

## Purpose of instruction manual

This instruction manual describes the function blocks for the PSx3xx EIP (with EtherNet/IP 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 | TS, FS

**7100.007064A Version 2.1 Instruction Manual Function Blocks PSx3xx EIP**

## 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> .....	6
2.3	<ST_Variables_PSx3xx> .....	7
3	Error description (<udiErrorID>) .....	8
4	Description of the function blocks .....	10
4.1	FB_MC_Move_PSx3xx .....	10
4.2	FB_MC_Error_PSx3xx.....	10
4.3	FC_MC_Error_ID_PSx3xx .....	11
4.4	FB_MC_ReadParameter_PSx3xx.....	11
4.5	FB_MC_WriteParameter_PSx3xx .....	11
4.6	FB_MC_Parametrization_PSx3xx.....	11
4.7	FB_MC_PosParametrization_PSx3xx.....	12
5	Description of the parameters .....	14
5.1	<xActive> .....	14
5.2	<diActualPosition> .....	14
5.3	<xDeliveryState> .....	15
5.4	<uiDirection> .....	15
5.5	<xDone> .....	15
5.6	<stDrive> .....	16
5.7	<xError>.....	16
5.8	<sErrorDescription> .....	16
5.9	<udiErrorID> .....	17
5.10	<uiErrorParameter> .....	17
5.11	<xExecute> .....	18
5.12	<diLowerLimit> .....	18
5.13	<xManualRunToLargerValues> .....	18
5.14	<xManualRunToSmallerValues> .....	19
5.15	<stParameter>.....	19
5.16	<uiParameterNumber> .....	19
5.17	<xSaveSettings> .....	20
5.18	<diSetPoint> .....	20
5.19	<uiStepsPerTurn>.....	20
5.20	<usiSubIndex> .....	20
5.21	<diTargetPosition> .....	21
5.22	<tTimeBusCycle> .....	21

5.23	<diUpperLimit>.....	22
5.24	<diValue>.....	22
5.25	<sValue>.....	22
6	Application of the function blocks.....	23
6.1	Project.....	23
6.2	Library/AOI .....	24
6.3	Interlock between the function blocks .....	25
7	Example programs .....	26
7.1	Example 1: positioning mode .....	26
7.2	Example 2: manual mode .....	27
7.3	Example 3: read actual value .....	27
7.4	Example 4: writing the delivery state .....	28
7.5	Example 5: Parameterize parameters.....	28
7.6	Example 6: Parameterize position parameters .....	29
7.7	Example 7: Error upper end limit reached .....	29
	List of Figures .....	30

# 1 Safety precautions




## 1.1 Appropriate use

The positioning systems PSx3xx EIP 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 EIP 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>

A data structure is used to store some of the drives' 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. Here, it will be prevented, that two FBs can access the parameter interface of a single drive. Furthermore, in this data structure the addresses for the input and output data of the corresponding drive has to be deposited.

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)
<iActiveFunktionBlock>	Int	Function blocks	Active function block
<xCommunicationError>	Bool	Function blocks	Communication error to IO-Device
<stPrivate>	ST_Private	Function blocks	Data structure for internal use

### 2.2 <ST\_FunctionBlocks\_PSx3xx>

For each drive, the variables for the assignment to the function blocks are stored in a data structure. This structure has several substructures, which are not explained further in the documentation. A global instance of this structure is required for each drive. 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 data structure comprises of the structures <ST\_FunctionBlocks\_PSx3xx> and <ST\_DriveData\_PSx3xx>. All inputs and outputs of the function blocks as well as those of the drive can be connected to this structure.

If multiple drives are used in a project, it is recommended to create an array of this structure with an entry for each drive.

### 3 Error description (<udiErrorID>)

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

<udiErrorID> (hex)	Description
<b>16#F0000 (Mask)</b>	<b>Function block</b>
16#0xxxx	No error code
16#1xxxx	Error in FB_MC_Move_Ps3xx
16#2xxxx	Error in FB_MC_Error_Ps3xx
16#3xxxx	Error in FB_MC_ReadParameter_Ps3xx
16#4xxxx	Error in FB_MC_WriteParameter_Ps3xx
16#5xxxx	Error in FB_MC_Parametrization_Ps3xx
16#6xxxx	Error in FB_MC_PosParametrization_Ps3xx
<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 (**FB\_MC\_Move\_PSx3xx**) with incorrect target value →  
<udiErrorID> = 16#10080
- Writing a parameter (**FB\_MC\_WriteParameter\_PSx3xx**) with invalid parameter number →  
<udiErrorID> = 16#40200

## 4 Description of the function blocks

A detailed description of all parameters is in chapter 6 Description of the parameters.

