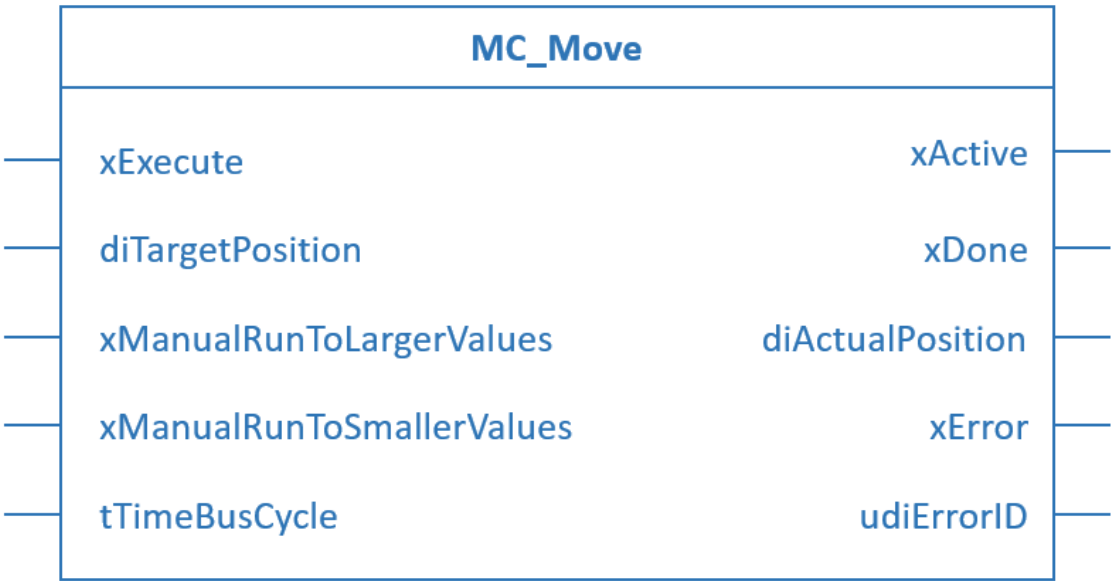


Funktionsbausteine PSx3xx für TwinCAT 3



Bedeutung der Betriebsanleitung

Diese Betriebsanleitung beschreibt die Funktionsbausteine für die Positioniersysteme PSx3xx 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
79199 Kirchzarten

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

© 2024

15.04.2024, TS & FS

7100.007004 Version 2.1.1 Betriebsanleitung Funktionsbausteine PSx3xx EC

Inhaltsverzeichnis

| | | |
|------|---|----|
| 1 | Sicherheitshinweise | 5 |
| 1.1 | Bestimmungsgemäße Verwendung | 5 |
| 1.2 | Warnsymbole..... | 5 |
| 2 | Benutzerspezifische Datentypen..... | 6 |
| 2.1 | <ST_DriveData_PSx3xx> | 6 |
| 2.2 | <ST_FunctionBlocks_PSx3xx> | 7 |
| 2.3 | <ST_Variables_PSx3xx> | 8 |
| 3 | <GVL_Data_Store_PSx3xx>..... | 9 |
| 4 | Fehlerbeschreibung (<udiErrorID>)..... | 10 |
| 5 | Beschreibung der Funktionsbausteine | 12 |
| 5.1 | FB_MC_Move_PSx3xx..... | 12 |
| 5.2 | FB_MC_Error_PSx3xx | 13 |
| 5.3 | FUN_MC_Error_ID_PSx3xx | 13 |
| 5.4 | FB_MC_ReadParameter_PSx3xx | 13 |
| 5.5 | FB_MC_WriteParameter_PSx3xx..... | 14 |
| 5.6 | FB_MC_Parametrization_PSx3xx | 14 |
| 5.7 | FB_MC_PosParametrization_PSx3xx..... | 15 |
| 6 | Parameterbeschreibung..... | 16 |
| 6.1 | <xActive> | 16 |
| 6.2 | <diActualPosition> | 16 |
| 6.3 | <xDeliveryState> | 17 |
| 6.4 | <uiDirection> | 17 |
| 6.5 | <xDone> | 17 |
| 6.6 | <stDrive> | 18 |
| 6.7 | <xError> | 18 |
| 6.8 | <sErrorDescription> | 18 |
| 6.9 | <udiErrorID> | 19 |
| 6.10 | <uiErrorParameter>..... | 19 |
| 6.11 | <xExecute> | 20 |
| 6.12 | <diLowerLimit>..... | 20 |
| 6.13 | <xManualRunToLargerValues> | 20 |
| 6.14 | <xManualRunToSmallerValues> | 21 |
| 6.15 | <sNetId> | 21 |
| 6.16 | <stParameter> | 22 |
| 6.17 | <uiParameterNumber> | 22 |
| 6.18 | <xSaveSettings>..... | 22 |

