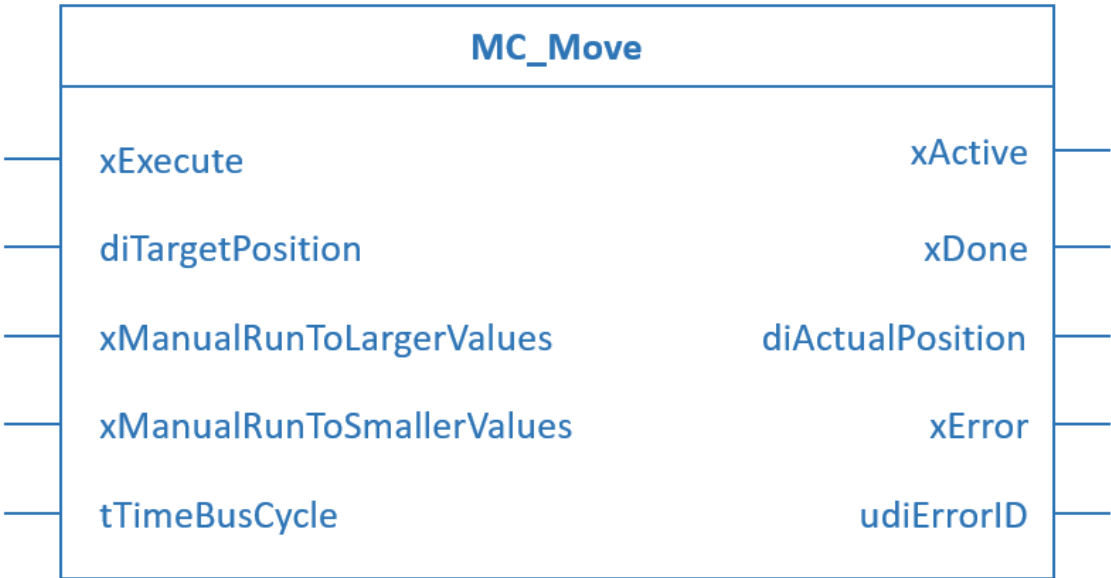




Meaning of the operating instructions

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 the 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 10-12
79199 Kirchzarten

Tel. +49 (7661) 39 63-0
info@halstrup-walcher.de
www.halstrup-walcher.de

© 2026

Translation of the original instructions

Table of contents

1	Safety notices.....	5
1.1	Intended use	5
1.2	Warning symbols.....	5
2	User-specific data types.....	6
2.1	<ST_DriveData_PSxHub>	6
2.2	<ST_DriveData_PSxHub_PSD4xx>	8
2.3	<ST_FunctionBlocks_PSxHub>	9
2.4	<ST_FunctionBlocks_PSxHub_PSD4xx>	10
2.5	<ST_Variables_PSxHub/PSxHub_PSD4xx>	10
3	<GVL_Data_Store_PSxHub>.....	10
4	Error description (<udiErrorID>).....	12
5	Description of the function blocks.....	14
5.1	FB_MC_ReadParameter_PSxHub.....	14
5.2	FB_MC_WriteParameter_PSxHub.....	14
5.3	FB_MC_MakeSynchronousGroups_PSxHub.....	14
5.4	FB_MC_SynchronousMovement_PSxHub.....	14
5.5	FB_MC_TryResolveSyncErrorGroup_PSxHub	15
5.6	FUN_MC_Communication_Drive	15
5.7	FB_MC_Move__PSxHub_Drive.....	16
5.8	FB_MC_Error_PSxHub_Drive	16
5.9	FC_MC_Error_ID_PSxHub/PSxHub_PSD4xx.....	16
5.10	FB_MC_ReadParameter_PSxHub_PSD4xx.....	17
5.11	FB_MC_WriteParameter_PSxHub_PSD4xx	17
5.12	FB_MC_Parametrization_PSxHub_PSD4xx	17
5.13	FB_MC_Parametrization_PSxHub_PSE3xx	18
5.14	FB_MC_PosParametrization_PSxHub_PSD4xx	19
5.15	FB_MC_PosParametrization_PSxHub_PSE3xx.....	19
6	Parameter description.....	20
6.1	<aGroupNumberForPort>.....	20
6.2	<xResetSynchronousMovementErrorBit>	20
6.3	<uiMaximumDifference>.....	20
6.4	<aExecuteGroup>	21
6.5	<aTargetPositionGroup>	21
6.6	<aTryResolveSyncErrorGroup>.....	21
6.7	<usiSyncGroupErrorResolveStrategy>	22
6.8	<xActive>	22
6.9	<diActualPosition>.....	23
6.10	<xDeliveryState>	23
6.11	<uiDirection>	23
6.12	<xDone>	24

6.13	<stDrive>.....	24
6.14	<stHub>	25
6.15	<stHubAndDrives>	25
6.16	<xError>	26
6.17	<sErrorDescription>.....	26
6.18	<udiErrorID>.....	27
6.19	<uiErrorParameter>.....	28
6.20	<xExecute>	28
6.21	<diLowerLimit>	29
6.22	<xManualRunToLargerValues>	30
6.23	<xManualRunToSmallerValues>	30
6.24	<stParameter>	31
6.25	<uiParameterNumber>	31
6.26	<xSaveSettings>	32
6.27	<diSetPoint>.....	32
6.28	<uiStepsPerTurn>	33
6.29	<usiSubIndex>	33
6.30	<diTargetPosition>	33
6.31	<tTimeBusCycle>	34
6.32	<diUpperLimit>.....	34
6.33	<diValue>.....	35
7	Use of the function blocks	36
7.1	Project.....	36
7.2	Library.....	36
7.3	Interlocking between the function blocks.....	38
7.4	Several PSxHubs in one project	39
7.5	Dependencies between function blocks	40
8	Example programs.....	41
8.1	Example 1: Positioning mode	41
8.2	Example 2: Manual mode	42
8.3	Example 3: Read current position	42
8.4	Example 4: Write factory default state.....	43
8.5	Example 5: Parameterize parameters.....	43
8.6	Example 6: Parameterize position parameters	44
8.7	Example 7: Error upper limit reached	44
8.8	Example 8: Creation of groups with synchronization monitoring	45
8.9	Example 9: Error handling when synchronization monitoring is triggered.....	46
	List of figures	47
	List of tables.....	47

1 Safety notices

1.1 Intended use

The PSxHub EC acts as an EtherCAT device via which up to 10 PSD IO-Link drives can be supplied and controlled. The process data of the individual drives is mapped in the process data of the hub.

1.2 Warning symbols

In these operating instructions the following highlights are used to indicate the hazards described thereafter when handling the system:

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

For each hub there is a data structure in which some data of the hub is stored. A global instance of this structure is required for each hub. This instance must be passed to every function block (FB) that acts on the hub itself (without _PSD4xx at the end). This is used, for example, to ensure that two accesses from different FBs to the parameter channel cannot be carried out at the same time. Furthermore, the addresses of the input and output data of the respective hub must be stored in this data structure.

Parameter name	Data type	written by	Description
<uiNodeAddress>	UInt	User	EtherCAT address
<sTargetNetId>	T_AmsNetId	User	NetId subdevices
<sMasterNetId>	T_AmsNetId	User	NetId hub
<sAxisName>	String[16]	User (optional)	Name of the axis
<sAxisDescription>	String[32]	User (optional)	Description (e.g. function, task of this axis)
<iActiveFunktionBlock>	Int	Function blocks	Active function block
<xAutomaticStop>	Bool	User	Motors are stopped on synchronization error
<xResetSynchronousMovementErrorBit>	Bool	User	Resets synchronization error bit
<xCommunicationError>	Bool	Function blocks	Communication error to IO-Device
<stPrivate>	ST_Private	Function blocks/ Linking by user	Data structure for internal use

Table 1 Structure of the ST_DriveData_PSxHub data type

The values for uiNodeAddress, sTargetNetId and sMasterNetId depend on the configuration in the project and should be read out in the online state. The assignment is shown below as an example.

PSxHub.example_PSxHub_TwinCAT3.GVL_Data_Store_PSxHub		
Expression	Type	Value
Hub	ST_Variables_PSxHubAndDrive	
PSxHub	ST_Variables_PSxHub	
stHubData_PSxHub	ST_HubData_PSxHub	
uiNodeAddress	UINT	1001
sTargetNetId	T_AmsNetId	'10.250.53.73.2.2'
sMasterNetId	T_AmsNetId	'192.168.233.1.2.1'
sAxisName	STRING(16)	"
sAxisDescription	STRING(32)	"
iActiveFunktionBlock	INT	0
xAutomaticStop	BOOL	FALSE
xResetSynchronousMovementErrorBit	BOOL	FALSE
xCommunicationError	BOOL	FALSE
stPrivate	ST_Private_PSxHub	
stFunctionBlocks_PSxHub	ST_FunctionBlocks_PSxHub	
PSD4xx	ARRAY [1.. 10] OF ST_Variables_PSD4xx	