### 4.1 FB\_MC\_Move\_Ps3xx

This function block is used to move the drive.

```
FB_MC_Move_Ps3xx(fbMC_Move_PSE1,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stInput_Move_Ps3xx.xExecute,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stInput_Move_Ps3xx.diTargetPosition,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stInput_Move_Ps3xx.xManualRunToLargerValues,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stInput_Move_Ps3xx.xManualRunToSmallerValues,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stInput_Move_Ps3xx.tTimeBusCycle,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stOutput_Move_Ps3xx.xActive,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stOutput_Move_Ps3xx.xDone,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stOutput_Move_Ps3xx.diCurrentPosition,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stOutput_Move_Ps3xx.xError,
    PSE_1.stFunctionBlocks_Ps3xx.stMove_Ps3xx.stOutput_Move_Ps3xx.udiErrorID,
    PSE_1.stDriveDate_Ps3xx);
```

Figure 1 Connection of variables from ST\_Variables\_Ps3xx to the inputs and outputs of FB\_MC\_Move\_Ps3xx

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 Ps3xx is a device with STO, then the error description in brackets must be taken into account! (<udiErrorID >: 16#xxx2x und 16#xxx5x)

### 4.2 FB\_MC\_Error\_Ps3xx

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

```
FB_MC_Error_Ps3xx(FB_MC_Error_PSE1,
    PSE_1.stFunctionBlocks_Ps3xx.stError_Ps3xx.stInput_Error_Ps3xx.xExecute,
    PSE_1.stFunctionBlocks_Ps3xx.stError_Ps3xx.stOutput_Error_Ps3xx.xError,
    PSE_1.stFunctionBlocks_Ps3xx.stError_Ps3xx.stOutput_Error_Ps3xx.udiErrorID,
    PSE_1.stFunctionBlocks_Ps3xx.stError_Ps3xx.stOutput_Error_Ps3xx.sErrorDescription,
    PSE_1.stDriveDate_Ps3xx);
```

Figure 2 Connection of variables from ST\_Variables\_Ps3xx to the inputs and outputs of FB\_MC\_Error\_Ps3xx

### 4.3 FC\_MC\_Error\_ID\_Ps3xx

This function is used internally as a sub function of the FBs **FB\_MC\_Move\_Ps3xx**, **FB\_MC\_ReadParameter\_Ps3xx**, **FB\_MC\_WriteParameter\_Ps3xx** and **FB\_MC\_Error\_Ps3xx**. It just has to be copied from the library into the application program. The setting of the inputs is done completely internally.

### 4.4 FB\_MC\_ReadParameter\_Ps3xx

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

```
FB_MC_ReadParameter_Ps3xx(fbMC_ReadParameter_PSE1,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stInput_ReadParameter_Ps3xx.xExecute,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stInput_ReadParameter_Ps3xx.uiParameterNumber,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stInput_ReadParameter_Ps3xx.usiSubindex,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stOutput_ReadParameter_Ps3xx.xActive,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stOutput_ReadParameter_Ps3xx.xDone,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stOutput_ReadParameter_Ps3xx.xError,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stOutput_ReadParameter_Ps3xx.udiErrorID,
    PSE_1.stFunctionBlocks_Ps3xx.stReadParameter_Ps3xx.stOutput_ReadParameter_Ps3xx.diValue,
    PSE_1.stDriveDate_Ps3xx);
```

Figure 3 Connection of variables from ST\_ Variables\_Ps3xx to the inputs and outputs of FB\_MC\_ReadParameter\_Ps3xx

