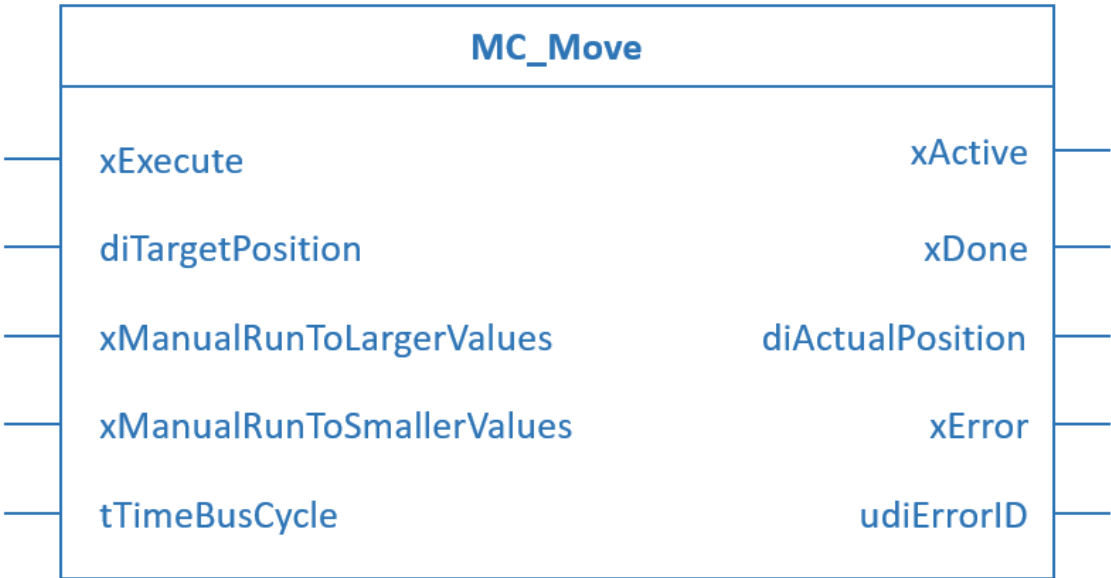




Bedeutung der Betriebsanleitung

Diese Betriebsanleitung beschreibt die Funktionsbausteine für den PSxHub EC (mit EtherCAT-Schnittstelle).

Von diesen Geräten können für Personen und Sachwerte Gefahren durch nicht bestimmungsgemäße Verwendung und durch Fehlbedienung ausgehen. Deshalb muss jede Person, die mit der Handhabung der Geräte betraut ist, eingewiesen sein und die Gefahren kennen. Die Betriebsanleitung und insbesondere die darin gegebenen Sicherheitshinweise müssen sorgfältig beachtet werden.

Wenden Sie sich unbedingt an den Hersteller, wenn Sie Teile davon nicht verstehen.

Der Hersteller behält sich das Recht vor, die Funktionsbausteine weiterzuentwickeln, ohne dies in jedem Einzelfall zu dokumentieren. Über die Aktualität dieser Betriebsanleitung gibt Ihnen Ihr Hersteller gerne Auskunft.

Das Urheberrecht an dieser Betriebsanleitung verbleibt beim Hersteller. Sie enthält technische Daten, Anweisungen und Zeichnungen zur Funktion und Handhabung der Software. Sie darf weder ganz noch in Teilen vervielfältigt oder Dritten zugänglich gemacht werden.

halstrup-walcher GmbH
Stegener Straße 10-12
79199 Kirchzarten

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

© 2026

Originalbetriebsanleitung

Inhaltsverzeichnis

1	Sicherheitshinweise	5
1.1	Bestimmungsgemäße Verwendung	5
1.2	Warnsymbole	5
2	Benutzerspezifische Datentypen	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>	11
4	Fehlerbeschreibung (<udiErrorID>)	12
5	Beschreibung der Funktionsbausteine	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	15
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	16
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	18
5.15	FB_MC_PosParametrization_PSxHub_PSE3xx	19
6	Parameterbeschreibung	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>	29
6.23	<xManualRunToSmallerValues>	29
6.24	<stParameter>	30
6.25	<uiParameterNumber>	30
6.26	<xSaveSettings>	31
6.27	<diSetPoint>.....	31
6.28	<uiStepsPerTurn>	32
6.29	<usiSubIndex>	32
6.30	<diTargetPosition>	32
6.31	<tTimeBusCycle>	33
6.32	<diUpperLimit>.....	33
6.33	<diValue>.....	34
7	Anwendung der Funktionsbausteine	35
7.1	Projekt.....	35
7.2	Bibliothek	35
7.3	Sperrung zwischen den Funktionsbausteinen	37
7.4	Mehrere PSxHubs in einem Projekt	38
7.5	Abhängigkeiten zwischen Funktionsbausteinen	39
8	Beispielprogramme.....	40
8.1	Beispiel 1: Positioniermodus.....	40
8.2	Beispiel 2: Handfahrmodus.....	41
8.3	Beispiel 3: Aktuelle Position lesen.....	41
8.4	Beispiel 4: Auslieferungszustand schreiben	42
8.5	Beispiel 5: Parameter parametrieren.....	42
8.6	Beispiel 6: Positionsparameter parametrieren	43
8.7	Beispiel 7: Fehler obere Endbegrenzung erreicht.....	43
8.8	Beispiel 8: Erstellung von Gruppen mit Gleichlaufüberwachung	44
8.9	Beispiel 9: Fehlerbehandlung bei ausgelöster Gleichlaufüberwachung.....	45
	Abbildungsverzeichnis.....	46
	Tabellenverzeichnis	46

1 Sicherheitshinweise

1.1 Bestimmungsgemäße Verwendung

Der PSxHub EC fungiert als EtherCAT Gerät, über welches sich bis zu 10 PSD IO-Link Antriebe versorgen und ansteuern lassen. Die Prozessdaten der einzelnen Antriebe sind in den Prozessdaten des Hubs abgebildet.

1.2 Warnsymbole

In dieser Betriebsanleitung wird mit folgenden Hervorhebungen auf die darauffolgend beschriebenen Gefahren bei der Handhabung der Anlage hingewiesen:

 GEFAHR!	GEFAHR! Bei Nichtbeachtung dieses Sicherheitshinweises werden Tod oder schwere Körperverletzung eintreten.
 WARNUNG!	WARNUNG! Bei Nichtbeachtung dieses Sicherheitshinweises können Tod oder schwere Körperverletzung eintreten.
 VORSICHT!	VORSICHT! Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körperverletzung eintreten.
HINWEIS	HINWEIS Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten.

2 Benutzerspezifische Datentypen

Es gibt viele benutzerspezifische Datentypen. Im Folgenden werden nur die drei wichtigsten Datentypen dokumentiert.

2.1 <ST_DriveData_PSxHub>

Für jeden Hub gibt es eine Datenstruktur, in der einige Daten des Hubs abgelegt sind. Für jeden Hub wird eine globale Instanz dieser Struktur benötigt. Diese Instanz muss jedem Funktionsbaustein (FB) übergeben werden, der auf den Hub selbst wirkt (ohne _PSD4xx am Ende). Hiermit soll z.B. sichergestellt werden, dass nicht gleichzeitig zwei Zugriffe aus unterschiedlichen FBs auf den Parameterkanal durchgeführt werden können. Des Weiteren müssen in dieser Datenstruktur die Adressen zu den Ein- und Ausgangsdaten des jeweiligen Hubs hinterlegt werden.

Parametername	Datentyp	geschrieben von	Beschreibung
<uiNodeAddress>	UInt	Benutzer	EtherCAT Adresse
<sTargetNetId>	T_AmsNetId	Benutzer	NetId Subdevices
<sMasterNetId>	T_AmsNetId	Benutzer	NetId Hub
<sAxisName>	String[16]	Benutzer (optional)	Name der Achse
<sAxisDescription>	String[32]	Benutzer (optional)	Beschreibung (z.B. Funktion, Aufgabe dieser Achse)
<iActiveFunktionBlock>	Int	Funktionsbausteine	Aktiver Funktionsbaustein
<xAutomaticStop>	Bool	Benutzer	Motoren werden bei Gleichlauffehler angehalten
<xResetSynchronousMovementErrorBit>	Bool	Benutzer	Setzt Gleichlauffehler-Bit zurück
<xCommunicationError>	Bool	Funktionsbausteine	Kommunikationsfehler zum IO-Device
<stPrivate>	ST_Private	Funktionsbausteine/ Verknüpfung durch Nutzer	Datenstruktur zur internen Verwendung

Tabelle 1: Aufbau der Datenstruktur ST_DriveData_PSxHub

Die Werte für uiNodeAddress, sTargetNetId und sMasterNetId sind abhängig von der Konfiguration im Projekt und sollten im Online-Zustand ausgelesen werden. Im Folgenden ist die Zuordnung exemplarisch dargestellt.

