID>: 16#01000).

- Only one of the function blocks **FB\_MC\_ReadParameter\_PSx3xx**, **FB\_MC\_WriteParameter\_PSx3xx**, **FB\_MC\_Parametrization\_PSx3xx** or **FB\_MC\_PosParametrization\_PSx3xx** can be active at the same time. If the input <xExecute> of a function block is set and the input <xExecute> of another function block is set, then there is the <errorID> 16#x1000.
- The function block **FB\_MC\_Error\_PSx3xx** can always be activated.

#### 7.4 Several PSx3xx in a project

There is only one PSx3xx in the example projects. If several drives are used, the project must be adjusted accordingly.

For example for three PSx3xx:

1. Initialize instance data blocks
  - Three instance data blocks of fb's **FB\_MC\_Move\_PSx3xx**, e.g.
    - fbMC\_Move\_PSx3xx\_1
    - fbMC\_Move\_PSx3xx\_2
    - fbMC\_Move\_PSx3xx\_3
  - Three instance data blocks of fb's **FB\_MC\_Error\_PSx3xx**, e.g.
    - fbMC\_Error\_PSx3xx\_1
    - fbMC\_Error\_PSx3xx\_2
    - fbMC\_Error\_PSx3xx\_3
  - any other required instance data blocks
2. Afterwards, the global variable list <GVL\_Data\_Store\_PSx3xx> is adapted. Three variables of the <ST\_Variables\_PSx3xx> type are created there, e.g. with the following designations:
  - „PSE\_1“
  - „PSE\_2“
  - „PSE\_3“
3. Finally, the variables from <GVL\_Data\_Store\_PSx3xx> must be assigned to the function blocks (see chapter 5 Description of the function blocks)