### 4.5 FB\_MC\_WriteParameter\_Ps3xx

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

```
FB_MC_WriteParameter_Ps3xx(fbMC_WriteParameter_PSE1,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stInput_WriteParameter_Ps3xx.xExecute,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stInput_WriteParameter_Ps3xx.uiParameterNumber,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stInput_WriteParameter_Ps3xx.diValue,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stOutput_WriteParameter_Ps3xx.xActive,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stOutput_WriteParameter_Ps3xx.xDone,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stOutput_WriteParameter_Ps3xx.xError,
    PSE_1.stFunctionBlocks_Ps3xx.stWriteParameter_Ps3xx.stOutput_WriteParameter_Ps3xx.udiErrorID,
    PSE_1.stDriveDate_Ps3xx);
```

Figure 4 Connection of variables from ST\_ Variables\_Ps3xx to the inputs and outputs of FB\_MC\_WriteParameter\_Ps3xx

### 4.6 FB\_MC\_Parametrization\_Ps3xx

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\_Ps3xx>.

```
FB_MC_Parametrization_Ps3xx(fbMC_Parametrization_PSE1,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stInput_Parametrization_Ps3xx.xExecute,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stInput_Parametrization_Ps3xx.xDeliveryState,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stInput_Parametrization_Ps3xx.xParameter,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stInput_Parametrization_Ps3xx.xSaveSettings,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stOutput_Parametrization_Ps3xx.xActive,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stOutput_Parametrization_Ps3xx.xDone,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stOutput_Parametrization_Ps3xx.xError,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stOutput_Parametrization_Ps3xx.udiErrorID,
    PSE_1.stFunctionBlocks_Ps3xx.stParametrization_Ps3xx.stOutput_Parametrization_Ps3xx.uiErrorParameter,
    PSE_1.stDriveDate_Ps3xx);
```

Figure 5 Connection of variables from ST\_ Variables\_Ps3xx to the inputs and outputs of FB\_MC\_Parametrization\_Ps3xx

**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

```
PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.stParameter.stTargetSpeed_52.xEnable := 1;
PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.stParameter.stTargetSpeed_52.uiValue := 20;
```

Figure 6 Parameter "loop length" is assigned value 20

- 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 26, 28, 30, 10, 32, 34, ...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.

## 4.7 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.

```
FB_MC_PosParametrization_PSx3xx(fbMC_PosParametrization_PSE1,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.xExecute,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.uiDirection,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.uiStepsPerTurn,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.diLowerLimit,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.diUpperLimit,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.diSetPoint,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stInput_PosParametrization_PSx3xx.xSaveSettings,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.xActive,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.xDone,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.xError,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.udiErrorID,
    PSE_1.stFunctionBlocks_PSx3xx.stPosParametrization_PSx3xx.stOutput_PosParametrization_PSx3xx.uiErrorParameter,
    PSE_1.stDriveDate_PSx3xx);
```

Figure 7 Connection of variables from ST\_ Variables\_PSx3xx to the inputs and outputs of 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 26) → <uiDirection>
  2. Position scaling numerator (Parameter 28) → 400
  3. Position scaling denominator (Parameter 30) → <uiStepsPerTurn>
  4. Actual value (Parameter 10) → <diSetPoint>
  5. If (<diSetPoint> > <diUpperLimit>):  
Upper mapping end (Parameter 34) = <diSetPoint> + (3 x <uiStepsPerTurn>)  
Else:  
Upper mapping end (parameter 34) = <diUpperLimit> + (3 x <uiStepsPerTurn>)
  6. Upper limit (parameter 36) → <diUpperLimit>
  7. Lower limit (parameter 38) → <diLowerLimit>

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

Condition	<udiErrorID>	<uiErrorParameter>
$\langle \text{uiStepsPerTurn} \rangle \geq 1$	16#61400	30
$\langle \text{uiStepsPerTurn} \rangle \leq 10000$	16#61400	30
$\langle \text{diLowerLimit} \rangle \leq \langle \text{diUpperLimit} \rangle$	16#61400	36
$(\langle \text{diUpperLimit} \rangle - \langle \text{diLowerLimit} \rangle) / \langle \text{uiStepsPerTurn} \rangle \leq 250$	16#61400	38
If $\langle \text{diSetPoint} \rangle < \langle \text{diLowerLimit} \rangle$ : $(\langle \text{diUpperLimit} \rangle - \langle \text{diSetPoint} \rangle) / \langle \text{uiStepsPerTurn} \rangle \leq 250$	16#61400	10
If $\langle \text{diSetPoint} \rangle > \langle \text{diUpperLimit} \rangle$ : $(\langle \text{diSetPoint} \rangle - \langle \text{diLowerLimit} \rangle) / \langle \text{uiStepsPerTurn} \rangle \leq 250$	16#61400	10

- 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.