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	

Abbildung 1 Beispielkonfiguration

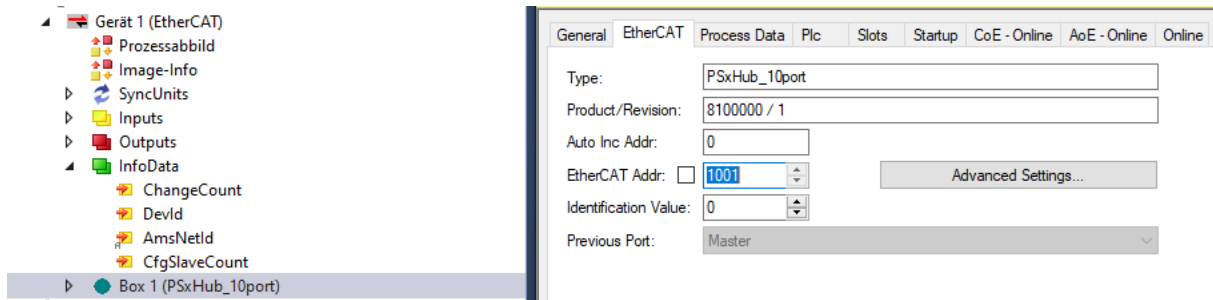


Abbildung 2 Auslesen von *uiNodeAddress*

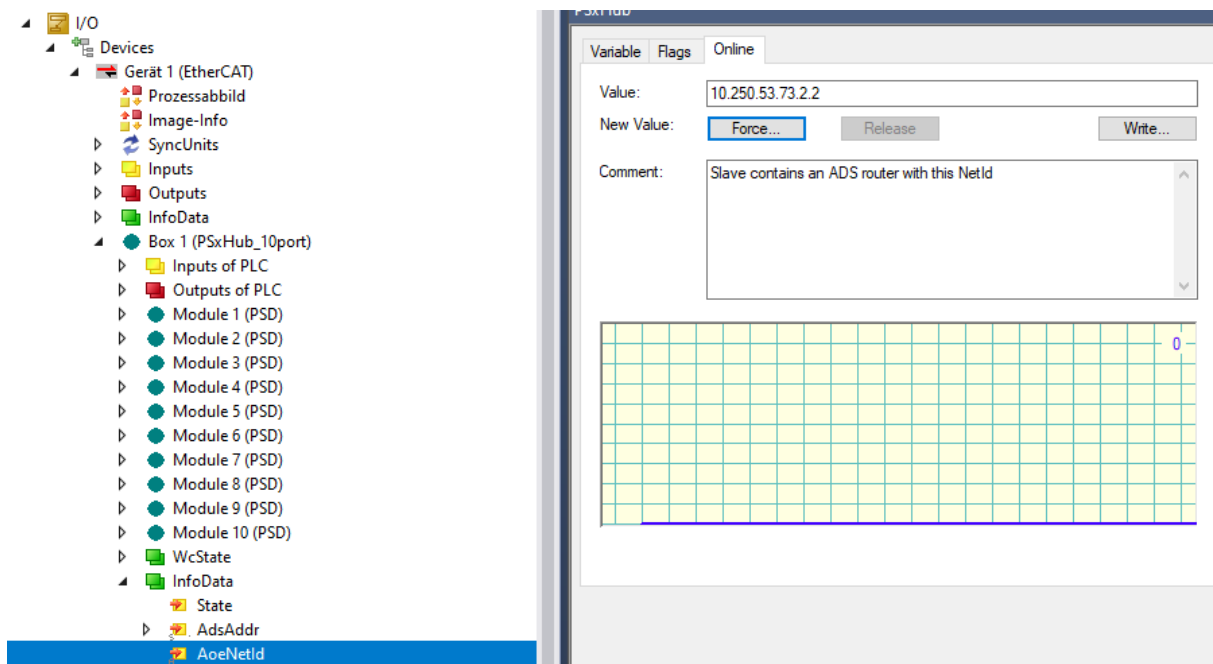


Abbildung 3 Auslesen von *sTargetNetId*

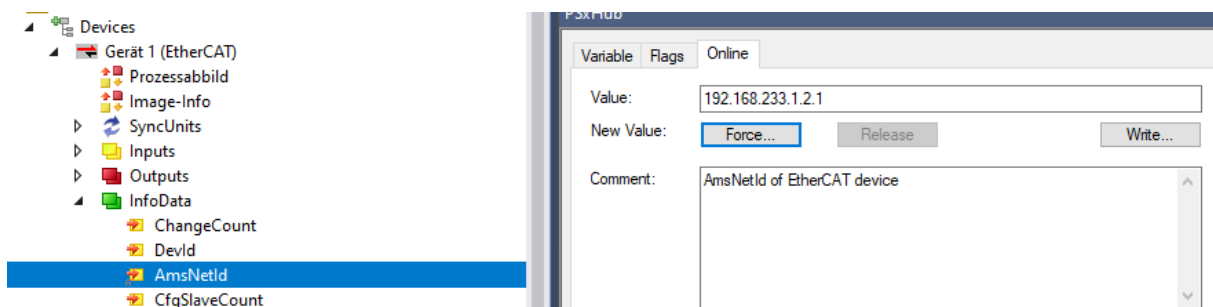


Abbildung 4 Auslesen von *sMasterNetId*

Die Verknüpfung der Prozessdaten mit stPrivate ist Tabelle 1 zu entnehmen.

Bemerkung	Name in Prozessdaten	Name in stPrivate
	StatusWord1	IudiStatus_1
	StatusWord2	IudiStatus_2
Aktuell nicht genutzt	PKW_PKE_in	
Aktuell nicht genutzt	PKW_IND_in	
Aktuell nicht genutzt	PKW_PWE_in	
	ControlWord1	OudiControlWord_1
	ControlWord2	OudiControlWord_2
Aktuell nicht genutzt	PKW_PKE_out	
Aktuell nicht genutzt	PKW_IND_out	
Aktuell nicht genutzt	PKW_PWE_out	

Tabelle 2 Zuordnung Hub Prozessdaten zu stPrivate

2.2 <ST_DriveData_PSxHub_PSD4xx>

Für jeden Antrieb gibt es eine Datenstruktur, in der einige Daten eines Antriebs abgelegt sind. Für jeden Antrieb wird eine globale Instanz dieser Struktur benötigt. Diese Instanz muss jedem Funktionsbaustein (FB) übergeben werden, der auf den betriebenen Antrieb wirkt. Hiermit soll z.B. sichergestellt werden, dass nicht gleichzeitig zwei Zugriffe aus unterschiedlichen FBs auf den Parameterkanal durchgeführt werden können. Des Weiteren müssen in dieser Datenstruktur die Adressen zu den Ein- und Ausgangsdaten des jeweiligen Antriebs hinterlegt werden.

Parametername	Datentyp	geschrieben von	Beschreibung
<sAxisName>	String[16]	Benutzer (optional)	Name der Achse
<sAxisDescription>	String[32]	Benutzer (optional)	Beschreibung (z.B. Funktion, Aufgabe dieser Achse)
<iActiveFunktionBlock>	Int	Funktionsbausteine	Aktiver Funktionsbaustein
<xCommunicationError>	Bool	Funktionsbausteine	Kommunikationsfehler zum IO-Device
<stPrivate>	ST_Private	Funktionsbausteine/ Verknüpfung durch Nutzer	Datenstruktur zur internen Verwendung

Tabelle 3: Aufbau der Datenstruktur ST_Drive

Es gibt 2 Varianten zur Verknüpfung der Prozessdaten mit stPrivate. Tabelle 2 enthält die Beschreibung zur Verwendung der Module PSD bzw. PSE, Tabelle 3 die Beschreibung zur Verwendung der Module PSD_PlainData bzw. PSE_PlainData. Beide Verknüpfungen ermöglichen dieselbe Funktionalität, die Verwendung der PlainData Module reduziert lediglich die Anzahl der notwendigen Verknüpfungen, erfordert allerdings für jeden Antrieb einen Aufruf der Funktion FUN_MC_Communication_Drive zur Übersetzung.

Die PKW-Schnittstelle ist lediglich als Alternative zum Zugriff über ADS vorgesehen, da dies von einigen Steuerungen (z.B. Lenze) nicht unterstützt wird. In den meisten Anwendungsfällen wird es nicht benötigt und die entsprechenden Bausteine (_PKW) können gelöscht werden.

Bemerkung	Name in Prozessdaten	Name in stPrivate
Nur wenn PKW genutzt wird	PKE_in	luiPKE
Nur wenn PKW genutzt wird	IND_in	luiIND
Nur wenn PKW genutzt wird	PWE_in	ludiPWE
	StatusWordNode	unStatus.uiStatusRaw
	ActualSpeedNode	iCurrentRPM
	ActualPositionNode	diCurrentPosition
	PKE_out	OuiPKE
	IND_out	OuiIND
	PWE_out	OudiPWE
	ControlWordNode	unControl.uiControlRaw
	TargetPositionNode	diTargetPosition
Nur relevant für Antriebe mit Softwaremodul „P“ oder „Z“	TargetSpeedNode	iTargetRPM

Tabelle 4 Verknüpfung der Prozessdaten mit stPrivate, Module: PSD/PSE

Bemerkung	Name in Prozessdaten	Name in stPrivate
Nur wenn PKW genutzt wird	PKW_In	stInputDataRow[0]
	ProcessData_In	stInputDataRow[1]
Nur wenn PKW genutzt wird	PKW_Out	stOutputDataRow[0]
	ProcessData_Out	stOutputDataRow[1]

Tabelle 5 Verknüpfung der Prozessdaten mit stPrivate, Module PSD_PlainData/PSE_PlainData

2.3 <ST_FunctionBlocks_PSxHub>

Für jeden PSxHub gibt es eine Datenstruktur, in der die Variablen für die Zuweisung an die Funktionsbausteine abgelegt sind. Diese Struktur hat mehrere Unterstrukturen, die in der Dokumentation nicht weiter ausgeführt werden. Für jeden Antrieb wird eine globale Instanz dieser Struktur benötigt. Um unnötige Speicherbelegung in der SPS zu vermeiden, sollte die Datenstruktur angepasst werden, wenn Funktionsbausteine nicht genutzt werden.

Parametername	Datentyp	geschrieben von	Beschreibung
<stSynchronousMovement_PSxHub>	ST_SynchronousMovement_PSxHub	Funktionsbausteine	Variablen für FB_MC_SynchronousMovement_PSxHub
<stMakeSynchronousGroups_PSxHub>	ST_MakeSynchronousGroups_PSxHub	Funktionsbausteine	Variablen für FB_MC_MakeSynchronousGroups_PSxHub
<stReadParameter_PSxHub>	ST_ReadParameter_PSxHub	Funktionsbausteine	Variablen für FB_MC_ReadParameter_PSxHub
<stWriteParameter_PSxHub>	ST_WriteParameter_PSxHub	Funktionsbausteine	Variablen für FB_MC_WriteParameter_PSxHub
<stTryResolveSyncErrorGroup_PSxHub>	ST_TryResolveSyncErrorGroup_PSxHub	Funktionsbausteine	Variablen für FB_MC_TryResolveSyncErrorGroup_PSxHub

Tabelle 6: Datenstruktur ST_FunctionBlocks_PSxHub

2.4 <ST_FunctionBlocks_PSxHub_PSD4xx>

Für jeden Antrieb gibt es eine Datenstruktur, in der die Variablen für die Zuweisung an die Funktionsbausteine abgelegt sind. Diese Struktur hat mehrere Unterstrukturen, die in der Dokumentation nicht weiter ausgeführt werden. Für jeden Antrieb wird eine globale Instanz dieser Struktur benötigt. Bei Nichtbenutzung von Funktionsbausteinen ist empfehlenswert, die Datenstruktur anzupassen, damit nicht unnötig Speicher in der SPS belegt wird.

Parametername	Datentyp	geschrieben von	Beschreibung
<stMove_PSxHub_PSD4xx>	ST_Move_PSxHub_PSD4xx	Funktionsbausteine	Variablen für FB_MC_Move_PSxHub_PSD4xx
<stError_PSxHub_PSD4xx>	ST_Error_PSxHub_PSD4xx	Funktionsbausteine	Variablen für FB_MC_Error_PSxHub_PSD4xx
<stReadParameter_PSxHub_PSD4xx>	ST_ReadParameter_PSxHub_PSD4xx	Funktionsbausteine	Variablen für FB_MC_ReadParameter_PSxHub_PSD4xx
<stWriteParameter_PSxHub_PSD4xx>	ST_WriteParameter_PSxHub_PSD4xx	Funktionsbausteine	Variablen für FB_MC_WriteParameter_PSxHub_PSD4xx
<stParametrization_PSxHub_PSD4xx>	ST_Parametrization_PSxHub_PSD4xx	Funktionsbausteine	Variablen für FB_MC_Parametrization_PSxHub_PSD4xx
<stPosParametrization_PSxHub_PSD4xx>	ST_PosParametrization_PSxHub_PSD4xx	Funktionsbausteine	Variablen für FB_MC_PosParametrization_PSxHub_PSD4xx

Tabelle 7: Datenstruktur ST_FunctionBlocks_PSxHub_PSD4xx

2.5 <ST_Variables_PSxHub/PSxHub_PSD4xx>

In dieser Datenstruktur werden die Strukturen <ST_Variables_PSxHub> und ein Array mit 10 Einträgen aus <ST_Variables_PSxHub> gebündelt. Diese Struktur wird für jeden PSxHub in der globalen Variablenliste <GVL_Data_Store_PSxHub> angelegt.

3 <GVL_Data_Store_PSxHub>

Die GVL (globale Variablenliste) <GVL_Data_Store_PSxHub> stellt die notwendigen Variablen bereit, um alle Ein- und Ausgänge aller zur Verfügung stehender Funktionsbausteine zuweisen zu können.