Figure 1: sample configuration

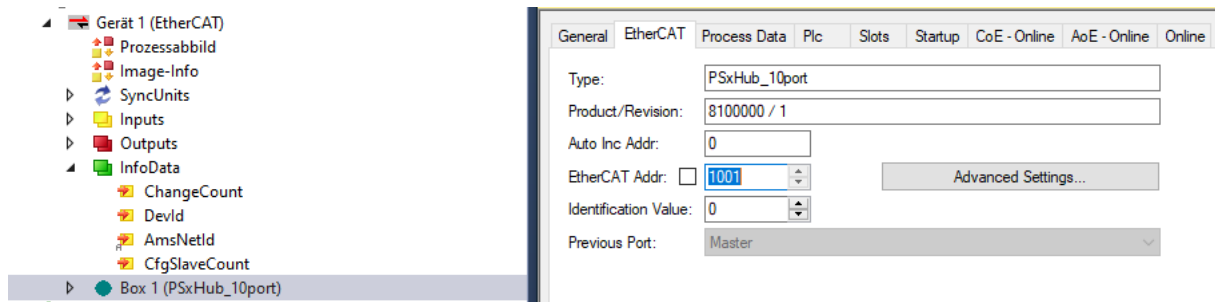


Figure 2: reading from *uiNodeAddress*

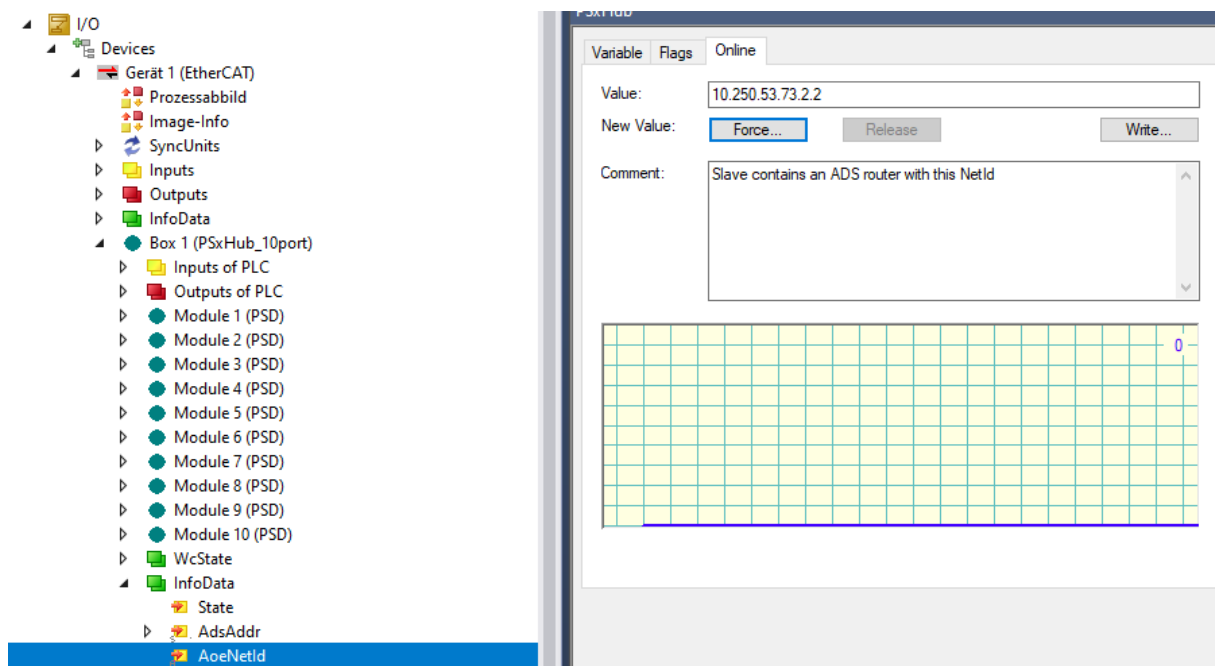


Figure 3: reading from *sTargetNetId*

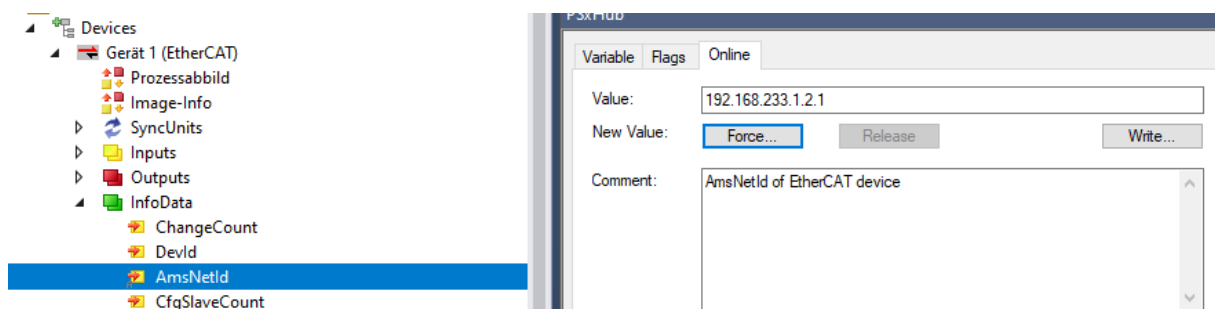


Figure 4: reading from *sMasterNetId*

The linking of the process data with stPrivate can be found in Table 1.

Remark	Name in process data	Name in stPrivate
	StatusWord1	ludiStatus_1
	StatusWord2	ludiStatus_2
Currently not used	PKW_PKE_in	
Currently not used	PKW_IND_in	
Currently not used	PKW_PWE_in	
	ControlWord1	OudiControlWord_1
	ControlWord2	OudiControlWord_2
Currently not used	PKW_PKE_out	
Currently not used	PKW_IND_out	
Currently not used	PKW_PWE_out	

Table 2 Assignment of hub process data to stPrivate

2.2 <ST_DriveData_PSxHub_PSD4xx>

For each drive there is a data structure in which some data of a drive is stored. A global instance of this structure is required for each drive. This instance must be passed to every function block (FB) that acts on the operated drive. This is used, for example, to ensure that two accesses from different FBs to the parameter channel cannot be carried out at the same time. Furthermore, the addresses of the input and output data of the respective drive must be stored in this data structure.

Parameter name	Data type	written by	Description
<sAxisName>	String[16]	User (optional)	Name of the 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/ Linking by user	Data structure for internal use

Table 3 Structure of the ST_DriveData_PSxHub_PSD4xxx data type

There are 2 variants for linking the process data with stPrivate. Table 2 contains the description for using the PSD or PSE modules, Table 3 the description for using the PSD_PlainData or PSE_PlainData modules. Both links enable the same functionality; using the PlainData modules merely reduces the number of necessary links, but requires a call to the function FUN_MC_Communication_Drive for each drive for conversion.

The PKW interface is only intended as an alternative to access via ADS, since this is not supported by some controllers (e.g. Lenze). In most use cases it is not needed and the corresponding blocks (_PKW) can be deleted.

Remark	Name in process data	Name in stPrivate
Only if PKW is used	PKE_in	luiPKE
Only if PKW is used	IND_in	luiIND
Only if PKW is used	PWE_in	ludiPWE
	StatusWordNode	unStatus.uiStatusRaw
	ActualSpeedNode	iCurrentRPM
	ActualPositionNode	diCurrentPosition
	PKE_out	OuiPKE
	IND_out	OuiIND
	PWE_out	OudiPWE
	ControlWordNode	unControl.uiControlRaw
	TargetPositionNode	diTargetPosition
Only relevant for drives with software module 'P' or 'Z'	TargetSpeedNode	iTargetRPM

Table 4 Linking of the process data with stPrivate, modules: PSD/PSE

Remark	Name in process data	Name in stPrivate
Only if PKW is used	PKW_In	stInputDataRow[0]
	ProcessData_In	stInputDataRow[1]
Only if PKW is used	PKW_Out	stOutputDataRow[0]
	ProcessData_Out	stOutputDataRow[1]

Table 5 Linking of the process data with stPrivate, modules PSD_PlainData/PSE_PlainData

2.3 <ST_FunctionBlocks_PSxHub>

For each PSxHub there is a data structure in which the variables for the assignment to the function blocks are stored. This structure has several sub-structures that are not further explained in the documentation. A global instance of this structure is required for each drive. To avoid unnecessary memory usage in the PLC, the data structure should be adjusted if function blocks are not used.

Parameter name	Data type	written by	Description
<stSynchronousMovement_PSxHub>	ST_SynchronousMovement_PSxHub	Function blocks	Variables for FB_MC_SynchronousMovement_PSxHub
<stMakeSynchronousGroups_PSxHub>	ST_MakeSynchronousGroups_PSxHub	Function blocks	Variables for FB_MC_MakeSynchronousGroups_PSxHub
<stReadParameter_PSxHub>	ST_ReadParameter_PSxHub	Function blocks	Variables for FB_MC_ReadParameter_PSxHub
<stWriteParameter_PSxHub>	ST_WriteParameter_PSxHub	Function blocks	Variables for FB_MC_WriteParameter_PSxHub
<stTryResolveSyncErrorGroup_PSxHub>	ST_TryResolveSyncErrorGroup_PSxHub	Function blocks	Variables for FB_MC_TryResolveSyncErrorGroup_PSxHub

Table 6 data structure ST_FunctionBlocks_PSxHub

2.4 <ST_FunctionBlocks_PSxHub_PSD4xx>

For each drive there is a data structure in which the variables for the assignment to the function blocks are stored. This structure has several sub-structures that are not further explained in the documentation. A global instance of this structure is required for each drive. If function blocks are not used, it is advisable to adapt the data structure so that no unnecessary memory is occupied in the PLC.

Parameter name	Data type	written by	Description
<stMove_PSxHub_PSD4xx>	ST_Move_PSxHub_PSD4xx	Function blocks	Variables for FB_MC_Move_PSxHub_PSD4xx
<stError_PSxHub_PSD4xx>	ST_Error_PSxHub_PSD4xx	Function blocks	Variables for FB_MC_Error_PSxHub_PSD4xx
<stReadParameter_PSxHub_PSD4xx>	ST_ReadParameter_PSxHub_PSD4xx	Function blocks	Variables for FB_MC_ReadParameter_PSxHub_PSD4xx
<stWriteParameter_PSxHub_PSD4xx>	ST_WriteParameter_PSxHub_PSD4xx	Function blocks	Variables for FB_MC_WriteParameter_PSxHub_PSD4xx
<stParametrization_PSxHub_PSD4xx>	ST_Parametrization_PSxHub_PSD4xx	Function blocks	Variables for FB_MC_Parametrization_PSxHub_PSD4xx
<stPosParametrization_PSxHub_PSD4xx>	ST_PosParametrization_PSxHub_PSD4xx	Function blocks	Variables for FB_MC_PosParametrization_PSxHub_PSD4xx

Table 7 data structure ST_FunctionBlocks_PSxHub_PSD4xx

2.5 <ST_Variables_PSxHub/PSxHub_PSD4xx>

In this data structure, the structures <ST_Variables_PSxHub> and an array with 10 entries of <ST_Variables_PSxHub> are bundled. This structure is created for each PSxHub in the global variable list <GVL_Data_Store_PSxHub>.

3 <GVL_Data_Store_PSxHub>

The GVL (global variable list) <GVL_Data_Store_PSxHub> provides the necessary variables to be able to assign all inputs and outputs of all available function blocks.

Each variable currently exists only once. If several PSxHubs are used in one project, the number of variables must be increased accordingly.