## 8 Example programs

Example programs are provided to demonstrate usage of the function blocks. The functions are programmed in ST and the respective function must be inserted into the main program in order to be able to call the function.

```

15 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
16
17 (* MAIN *)
18
19 //call of function
20 PRG_Example_7_Error_Upper_Position_Limit_Exceeded();
21
22 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

*Figure 16: Calling function in the main program*

### Similarity of all example programs

- for all programs, the required instances are defined and called at the beginning.
- For all programs the variable <xStartProgram> must be set to TRUE to start execution the program.
- For each sample program, the first step after starting the program is to set all relevant variables to FALSE for initialization.
- In each example program, the first step after starting the program is to set all relevant variables to FALSE for the initialization.
- The program can be stopped with <xStopProgram>.

### 8.1 Example 1: positioning mode

The positioning mode of the **FB\_MC\_Move\_PSx3xx** is presented in this example program. In this example, the drive moves an endless loop between the values 20000 and 10000.

#### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The target position of 20000 and the movement command are set. If the actual value is at $20000 \pm 2$ (see parameter 16#2006 "Positioning window") and the travel was successfully completed, the movement command is reset and the state machine is increased by one.
2	The target position of 10000 and the movement command are set. If the actual value is at $10000 \pm 2$ (see parameter 16#2006 "Positioning window") and the movement has been successfully completed, the movement command is reset and the State Machine is set to 1.

## 8.2 Example 2: manual mode

This sample program presents the manual mode of the **FB\_MC\_Move\_PSx3xx**. In this example, the drive moves to a start position with the positioning mode and then with the manual mode.

### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The start position of 20000 and the movement command are set. If the current value is at $20000 \pm 2$ (see parameter 16#2006 "Positioning window") and the movement was successfully completed, then the movement command is reset and the State Machine is increased by one.
2	With manual mode, movement is performed over the target of 30000. Only when the actual value corresponds to 30000, the movement command for manual move is reset and the state machine is set to 100.

## 8.3 Example 3: read actual value

This sample program introduces the **FB\_MC\_ReadParameter\_PSx3xx**. In this example, the drive moves to a target position and the parameter "actual value" is read.

### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The start position of 20000 and the movement command are set. If the current value is at $20000 \pm 2$ (see parameter 16#2006 "positioning window") and the movement was successfully completed, then the movement command is reset and the State Machine is increased by one.
2	The parameter "actual value" is read. If the value is $20000 \pm 2$ (see parameter 16#2006 "Positioning window"), then the read command is reset and the state machine is set to 100.

#### 8.4 Example 4: writing the delivery state

This sample program introduces the **FB\_MC\_WriteParameter\_PSx3xx**. The "Delivery state" parameter is written in this example and the drive then moves to its range center.

##### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The start position of 20000 and the movement command are set. If the current value is at $20000 \pm 2$ (see parameter 16#2006 "positioning window") and the travel was successfully completed, then the movement command is reset and the State Machine is increased by one.
2	The value of the "Delivery state" parameter is written to -5. The drive is reset to its delivery state, the parameters are set to the default value and additionally a positioning to the measuring range center (position 51200) takes place. If the write command was executed successfully, the write command is reset and the state machine is increased by one. (The drive still moves to the center of the measuring range).

#### 8.5 Example 5: Parameterize parameters

This sample program presents the **FB\_MC\_Parametrization\_PSx3xx**. The parameters "loop length" and "target speed" are parameterized in this example and then a positioning run is performed.

##### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The start position of 20000 and the movement command are set. If the current value is at $20000 \pm 2$ (see parameter 16#2006 "Positioning window") and the travel was successfully completed, then the movement command is reset and the State Machine is increased by one.
2	The "Loop length" and "target speed" parameters are parameterized. The <xEnable> of the parameters must be set. If the parameterization command was executed successfully, the parameterization command is reset and the state machine is increased by one.
3	The target position of 10000 and the movement command are set. During this positioning travel, no loop travel is performed and travel is performed at 20 rpm. If the current value is at $10000 \pm 2$ (see parameter 44 "Positioning window") and the travel was successfully completed, then the movement command is reset and the state machine is set to 100.

## 8.6 Example 6: Parameterize position parameters

In this sample program the **FB\_MC\_PosParametrization\_PSx3xx** is presented. Parameters are set and the "current actual position" parameter is read.

### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The start position of 20000 and the movement command are set. If the current value is at $20000 \pm 2$ (see parameter 16#2006 "Positioning window") and the travel was successfully completed, then the movement command is reset and the State Machine is increased by one.
2	All position parameters are parameterized. The parameter "Actual position" is set to 0, the parameter "Upper end limit" is set to 10000 and the parameter "Lower end limit" is set to -10000. The rest of the parameters are not changed and the parameter values are not to be saved. If the parameterization command of the position parameters was executed successfully, then the parameterization command of the position parameters is reset and the State Machine is increased by one.
3	The parameter "current position" is read. If the value is $0 \pm 2$ , then the read command is reset and the state machine is set to 100.

## 8.7 Example 7: Error upper end limit reached

In this sample program the **FB\_MC\_Error\_PSx3xx** is presented. The drive is positioned to the upper end limit and the error message is read out.

### Sequence of the program

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always useful to set all relevant variables to the initial value at the beginning of a program).
1	The start position of 20000 and the movement command are set. If the current value is at $20000 \pm 2$ (see parameter 16#2006 "Positioning window") and the travel was successfully completed, then the movement command is reset and the State Machine is increased by one.
2	The parameter "Upper end limit" is written to the value 25000. If the write command was executed successfully, then the write command is reset and the state machine is incremented by one.
3	The function block <b>FB_MC_Error_PSx3xx</b> is activated to detect errors.
4	With manual mode the upper end limit is approached. Only when the actual value corresponds to 25000, the error description is displayed as a string. Then the move command for manual mode is reset and the index is set to 100. (It would also be possible to receive this error from FB_MC_Move_PSx3xx (<udiErrorID>). The <udiErrorID> would then be 0x100B0.)

## List of Figures

Figure 1: Address of EtherCAT-Master .....	6
Figure 2: Address of EtherCAT-Slaves .....	7
Figure 3: Allocation of the variables from <GVL_Data_Store_PSx3xx> to FB_MC_Move_PSx3xx.....	8
Figure 4: <GVL_Data_Store_PSx3xx> with the variables for a drive .....	8
Figure 5: Allocation of control word with a variable from <GVL_Data_Store_PSx3xx>.....	11
Figure 6: Allocation of variables from DB <Data_Store> to FB_MC_Move_PSx3xx .....	11
Figure 7: Allocation of variables from DB <GVL_Data_Store_PSx3xx> to FB_MC_Error_PSx3xx .....	12
Figure 8: Allocation of variables from <GVL_Data_Store_PSx3xx> to FB_MC_ReadParameter_PSx3xx .....	12
Figure 9: Allocation of variables from <GVL_Data_Store_PSx3xx> to FB_MC_WriteParameter_PSx3xx .....	13
Figure 10: Allocation of variables from <GVL_Data_Store_PSx3xx> to FB_MC_Parametrization_PSx3xx .....	13
Figure 11: Parameter „loop length“ with the value of 0 .....	13
Figure 12: Allocation of variables from <GVL_Data_Store_PSx3xx> to FB_MC_PosParametrization_PSx3xx .....	14
Figure 13: Installing the library in the library repository.....	25
Figure 14: Inserting the library.....	25
Figure 15: Files included in the library .....	26
Figure 16: Calling function in the main program.....	28