Jede Variable ist aktuell nur einfach vorhanden. Bei mehreren PSxHub in einem Projekt muss die Anzahl der Variablen entsprechend erhöht werden.

Beispiel

```
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);
```

Abbildung 5 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_ReadParameter_PSxHub

Falls die GVL in das Projekt übernommen wird, ist es empfehlenswert, die nicht verwendeten Variablen zu löschen (damit nicht unnötig Speicher in der SPS belegt wird) und den Baustein auf die Anzahl der tatsächlich vorhandenen Antriebe anzupassen.

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	

Abbildung 6 <GVL_Data_Store_PSxHub> mit den Variablen für einen Antrieb

4 Fehlerbeschreibung (<udiErrorID>)

Nachfolgend sind die Fehlercodes beschrieben, die von den Funktionsbausteinen ausgegeben werden:

<udiErrorID> (hex)	Beschreibung
16#F0000 (Maske)	Funktionsbaustein
16#0xxxx	<u>Kein</u> Fehlercode
16#1xxxx	Fehler in FB_MC_Move_PSxHub_Drive
16#2xxxx	Fehler in FB_MC_Error_PSxHub_Drive
16#3xxxx	Fehler in FB_MC_ReadParameter_PSxHub_PSD4xx
16#4xxxx	Fehler in FB_MC_WriteParameter_PSxHub_PSD4xx
16#5xxxx	Fehler in FB_MC_Parametrization_PSxHub_PSD4xx
16#6xxxx	Fehler in FB_MC_PosParametrization_PSxHub_PSD4xx
16#7xxxx	Fehler in FB_MC_SynchronousMovement_PSxHub
16#8xxxx	Fehler in FB_MC_MakeSynchronousGroups_PSxHub
16#9xxxx	Fehler in FB_MC_ReadParameter_PSxHub
16#Axxxx	Fehler in FB_MC_WriteParameter_PSxHub
16#Bxxxx	Fehler in FB_MC_TryResolveSyncErrorGroup_PSxHub
16#0F000 (Maske)	Interne Fehler in den Funktionsbausteinen und Prozessdatenfehler
16#x0xxx	<u>Kein</u> interner Fehler in den Funktionsbausteinen und Prozessdatenfehler
16#x1xxx	Fehler in der Zustandsmaschine (Verriegelung)
16#x2xxx	Ungültige Eingangsadresse der Prozessdaten
16#x3xxx	Ungültige Ausgangsadresse der Prozessdaten
16#x4xxx	Kommunikationsfehler beim Lesen der Prozessdaten
16#x5xxx	Kommunikationsfehler beim Schreiben der Prozessdaten
16#x6xxx	Ungültiger Schreibbefehl
16#00F00 (Maske)	Fehler in den Parametern
16#xx0xx	<u>Kein</u> Fehlverhalten in den Parametern
16#xx1xx	Parameter: Kommunikationsauszeit (1000 ms)
16#xx2xx	Parameter: ungültige Parameternummer
16#xx3xx	Parameter: Parameternummer ist nur lesbar
16#xx4xx	Parameter: Wert ist unterhalb des Minimums oder oberhalb des Maximums
16#xx5xx	Parameter: ungültiger Subindex
16#xx6xx	Parameter: kein Array
16#xx7xx	Parameter: ungültiger Datentyp
16#xx8xx	Parameter: kein Schreibzugriff
16#xx9xx	Parameter: Anfrage kann wegen des aktuellen Betriebszustands nicht bearbeitet werden
16#xxAxx	Andere Fehler

<udiErrorID> (hex)	Beschreibung
16#000F0 (Maske)	Antriebsfehler
16#xxx0x	<u>Kein</u> Antriebsfehler
16#xxx1x	Schleppfehler
16#xxx2x	Unter- oder Überspannung der Motorversorgung
16#xxx3x	Positionierung wurde abgebrochen
16#xxx4x	Temperaturüberschreitung
16#xxx5x	Messsystemfehler
16#xxx6x	Positionierfehler (Blockieren)
16#xxx7x	Manuelles Verdrehen
16#xxx8x	Zielposition falsch
16#xxx9x	Unter- oder Überspannung der Motorspannung/ Ausfall der Spannungsüberwachung
16#xxxAx	Untere Endbegrenzung unterschritten
16#xxxBx	Obere Endbegrenzung überschritten
16#0000F (Maske)	Fehler PSxHub/Gleichlauf
16#xxxx0	<u>Kein</u> Fehler am PSxHub bzw. beim Gleichlauf
16#xxxx1	Gleichlauffehler Gruppe 1
16#xxxx2	Gleichlauffehler Gruppe 2
16#xxxx3	Gleichlauffehler Gruppe 3
16#xxxx4	Gleichlauffehler Gruppe 4
16#xxxx5	Gleichlauffehler Gruppe 5
16#xxxx6	Positionierfehler (Blockieren)
16#xxxx7	Unterspannung Steuerung
16#xxxx8	Überspannung Steuerung
16#xxxx9	Temperaturüberschreitung

Tabelle 8: Fehlercodes

Die Fehler der Kategorie „Antriebsfehler“ (Maske 16#000F0) sind eine Abbildung der Fehlerbits im Statuswort der Antriebe.

Beispiele

- Fahrauftrag (**FB_MC_Move_PSxHub_Drive**) mit falscher Zielposition
→ <udiErrorID> = 16#10080
- Parameter schreiben (**FB_MC_WriteParameter_PSxHub_PSD4xx**) mit ungültiger Parameternummer → <udiErrorID> = 16#40200

5 Beschreibung der Funktionsbausteine

Eine ausführliche Beschreibung aller Parameter ist im Kapitel 6. **Parameterbeschreibung** zu finden.

5.1 FB_MC_ReadParameter_PSxHub

Mit diesem FB können Werte von Parametern aus dem Hub ausgelesen werden.

```
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);
```

Abbildung 7 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_ReadParameter_PSxHub

5.2 FB_MC_WriteParameter_PSxHub

Mit diesem FB können Werte von Parametern in den Hub geschrieben werden.

```
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);
```

Abbildung 8 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_WriteParameter_PSxHub

5.3 FB_MC_MakeSynchronousGroups_PSxHub

Dieser Funktionsbaustein wird zur Zuweisung der Antriebe zu Gleichlaufgruppen verwendet. Es stehen die Gruppen 1-5 zur Auswahl.

```
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);
```

Abbildung 9 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_MakeSynchronousGroups_PSxHub

5.4 FB_MC_SynchronousMovement_PSxHub

Dieser Funktionsbaustein dient zum synchronen Verfahren der Gleichlaufgruppen.

```
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.PSxHub.stHubData_PSxHub);
```

Abbildung 10 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_SynchronousMovement_PSxHub

5.5 FB_MC_TryResolveSyncErrorGroup_PSxHub

Mit diesem Funktionsbaustein können aufgetretene Gleichlauferfehler ausgeglichen werden, in dem entweder auf die Position des höchsten oder tiefsten Antriebs der Gruppe gefahren wird.

```
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);
```

Abbildung 11 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum
FB_MC_TryResolveSyncErrorGroup_PSxHub

5.6 FUN_MC_Communication_Drive

Diese Funktion dient der Kommunikation zwischen Antrieben und Bausteinen, wenn die PlainData Versionen der Antriebe verwendet werden. In dieser Variante reduziert sich der Verknüpfungsaufwand entsprechend Tabelle 4.

Bemerkung	Name in Prozessdaten	Name in stPrivate
Nur wenn PKW genutzt wird	PKW_In	stInputDataRow[0]
	ProcessData_In	stInputDataRow[1]
Nur wenn PKW genutzt wird	PKW_Out	stOutputDataRow[0]
	ProcessData_Out	stOutputDataRow[1]

Tabelle 9 Verknüpfung der Prozessdaten mit stPrivate

Durch den Aufruf der Funktion werden die Daten in das jeweils notwendige Format gebracht.

5.7 FB_MC_Move_PSxHub_Drive

Dieser Funktionsbaustein wird zur Positionierung eines Antriebs verwendet.

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

Abbildung 12 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_Move_PSxHub_PSD4xx

Falls ein Antrieb mehrere Fehler meldet, wird die <udiErrorID> mit der höchsten Priorität ausgegeben. Dies gilt für alle FBs. Die Priorität der Ausgabe entspricht der Reihenfolge in der folgenden Tabelle (höchste Priorität hat 16#x1xxx):

<udiErrorID>	Beschreibung
16#x1xxx	Fehler in der Zustandsmaschine (Verriegelung)
16#x2xxx	Ungültige Eingangsadresse der Prozessdaten
16#x3xxx	Ungültige Ausgangsadresse der Prozessdaten
16#x4xxx	Kommunikationsfehler beim Lesen der Prozessdaten
16#x5xxx	Kommunikationsfehler beim Schreiben der Prozessdaten
16#xxx2x	Unter- oder Überspannung der Motorversorgung (STO-Freigabe inaktiv*)
16#xxx4x	Temperaturüberschreitung
16#xxx5x	Messsystemfehler (Messsystem- oder STO-Hardwarefehler*)
16#xxx8x	Zielposition falsch
16#xxx9x	Unter- oder Überspannung der Motorspannung/ Ausfall der Spannungsüberwachung
16#xxx6x	Positionierfehler (Blockieren)
16#xxx7x	Manuelles Verdrehen
16#xxxAx	Untere Endbegrenzung unterschritten
16#xxxBx	Obere Endbegrenzung überschritten
16#xxx3x	Positionierung wurde abgebrochen
16#xxx1x	Schleppfehler

Tabelle 10: Positionierung des Antriebs

*Falls das PSD4xx ein Gerät mit STO ist, dann muss die Fehlerbeschreibung in Klammern berücksichtigt werden! (<udiErrorID>: 16#xxx2x und 16#xxx5x)

5.8 FB_MC_Error_PSD4xx_Drive

Dieser FB gibt den Status des Antriebs und des FBs als Fehlerbit, Fehlerkennung (<udiErrorID>) und als Textausgabe aus.

```
//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);
```

Abbildung 13 Zuweisung der Variablen von <GVL_Data_Store_PSD4xx> zum FB_MC_Error_PSD4xx_Drive

5.9 FC_MC_Error_ID_PSD4xx/PSD4xx_PSD4xx

Diese Funktion wird intern als Unterfunktion der anderen Funktionsbausteine eingesetzt. Sie ist lediglich aus der Bibliothek in das Anwenderprogramm zu kopieren. Die Beschaltung findet schon intern statt.