Example

```
fbMC_ReadParameter_PSxHub(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stInput_ReadParameter_PSxHub.xExecute,
  uiParameterIndex:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stInput_ReadParameter_PSxHub.uiParameterIndex,
  byParameterSubIndex := GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stInput_ReadParameter_PSxHub.byParameterSubIndex,
  xActive=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.xActive,
  xDone=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.xDone,
  diValue=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.diValue,
  xError=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.xError,
  udiErrorID=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.udiErrorID,
  aStatus=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.aStatus,
  udiSdoAbort=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.udiSdoError,
  stHub:= GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 5: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_ReadParameter_PSxHub

If the GVL is adopted into the project, it is recommended to delete the unused variables (to avoid unnecessary memory allocation in the PLC) and to adapt the function block to the actual number of drives present.

PSxHub.example_PSxHub_TwinCAT3.GVL_Data_Store_PSxHub		
Expression	Type	Value
Hub	ST_Variables_PSxHu...	
PSxHub	ST_Variables_PSxHub	
stHubData_PSxHub	ST_HubData_PSxHub	
stFunctionBlocks_PSxHub	ST_FunctionBlocks_...	
PSD4xx	ARRAY [1..10] OF S...	
PSD4xx[1]	ST_Variables_PSD4xx	
stDriveData_PSD4xx	ST_DriveData_PSD4xx	
sAxisName	STRING(16)	"
sAxisDescription	STRING(32)	"
iActiveFunctionBlock	INT	0
xCommunicationError	BOOL	FALSE
usiPort	USINT	0
stPrivate	ST_ProcessData	
stFunctionBlocks_PSD4xx	ST_FunctionBlocks_...	
stMove_PSD4xx	ST_Move_PSD4xx	
stError_PSD4xx	ST_Error_PSD4xx	
stReadParameter_PSD4xx	ST_ReadParameter_...	
stWriteParameter_PSD4xx	ST_WriteParameter...	
stParametrization_PSD4xx	ST_Parametrization_...	
stPosParametrization_PSD4xx	ST_PosParametrizati...	
PSD4xx[2]	ST_Variables_PSD4xx	
PSD4xx[3]	ST_Variables_PSD4xx	

Figure 6: <GVL_Data_Store_PSxHub> with variables for the drive

4 Error description (<udiErrorID>)

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

<udiErrorID> (hex)	Description
16#F0000 (mask)	Function block
16#0xxxx	<u>No</u> error code
16#1xxxx	Error in FB_MC_Move_PSxHub_Drive
16#2xxxx	Error in FB_MC_Error_PSxHub_Drive
16#3xxxx	Error in FB_MC_ReadParameter_PSxHub_PSD4xx
16#4xxxx	Error in FB_MC_WriteParameter_PSxHub_PSD4xx
16#5xxxx	Error in FB_MC_Parametrization_PSxHub_PSD4xx
16#6xxxx	Error in FB_MC_PosParametrization_PSxHub_PSD4xx
16#7xxxx	Error in FB_MC_SynchronousMovement_PSxHub
16#8xxxx	Error in FB_MC_MakeSynchronousGroups_PSxHub
16#9xxxx	Error in FB_MC_ReadParameter_PSxHub
16#Axxxx	Error in FB_MC_WriteParameter_PSxHub
16#Bxxxx	Error in FB_MC_TryResolveSyncErrorGroup_PSxHub
16#0F000 (mask)	Internal errors in the function blocks and process data errors
16#x0xxx	<u>No</u> internal error in the function blocks and process data errors
16#x1xxx	Error in the state machine (interlocking)
16#x2xxx	Invalid input address of the process data
16#x3xxx	Invalid output address of the process data
16#x4xxx	Communication error when reading the process data
16#x5xxx	Communication error when writing the process data
16#x6xxx	Invalid write command
16#00F00 (mask)	Error in the parameters
16#xx0xx	<u>No</u> fault behavior in the parameters
16#xx1xx	Parameter: communication timeout (1000 ms)
16#xx2xx	Parameter: invalid parameter number
16#xx3xx	Parameter: parameter number is read-only
16#xx4xx	Parameter: value is below the minimum or above the maximum
16#xx5xx	Parameter: invalid sub-index
16#xx6xx	Parameter: not an array
16#xx7xx	Parameter: invalid data type
16#xx8xx	Parameter: no write access
16#xx9xx	Parameter: request cannot be processed due to the current operating state
16#xxAxx	Other errors
16#000F0 (mask)	Drive error
16#xxx0x	<u>No</u> drive error
16#xxx1x	Following error
16#xxx2x	Under- or overvoltage of the motor supply
16#xxx3x	Positioning was aborted
16#xxx4x	Overtemperature

<udiErrorID> (hex)	Description
16#000F0 (mask)	Drive error
16#xxx5x	Measuring system error
16#xxx6x	Positioning error (blocking)
16#xxx7x	Manual rotation
16#xxx8x	Target position incorrect
16#xxx9x	Under- or overvoltage of the motor voltage / failure of the voltage monitoring
16#xxxAx	Lower limit undershot
16#xxxBx	Upper limit exceeded
16#0000F (mask)	Error PSxHub/synchronization
16#xxxx0	<u>No error at the PSxHub or synchronization</u>
16#xxxx1	Synchronization error group 1
16#xxxx2	Synchronization error group 2
16#xxxx3	Synchronization error group 3
16#xxxx4	Synchronization error group 4
16#xxxx5	Synchronization error group 5
16#xxxx6	Positioning error (blocking)
16#xxxx7	Undervoltage control
16#xxxx8	Overvoltage control
16#xxxx9	Overtemperature

Table 8: Error codes

The 'drive errors' are a mapping of the error bits in the status word of the drives.

Examples

- Move command (FB_MC_Move_PSxHub_Drive) with incorrect target position
→ <udiErrorID> = 16#10080
- Write parameter (FB_MC_WriteParameter_PSxHub_PSD4xx) with invalid parameter number → <udiErrorID> = 16#40200

5 Description of the function blocks

A detailed description of all parameters can be found in Chapter 6. Parameter description.

5.1 FB_MC_ReadParameter_PSxHub

This FB can be used to read values of parameters from the hub.

```
fbMC_ReadParameter_PSxHub(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stInput_ReadParameter_PSxHub.xExecute,
  uiParameterIndex:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stInput_ReadParameter_PSxHub.uiParameterIndex,
  byParameterSubIndex := GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stInput_ReadParameter_PSxHub.byParameterSubIndex,
  xActive=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.xActive,
  xDone=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.xDone,
  diValue:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.diValue,
  xError=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.xError,
  udiErrorID=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.udiErrorID,
  aStatus:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.aStatus,
  udiSdoAbort=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stReadParameter_PSxHub.stOutput_ReadParameter_PSxHub.udiSdoError,
  stHub:= GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 7: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_ReadParameter_PSxHub

5.2 FB_MC_WriteParameter_PSxHub

This FB can be used to write values of parameters to the hub.

```
fbMC_WriteParameter_PSxHub(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stInput_WriteParameter_PSxHub.xExecute,
  uiParameterIndex:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stInput_WriteParameter_PSxHub.uiParameterIndex,
  byParameterSubIndex := GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stInput_WriteParameter_PSxHub.byParameterSubIndex,
  diValue:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stInput_WriteParameter_PSxHub.diValue,
  xActive=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stOutput_WriteParameter_PSxHub.xActive,
  xDone=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stOutput_WriteParameter_PSxHub.xDone,
  xError=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stOutput_WriteParameter_PSxHub.xError,
  udiErrorID=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stOutput_WriteParameter_PSxHub.udiErrorID,
  udiSdoAbort=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stOutput_WriteParameter_PSxHub.udiSdoError,
  aStatus:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stWriteParameter_PSxHub.stOutput_WriteParameter_PSxHub.aStatus,
  stHub:= GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 8: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_WriteParameter_PSxHub

5.3 FB_MC_MakeSynchronousGroups_PSxHub

This function block is used to assign drives to synchronous groups. Groups 1-5 are available for selection.

```
fbMC_MakeSynchronousGroups_PSxHub(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stInput_MakeSynchronousGroups_PSxHub.xExecute,
  aGroupNumberForPort:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stInput_MakeSynchronousGroups_PSxHub.aGroupNumberForPort,
  uiMaximumDifference:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stInput_MakeSynchronousGroups_PSxHub.uiMaximumDifference,
  xSaveSettings:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stInput_MakeSynchronousGroups_PSxHub.xSaveSettings,
  xActive=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stOutput_MakeSynchronousGroups_PSxHub.xActive,
  xDone=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stOutput_MakeSynchronousGroups_PSxHub.xDone,
  xError=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stOutput_MakeSynchronousGroups_PSxHub.xError,
  udiErrorID=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stOutput_MakeSynchronousGroups_PSxHub.udiErrorID,
  uiErrorParameter:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stMakeSynchronousGroups_PSxHub.stOutput_MakeSynchronousGroups_PSxHub.uiErrorParameter,
  stHub:= GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 9: Assignment of variable from <GVL_Data_Store_PSxHub> to <FB_MC_MakeSynchronousGroups_PSxHub>

5.4 FB_MC_SynchronousMovement_PSxHub

This function block is used for synchronous movement of the synchronous groups.

```
fbMC_SynchronousMovement_PSxHub(
  aExecuteGroup:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stSynchronousMovement_PSxHub.stInput_SynchronousMovement_PSxHub.aExecuteGroup,
  aTargetPositionGroup:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stSynchronousMovement_PSxHub.stInput_SynchronousMovement_PSxHub.aTargetPositionGroup,
  xError=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stSynchronousMovement_PSxHub.stOutput_SynchronousMovement_PSxHub.xError,
  udiErrorID=> GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stSynchronousMovement_PSxHub.stOutput_SynchronousMovement_PSxHub.udiErrorID,
  stHubAndDrives:= GVL_Data_Store_PSxHub.Hub);
```

Figure 10: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_SynchronousMovement_PSxHub

5.5 FB_MC_TryResolveSyncErrorGroup_PSxHub

This function block can be used to compensate for synchronization errors that have occurred by moving either to the position of the highest or lowest drive in the group.

```
fbMC_TryResolveSyncErrorGroup_PSxHub(
  aTryResolveSyncErrorGroup:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stTryResolveSyncErrorGroup_PSxHub.stInput_TryResolveSyncErrorGroup_PSxHub.aTryResolveSyncErrorGroup,
  usiSyncGroupErrorResolveStrategy:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stTryResolveSyncErrorGroup_PSxHub.stInput_TryResolveSyncErrorGroup_PSxHub.usiSyncGroupErrorResolveStrategy,
  xError:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stTryResolveSyncErrorGroup_PSxHub.stOutput_TryResolveSyncErrorGroup_PSxHub.xError,
  udiErrorID:= GVL_Data_Store_PSxHub.Hub.PSxHub.stFunctionBlocks_PSxHub.stTryResolveSyncErrorGroup_PSxHub.stOutput_TryResolveSyncErrorGroup_PSxHub.udiErrorID,
  stHubAndDrives:= GVL_Data_Store_PSxHub.Hub);
```

Figure 11: Assignment from variable <GVL_Data_Store_PSxHub> to FB_MC_TryResolveSyncErrorGroup_PSxHub

5.6 FUN_MC_Communication_Drive

This function is used for communication between drives and blocks when the PlainData versions of the drives are used. In this variant the linking effort is reduced as shown in Table 4.

Remark	Name in process data	Name in stPrivate
Only if PKW is used	PKW_In	stInputDataRow[0]
	ProcessData_In	stInputDataRow[1]
Only if PKW is used	PKW_Out	stOutputDataRow[0]
	ProcessData_Out	stOutputDataRow[1]

Table 9: linking process data with stPrivate

By calling the function, the data is converted into the respective required format.