## 5 Description of the parameters

### 5.1 <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 5.22 &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.		

### 5.2 <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.		

### 5.3 <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.	

### 5.4 <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 26 "direction of rotation".</p> <p>Direction in which the drive should rotate for larger values (when looking at the output shaft):</p> <p>(0 → clockwise; 1 → counterclockwise)</p>	

### 5.5 <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></p> <p>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 5.22 &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></p> <p>The bit is set as soon as the process has been successfully completed. It is reset at the start of the respective process.</p>	

## 5.6 <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.	

## 5.7 <xError>

<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>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>	

## 5.8 <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	



## 5.9 <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 3 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**.

## 5.10 <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.	

### 5.11 <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.		

### 5.12 <diLowerLimit>

Block	FB_MC_PosParametrization_PSx3xx	
Lower limit		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to parameter 38 “lower limit”.		

### 5.13 <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>.

#### 5.14 <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>.

#### 5.15 <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-EIP.	

#### 5.16 <uiParameterNumber>

<b>Block</b>	<b>FB_MC_WriteParameter_PSx3xx, FB_MC_ReadParameter_PSx3xx</b>
<b>Parameter number of the parameter to be read or written</b>	
<b>Data type</b>	UInt
<b>Default</b>	0
<b>Type</b>	Input
<b>Description</b>	
A read or write process is started with a rising edge. A rising edge must be generated again for a new process. If the bit is reset, the outputs assume the specified initial values.	

### 5.17 <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 113 „delivery state. (function <Writing “1”>)		

### 5.18 <diSetPoint>

Block	FB_MC_PosParametrization_PSx3xx	
Setpoint of the measuring system		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to parameter 10 “actual value”.		

### 5.19 <uiStepsPerTurn>

Block	FB_MC_PosParametrization_PSx3xx	
Steps per rotation on the output shaft (resolution)		
Data type	Int	
Default	0	
Type	Input	
Description		
The number of steps per rotation <uiStepsPerTurn> directly results in the value of the parameter “position scaling denominator” (Parameter 30). It is assumed that the value of “position scaling nominator” (Parameter 28) is in the delivery state (400).		

### 5.20 <usiSubIndex>

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

## 5.21 <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

## 5.22 <tTimeBusCycle>

<b>Block</b>	<b>FB_MC_Move_PSx3xx</b>
<b>Cycle time of the EtherNet/IP 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 EtherNet/IP-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>	

### 5.23 <diUpperLimit>

Block	FB_MC_PosParametrization_PSx3xx	
Upper limit		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to parameter 36 “upper limit”.		

### 5.24 <diValue>

Block	FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx	
value to be written or read value of a parameter		
Data type	DInt	
Default	0	
Type	Input for <b>FB_MC_WriteParameter_PSx3xx</b>  Output for <b>FB_MC_ReadParameter_PSx3xx</b>	
Description		
<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.		

### 5.25 <sValue>

Block	FB_MC_ReadParameter_PSx3xx	
Value of a parameter to be read		
Data type	String[31]	
Default	0	
Type	Output	
Description		
If xDone is set this value represents the value of the read string parameter (currently only Parameter “Device model” (23)).		

## 6 Application of the function blocks

The function blocks are programmed in Studio5000 v33. Upgrades to higher versions can be done without any problems.

There are two ways of using the function blocks:

- As Project or
- As Add On Instruction (AOI)

### 6.1 Project

The example project ControlLogix\_Ps3xx\_2\_2\_AOI.L5K is already configured to contain all function blocks, a drive and some example programs.