5.10 FB_MC_ReadParameter_PSD4xx

Mit diesem FB können Werte von Parametern (ISDUs) aus einem Antrieb ausgelesen werden.

```
//fbMC_ReadParameter_PSD4xx_1 --> instance of FB_MC_ReadParameter_PSD4xx (1. PSD4xx)
fbMC_ReadParameter_PSD4xx_1(
  xExecute:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stInput_ReadParameter_PSD4xx.xExecute,
  uiParameterNumber:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stInput_ReadParameter_PSD4xx.uiParameterNumber,
  usiSubindex:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stInput_ReadParameter_PSD4xx.usiSubindex,
  xActive=> GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.xActive,
  xDone=> GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.xDone,
  xError=> GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.xError,
  udiErrorID=> GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.udiErrorID,
  diValue=> GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stReadParameter_PSD4xx.stOutput_ReadParameter_PSD4xx.diValue,
  stDrive:= GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stDriveData_PSD4xx,
  stHub := GVL_Data_Store_PSD4xx.Hub.PSD4xx[1].stHubData_PSD4xx);
```

Abbildung 14 Zuweisung der Variablen von <GVL_Data_Store_PSD4xx> zum FB_MC_ReadParameter_PSD4xx

5.11 FB_MC_WriteParameter_PSxHub_PSD4xx

Mit diesem FB können Parameterwerte (ISDUs) in einen Antrieb geschrieben werden.

```
//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);
```

Abbildung 15 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum
FB_MC_WriteParameter_PSxHub_PSD4xx

5.12 FB_MC_Parametrization_PSxHub_PSD4xx

Mit diesem FB können sämtliche Parameter (ISDUs) des Antriebs auf einmal geschrieben werden. Das ist vor allem für die Erstinbetriebnahme von Nutzen. Dabei wurde eine übersichtliche Struktur gewählt, da sämtliche Parameter als Block mit dem benutzerspezifischen Datentyp <ST_Parameter_PSD4xx> übergeben werden.

```
//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);
```

Abbildung 16 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum
FB_MC_Parametrization_PSxHub_PSD4xx

Folgendes ist bei der Nutzung des FBs zu beachten:

Zu jedem Parameter gibt es eine Variable <xEnable> und eine Variable <diValue> (abhängig von dem Datentyp des Parameters).

Beispiel

```
2: //parameterize length of loop and target speed
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stTargetSpeed_137.xEnable := TRUE;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stTargetSpeed_137.diValue := 20;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stLoopLength_124.xEnable := TRUE;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.stParameter.stLoopLength_124.diValue := 0;
GVL_Data_Store_PSxHub.Hub.PSD4xx[1].stFunctionBlocks_PSD4xx.stParametrization_PSD4xx.stInput_Parametrization_PSD4xx.xExecute := TRUE;
```

Abbildung 17 Parameter „Schleifenlänge“ auf 0 setzen, Parameter „Solldrehzahl Positionieren“ auf 20

- Durch das Setzen des <xEnable> = FALSE, wird der jeweilige Parameter nicht geschrieben.
- Wahlweise kann vor dem Setzen einzelner Parameter ein Auslieferungszustand angefordert werden. Dazu muss der Eingang <xDeliveryState> auf TRUE gesetzt werden. Dadurch werden die Werte aller Parameter auf den Auslieferungszustand gesetzt (zunächst ohne zu speichern).
- Wahlweise können die geschriebenen Werte am Ende auch gespeichert werden. Dazu muss der Eingang <xSaveSettings> auf TRUE gesetzt werden.
- Die Reihenfolge der Schreibzugriffe ist wie folgt:
<xDeliveryState>, Parameter 115, 116, 117, 68, ...und <xSaveSettings>.
- Bei einem Schreibfehler eines Parameters werden die nachfolgenden Parameter nicht mehr geschrieben und es erfolgt auch kein Speichern der Werte, falls der Eingang <xSaveSettings> gesetzt ist.

5.13 FB_MC_Parametrization_PSxHub_PSE3xx

Dieser Baustein entspricht in Bedienung und Verhalten dem Baustein FB_MC_Parametrization_PSxHub_PSD4xx.

Die Parameter der PSx3 Familie unterscheiden sich geringfügig von denen der PSD4 Familie, daher wird für den stParameter Eingang der Datentyp ST_Parameter_PSxHub_PSE3xx genutzt, um dem Anwender alle PSx3 Parameter verfügbar zu machen.

```
//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.PSxHub.stHubData_PSxHub);
```

Abbildung 18 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_Parametrization_PSE3xx

5.14 FB_MC_PosParametrization_PSxHub_PSD4xx

Mit diesem FB kann die Parametrierung der Positionsdaten vorgenommen werden (Parameter, die die angezeigte Istposition beeinflussen). Das ist vor allem für die Erstinbetriebnahme von Nutzen.

```
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);
```

Abbildung 19 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum
FB_MC_PosParametrization_PSxHub_PSD4xx

Folgendes ist bei der Nutzung des FBs zu beachten:

- Es müssen alle Werte gesetzt werden und die Werte müssen in einem sinnvollen Bezug zueinander stehen. Alle Bedingungen sind in der unten aufgeführten Tabelle beschrieben. Alle Werte werden verarbeitet, danach werden die folgenden Parameter in der angegebenen Reihenfolge geschrieben:
 1. Drehsinn (ISDU 115) → <uiDirection>
 2. Istwertbewertung Zähler (ISDU 116) → 400
 3. Istwertbewertung Nenner (ISDU 117) → <uiStepsPerTurn>
 4. Istwert (ISDU 68) → <diSetPoint>
 5. Falls (diSetPoint > diUpperLimit):
Oberes Mapping-Ende (ISDU 120) = <diSetPoint> + (3 x <uiStepsPerTurn>)
sonst:
Oberes Mapping-Ende (ISDU 120) = <diUpperLimit> + (3 x <uiStepsPerTurn>)
 6. Obere Endbegrenzung (ISDU 121) → <diUpperLimit>
 7. Untere Endbegrenzung (ISDU 122) → <diLowerLimit>

Nachfolgend sind die Bedingungen und die Fehlermeldungen aufgeführt, die bei nicht erfüllter Voraussetzung ausgegeben werden.

Bedingung	<udiErrorID>	<uiErrorParameter>
<uiStepsPerTurn> ≥ 1	16#61400	116
<uiStepsPerTurn> ≤ 10000	16#61400	116
<diLowerLimit> $\leq$ <diUpperLimit>	16#61400	121
(<diUpperLimit> – <diLowerLimit>) / <uiStepsPerTurn> $\leq$ <rRotations>	16#61400	122
Falls <diSetPoint> < <diLowerLimit>: (<diUpperLimit> – <diSetPoint>) / <uiStepsPerTurn> $\leq$ <rRotations>	16#61400	68
Falls <diSetPoint> > <diUpperLimit>: (<diSetPoint> – <diLowerLimit>) / <uiStepsPerTurn> $\leq$ <rRotations>	16#61400	68

Tabelle 11: Bedingung und Fehlermeldung

- Wahlweise können die geschriebenen Werte am Ende auch gespeichert werden. Dazu muss der Eingang <xSaveSettings> auf TRUE gesetzt werden.
- Bei einem Schreibfehler eines Parameters werden die nachfolgenden Parameter nicht mehr geschrieben und es erfolgt auch kein Speichern der Werte, falls der Eingang <xSaveSettings> gesetzt ist.

5.15 FB_MC_PosParametrization_PSxHub_PSE3xx

Dieser Baustein entspricht in Bedienung und Verhalten dem Baustein

FB_MC_PosParametrization_PSxHub_PSD4xx, lediglich die intern verwendeten Parameternummern unterscheiden sich.

6 Parameterbeschreibung

6.1 <aGroupNumberForPort>

Baustein	FB_MC_MakeSynchronousGroups_PSxHub
Gleichlaufgruppe des Ports	
Datentyp	Array[1..10] of UInt
Default	0
Art	Input
Beschreibung	
Über diesen Eingang kann jedem Port eine Gleichlaufgruppe zugewiesen werden. Sollen beispielsweise Ports 1-3 in der Gleichlaufgruppe 1 zusammengefasst werden, müssten den Einträgen 1-3 die 1 zugewiesen werden. Zulässige Werte sind 0-5. 0 bedeutet der Port ist nicht Teil einer Gleichlaufgruppe, die Werte 1-5 korrespondieren mit den Gleichlaufgruppen 1-5.	

6.2 <xResetSynchronousMovementErrorBit>

Baustein	
Setzt das Bit 8 („Fehler Gleichlauf erkannt“) des Status zurück	
Datentyp	Bool
Default	-
Art	Output
Beschreibung	
Setzt das Bit 8 („Fehler Gleichlauf erkannt“) des Status zurück. Das Gleichlauffehler Bit der Gruppe wird automatisch gelöscht, wenn sich die Differenz der Antriebe wieder im tolerierten Bereich befindet.	

6.3 <uiMaximumDifference>

Baustein	FB_MC_MakeSynchronousGroups_PSxHub
Erlaubte Differenz ohne Gleichlauffehler	
Datentyp	UInt
Default	-
Art	Input
Beschreibung	
Dieser Eingang legt die maximale Schrittdifferenz fest, die in einer Gleichlaufgruppe toleriert wird. Wenn die größte Differenz der Gruppe diesen Wert überschreitet, wird ein Gleichlauffehler erkannt (Bit 8 Statuswort PSxHub) und das Gleichlauffehlerbit der Gruppe gesetzt (Bit 9-13 Statuswort PSxHub).	

6.4 <aExecuteGroup>

Baustein	FB_MC_SynchronousMovement_PSxHub
Startet die Positionierung der Gleichlaufgruppe	
Datentyp	Array[1..5] of Bool
Default	-
Art	Input
Beschreibung	
Startet die Positionierung der jeweiligen Gleichlaufgruppe auf den in <aTargetPositionGroup> eingestellten Wert. Die Positionierung ist abgeschlossen, wenn alle Antriebe der Gruppe das <xDone> Bit gesetzt haben und auf der Zielposition stehen. Während der Fahrt sollten die Statuswörter der Antriebe und des PSxHubs überwacht werden, um Fehler der Antriebe und Gleichlauffehler zu erkennen.	

6.5 <aTargetPositionGroup>

Baustein	FB_MC_SynchronousMovement_PSxHub
Setzt Zielposition der Gleichlaufgruppe	
Datentyp	Array[1..5] of DInt
Default	-
Art	Input
Beschreibung	
Legt die Zielposition für die jeweilige Gleichlaufgruppe fest.	

6.6 <aTryResolveSyncErrorGroup>

Baustein	FB_MC_TryResolveSyncErrorGroup_PSxHub
Versucht den Gleichlauffehler der Gruppe zu lösen	
Datentyp	Array[1..5] of Bool
Default	-
Art	Input
Beschreibung	
Ist der Eingang der jeweiligen Gruppe gesetzt wird versucht den Gleichlauffehler der Gruppe durch Verfahren aller Antriebe auf eine gemeinsame Zielposition zu lösen. Der Erfolg kann an der Löschung der Statusbits des Hubs (Bit 8 „Gleichlauffehler erkannt“ sowie Bits 9-13 „Abweichung in Gleichlaufgruppe X erkannt“) abgelesen werden.	



VORSICHT!

Durch die Schleifenfahrt kann es dabei temporär zu größeren Abweichungen kommen. Parameter Schleifenlänge berücksichtigen.

6.7 <usiSyncGroupErrorResolveStrategy>

Baustein	FB_MC_TryResolveSyncErrorGroup_PSxHub
Die zu verwendende Strategie zum Auflösen von Gleichlauffehlern	
Datentyp	USInt
Default	-
Art	Input
Beschreibung	
Dieser Eingang legt die Strategie fest, die zur Auflösung von Gleichlauffehlern angewendet wird. Bei 0 fahren alle Antriebe der Gruppe auf die Position des höchsten Antriebs. Bei 1 fahren alle Antriebe der Gruppe auf die Position des niedrigsten Antriebs.	