5.7 FB_MC_Move_PSD4xx_Drive

This function block is used to position a drive.

```
//fbMC_Move_PSD4xx_1 --> instance of FB_MC_Move_PSD4xx (1. PSD4xx)
fbMC_Move_PSD4xx_1(
  xExecute:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stInput_Move_PSD4xx.xExecute,
  diTargetPosition:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stInput_Move_PSD4xx.diTargetPosition,
  xManualRunToLargerValues:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stInput_Move_PSD4xx.xManualRunToLargerValues,
  xManualRunToSmallerValues:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stInput_Move_PSD4xx.xManualRunToSmallerValues,
  tTimeBusCycle:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stInput_Move_PSD4xx.tTimeBusCycle,
  xActive:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stOutput_Move_PSD4xx.xActive,
  xDone:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stOutput_Move_PSD4xx.xDone,
  diActualPosition:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stOutput_Move_PSD4xx.diActualPosition,
  xError:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stOutput_Move_PSD4xx.xError,
  udiErrorID:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stMove_PSD4xx.stOutput_Move_PSD4xx.udiErrorID,
  stDrive:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stDriveData_PSD4xx);
```

Figure 12: Assignment from variable <GVL_Data_Store_PSD4xx> to FB_MC_Move_PSD4xx

If a drive reports several errors, the <udiErrorID> with the highest priority is output. This applies to all FBs. The priority of the output corresponds to the order in the following table (highest priority is 16#x1xxx):

<udiErrorID>	Description
16#x1xxx	Error in the state machine (interlocking)
16#x2xxx	Invalid input address of the process data
16#x3xxx	Invalid output address of the process data
16#x4xxx	Communication error when reading the process data
16#x5xxx	Communication error when reading the process data
16#xxx2x	Under- or overvoltage of the motor supply (STO enable inactive*)
16#xxx4x	Overtemperature
16#xxx5x	Measuring system error (measuring system or STO hardware error*)
16#xxx8x	Target position incorrect
16#xxx9x	Under- or overvoltage of the motor voltage / failure of the voltage monitoring
16#xxx6x	Positioning error (blocking)
16#xxx7x	Manual rotation
16#xxxAx	Lower limit undershot
16#xxxBx	Upper limit exceeded
16#xxx3x	Positioning was aborted
16#xxx1x	Following error

Table 10 drive positioning

*If the PSD4xx is a device with STO, the error description in brackets must be observed!
(<udiErrorID>: 16#xxx2x and 16#xxx5x)

5.8 FB_MC_Error_PSD4xx_Drive

This FB outputs the status of the drive and of the FB as an error bit, error ID (<udiErrorID>) and as a text output.

```
//fbMC_Error_PSD4xx_1 --> instance of FB_MC_Error_PSD4xx (1. PSD4xx)
fbMC_Error_PSD4xx_1(
  xExecute:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stError_PSD4xx.stInput_Error_PSD4xx.xExecute,
  xError:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stError_PSD4xx.stOutput_Error_PSD4xx.xError,
  udiErrorID:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stError_PSD4xx.stOutput_Error_PSD4xx.udiErrorID,
  sErrorDescription:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stError_PSD4xx.stOutput_Error_PSD4xx.sErrorDescription,
  stDrive:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stDriveData_PSD4xx);
```

Figure 13: Assignment from variable <GVL_Data_Store_PSD4xx> to FB_MC_Error_PSD4xx_Drive

5.9 FC_MC_Error_ID_PSD4xx/PSD4xx

This function is used internally as a sub-function of the other function blocks. It only has to be copied from the library into the user program. The wiring is performed internally.

5.10 FB_MC_ReadParameter_PSxHub_PSD4xx

This FB can be used to read values of parameters (ISDUs) from a drive.

```
//fbMC_ReadParameter_PSD4xx_1 --> instance of FB_MC_ReadParameter_PSD4xx (1. PSD4xx)
fbMC_ReadParameter_PSD4xx_1(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stInput_ReadParameter_PSD4xx.xExecute,
  uiParameterNumber:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stInput_ReadParameter_PSD4xx.uiParameterNumber,
  usiSubindex:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stInput_ReadParameter_PSD4xx.usiSubindex,
  xActive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.xActive,
  xDone:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.xDone,
  xError:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.xError,
  udiErrorID:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.udiErrorID,
  diValue:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.diValue,
  stDrive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stDriveData_PSD4xx,
  stHub := GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 14: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_ReadParameter_PSxHub_PSD4xx

5.11 FB_MC_WriteParameter_PSxHub_PSD4xx

This FB can be used to write parameter values (ISDUs) to a drive.

```
//fbMC_WriteParameter_PSD4xx_1 --> instance of FB_MC_WriteParameter_PSD4xx (1. PSD4xx)
fbMC_WriteParameter_PSD4xx_1(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stInput_WriteParameter_PSD4xx.xExecute,
  uiParameterNumber:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stInput_WriteParameter_PSD4xx.uiParameterNumber,
  diValue:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stInput_WriteParameter_PSD4xx.diValue,
  xActive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stOutput_WriteParameter_PSD4xx.xActive,
  xDone:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stOutput_WriteParameter_PSD4xx.xDone,
  xError:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stOutput_WriteParameter_PSD4xx.xError,
  udiErrorID:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stWriteParameter_PSD4xx.stOutput_WriteParameter_PSD4xx.udiErrorID,
  stDrive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stDriveData_PSD4xx,
  stHub := GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 15: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_WriteParameter_PSxHub_PSD4xx

5.12 FB_MC_Parametrization_PSxHub_PSD4xx

This FB can be used to write all parameters (ISDUs) of the drive at once. This is particularly useful for initial commissioning. A clear structure was chosen here, since all parameters are passed as a block with the user-specific data type <ST_Parameter_PSD4xx>.

```
//fbMC_Parametrization_PSD4xx_1 --> instance of FB_MC_Parametrization_PSD4xx (1. PSD4xx)
fbMC_Parametrization_PSD4xx_1(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.xExecute,
  xDeliveryState:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.xDeliveryState,
  stParameter:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter,
  xSaveSettings:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.xSaveSettings,
  xActive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stOutput_Parametrization_PSD4xx.xActive,
  xDone:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stOutput_Parametrization_PSD4xx.xDone,
  xError:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stOutput_Parametrization_PSD4xx.xError,
  udiErrorID:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stOutput_Parametrization_PSD4xx.udiErrorID,
  uiErrorParameter:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stOutput_Parametrization_PSD4xx.uiErrorParameter,
  stDrive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stDriveData_PSD4xx,
  stHub := GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 16: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_Parametrization_PSxHub_PSD4xx

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

```
2: //parameterize length of loop and target speed
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stTargetSpeed_l37.xEnable := TRUE;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stTargetSpeed_l37.uiValue := 20;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stLoopLength_l24.xEnable := TRUE;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stLoopLength_l24.diValue := 0;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.xExecute := TRUE;
```

Figure 17: parameter 'loop length' set to 0, Parameter 'targetSpeed' to 20

- By setting <xEnable> = FALSE, the respective parameter is not written.
- Optionally, a factory default state can be requested before setting individual parameters. For this, the input <xDeliveryState> must be set to TRUE. This sets the values of all parameters to the factory default state (initially without saving).
- Optionally, the written values can also be saved at the end. For this, the input <xSaveSettings> must be set to TRUE.
- The order of the write accesses is as follows:
<xDeliveryState>, parameter 115, 116, 117, 68, ... and <xSaveSettings>.
- In the event of a write error for a parameter, the subsequent parameters are no longer written and the values are not saved either, even if the input <xSaveSettings> is set.

5.13 FB_MC_Parametrization_PSxHub_PSE3xx

In terms of operation and behavior, this block corresponds to the block FB_MC_Parametrization_PSxHub_PSD4xx.

The parameters of the PSx3 family differ slightly from those of the PSD4 family; the data type ST_Parameter_PSxHub_PSE3xx is therefore used for the stParameter input in order to make all PSx3 parameters available to the user.

```
//fbMC_Parametrization_PSE3xx --> instance of FB_MC_Parametrization_PSE3xx
fbMC_Parametrization_PSE3xx(
    xExecute:= GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stInput_Parametrization_PSE3xx.xExecute,
    xDeliveryState:= GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stInput_Parametrization_PSE3xx.xDeliveryState,
    stParameter:= GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stInput_Parametrization_PSE3xx.stParameter,
    xSaveSettings:= GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stInput_Parametrization_PSE3xx.xSaveSettings,
    xActive=> GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stOutput_Parametrization_PSE3xx.xActive,
    xDone=> GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stOutput_Parametrization_PSE3xx.xDone,
    xError=> GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stOutput_Parametrization_PSE3xx.xError,
    udiErrorID:= GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stOutput_Parametrization_PSE3xx.udiErrorID,
    uiErrorParameter:= GVL_Data_Store_PSxHub.Hub.PSE.stFunctionBlocks_PSE3xx.stParametrization_PSE3xx.stOutput_Parametrization_PSE3xx.uiErrorParameter,
    stDrive:= GVL_Data_Store_PSxHub.Hub.PSE.stDriveData_PSE3xx,
    stHub := GVL_Data_Store_PSxHub.Hub.PSE.stHubData_PSxHub);
```

Figure 18: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_Parametrization_PSE3xx

5.14 FB_MC_PosParametrization_PSxHub_PSD4xx

This FB can be used to parameterize the position data (parameters that influence the displayed actual position). This is particularly useful for initial commissioning.