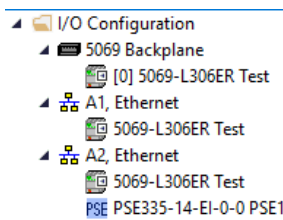


Figure 9 Drive

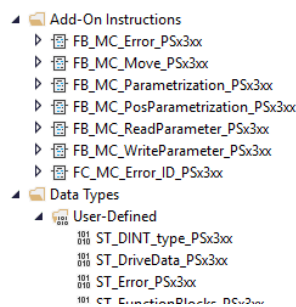


Figure 8 Function Blocks and UDTs

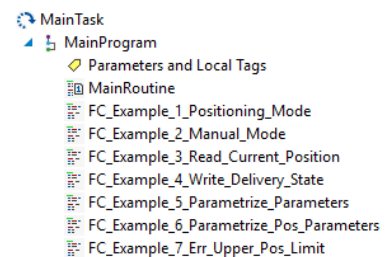


Figure 10 Example programs

When switching to another PS3xx variant, the currently configured drive has to be replaced. "PSE1" should be chosen as the new name, otherwise the assignments in Communication\_Routine will have to be updated accordingly.

To switch between different example programs, comment out the calls that should be inactive.

```

10 //call example program
11 //JSR(FC_Example_1_Positioning_Mode,0);
12 //JSR(FC_Example_2_Manual_Mode,0);
13 //JSR(FC_Example_3_Read_Current_Position,0);
14 //JSR(FC_Example_4_Write_Delivery_State,0);
15 //JSR(FC_Example_5_Parametrize_Parameters, 0);
16 //JSR(FC_Example_6_Parametrize_Pos_Parameters, 0);
17 JSR(FC_Example_7_Err_Upper_Pos_Limit, 0);

```

Figure 11 Selection of example program

To start execution of the selected program, set the variable xStartProgram to 1.

Via the view „Monitor Tags“ in MainProgram, all inputs of the function blocks can be assigned and the inputs read out. This way the functionality of the function blocks can be explored online on the PLC.

▲ PSE_1	Local	{...}
▶ PSE_1.stDriveDate_P5x3xx		{...}
▲ PSE_1.stFunctionBlocks_P5x3xx		{...}
▲ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx		{...}
▲ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx		{...}
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx.execute		1
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx.uiParameterNumber		10
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx.usiSubindex		0
▲ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx		{...}
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.Active		0
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.Done		1
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.Error		0
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.udiErrorID		0
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.dIValue		25000
▶ PSE_1.stFunctionBlocks_P5x3xx.stWriteParameter_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stError_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stMove_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stParametrization_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stPosParametrization_P5x3xx		{...}

Figure 12 Read out of parameter 10 (actual value) via the FB\_MC\_ReadParameter\_P5x3xx function block

## 6.2 Library/AOI

Alternatively, the function blocks can be imported into an existing project. To achieve this, import the File AOI\_PSx3xx.L5X. To simplify usage of the function block, it is advisable to also import the UDT "ST\_Variables\_PSx3xx" contained in AOI\_PSx3xx\_Datatypes.L5X.

For every drive, a new tag of type "ST\_DriveData\_PSx3xx" has to be created and the members of "ST\_DriveData\_PSx3xx" have to be connected to the appropriate members of the process data of the drive.

```
//copy input data from drive into struct
PSE_1.stDriveDate_PSx3xx.xCommunicationError := PSE1:I.ConnectionFaulted;
PSE_1.stDriveDate_PSx3xx.stPrivate.stInputModule.luiStatusWord := PSE1:I.status_word;
PSE_1.stDriveDate_PSx3xx.stPrivate.stInputModule.liCurrentSpeed := PSE1:I.actual_speed;
PSE_1.stDriveDate_PSx3xx.stPrivate.stInputModule.lidiCurrentPosition := PSE1:I.actual_position;
PSE_1.stDriveDate_PSx3xx.stPrivate.stInputModule.IuiPKE := PSE1:I.Parameter_ID;
PSE_1.stDriveDate_PSx3xx.stPrivate.stInputModule.IuiIND := PSE1:I.Subindex;
PSE_1.stDriveDate_PSx3xx.stPrivate.stInputModule.IudiPWE := PSE1:I.Parameter_value;
//copy output data from struct to output
PSE1:O.control_word := PSE_1.stDriveDate_PSx3xx.stPrivate.stOutputModule.OuiControlWord;
PSE1:O.target_position := PSE_1.stDriveDate_PSx3xx.stPrivate.stOutputModule.OdiTargetPosition;
PSE1:O.Parameter_ID := PSE_1.stDriveDate_PSx3xx.stPrivate.stOutputModule.OuiPKE;
PSE1:O.Subindex:= PSE_1.stDriveDate_PSx3xx.stPrivate.stOutputModule.OuiIND;
PSE1:O.Parameter_value:= PSE_1.stDriveDate_PSx3xx.stPrivate.stOutputModule.OudiPWE;
```