6.8 <xActive>

Baustein	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,
FBs ist aktiv oder Fahrauftrag bzw. Fahrt ist aktiv	
Datentyp	Bool
Default	-
Art	Output
Beschreibung	
<u>FB MC Move Drive</u> Dieser Ausgang wird gesetzt, wenn: • die Freigabe <xExecute> von FALSE auf TRUE gesetzt wird • die Freigabe <xExecute> schon vorhanden ist und sich die Zielposition ändert, das Bit „Antrieb läuft“ im Status des Antriebs gesetzt ist (z.B. beim Nachregeln des Antriebs) Dieser Ausgang wird zurückgesetzt, wenn: • am Ende einer Fahrt das Bit „Antrieb läuft“ im Status des Antriebs nicht mehr gesetzt ist und die Zeit des Parameters Buszyklus (siehe 6.33 <tTimeBusCycle>) abgelaufen ist • ein Kommunikationsfehler auftritt <u>Alle anderen Bausteine</u> Das Bit wird gesetzt, solange der jeweilige Funktionsbaustein läuft. Sobald der Vorgang abgeschlossen wurde oder ein Fehler aufgetreten ist, wird das Bit zurückgesetzt.	

6.9 <diActualPosition>

Baustein	FB_MC_Move_PSxHub_Drive
Istwert der Position	
Datentyp	DInt
Default	-
Art	Output
Beschreibung	
Dieser Wert ist eine Abbildung der Istposition. Falls ein Kommunikationsfehler auftritt, wird der Wert auf 0 gesetzt.	

6.10 <xDeliveryState>

Baustein	FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx
Laden der Werkseinstellungen (zunächst ohne Speichern)	
Datentyp	Bool
Default	FALSE
Art	Input
Beschreibung	
Vor dem Schreiben der aktivierten Parameter, werden zunächst alle Parameter auf Werkseinstellungen zurückgesetzt.	

6.11 <uiDirection>

Baustein	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Drehrichtung der Abtriebswelle	
Datentyp	UInt
Default	0
Art	Input
Beschreibung	
Dieser Parameter entspricht ISDU 115(PSD) bzw. 123(PSE) „Drehsinn“. Richtung, in der der Antrieb bei größeren Werten drehen soll (bei Sicht auf die Abtriebswelle): (0 → im Uhrzeigersinn; 1 → gegen Uhrzeigersinn)	

6.12 <xDone>

Baustein	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
Zielposition erreicht oder FBs hat Vorgang abgeschlossen	
Datentyp	Bool
Default	-
Art	Output
Beschreibung	
<u>FB MC Move Drive</u> Dieser Ausgang ist eine Abbildung des Statusbits „Sollposition erreicht“. Zusätzlich wird der Ausgang für die Dauer des Parameters „Buszyklus“ (siehe 6.33 <tTimeBusCycle>) nach dem Start durch <xExecute> auf 0 gesetzt. Falls ein Kommunikationsfehler auftritt, wird er zurückgesetzt.	
<u>Alle anderen Bausteine</u> Das Bit wird gesetzt, sobald der Vorgang erfolgreich abgeschlossen wurde. Es wird beim Start des jeweiligen Vorgangs zurückgesetzt.	

6.13 <stDrive>

Baustein	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_PSE3xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx, FUN_MC_Communication_Drive
Datenstruktur zur Kommunikation	
Datentyp	ST_DriveData_PSD4xx
Art	Input & Output
Beschreibung	
Für jeden Antrieb wird eine globale Instanz dieser Struktur benötigt. Diese Instanz wird jedem Funktionsbaustein (FB) übergeben, der auf den betriebenen Antrieb wirkt.	

6.14 <stHub>

Baustein	FB_MC_Error_PSxHub_Drive, FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_ReadParameter_PSxHub, FB_MC_WriteParameter_PSxHub,
Datenstruktur zur Kommunikation	
Datentyp	ST_HubData_PSxHub
Art	Input & Output
Beschreibung	
Für jeden Hub wird eine globale Instanz dieser Struktur benötigt. Diese Instanz wird jedem Funktionsbaustein (FB) übergeben, der auf den betriebenen Hub wirkt.	

6.15 <stHubAndDrives>

Baustein	FB_MC_SynchronousMovement_PSxHub, FB_MC_TryResolveSyncErrorGroup_PSxHub
Datenstruktur zur Kommunikation	
Datentyp	ST_Variables_PSxHub/PSxHub_PSD4xx
Art	Input & Output
Beschreibung	
Für jeden Hub wird eine globale Instanz dieser Struktur benötigt. Diese Instanz wird den Funktionsbausteinen (FB) übergeben, die auf den PSxHub und mehrere Antriebe wirken.	

6.16 <xError>

Baustein	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
Fehler bei der Ausführung des FBs oder Fehler im Antrieb	
Datentyp	Bool
Default	FALSE
Art	Output
Beschreibung	
<u>FB MC Move PSxHub Drive</u>	
Das Fehlerbit wird jeden Zyklus aktualisiert und wird gesetzt, wenn während der Ausführung des FBs ein Fehler auftritt. Es kann auch gesetzt sein, während der Antrieb aktiv ist (z.B. Schleppfehler).	
<u>FB MC Error PSxHub Drive</u>	
Der Ausgang <xError> wird jeden Zyklus aktualisiert, solange <xExecute> gesetzt ist. Wird <xExecute> zurückgesetzt, so nimmt dieser Ausgang den angegebenen Defaultwert an.	
<u>FB MC SynchronousMovement PSxHub</u>	
Das Fehlerbit wird jeden Zyklus aktualisiert und wird gesetzt, wenn während der Ausführung einer Gleichlaufgruppe ein Fehler auftritt.	
<u>Alle anderen Bausteine</u>	
Das Fehlerbit wird jeden Zyklus aktualisiert und wird gesetzt, wenn während der Ausführung des FBs ein Fehler aufgetreten ist.	

6.17 <sErrorDescription>

Baustein	FB_MC_Error_PSxHub_Drive
Fehlerbeschreibung als Textausgabe	
Datentyp	String[254]
Default	„
Art	Output
Beschreibung	
Eine Fehlerbeschreibung als Textausgabe	

6.18 <udiErrorID>

Baustein	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
Fehlerkennung (siehe Kapitel 4 Fehlerbeschreibung (<udiErrorID>))	
Datentyp	UDint
Default	0
Art	Output
Beschreibung	
Die Fehlerkennung kann auch gesetzt sein, während der Antrieb fährt (z.B. Schleppfehler) <u>FB MC Move PSxHub Drive</u> Die Fehlerkennung wird jeden Zyklus aktualisiert und wird ausgegeben, wenn während der Ausführung des FBs ein Fehler auftritt. Es kann auch ausgegeben werden, während der Antrieb aktiv ist (z.B. Schleppfehler). <u>FB MC Error PSxHub Drive</u> Der Ausgang <udiErrorID> wird jeden Zyklus aktualisiert, solange <xExecute> gesetzt ist. Wird <xExecute> zurückgesetzt, so nehmen diese Ausgänge den angegebenen Anfangswert an. <u>Alle anderen Bausteine</u> Die Fehlerkennung wird jeden Zyklus aktualisiert und wird gesetzt, wenn während der Ausführung des FBs ein Fehler aufgetreten ist.	

VORSICHT!

Die Ausgänge <xError> und <udiErrorID> der Funktionsbausteine werden bei dem **FB_MC_Move_Drive** stets aktualisiert – auch dann, wenn der Eingang <xExecute> nicht gesetzt ist.

6.19 <uiErrorParameter>

Baustein	FB_MC_MakeSynchronousGroups_PSxHub, FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSE3xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Parameternummer, bei dem im Fall eines Fehlers der Fehler aufgetreten ist	
Datentyp	UInt
Default	0
Art	Output
Beschreibung	
Falls es keinen Fehler gab, wird der Wert 0 ausgegeben.	

6.20 <xExecute>

Baustein	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
Freigabe des Antriebs	
Datentyp	Bool
Default	FALSE
Art	Input
Beschreibung	
<u>FB MC Move PSxHub Drive</u> Ein Sollwert wird erst angefahren, wenn dieser Eingang gesetzt ist. Dieser Eingang wirkt direkt auf das Freigabebit (Bit 4) im Steuerwort. Bleibt der Eingang gesetzt und ist z.B. das Nachregeln im Antrieb aktiv, so regelt der Antrieb automatisch nach. Ist der Eingang gesetzt und wird der Sollwert geändert, so fährt der Antrieb diesen sofort an. Eine Flanke ist nicht erforderlich. Wird der Eingang während der Fahrt zurückgesetzt, stoppt der Antrieb. <u>Alle anderen Bausteine</u> Bei einer steigenden Flanke wird der Vorgang des jeweiligen Funktionsbausteins durchgeführt. Für einen neuen Vorgang muss wieder eine steigende Flanke generiert werden. Wird das Bit zurückgesetzt, so nehmen die Ausgänge die angegebenen Anfangswerte an.	

6.21 <diLowerLimit>

Baustein	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx		
Untere Endbegrenzung			
Datentyp	DInt		
Default	0		
Art	Input		
Beschreibung			
Dieser Parameter entspricht den ISDUs 122(PSD) bzw. 130(PSE) „untere Endbegrenzung“.			

6.22 <xManualRunToLargerValues>

Baustein	FB_MC_Move_PSxHub_Drive	
Handfahrt zu größeren Werten		
Datentyp	Bool	
Default	FALSE	
Art	Input	
Beschreibung		
Es wird mit der Handfahrt zu größeren Werten bis zur oberen Endbegrenzung gefahren. Der Eingang <xExecute> muss zusätzlich gesetzt werden.		

VORSICHT!

Beim Zurücksetzen des Eingangs <xManualRunToLargerValues> muss auch <xExecute> zurückgesetzt werden, da der Antrieb ansonsten die Zielposition <diTargetPosition> anfährt.

6.23 <xManualRunToSmallerValues>

Baustein	FB_MC_Move_PSxHub_Drive	
Handfahrt zu kleineren Werten		
Datentyp	Bool	
Default	FALSE	
Art	Input	
Beschreibung		
Es wird mit der Handfahrt zu kleineren Werten bis zur unteren Endbegrenzung gefahren. Der Eingang <xExecute> muss zusätzlich gesetzt werden.		



VORSICHT!

Beim Zurücksetzen des Eingangs <xManualRunToSmallerValues> muss auch <xExecute> zurückgesetzt werden, da der Antrieb ansonsten die Zielposition <diTargetPosition> anfährt.

6.24 <stParameter>

Baustein	FB_MC_Parametrization_PSxHub_PSD4xx, FB_MC_Parametrization_PSxHub_PSD4xx	
Übergabe des Parametersatzes		
Datentyp	ST_Parameter_PSxHub_PSD4xx	
Default	-	
Art	Input	
Beschreibung		
Die Parameternummer des Antriebs (ISDU) ist hinter dem Parameternamen angegeben. Der Datentyp, eine Beschreibung sowie der Wertebereich können der Busbeschreibung des PSD4 bzw. PSE3 IOLink entnommen werden.		

6.25 <uiParameterNumber>

Baustein	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	
	Parameternummer des zu lesenden bzw. schreibenden Parameters	
	Datentyp	UInt
	Default	0
	Art	Input
Beschreibung		
Bei einer steigenden Flanke wird ein Lese- bzw. Schreibvorgang gestartet. Für einen neuen Vorgang muss erneut eine steigende Flanke generiert werden. Wird das Bit zurückgesetzt, so nehmen die Ausgänge die angegebenen Anfangswerte an.		