```
fbMC_PosParametrization_PSD4xx_3(
  xExecute:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.xExecute,
  uiDirection:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.uiDirection,
  uiStepsPerTurn:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.uiStepsPerTurn,
  diLowerLimit:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.diLowerLimit,
  diUpperLimit:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.diUpperLimit,
  diSetPoint:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.diSetPoint,
  xSaveSettings:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stInput_PosParametrization_PSD4xx.xSaveSettings,
  xActive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stOutput_PosParametrization_PSD4xx.xActive,
  xDone:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stOutput_PosParametrization_PSD4xx.xDone,
  xError:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stOutput_PosParametrization_PSD4xx.xError,
  udiErrorID:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stOutput_PosParametrization_PSD4xx.udiErrorID,
  uiErrorParameter:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stFunctionBlocks_PSD4xx.stPosParametrization_PSD4xx.stOutput_PosParametrization_PSD4xx.uiErrorParameter,
  stDrive:= GVL_Data_Store_PSxHub.Hub.PSD4xx[3].stDriveData_PSD4xx,
  stHub := GVL_Data_Store_PSxHub.Hub.PSxHub.stHubData_PSxHub);
```

Figure 19: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_PosParametrization_PSxHub_PSD4xx

The following must be observed when using the FB:

- All values must be set and must have a meaningful relationship to each other. All conditions are described in the table below. All values are processed, then the following parameters are written in the order specified:
 - Rotation direction (ISDU 115) <uiDirection>
 - Actual value evaluation numerator (ISDU116) 400
 - Actual value evaluation denominator (ISDU 117) <uiStepsPerTurn>
 - Actual value (ISDU 68) <diSetPoint>
 - If (<diSetPoint> > <diUpperLimit>):
Upper mapping end (ISDU 120) = <diSetPoint> + (3 x <uiStepsPerTurn>)
otherwise:
Upper mapping end (ISDU 120) = <diUpperLimit> + (3 x <uiStepsPerTurn>)
 - Upper limit (ISDU 121) <diUpperLimit>
 - Lower limit (ISDU 122) <diLowerLimit>

The conditions and the error messages that are output if a requirement is not met are listed below.

Condition	<udiErrorID>	<uiErrorParameter>
<uiStepsPerTurn> ≥ 1	16#61400	116
<uiStepsPerTurn> ≤ 10000	16#61400	116
<diLowerLimit> ≤ <diUpperLimit>	16#61400	121
(<diUpperLimit> – <diLowerLimit>) / <uiStepsPerTurn> ≤ <rRotations>	16#61400	122
If <diSetPoint> < <diLowerLimit>: (<diUpperLimit> – <diSetPoint>) / <uiStepsPerTurn> ≤ <rRotations>	16#61400	68
If <diSetPoint> > <diUpperLimit>: (<diSetPoint> – <diLowerLimit>) / <uiStepsPerTurn> ≤ <rRotations>	16#61400	68

Table 11 Condition and Errormessage

- Optionally, the written values can also be saved at the end. For this, the input <xSaveSettings> must be set to TRUE.
- In the event of a write error for a parameter, the subsequent parameters are no longer written and the values are not saved either, even if the input <xSaveSettings> is set.

5.15 FB_MC_PosParametrization_PSxHub_PSE3xx

In terms of operation and behavior, this block corresponds to the block FB_MC_PosParametrization_PSxHub_PSD4xx; only the internally used parameter numbers differ.

6 Parameter description

6.1 <aGroupNumberForPort>

Block	FB_MC_MakeSynchronousGroups_PSxHub	
Synchronous group of the port		
Data type	Array[1..10] of UInt	
Default	0	
Type	Input	
Description		
This input can be used to assign a synchronous group to each port. If, for example, ports 1-3 are to be combined in synchronous group 1, the entries 1-3 would have to be assigned the value 1. Permissible values are 0-5. 0 means the port is not part of a synchronous group; the values 1-5 correspond to synchronous groups 1-5.		

6.2 <xResetSynchronousMovementErrorBit>

Block		
Resets bit 8 ('synchronization error detected') of the status		
Data type	Bool	
Default	-	
Type	Output	
Description		
Resets bit 8 ('synchronization error detected') of the status. The synchronization error bit of the group is cleared automatically when the difference between the drives is back within the tolerated range.		

6.3 <uiMaximumDifference>

Block	FB_MC_MakeSynchronousGroups_PSxHub	
Permitted difference without synchronization error		
Data type	UInt	
Default	-	
Type	Input	
Description		
This input defines the maximum step difference that is tolerated in a synchronous group. If the largest difference in the group exceeds this value, a synchronization error is detected (bit 8 status word PSxHub) and the synchronization error bit of the group is set (bits 9-13 status word PSxHub).		

6.4 <aExecuteGroup>

Block	FB_MC_SynchronousMovement_PSxHub
Starts the positioning of the synchronous group	
Data type	Array[1..5] of Bool
Default	-
Type	Input
Description	
Starts the positioning of the respective synchronous group to the value set in <aTargetPositionGroup>. Positioning is completed when all drives of the group have set the <xDone> bit and are at the target position. During movement the status words of the drives and of the PSxHub should be monitored in order to detect drive errors and synchronization errors.	

6.5 <aTargetPositionGroup>

Block	FB_MC_SynchronousMovement_PSxHub
Sets target position of the synchronous group	
Data type	Array[1..5] of DInt
Default	-
Type	Input
Description	
Defines the target position for the respective synchronous group.	

6.6 <aTryResolveSyncErrorGroup>

Block	FB_MC_TryResolveSyncErrorGroup_PSxHub
Attempts to resolve the synchronization error of the group	
Data type	Array[1..5] of Bool
Default	-
Type	Input
Description	
If the input of the respective group is set, an attempt is made to resolve the synchronization error of the group by moving all drives to a common target position. Success can be seen from the clearing of the hub's status bits (bit 8 'synchronization error detected' and bits 9-13 'deviation in synchronous group X detected').	

CAUTION!

The reference loop may cause temporary significant deviations. Take the loop length parameter into account.

6.7 <usiSyncGroupErrorResolveStrategy>

Block	FB_MC_TryResolveSyncErrorGroup_PSxHub
The strategy to be used for resolving synchronization errors	
Data type	USInt
Default	-
Type	Input
Description	
This input defines the strategy that is applied to resolve synchronization errors. With 0, all drives of the group move to the position of the highest drive. With 1, all drives of the group move to the position of the lowest drive.	

6.8 <xActive>

Block	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub, FB_MC_Move_PSxHub_Drive, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx,
FB is active or move command or movement is active	
Data type	Bool
Default	-
Type	Output
Description	
<u>FB MC Move Drive</u> This output is set when: - the approval <xExecute> is set from FALSE to TRUE - the approval <xExecute> is already present and the target position changes, the bit 'drive running' in the status of the drive is set (e.g. during readjustment of the drive) This output is reset when: - at the end of a movement the bit 'drive running' in the status of the drive is no longer set and the time of the parameter bus cycle (see 6.33 <tTimeBusCycle>) has elapsed - a communication error occurs <u>All other blocks</u> The bit is set as long as the respective function block is running. As soon as the operation has been completed or an error has occurred, the bit is reset.	

6.9 <diActualPosition>

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

6.10 <xDeliveryState>

Block	FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx
Loading the factory settings (initially without saving)	
Data type	Bool
Default	FALSE
Type	Input
Description	
Before writing the enabled parameters, all parameters are first reset to factory settings.	

6.11 <uiDirection>

Block	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Direction of rotation of the output shaft	
Data type	UInt
Default	0
Type	Input
Description	