Figure 13 Connection of drive process data to the PLC variables



This instance of "ST\_DriveData\_PSD4xx" has to be connected to the *StDrive* InOut connectors of each function block. It is advisable to connect the other members of "ST\_DriveData\_PSD4xx" to the other connectors.

Additionally instances of the function blocks have to be created for each drive. However, it is not necessary to create an instance of a function block if its functionality isn't needed.

### 6.3 Interlock between the function blocks

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

Es gelten die folgenden Regeln:

There are following rules:

- If the input <xExecute> of **FB\_MC\_Move\_PSx3xx** is set, the function blocks **FB\_MC\_Parametrization\_PSx3xx** und **FB\_MC\_PosParametrization\_PSx3xx** cannot be activated (<udiErrorID>: 16#x1000).
- On the other hand, it is possible to activate **FB\_MC\_ReadParameter\_PSx3xx** or **FB\_MC\_WriteParameter\_PSx3xx** 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\_PSx3xx** or **FB\_MC\_PosParametrization\_PSx3xx** is set, the function block **FB\_MC\_Move\_PSx3xx** 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 <udiErrorID> 16#x1000.
- The function block **FB\_MC\_Error\_PSx3xx** can always be activated.

## 7 Example programs

Example programs are programmed to demonstrate, how the function blocks are to be used. The functions are programmed in SCL and the respective function must be inserted into the main program in order to be able to use the function (see Figure 11).

### 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 for starting 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 ended with <xStopProgram>.

### 7.1 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 40 "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 40 "Positioning window") and the travel has been carried out successfully, the movement command is reset and the State Machine is set to 1.

## 7.2 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 40 "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, travel 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.

## 7.3 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 40 "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 40 "Positioning window"), then the read command is reset and the state machine is set to 100.

#### 7.4 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 40 "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).

#### 7.5 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 40 "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 40 "Positioning window") and the travel was successfully completed, then the movement command is reset and the state machine is set to 100.

## 7.6 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 40 "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.

## 7.7 Example 7: Error upper end limit reached

In this sample program the **FB\_MC\_Error\_PSx3xx** is presented. It is moved 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 40 "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 <b>FB_MC_Move_PSx3xx</b> (<udiErrorID>). The <udiErrorID> would then be 0x100B0.)

## List of Figures

Figure 1 Connection of variables from ST_Variables_PSx3xx to the inputs and outputs of FB_MC_Move_PSx3xx .....	10
Figure 2 Connection of variables from ST_Variables_PSx3xx to the inputs and outputs of FB_MC_Error_PSx3xx .....	10
Figure 3 Connection of variables from ST_Variables_PSx3xx to the inputs and outputs of FB_MC_ReadParameter_PSx3xx.....	11
Figure 4 Connection of variables from ST_Variables_PSx3xx to the inputs and outputs of FB_MC_WriteParameter_PSx3xx .....	11
Figure 5 Connection of variables from ST_Variables_PSx3xx to the inputs and outputs of FB_MC_Parametrization_PSx3xx.....	11
Figure 6 Parameter "loop length" is assigned value 20.....	12
Figure 7 Connection of variables from ST_Variables_PSx3xx to the inputs and outputs of FB_MC_PosParametrization_PSx3xx.....	12
Figure 8 Function Blocks and UDTs.....	23
Figure 9 Drive .....	23
Figure 10 Example programs.....	23
Figure 11 Selection of example program .....	23
Figure 12 Read out of parameter 10 (actual value) via the FB_MC_ReadParameter_PSx3xx function block.....	24
Figure 13 Connection of drive process data to the PLC variables .....	24