6.26 <xSaveSettings>

Baustein	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
Speichern der Parametereinstellungen	
Datentyp	Bool
Default	FALSE
Art	Input
Beschreibung	
<u>FB_MC_MakeSynchronousGroups_PSxHub:</u> Dieser Parameter entspricht dem Parameter 0xF58F „Auslieferungszustand“. (Funktion <Schreiben einer „1“>)	
<u>Funktionsbausteine der Antriebe:</u> Dieser Parameter entspricht ISDU 194(PSD) bzw. 193(PSE) „Auslieferungszustand“ des Antriebs. (Funktion <Schreiben einer „1“>)	

6.27 <diSetPoint>

Baustein	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Istposition des Messsystems	
Datentyp	DInt
Default	0
Art	Input
Beschreibung	
Dieser Parameter entspricht ISDU 68 (PSD) bzw. 67 (PSE) „Istwert“.	

6.28 <uiStepsPerTurn>

Baustein	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx
Schritte pro Umdrehung an der Abtriebswelle (Auflösung)	
Datentyp	Int
Default	0
Art	Input
Beschreibung	
Die Anzahl der Schritte pro Umdrehung <uiStepsPerTurn> ergibt unmittelbar den Wert des ISDU 117 (PSD) bzw. 125(PSE) „Istwertbewertung Nenner“.	

6.29 <usiSubIndex>

Baustein	FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
Subindex des Parameters	
Datentyp	USInt
Default	0
Art	Input
Beschreibung	
Der Wert des Parameters <usiSubIndex> kann auf den Defaultwert gesetzt bleiben, wenn der Parameter keinen Subindex besitzt.	

6.30 <diTargetPosition>

Baustein	FB_MC_Move_PSxHub_Drive
Anzufahrende Zielposition	
Datentyp	DInt
Default	0
Art	Input
Beschreibung	
Wird während einer Fahrt eine neue Zielposition übertragen, wird diese sofort angefahren. Ist nach Fahrtende <xExecute> noch gesetzt und wird die Zielposition geändert, so fährt der Antrieb diesen sofort an.	

HINWEIS

Um die gleiche Zielposition z.B. nach einem Blockieren anzufahren, muss der Freigabeparameter <xExecute> zurückgesetzt und erneut gesetzt werden.

6.31 <tTimeBusCycle>

Baustein	FB_MC_Move_PSxHub_Drive	
Aktualisierungszeit des EtherCAT Buszyklus		
Datentyp	Time	
Default	40ms	
Art	Input	
Beschreibung		
Bei langen Zykluszeiten auf dem EtherCAT-Bus kann es passieren, dass die SPS keinen Wechsel des Bits „Antrieb läuft“ empfängt. Dies ist dann der Fall, wenn die Fahrdauer kürzer als der Buszyklus ist. Um dem zu begegnen, wurde der Parameter <tTimeBusCycle> eingeführt. Als Anfangswert werden 40 ms für einen Buszyklus angesetzt. Für diese Dauer wird nach einem Fahrauftrag der Ausgang <xActive> auf jeden Fall gesetzt und der Ausgang <xDone> zurückgesetzt. Danach nehmen diese Ausgänge in Abhängigkeit der Rückmeldung des Statusworts (Bit „Antrieb läuft“ und Bit „Sollposition ist erreicht“) den entsprechenden Zustand an. Bei längeren Buszykluszeiten empfiehlt es sich, anstatt dem Anfangswert die tatsächliche Dauer des Zyklus multipliziert mit 2 einzugeben. Wenn eine kürzere Buszykluszeit garantiert eingehalten wird, die Zeit für die Positionierung sehr kurz ist und nach abgeschlossener Positionierung der übergeordnete Ablauf schnell fortgesetzt werden soll, kann der Wert für <tTimeBusCycle> auch verringert werden.		

6.32 <diUpperLimit>

Baustein	FB_MC_PosParametrization_PSxHub_PSD4xx, FB_MC_PosParametrization_PSxHub_PSE3xx,	
Obere Endbegrenzung		
Datentyp	DInt	
Default	0	
Art	Input	
Beschreibung		
Dieser Parameter entspricht dem Parameter 129(PSE) bzw. 121(PSD) „obere Endbegrenzung“.		

6.33 <diValue>

Baustein	FB_MC_ReadParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_ReadParameter_PSxHub_PSE3xx, FB_MC_WriteParameter_PSxHub_PSE3xx
zu schreibender oder zu lesender Wert eines Parameters	
Datentyp	DInt
Default	0
Art	Input für FB_MC_WriteParameter_PSxHub, FB_MC_WriteParameter_PSxHub_PSD4xx, FB_MC_WriteParameter_PSxHub_PSD4xx Output für FB_MC_ReadParameter_PSxHub, FB_MC_ReadParameter_PSxHub_PSD4xx
Beschreibung	
Bei einer steigenden Flanke wird ein Lese- bzw. Schreibvorgang gestartet. Für einen neuen Vorgang muss erneut eine steigende Flanke generiert werden.	

7 Anwendung der Funktionsbausteine

Die Funktionsbausteine sind in TwinCAT 3.1 (Build 4024) programmiert. Ein Upgrade auf höhere Versionen kann problemlos durchgeführt werden.

Dabei gibt es zwei Möglichkeiten die Funktionsbausteine zu nutzen.

- als Projekt oder
- als eingebundene Bibliothek

7.1 Projekt

Im Projekt ist ein Hub mit 10 PSD an einer Soft-SPS enthalten. Gegebenenfalls, müssen noch die Adressen gemäß 2.1 angepasst werden, um die Kommunikation zu ermöglichen.

Die Ein- und Ausgänge der Funktionsbausteine sind in der Funktion „Instances()“ mit den Variablen in „GVL_Data_Store_PSxHub“ verbunden, sodass über das Forcen dieser Werte die Funktionsweise der Bausteine getestet werden kann.

Im Projekt sind zudem einige Beispiele enthalten, diese sind in Kapitel 8 **Beispielprogramme** näher beschrieben. Falls eines der Beispielprojekte aktiv ist, funktioniert die Ansteuerung über „GVL_Data_Store_PSxHub“ möglicherweise nur noch eingeschränkt, da in diesen nur die jeweils benötigten Bausteine aufgerufen werden.

7.2 Bibliothek

Zuerst muss die Bibliothek in dem Bibliotheksrepository installiert werden.

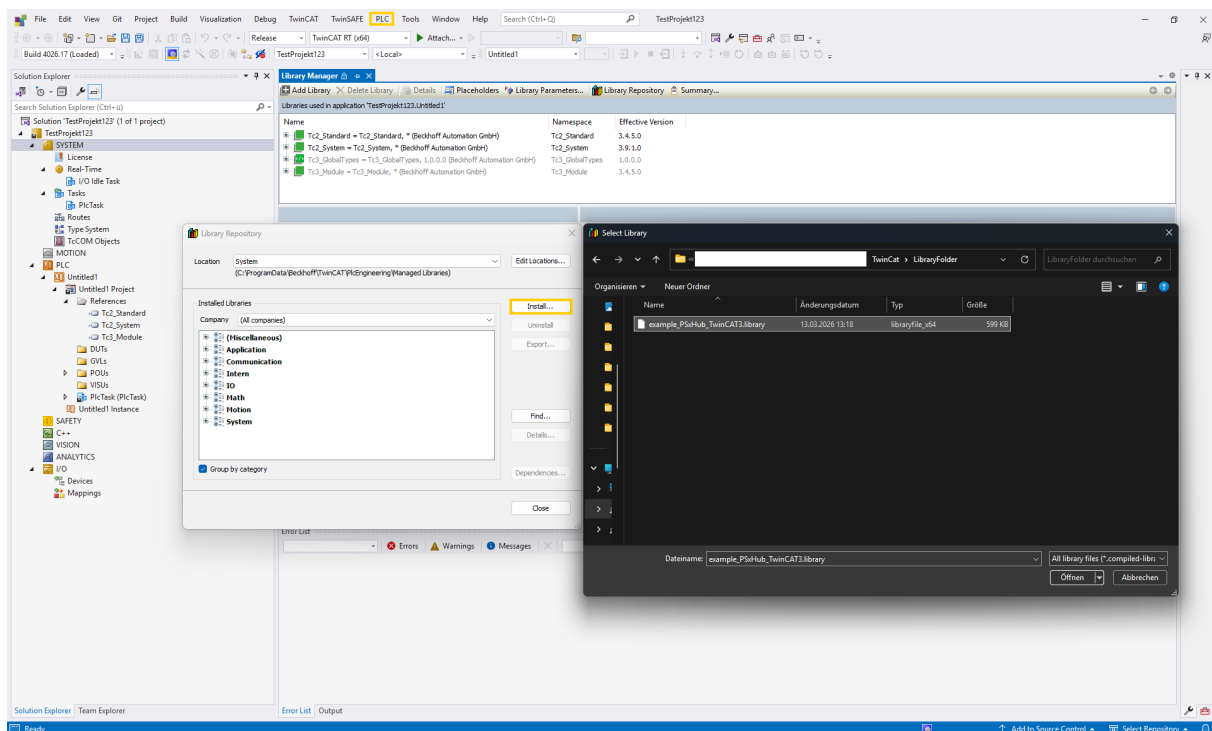


Abbildung 20 Installation der Bibliothek im Bibliotheksrepository

Zusätzlich kann die globale Variablenliste <GVL_Data_Store_PSD4xx> als Vorlage für die Anbindung an die Funktionsbausteine übernommen werden. Dafür sind jedoch alle Datentypen (DUTs_halstrup_walcher) notwendig.

Für das Einbinden der Beispielprogramme sollte die globale Variablenliste <GVL_VariablesForProgram> übernommen werden.

7.3 Sperrung zwischen den Funktionsbausteinen

Die Bausteine sind zum Teil gegeneinander gesperrt. Dadurch ist z.B. sichergestellt, dass nicht gleichzeitig zwei Zugriffe aus unterschiedlichen FBs eines Antriebs durchgeführt werden können.

Es gelten die folgenden Regeln für die Bausteine eines Antriebs:

- Wenn der Eingang <xExecute> von **FB_MC_Move_Drive** gesetzt ist, können die Bausteine **FB_MC_Parametrization_PSD4xx** und **FB_MC_PosParametrization_PSD4xx** nicht aktiviert werden (<udiErrorID>: 16#1000).
- Dagegen ist es möglich, während einer Bewegung **FB_MC_ReadParameter_PSD4xx** oder **FB_MC_WriteParameter_PSD4xx** aufzurufen (z.B. um das aktuelle Drehmoment auszulesen oder während der Fahrt die Solldrehzahl zu ändern).
- Wenn der Eingang <xExecute> von **FB_MC_Parametrization_PSD4xx** oder **FB_MC_PosParametrization_PSD4xx** gesetzt ist, kann der Baustein **FB_MC_Move_Drive** nicht aktiviert werden (<udiErrorID>: 16#01000).
- Es kann stets nur einer der Bausteine **FB_MC_ReadParameter_PSD4xx**, **FB_MC_WriteParameter_PSD4xx**, **FB_MC_Parametrization_PSD4xx** oder **FB_MC_PosParametrization_PSD4xx** aktiv sein. Wenn der Eingang <xExecute> einer dieser Bausteine gesetzt ist und der Eingang <xExecute> eines weiteren Bausteins gesetzt wird, gibt dieser den Fehler 16#1000 aus.
- Der Baustein **FB_MC_Error_Drive** kann unabhängig von den anderen Bausteinen stets aktiviert sein.