This parameter corresponds to ISDU 115(PSD) or 123(PSE) 'rotation direction'. Direction in which the drive should rotate for larger values (when viewed from the output shaft): (0 clockwise; 1 counter clockwise)	

6.12 <xDone>

Block	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub, FB_MC_Move_PSxHub_Drive, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx	
Target position reached or FB has completed operation		
Data type	Bool	
Default	-	
Type	Output	
Description		
<u>FB_MC_Move_Drive</u> This output is a mapping of the status bit ‘target position reached’. In addition, the output is set to 0 for the duration of the parameter ‘bus cycle’ (see 6.33 <tTimeBusCycle>) after being started by <xExecute>. If a communication error occurs, it is reset.		
<u>All other blocks</u> The bit is set as soon as the operation has been successfully completed. It is reset at the start of the respective operation.		

6.13 <stDrive>

Block	FB_MC_Error_PSxHub_Drive, FB_MC_Move_PSxHub_Drive, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx, FUN_MC_Communication_Drive	
Data structure for communication		
Data type	ST_DriveData_PSD4xx	
Type	Input & Output	
Description		
A global instance of this structure is required for each drive. This instance is passed to every function block (FB) that acts on the operated drive.		

6.14 <stHub>

Block	FB_MC_Error_PSxHub_Drive, FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub,	
Data structure for communication		
Data type	ST_HubData_PSxHub	
Type	Input & Output	
Description		
A global instance of this structure is required for each hub. This instance is passed to every function block (FB) that acts on the operated hub.		

6.15 <stHubAndDrives>

Block	FB_MC_SynchronousMovement_PSxHub, FB_MC_TryResolveSyncErrorGroup_PSxHub	
Data structure for communication		
Data type	ST_Variables_PSxHub/PSxHub_PSD4xx	
Type	Input & Output	
Description		
A global instance of this structure is required for each hub. This instance is passed to the function blocks (FB) that act on the PSxHub and several drives.		

6.16 <xError>

Block	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_SynchronousMovement_PSxHub, FB_MC_TryResolveSyncErrorGroup_PSxHub, FB_MC_Error_PSxHub_Drive, FB_MC_Move_PSxHub_Drive, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
Error during execution of the FB or error in the drive	
Data type	Bool
Default	FALSE
Type	Output
Description	
<u>FB MC Move PSxHub Drive</u> 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. following error).	
<u>FB MC Error PSxHub Drive</u> The output <xError> is updated every cycle as long as <xExecute> is set. If <xExecute> is reset, this output assumes the specified default value.	
<u>FB MC SynchronousMovement PSxHub</u> The error bit is updated every cycle and is set if an error occurs during the execution of a synchronous group.	
<u>All other blocks</u> The error bit is updated every cycle and is set if an error has occurred during execution of the FB.	

6.17 <sErrorDescription>

Block	FB_MC_Error_PSxHub_Drive
Error description as text output	
Data type	String[254]
Default	""
Type	Output
Description	
An error description as text output	

6.18 <udiErrorID>

Block	FB_MC_SynchronousMovement_PSxHub, FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub, FB_MC_TryResolveSyncErrorGroup_PSxHub, FB_MC_Error_PSxHub_Drive, FB_MC_Move_PSxHub_Drive, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
Error ID (see Chapter 4 Error description (<udiErrorID>))	
Data type	UDint
Default	0
Type	Output
Description	
The error ID can also be set while the drive is moving (e.g. following error) <u>FB MC Move PSxHub Drive</u> The error ID is updated every cycle and is output if an error occurs during execution of the FB. It can also be output while the drive is active (e.g. following error). <u>FB MC Error PSxHub Drive</u> The output <udiErrorID> is updated every cycle as long as <xExecute> is set. If <xExecute> is reset, these outputs assume the specified initial value. <u>All other blocks</u> The error ID is updated every cycle and is set if an error has occurred during execution of the FB.	



CAUTION!

The outputs <xError> and <udiErrorID> of the function blocks are always updated in **FB_MC_Move_Drive** — even when the input <xExecute> is not set.

6.19 <uiErrorParameter>

Block	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Parameter number at which the error occurred in the event of an error	
Data type	UInt
Default	0
Type	Output
Description	
If there was no error, the value 0 is output.	

6.20 <xExecute>

Block	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub, FB_MC_Error_PSxHub_Drive, FB_MC_Move_PSxHub_Drive, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
Approval of the drive	
Data type	Bool
Default	FALSE
Type	Input
Description	
<u>FB MC Move PSxHub Drive</u> 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, for example, readjustment is active in the drive, the drive readjusts automatically. If the input is set and the setpoint is changed, the drive immediately moves to it. An edge is not required. If the input is reset during movement, the drive stops.	
<u>All other blocks</u> On a rising edge, the operation of the respective function block is carried out. For a new operation a rising edge must be generated again. If the bit is reset, the outputs assume the specified initial values.	

6.21 <diLowerLimit>

Block	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx	
Lower limit		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to ISDUs 122(PSD) or 130(PSE) 'lower limit'.		

6.22 <xManualRunToLargerValues>

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

CAUTION!

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

6.23 <xManualRunToSmallerValues>

Block	FB_MC_Move_PSxHub_Drive
Manual run to smaller values	
Data type	Bool
Default	FALSE
Type	Input
Description	
Manual run to smaller values is performed up to the lower limit. The input <xExecute> must additionally be set.	

CAUTION!

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

6.24 <stParameter>

Block	FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSD4xx
Transfer of the parameter set	
Data type	ST_Parameter_PSxHub_PSD4xx
Default	-
Type	Input
Description	
The parameter number of the drive (ISDU) is given after the parameter name. The data type, a description and the value range can be found in the bus description of the PSD4 or PSE3 IOLink.	

6.25 <uiParameterNumber>

Block	FB_MC_WriteParameter_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx
Parameter number of the parameter to be read or written	
Data type	UInt
Default	0
Type	Input
Description	
On a rising edge a read or write operation is started. For a new operation a rising edge must be generated again. If the bit is reset, the outputs assume the specified initial values.	

6.26 <xSaveSettings>

Block	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Saving the parameter settings	
Data type	Bool
Default	FALSE
Type	Input
Description	
<u>FB_MC_MakeSynchronousGroups_PSxHub:</u> This parameter corresponds to parameter 0xF58F 'factory default state'. (function <writing a '1'>)	
<u>Function blocks of the drives:</u> This parameter corresponds to ISDU 194(PSD) or 193(PSE) 'factory default state' of the drive. (function <writing a '1'>)	

6.27 <diSetPoint>

Block	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Actual position of the measuring system	
Data type	DInt
Default	0
Type	Input
Description	
This parameter corresponds to ISDU 68 (PSD) or 67 (PSE) 'actual value'.	

6.28 <uiStepsPerTurn>

Block	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Steps per revolution at the output shaft (resolution)	
Data type	Int
Default	0
Type	Input
Description	
The number of steps per revolution <uiStepsPerTurn> directly produces the value of ISDU 117 (PSD) or 125(PSE) 'actual value evaluation denominator'.	

6.29 <usiSubIndex>

Block	FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
Sub-index of the parameter	
Data type	USInt
Default	0
Type	Input
Description	
The value of the parameter <usiSubIndex> can remain at the default value if the parameter has no sub-index.	

6.30 <diTargetPosition>

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