Von den Bausteinen

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

darf zu jeder Zeit nur einer aktiv sein. Wird ein weiterer Baustein aktiviert, gibt dieser den Fehler 16#1000 aus.

7.4 Mehrere PSxHubs in einem Projekt

In den Beispielprojekten befindet sich nur ein PSxHub. Wenn mehrere PSxHubs genutzt werden, muss das Projekt entsprechend angepasst werden.

Beispielsweise für drei PSxHub mit jeweils 10 angeschlossenen Antrieben:

1. Nach Bedarf Instanzen für Hub initialisieren
 - drei Instanzen vom Typ **FB_MC_WriteParameter_PSxHub** z.B.
 - fbMC_WriteParameter_PSxHub_1
 - fbMC_WriteParameter_PSxHub_2
 - fbMC_WriteParameter_PSxHub_3
 - drei Instanzen vom Typ **FB_MC_ReadParameter_PSxHub** z.B.
 - fbMC_ReadParameter_PSxHub_1
 - fbMC_ReadParameter_PSxHub_2
 - fbMC_ReadParameter_PSxHub_3
 - weitere benötigte Instanzen
2. Nach Bedarf Instanzen für jeden angeschlossenen Antrieb initialisieren
 - Dreißig Instanzen vom Typ **FB_MC_Move_Drive**
 - fbMC_Move_PSD4xx_1
 - ...
 - fbMC_Move_PSD4xx_30
 - weitere benötigte Instanzen
3. Anschließend wird die globale Variablenliste <GVL_Data_Store_PSxHub> angepasst. Darin werden drei Variablen vom Typ <ST_Variables_PSxHubAndDrive> angelegt, z.B. mit den folgenden Bezeichnungen:
 - „Hub_1“
 - „Hub_2“
 - „Hub_3“
4. Falls die PlainData Module genutzt werden, muss für jeden Antrieb die Funktion **FUN_MC_Communication_Drive** aufgerufen werden
5. Zum Schluss müssen die Variablen von <GVL_Data_Store_PSxHub> den Funktionsbausteinen zugewiesen werden. (siehe Kapitel 5 Beschreibung der Funktionsbausteine)

7.5 Abhängigkeiten zwischen Funktionsbausteinen

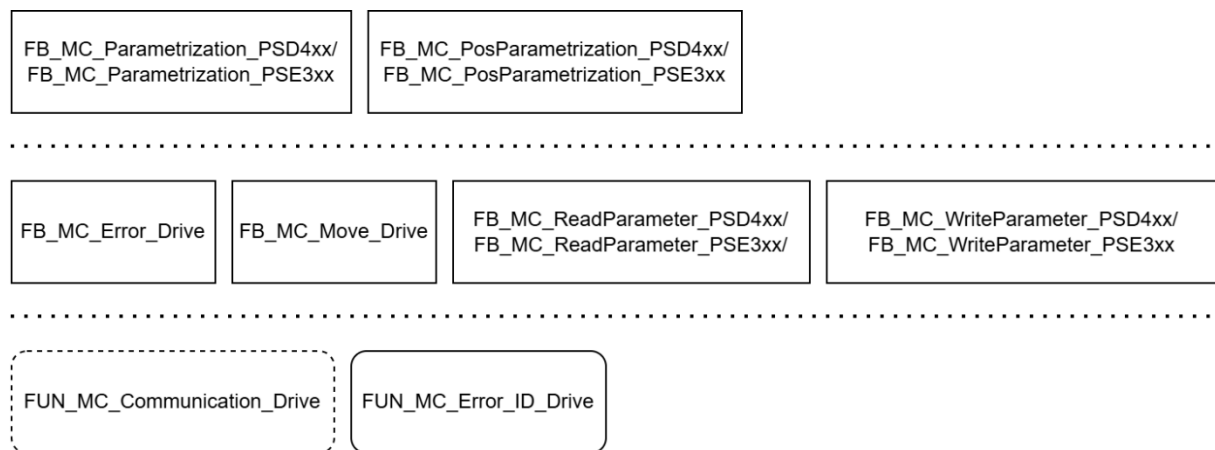


Abbildung 23: Unterteilung der Bausteine & Funktionen zur Kommunikation mit den Antrieben nach Komplexität (FUN_MC_Communication_Drive ist optional)

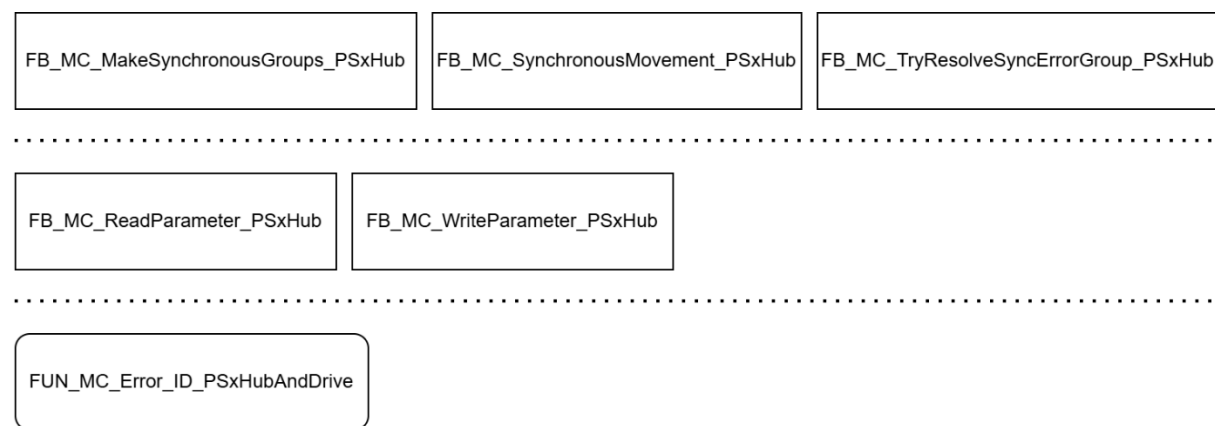


Abbildung 24: Unterteilung der Bausteine & Funktionen zur Kommunikation mit dem Hub nach Komplexität

Je nach Funktionsumfang der SPS-Anwendung, werden nicht alle Bausteine in allen Projekten benötigt. Nicht genutzte Bausteine, können entfernt werden, wenn nicht andere Bausteine auf ihrer Funktion aufbauen. Abbildung 23 ist eine grobe Unterteilung der Funktionsbausteine, zur Steuerung der Antriebe, nach Komplexität zu entnehmen. Abbildung 24 zeigt dies für die Bausteine zur Ansteuerung der Hubspezifischen Funktionen. Höhere Ebenen können dabei problemlos entfernt werden, ohne die Funktionalität der darunterliegenden Ebenen zu beeinträchtigen. Höhere Ebenen haben Abhängigkeiten zu mindestens einem Teil der darunterliegenden Ebenen, werden diese entfernt, führt dies zu Fehlern im Buildprozess.

In vielen Projekten werden die Hubspezifischen Funktionen nicht benötigt und die Bausteine können aus dem Projekt gelöscht werden.

8 Beispielprogramme

Um den Einsatz der Funktionsbausteine besser kennen zu lernen, können die Beispielprogramme verwendet werden.

Es sind Beispielprogramme im Projektbaum zu finden, um zu veranschaulichen, wie die Funktionsbausteine einzusetzen sind. Die Beispiele sind im Strukturierten Text programmiert und das jeweilige Programm muss in das Hauptprogramm eingefügt werden, um das Programm aufrufen zu können.

```
//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();
```

Abbildung 25 Aufgerufenes Programm im Hauptprogramm

Gemeinsamkeiten aller Beispielprogramme

- Bei allen Programmen werden am Anfang die benötigten Instanzen definiert und aufgerufen.
- Bei allen Programmen muss die Variable <xStartProgram> für das Starten des Programms auf TRUE gesetzt werden.
- Bei jedem Beispielprogramm ist der erste Schritt nach dem Start des Programms, dass alle relevanten Variablen für die Initialisierung auf FALSE gesetzt werden.
- Mit <xStopProgram> können die Programme beendet werden.

8.1 Beispiel 1: Positioniermodus

In diesem Beispielprogramm wird der Positioniermodus des FBs **FB_MC_Move_Drive** vorgestellt. Der Antrieb an Port 1 fährt in diesem Beispiel eine Endlosschleife zwischen den Werten 20000 und 10000.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Zielposition von 20000 und der Fahrbefehl werden gesetzt. Wenn die aktuelle Position bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Die Zielposition von 10000 und der Fahrbefehl werden gesetzt. Wenn die aktuelle Position bei 10000 ± 2 (siehe Parameter „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, wird der Fahrbefehl zurückgesetzt und die State Machine auf 1 gesetzt.

8.2 Beispiel 2: Handfahrmodus

In diesem Beispielprogramm wird der Handmodus des FBs **FB_MC_Move_Drive** vorgestellt. Der Antrieb an Port 1 fährt in diesem Beispiel auf eine Startposition mit dem Positioniermodus und danach mit dem Handmodus.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Mit Handmodus wird über das Ziel von 30000 gefahren. Erst wenn die aktuelle Position 30000 entspricht, wird der Fahrbefehl für die Handfahrt zurückgesetzt und die State Machine auf 100 gesetzt.

8.3 Beispiel 3: Aktuelle Position lesen

In diesem Beispielprogramm wird der FB **FB_MC_ReadParameter_PSD4xx** vorgestellt. Der Antrieb an Port 1 fährt in diesem Beispiel auf eine Zielposition und der Parameter „aktuelle Position“ wird gelesen.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Der Parameter „aktuelle Position“ wird gelesen. Wenn der Wert bei 20000 ± 2 liegt (siehe Parameter „Positionierfenster“), dann wird der Lesebefehl zurückgesetzt und die State Machine auf 100 gesetzt.

8.4 Beispiel 4: Auslieferungszustand schreiben

In diesem Beispielprogramm wird der FB **FB_MC_WriteParameter_PSD4xx** vorgestellt. Der Parameter „Auslieferungszustand“ wird in diesem Beispiel geschrieben und der Antrieb an Port 1 fährt anschließend in seine Bereichsmittle.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Der Wert des Parameters „Auslieferungszustand“ wird auf -5 geschrieben. Der Antrieb wird auf seinen Auslieferungszustand zurückgesetzt, die Parameter auf den Defaultwert gesetzt und zusätzlich findet eine Positionierung auf die Messbereichsmittle (Position 51200) statt. Wenn der Schreibbefehl erfolgreich durchgeführt wurde, dann wird der Schreibbefehl zurückgesetzt und die State Machine um eins erhöht. (Der Antrieb fährt hierbei noch auf die Messbereichsmittle.)

8.5 Beispiel 5: Parameter parametrieren

In diesem Beispielprogramm wird der FB **FB_MC_Parametrization_PSD4xx** vorgestellt. Die Parameter „Schleifenlänge“ und „Solldrehzahl“, des Antriebs an Port 1, werden in diesem Beispiel parametrieren und danach eine Positionierfahrt durchgeführt.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Die Parameter „Schleifenlänge“ und „Solldrehzahl“ werden parametrieren. Dabei müssen die <xEnable> der Parameter gesetzt sein. Wenn der Parametrierbefehl erfolgreich durchgeführt wurde, dann wird der Parametrierbefehl zurückgesetzt und die State Machine um eins erhöht.
3	Die Zielposition von 10000 und der Fahrbefehl werden gesetzt. Bei dieser Positionierfahrt wird keine Schleifenfahrt durchgeführt und es wird mit 20 U/min gefahren. Wenn der aktuelle Wert bei 10000 ± 2 (siehe Parameter „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine auf 100 gesetzt.

8.6 Beispiel 6: Positionsparameter parametrieren

In diesem Beispielprogramm wird der FB **FB_MC_PosParametrization_PSD4xx** vorgestellt. Der Antrieb an Port 1 wird parametriert und der Parameter „aktuelle Istposition“ gelesen.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Alle Positionsparameter werden parametriert. Der Parameter „Istposition“ wird auf 0, der Parameter „obere Endbegrenzung“ auf 10000 und der Parameter „untere Endbegrenzung“ auf -10000 gesetzt. Der Rest der Parameter wird nicht verändert und die Parameterwerte sollen nicht gespeichert werden. Wenn der Parametrierbefehl der Positionsparameter erfolgreich durchgeführt wurde, dann wird der Parametrierbefehl der Positionsparameter zurückgesetzt und die State Machine um eins erhöht.
3	Der Parameter „aktuelle Position“ wird gelesen. Wenn der Wert bei 0 ± 2 liegt, wird der Lesebefehl zurückgesetzt und die State Machine auf 100 gesetzt.

8.7 Beispiel 7: Fehler obere Endbegrenzung erreicht

In diesem Beispielprogramm wird der FB **FB_MC_Error_Drive** vorgestellt. Der Antrieb an Port 1 wird auf die obere Endbegrenzung gefahren und die Fehlermeldung ausgelesen.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Der Parameter „Obere Endbegrenzung“ wird auf den Wert 25000 geschrieben. Wenn der Schreibbefehl erfolgreich durchgeführt wurde, dann wird der Schreibbefehl zurückgesetzt und die State Machine um eins erhöht.
3	Der Funktionsbaustein FB_MC_Error_Drive wird aktiviert, um Fehler zu erkennen.
4	Mit Handmodus wird die obere Endbegrenzung angefahren. Erst wenn die aktuelle Position 25000 entspricht, wird die Fehlerbeschreibung als String angezeigt. Danach wird der Fahrbefehl für die Handfahrt zurückgesetzt und der Index auf 100 gesetzt. (Auch wäre es möglich, diesen Fehler vom FB FB_MC_Move_Drive (<udiErrorID>) zu erhalten. Die <udiErrorID> wäre dann 0x100B0.)

8.8 Beispiel 8: Erstellung von Gruppen mit Gleichlaufüberwachung

In diesem Beispielprogramm wird die Funktionsweise der Bausteine

FB_MC_MakeSynchronousGroups_PSxHub und **FB_MC_SynchronousMovement_PSxHub** vorgestellt. Die Antriebe an Ports 1&2 werden einer Gleichlaufüberwachungsgruppe hinzugefügt und verfahren.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und die State Machine wird um eins erhöht. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 20000 und der Fahrbefehl werden für Antrieb 1 gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist (siehe Parameter „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
2	Die Startposition von 20000 und der Fahrbefehl werden für Antrieb 2 gesetzt. Wenn der aktuelle Wert bei 20000 ± 2 ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht.
3	Port 1&2 werden mittels FB_MC_MakeSynchronousGroups_PSxHub der Gleichlaufüberwachungsgruppe 1 hinzugefügt, die Schrittgrenze für das Auslösen der Überwachung auf 30 festgelegt. Wenn die Antriebe erfolgreich der Gruppe hinzugefügt wurden, wird der Schreibbefehl zurückgesetzt und die State Machine um eins erhöht.
4	Der automatische Stopp im Falle der Überschreitung der Schrittgrenze wird aktiviert, beiden Antrieben wird mittels FB_MC_SynchronousMovement_PSxHub die Zielposition 10000 vorgegeben und der Fahrbefehl gesetzt. Wenn der Istwert der Antriebe jeweils 10000 ± 2 entspricht und weder der SynchronousMovement, noch die Move Bausteine Fehler melden, wird der Fahrbefehl zurückgesetzt und die State Machine um 1 erhöht.

8.9 Beispiel 9: Fehlerbehandlung bei ausgelöster Gleichlaufüberwachung

Die Funktionsweise der Bausteine **FB_MC_TryResolveSyncErrorGroup_PSxHub** und **FB_MC_SynchronousMovement_PSxHub** wird in diesem Beispielpogramm vorgestellt. Die Antriebe an Ports 1&2 werden einer Gleichlaufüberwachungsgruppe hinzugefügt und verfahren.

Programmablauf

Case	Beschreibung
INIT (0)	Alle relevanten xExecute-Signale der Funktionalbausteine (Move, WriteParameter, MakeSynchronousGroups, TryResolveSyncErrorGroup, SynchronousMovement) werden auf FALSE gesetzt. Die automatische Stoppfunktion wird deaktiviert. Danach wird die State Machine um 1 erhöht.
1	Antrieb 1 erhält die Zielposition 20000 und den Fahrbefehl. Wenn der Antrieb die Position 20000 ± 2 erreicht hat, kein Fehler anliegt und der Move-FB „Done“ meldet, wird der Fahrbefehl zurückgesetzt und die State Machine erhöht.
2	Antrieb 2 erhält ebenfalls die Zielposition 20000 und den Fahrbefehl. Sobald auch hier 20000 ± 2 erreicht ist und die Fahrt fehlerfrei beendet wurde, wird der Befehl zurückgesetzt, außerdem der Synchronfehler zurückgesetzt, und die State Machine erhöht.
3	Antrieb 1 wird über den Parameter Solldrehzahl Positionieren (137) per FB_MC_WriteParameter_PSD4xx auf 20 U/Min konfiguriert. Sobald der Schreibvorgang abgeschlossen ist, wird xExecute zurückgesetzt und weitergeschaltet.
4	Antrieb 2 wird ebenfalls über Parameter Solldrehzahl Positionieren (137) auf 40 U/Min eingestellt, um bewusst eine Positionsabweichung während der synchronen Bewegung zu provozieren. Nach erfolgreichem Schreiben wird xExecute zurückgesetzt und weitergeschaltet.
5	Für Antrieb 1 wird der Parameter Schleifenlänge (124) auf 0 gesetzt, damit bei der Angleichung der Positionen keine großen Positionssprünge durch die Schleifenfahrt entstehen können. Nach erfolgreichem Schreiben wird weitergeschaltet.
6	Für Antrieb 2 wird der Parameter Schleifenlänge (124) ebenfalls auf 0 gesetzt. Nach erfolgreichem Schreiben wird weitergeschaltet.
7	Port 1&2 werden mittels FB_MC_MakeSynchronousGroups_PSxHub der Gleichlaufüberwachungsgruppe 1 hinzugefügt, die Schrittgrenze für das Auslösen der Überwachung auf 30 festgelegt. Wenn die Antriebe erfolgreich der Gruppe hinzugefügt wurden, wird der Schreibbefehl zurückgesetzt und die State Machine um eins erhöht.
SYNC_MOVE (8)	Der automatische Stopp wird aktiviert. Über FB_MC_SynchronousMovement_PSxHub wird der Gruppe die Zielposition 10000 zugewiesen und der Fahrbefehl gesetzt. - Wenn ein Synchronfehler erkannt wird und beide Motoren stehen, springt das Programm in den RESOLVE_SYNC_ERROR-Zustand. Wenn beide Antriebe 10000 ± 2 erreicht haben, kein Fehler anliegt und beide Move-FBs „Done“ melden, wird der Fahrbefehl zurückgesetzt und das Programm beendet.
RESOLVE_SYNC_ERROR (9)	Mittels FB_MC_TryResolveSyncErrorGroup_PSxHub wird eine Angleichung der Motorpositionen angestoßen, dabei wird der höhere Antrieb auf den Wert des niedrigeren gefahren. Wenn beide Antriebe „Done“ melden und keiner der Bausteine einen Fehler meldet, wird wieder in den Zustand SYNC_MOVE gewechselt.

Abbildungsverzeichnis

Abbildung 1 Beispielkonfiguration	6
Abbildung 2 Auslesen von uiNodeAddress	7
Abbildung 3 Auslesen von sTargetNetId	7
Abbildung 4 Auslesen von sMasterNetId	7
Abbildung 5 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_ReadParameter_PSxHub	11
Abbildung 6 <GVL_Data_Store_PSxHub> mit den Variablen für einen Antrieb	11
Abbildung 7 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_ReadParameter_PSxHub	14
Abbildung 8 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_WriteParameter_PSxHub	14
Abbildung 9 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_MakeSynchronousGroups_PSxHub>	14
Abbildung 10 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_SynchronousMovement_PSxHub	14
Abbildung 11 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_TryResolveSyncErrorGroup_PSxHub	15
Abbildung 12 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_Move_PSxHub_PSD4xx	15
Abbildung 13 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_Error_PSxHub_Drive	16
Abbildung 14 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_ReadParameter_PSxHub_PSD4xx	16
Abbildung 15 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_WriteParameter_PSxHub_PSD4xx	17
Abbildung 16 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_Parametrization_PSxHub_PSD4xx	17
Abbildung 17 Parameter „Schleifenlänge“ auf 0 setzen, Parameter „Soll Drehzahl Positionieren“ auf 20	17
Abbildung 18 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_Parametrization_PSE3xx	18
Abbildung 19 Zuweisung der Variablen von <GVL_Data_Store_PSxHub> zum FB_MC_PosParametrization_PSxHub_PSD4xx	18
Abbildung 20 Installation der Bibliothek im Bibliotheksrepository	35
Abbildung 21 Einfügen der Bibliothek	36
Abbildung 22 Enthaltene Dateien in der Bibliothek	36
Abbildung 23 Unterteilung der Bausteine & Funktionen zur Kommunikation mit den Antrieben nach Komplexität (FUN_MC_Communication_Drive ist optional)	Fehler! Textmarke nicht definiert.
Abbildung 24 Unterteilung der Bausteine & Funktionen zur Kommunikation mit dem Hub nach Komplexität	Fehler!
Textmarke nicht definiert.	
Abbildung 25 Aufgerufenes Programm im Hauptprogramm	40

Tabellenverzeichnis

Tabelle 1: Aufbau der Datenstruktur ST_DriveData_PSxHub	6
Tabelle 2 Zuordnung Hub Prozessdaten zu stPrivate	8
Tabelle 3: Aufbau der Datenstruktur ST_Drive	8
Tabelle 4 Verknüpfung der Prozessdaten mit stPrivate, Module: PSD/PSE	9
Tabelle 5 Verknüpfung der Prozessdaten mit stPrivate, Module PSD_PlainData/PSE_PlainData	9
Tabelle 6: Datenstruktur ST_FunctionBlocks_PSxHub	9
Tabelle 7: Datenstruktur ST_FunctionBlocks_PSxHub_PSD4xx	10
Tabelle 8: Fehlercodes	13
Tabelle 9 Verknüpfung der Prozessdaten mit stPrivate	15
Tabelle 10: Positionierung des Antriebs	16
Tabelle 11: Bedingung und Fehlermeldung	19