NOTE

To approach the same target position, e.g. after blocking, the approval parameter <xExecute> must be reset and set again.

6.31 <tTimeBusCycle>

Block	FB_MC_Move_PSxHub_Drive	
Update time of the EtherCAT bus cycle		
Data type	Time	
Default	40ms	
Type	Input	
Description		
With long cycle times on the EtherCAT bus it may happen that the PLC does not receive a change of the bit 'drive running'. This is the case when the movement duration is shorter than the bus cycle. To counter this, the parameter <tTimeBusCycle> was introduced. An initial value of 40 ms for one bus cycle is assumed. For this duration, after a move command the output <xActive> is always set and the output <xDone> is reset. After that these outputs assume the corresponding state depending on the feedback from the status word (bit 'drive running' and bit 'target position reached'). With longer bus cycle times it is advisable, instead of the initial value, to enter the actual cycle duration multiplied by 2. If a shorter bus cycle time is guaranteed to be maintained, the positioning time is very short and the higher-level process is to continue quickly after the positioning has been completed, the value for <tTimeBusCycle> can also be reduced.		

6.32 <diUpperLimit>

Block	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx,	
Upper limit		
Data type	DInt	
Default	0	
Type	Input	
Description		
This parameter corresponds to parameter 129(PSE) or 121(PSD) ‘upper limit’.		

6.33 <diValue>

Block	FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
Value of a parameter to be written or read	
Data type	DInt
Default	0
Type	Input for FB_MC_WriteParameter_PSxHub, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx Output for FB_MC_ReadParameter_PSxHub, FB_MC_ReadParameter_PSxHub_PSD4xx
Description	
On a rising edge a read or write operation is started. For a new operation a rising edge must be generated again.	

7 Use of the function blocks

The function blocks are programmed in TwinCAT 3.1 (Build 4024). An upgrade to higher versions can be carried out without problems.

There are two ways to use the function blocks.

- as a project or
- as an integrated library

7.1 Project

The project contains one hub with 10 PSD connected to a soft PLC. If necessary, the addresses must be adjusted according to 2.1 in order to approve communication.

The inputs and outputs of the function blocks are connected in the function 'Instances()' with the variables in 'GVL_Data_Store_PSxHub', so that the functionality of the blocks can be tested by forcing these values.

The project also contains several examples, which are described in more detail in Chapter 8 **Example programs**. If one of the example projects is active, control via 'GVL_Data_Store_PSxHub' may only work to a limited extent, since only the respective required blocks are called in them.

7.2 Library

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

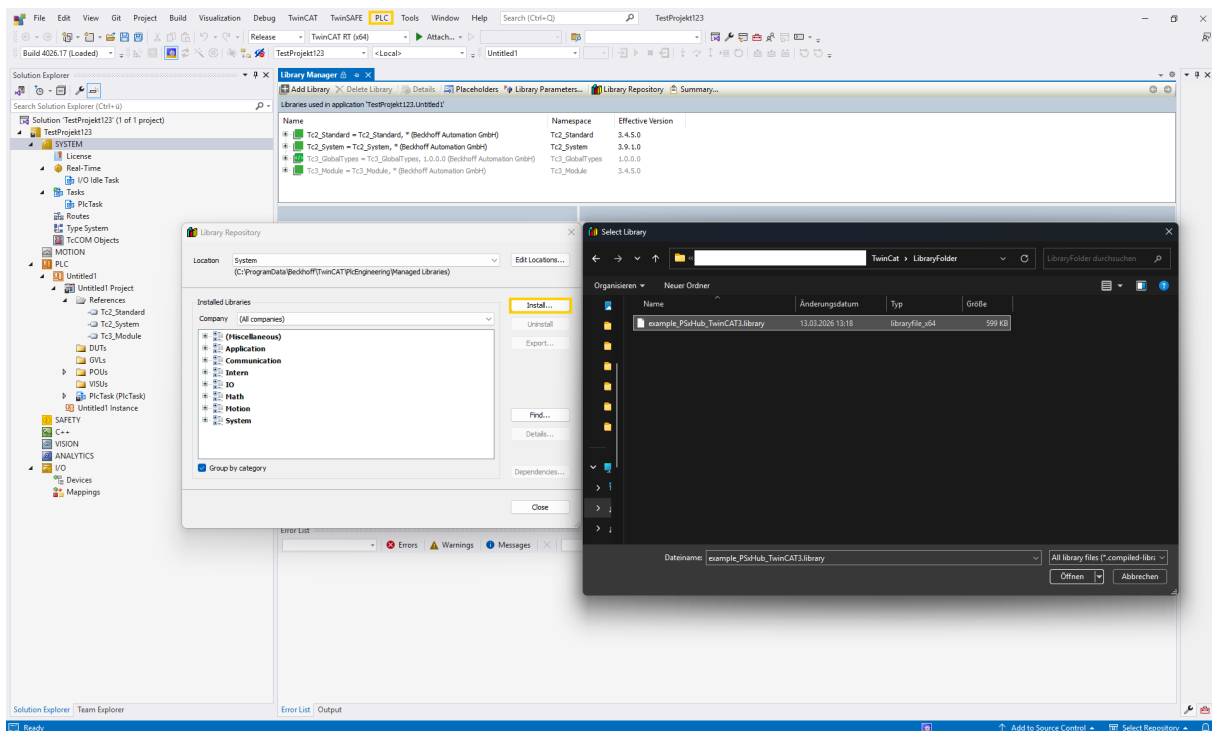


Figure 20: Installation of the library in the library repository

The library can then be selected and added.

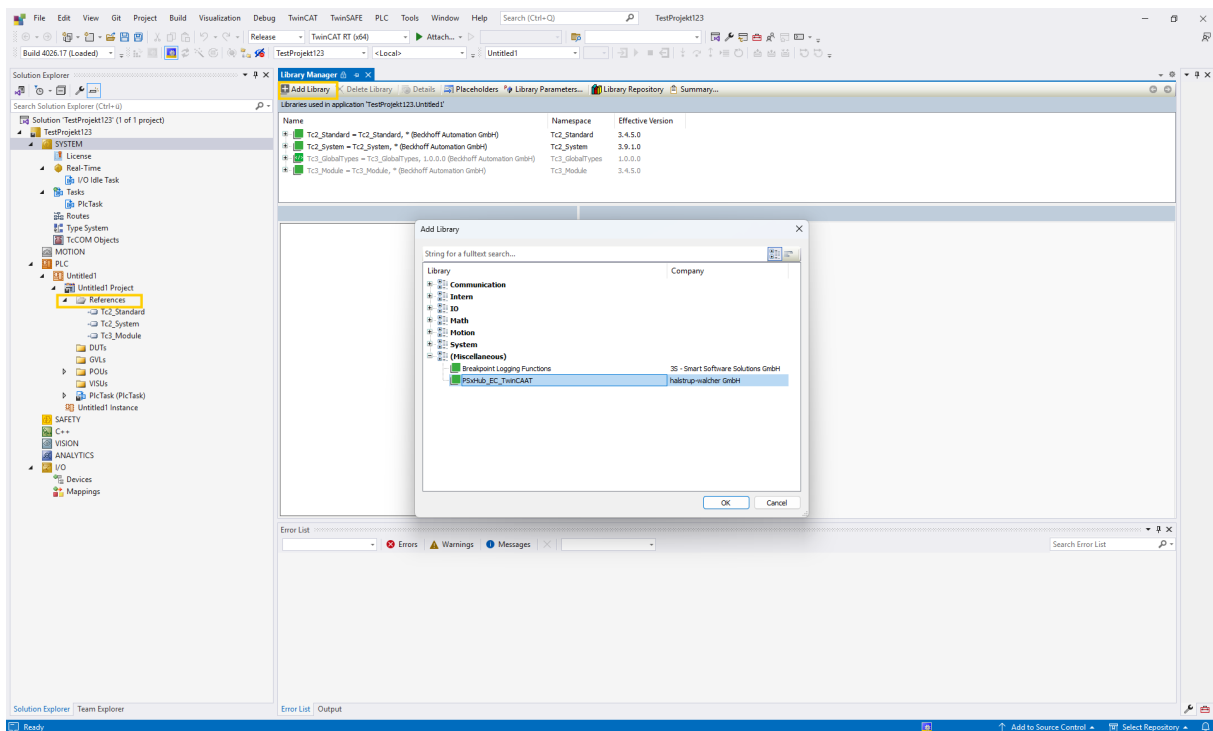


Figure 21 Inserting the library

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

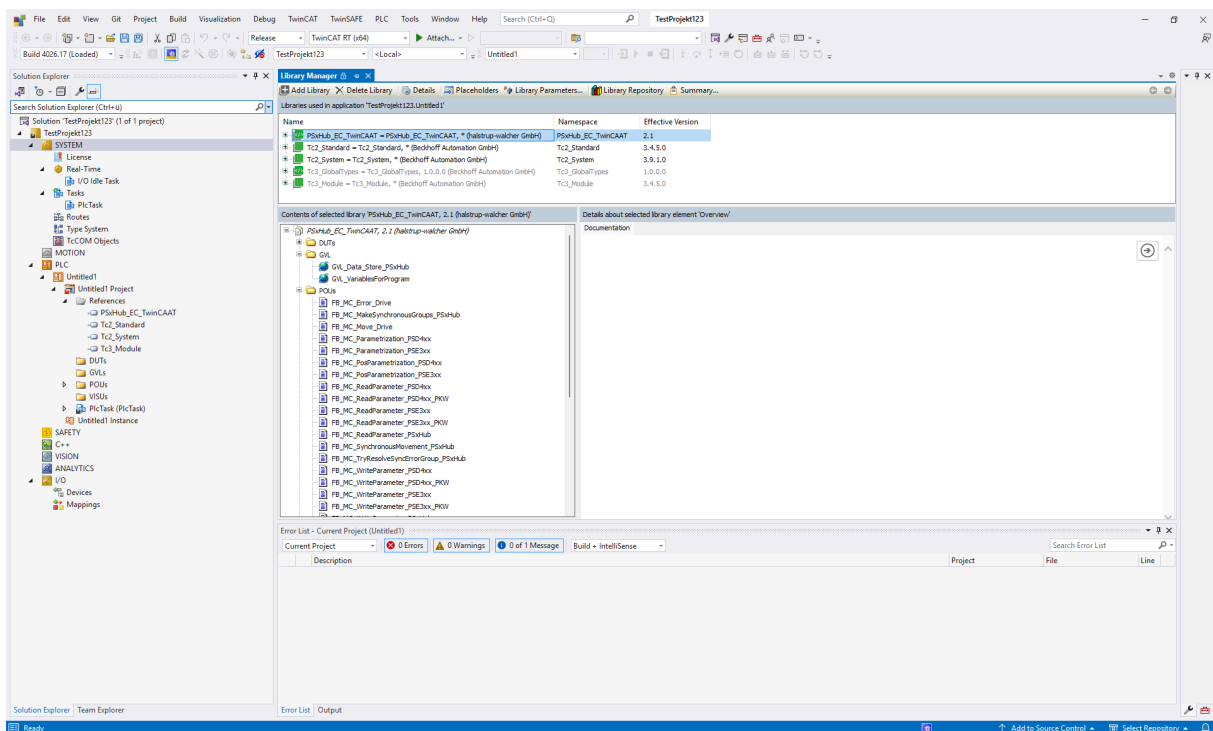


Figure 22: files contained in the library

Depending on the required range of functions, all or only a subset of the blocks can be integrated into the project. Note that some blocks require the presence of other blocks (see 7.5 Dependencies between function blocks).

In addition, the global variable list <GVL_Data_Store_PSD4xx> can be adopted as a template for the connection to the function blocks. However, all data types (DUTs_halstrup_walcher) are required for this.

The global variable list <GVL_VariablesForProgram> should be used for integrating the example programs.

7.3 Interlocking between the function blocks

Some of the blocks are interlocked against each other. This ensures, for example, that two accesses from different FBs of a drive cannot be carried out at the same time.

The following rules apply to the blocks of a drive:

- If the input <xExecute> of **FB_MC_Move_Drive** is set, the blocks **FB_MC_Parametrization_PSD4xx** and **FB_MC_PosParametrization_PSD4xx** cannot be activated (<udiErrorID>: 16#1000).
- However, it is possible to call **FB_MC_ReadParameter_PSD4xx** or **FB_MC_WriteParameter_PSD4xx** during a movement (e.g. to read the current torque or to change the target speed during movement).
- If the input <xExecute> of **FB_MC_Parametrization_PSD4xx** or **FB_MC_PosParametrization_PSD4xx** is set, the block **FB_MC_Move_Drive** cannot be activated (<udiErrorID>: 16#01000).
- Only one of the blocks **FB_MC_ReadParameter_PSD4xx**, **FB_MC_WriteParameter_PSD4xx**, **FB_MC_Parametrization_PSD4xx** or **FB_MC_PosParametrization_PSD4xx** can be active at any time. If the input <xExecute> of one of these blocks is set and the input <xExecute> of another block is set, the latter outputs the error 16#1000.
- The block **FB_MC_Error_Drive** can always be active independently of the other blocks.

Of the blocks

- **FB_MC_MakeSynchronousGroups_PSxHub**
- **FB_MC_SynchronousMovement_PSxHub**
- **FB_MC_TryResolveSyncErrorGroup_PSxHub**
- **FB_MC_ReadParameter_PSxHub**
- **FB_MC_WriteParameter_PSxHub**

only one may be active at any time. If another block is activated, it outputs the error 16#1000.

7.4 Several PSxHubs in one project

The example projects contain only one PSxHub. If several PSxHubs are used, the project must be adapted accordingly.

For example, for three PSxHubs with 10 connected drives each:

1. Initialize instances for hub as required
 - three instances of type **FB_MC_WriteParameter_PSxHub**, e.g.
 - fbMC_WriteParameter_PSxHub_1
 - fbMC_WriteParameter_PSxHub_2
 - fbMC_WriteParameter_PSxHub_3
 - three instances of type **FB_MC_ReadParameter_PSxHub**, e.g.
 - fbMC_ReadParameter_PSxHub_1
 - fbMC_ReadParameter_PSxHub_2
 - fbMC_ReadParameter_PSxHub_3
 - further required instances
2. Initialize instances for each connected drive as required
 - Thirty instances of type **FB_MC_Move_Drive**
 - fbMC_Move_PSD4xx_1
 - ...
 - fbMC_Move_PSD4xx_30
 - further required instances
3. The global variable list <GVL_Data_Store_PSxHub> is then adapted. Three variables of type <ST_Variables_PSxHubAndDrive> are created in it, e.g. with the following designations:
 - 'Hub_1'
 - 'Hub_2'
 - 'Hub_3'
4. If the PlainData modules are used, the function **FUN_MC_Communication_Drive** must be called for each drive
5. Finally, the variables of <GVL_Data_Store_PSxHub> must be assigned to the function blocks. (see Chapter 5 Description of the function blocks)

7.5 Dependencies between function blocks

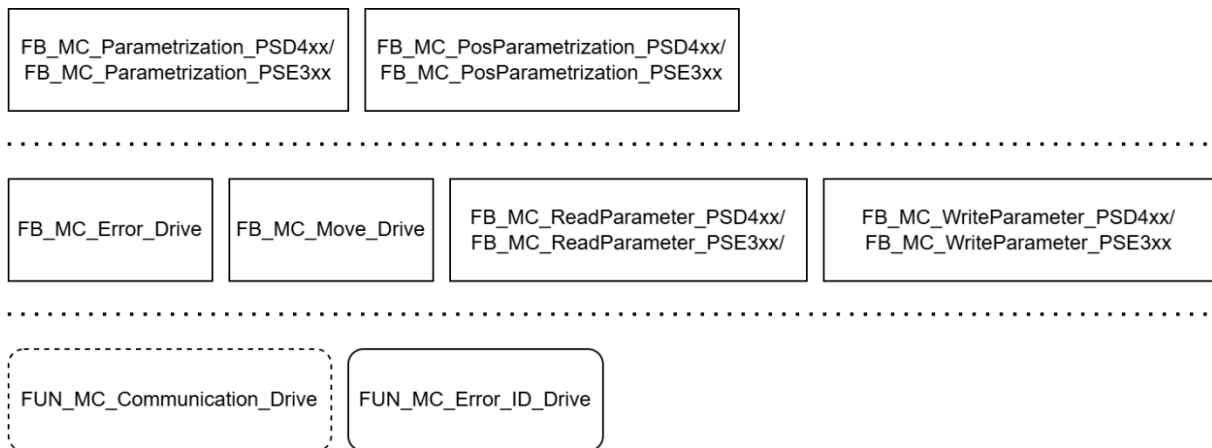


Figure 23: Subdivision of modules and functions for communicating with drives by complexity
(FUN_MC_Communication_Drive is optional)

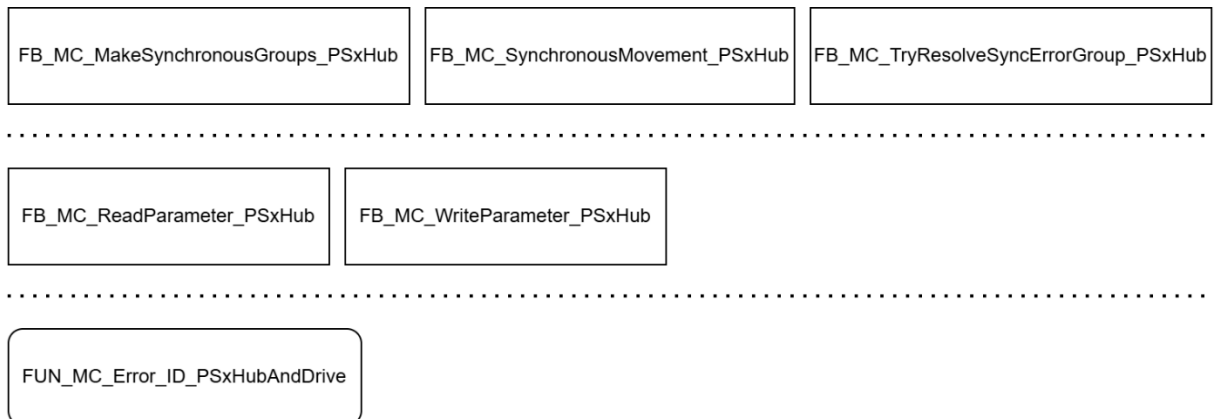


Figure 24: Subdivision of modules and functions for communicating with the hub by complexity

Depending on the scope of functions of the PLC application, not all blocks are required in every project. Unused blocks can be removed if no other blocks build on their function. Figure 23 shows a rough subdivision of the function blocks for controlling the drives according to complexity. Figure 24 shows this for the blocks for controlling the hub-specific functions. Higher levels can easily be removed without impairing the functionality of the underlying levels.

Higher levels have dependencies on at least one of the parts of the underlying levels; if these are removed, this leads to errors in the build process.

In many projects, the hub-specific functions are not required and the blocks can be deleted from the project.

8 Example programs

The example programs can be used to better understand the use of the function blocks.

Example programs can be found in the project tree to illustrate how the function blocks are to be used. The examples are programmed in structured text and the respective program must be inserted into the main program in order to be able to call the program.

```
//Currently this is configured to allow manual forcing of the inputs using the GVL_Data_Store_PSxHub
//if you like to use one of the examples, comment out everything but the function call
//the examples are controlled using GVL_VariablesForProgram

//PRG_Example_1_Positioning_Mode();
//PRG_Example_2_Manual_Mode();
//PRG_Example_3_Read_Current_Position();
//PRG_Example_4_Write_Delivery_State();
//PRG_Example_5_Parameterize_Parameters();
//PRG_Example_6_Parameterize_Position_Parameters();
//PRG_Example_7_Error_Upper_Position_Limit_Exceeded();
//PRG_Example_8_Make_Synchronous_Group();
//PRG_Example_9_Handle_Sync_Deviation();
Instances();
```

Figure 25: Program called from the main program

Common features 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 the program.
- For every example program, the first step after starting the program is that all relevant variables are set to FALSE for initialization.
- The programs can be ended with <xStopProgram>.

8.1 Example 1: Positioning mode

This example program presents the positioning mode of the FB **FB_MC_Move_Drive**. In this example, the drive at Port 1 runs an endless loop between the values 20000 and 10000.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The target position of 20000 and the move command are set. If the current position is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	The target position of 10000 and the move command are set. If the current position is at 10000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is set to 1.

8.2 Example 2: Manual mode

This example program presents the manual mode of the FB **FB_MC_Move_Drive**. In this example, the drive at Port 1 moves to a start position using the positioning mode and then with the manual mode.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	In manual mode, the drive moves over the target of 30000. Only when the current position equals 30000 is the move command for the manual run reset and the state machine set to 100.

8.3 Example 3: Read current position

This example program presents the FB **FB_MC_ReadParameter_PSD4xx**.

In this example, the drive at Port 1 moves to a target position and the parameter 'current position' is read.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	The parameter 'current position' is read. If the value is at 20000 ± 2 (see parameter 'positioning window'), the read command is reset and the state machine is set to 100.

8.4 Example 4: Write factory default state

This example program presents the FB **FB_MC_WriteParameter_PSD4xx**. In this example, the parameter 'factory default state' is written and the drive at Port 1 then moves to the center of its range.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	The value of the parameter 'factory default state' is written to -5. The drive is reset to its factory default state, the parameters are set to the default value and, in addition, positioning to the center of the measuring range (position 51200) takes place. If the write command was carried out successfully, the write command is reset and the state machine is incremented by one. (The drive still moves to the center of the measuring range in the process.)

8.5 Example 5: Parameterize parameters

This example program presents the FB **FB_MC_Parametrization_PSD4xx**. In this example, the parameters 'loop length' and 'target speed' of the drive at Port 1 are parameterized and then a positioning movement is carried out.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	The parameters 'loop length' and 'target speed' are parameterized. The <xEnable> of the parameters must be set. If the parametrization command was carried out successfully, the parametrization command is reset and the state machine is incremented by one.
3	The target position of 10000 and the move command are set. During this positioning movement no loop run is performed and the drive moves at 20 1/min. If the current value is at 10000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is set to 100.

8.6 Example 6: Parameterize position parameters

This example program presents the FB **FB_MC_PosParametrization_PSD4xx**. The drive at Port 1 is parameterized and the parameter 'current actual position' is read.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	All position parameters are parameterized. The parameter 'actual position' is set to 0, the parameter 'upper limit' to 10000 and the parameter 'lower limit' to -10000. The remaining parameters are not changed and the parameter values are not to be saved. If the parametrization command of the position parameters was carried out successfully, the parametrization command of the position parameters is reset and the state machine is incremented by one.
3	The parameter 'current position' is read. If the value is at 0 ± 2 , the read command is reset and the state machine is set to 100.

8.7 Example 7: Error upper limit reached

This example program presents the FB **FB_MC_Error_Drive**. The drive at Port 1 is moved to the upper limit and the error message is read out.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	The parameter 'upper limit' is written to the value 25000. If the write command was carried out successfully, the write command is reset and the state machine is incremented by one.
3	The function block FB_MC_Error_Drive is activated in order to detect errors.
4	In manual mode the upper limit is approached. Only when the current position equals 25000 is the error description displayed as a string. After that the move command for the manual run is reset and the index is set to 100. (It would also be possible to obtain this error from the FB FB_MC_Move_Drive (<udiErrorID>). The <udiErrorID> would then be 0x100B0.)

8.8 Example 8: Creation of groups with synchronization monitoring

This example program presents the functionality of the blocks

FB_MC_MakeSynchronousGroups_PSxHub and **FB_MC_SynchronousMovement_PSxHub**. The drives at ports 1&2 are added to a synchronization monitoring group and moved.

Program flow

Case	Description
0	All relevant variables are set to FALSE for initialization and the state machine is incremented by one. (It is always advisable at the start of a program to set all relevant variables to the initial value.)
1	The start position of 20000 and the move command are set for drive 1. If the current value is at 20000 ± 2 (see parameter 'positioning window') and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
2	The start position of 20000 and the move command are set for drive 2. If the current value is at 20000 ± 2 and the movement was carried out successfully, the move command is reset and the state machine is incremented by one.
3	Ports 1&2 are added to synchronization monitoring group 1 by means of FB_MC_MakeSynchronousGroups_PSxHub , and the step limit for triggering monitoring is set to 30. If the drives have been successfully added to the group, the write command is reset and the state machine is incremented by one.
4	Automatic stop is activated in case the step limit is exceeded, both drives are given the target position 10000 via FB_MC_SynchronousMovement_PSxHub and the move command is set. If the actual value of each drive equals 10000 ± 2 and neither the SynchronousMovement nor the Move blocks report errors, the move command is reset and the state machine is incremented by 1.

8.9 Example 9: Error handling when synchronization monitoring is triggered

The functionality of the blocks **FB_MC_TryResolveSyncErrorGroup_PSxHub** and **FB_MC_SynchronousMovement_PSxHub** is presented in this example program. The drives at ports 1&2 are added to a synchronization monitoring group and moved.

Program flow

Case	Description
INIT (0)	All relevant xExecute signals of the function blocks (Move, WriteParameter, MakeSynchronousGroups, TryResolveSyncErrorGroup, SynchronousMovement) are set to FALSE . The automatic stop function is deactivated. After that, the state machine is incremented by 1.
1	Drive 1 receives the target position 20000 and the move command. When the drive has reached the position 20000 ± 2 , no error is present and the Move FB reports 'Done', the move command is reset and the state machine is incremented.
2	Drive 2 also receives the target position 20000 and the move command. As soon as 20000 ± 2 has also been reached here and the movement has been completed without error, the command is reset, the synchronization error is reset as well, and the state machine is incremented.
3	Drive 1 is configured to 20 1/min via the parameter target speed positioning (137) using FB_MC_WriteParameter_PSD4xx. As soon as the write operation is complete, xExecute is reset and advanced.
4	Drive 2 is likewise set to 40 1/min via the parameter target speed positioning (137) in order to deliberately provoke a position deviation during the synchronous movement. After successful writing, xExecute is reset and advanced.
5	For drive 1 the parameter loop length (124) is set to 0 so that no large position jumps due to the loop run can occur during the alignment of the positions. After successful writing, the sequence advances.
6	For drive 2 the parameter loop length (124) is also set to 0. After successful writing, the sequence advances.
7	Ports 1&2 are added to synchronization monitoring group 1 by means of FB_MC_MakeSynchronousGroups_PSxHub , and the step limit for triggering monitoring is set to 30. If the drives have been successfully added to the group, the write command is reset and the state machine is incremented by one.
SYNC_MOVE (8)	Automatic stop is activated. The target position 10000 is assigned to the group via FB_MC_SynchronousMovement_PSxHub and the move command is set. - If a synchronization error is detected and both motors are at standstill, the program jumps to the RESOLVE_SYNC_ERROR state. When both drives have reached 10000 ± 2 , no error is present and both Move FBs report 'Done', the move command is reset and the program ends.
RESOLVE_SYNC_ERROR (9)	An alignment of the motor positions is initiated by means of FB_MC_TryResolveSyncErrorGroup_PSxHub , whereby the higher drive is moved to the value of the lower one. When both drives report 'Done' and none of the blocks reports an error, the program switches back to the state SYNC_MOVE.

List of figures

Figure 1: sample configuration	6
Figure 2: reading from uiNodeAddress	7
Figure 3: reading from sTargetNetId.....	7
Figure 4: reading from sMasterNetId.....	7
Figure 5: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_ReadParameter_PSxHub	11
Figure 6: <GVL_Data_Store_PSxHub> with variables for the drive.....	11
Figure 7: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_ReadParameter_PSxHub	14
Figure 8: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_WriteParameter_PSxHub.....	14
Figure 9: Assignment of variable from <GVL_Data_Store_PSxHub> to <FB_MC_MakeSynchronousGroups_PSxHub>.....	14
Figure 10: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_SynchronousMovement_PSxHub.....	14
Figure 11: Assignment from variable <GVL_Data_Store_PSxHub> to FB_MC_TryResolveSyncErrorGroup_PSxHub	15
Figure 12: Assignment from variable <GVL_Data_Store_PSxHub> to FB_MC_Move_PSxHub_PSD4xx.....	16
Figure 13: Assignment from variable <GVL_Data_Store_PSxHub> to FB_MC_Error_PSxHub_Drive	16
Figure 14: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_ReadParameter_PSxHub_PSD4xx	17
Figure 15: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_WriteParameter_PSxHub_PSD4xx.....	17
Figure 16: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_Parametrization_PSxHub_PSD4xx.....	17
Figure 17: parameter ,loop length' set to 0, Parameter ,targetSpeed' to 20	17
Figure 18: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_Parametrization_PSE3xx.....	18
Figure 19: Assignment of variable from <GVL_Data_Store_PSxHub> to FB_MC_PosParametrization_PSxHub_PSD4xx.....	19
Figure 20: Installation of the library in the library repository	36
Figure 21 Inserting the library.....	37
Figure 22: files contained in the library	37
Figure 23: Subdivision of modules and functions for communicating with drives by complexity (FUN_MC_Communication_Drive is optional).....	40
Figure 24: Subdivision of modules and functions for communicating with the hub by complexity.....	40
Figure 25: Program called from the main program.....	41

List of tables

Table 1 Structure of the ST_DriveData_PSxHub data type	6
Table 2 Assignment of hub process data to stPrivate	8
Table 3 Structure of the ST_DriveData_PSxHub_PSD4xxx data type.....	8
Table 4 Linking of the process data with stPrivate, modules: PSD/PSE	9
Table 5 Linking of the process data with stPrivate, modules PSD_PlainData/PSE_PlainData.....	9
Table 6 data structure ST_FunctionBlocks_PSxHub.....	9
Table 7 data structure ST_FunctionBlocks_PSxHub_PSD4xx	10
Table 8: Error codes	13
Table 9: linking process data with stPrivate.....	15
Table 10 drive positioning	16
Table 11 Condition and ErrorMessage	19