| | | |
|------|--|----|
| 6.19 | <diSetPoint>..... | 22 |
| 6.20 | <uiSlaveAddr>..... | 23 |
| 6.21 | <uiStepsPerTurn>..... | 23 |
| 6.22 | <usiSubIndex> | 23 |
| 6.23 | <diTargetPosition> | 24 |
| 6.24 | <tTimeBusCycle> | 24 |
| 6.25 | <diUpperLimit> | 25 |
| 6.26 | <diValue> | 25 |
| 6.27 | <sValue> | 25 |
| 7 | Anwendung der Funktionsbausteine | 26 |
| 7.1 | Projekt..... | 26 |
| 7.2 | Bibliothek..... | 26 |
| 7.3 | Sperrung zwischen den Funktionsbausteinen..... | 27 |
| 7.4 | Mehrere PSx3xx in einem Projekt..... | 28 |
| 8 | Beispielprogramme..... | 29 |
| 8.1 | Beispiel 1: Positioniermodus | 29 |
| 8.2 | Beispiel 2: Handfahrmodus | 30 |
| 8.3 | Beispiel 3: Aktuelle Position lesen | 30 |
| 8.4 | Beispiel 4: Auslieferungszustand schreiben | 31 |
| 8.5 | Beispiel 5: Parameter parametrisieren | 31 |
| 8.6 | Beispiel 6: Positionsparameter parametrisieren | 32 |
| 8.7 | Beispiel 7: Fehler obere Endbegrenzung erreicht..... | 32 |
| | Abbildungsverzeichnis..... | 33 |

1 Sicherheitshinweise




1.1 Bestimmungsgemäße Verwendung

Die Positioniersysteme PSx3xx EC eignen sich besonders zur automatischen Einstellung von Werkzeugen, Anschlägen oder Spindeln bei Holzverarbeitungsmaschinen, Verpackungsmaschinen, Druckmaschinen, Abfüllanlagen und bei Sondermaschinen.

Die PSx3xx EC sind nicht als eigenständige Geräte zu betreiben, sondern dienen ausschließlich zum Anbau an eine Maschine.

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 Körpverletzung eintreten. |
|  WARNUNG | WARNUNG! Bei Nichtbeachtung dieses Sicherheitshinweises können Tod oder schwere Körpverletzung eintreten. |
|  VORSICHT | VORSICHT! Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körpverletzung eintreten. |
| HINWEIS | HINWEIS Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten. |

2 Benutzerspezifische Datentypen

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

2.1 <ST_DriveData_PSx3xx>

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 Adresse des EtherCAT-Masters und die Adresse des jeweiligen EtherCAT-Slaves (PSx3xx) 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) |
| <iActiveFunctionBlock> | Int | Funktionsbausteine | Aktiver Funktionsbaustein |
| <xCommunicationError> | Bool | Funktionsbausteine | Kommunikationsfehler zum IO-Device |
| <sNetId> | T_AmsNetId | Benutzer | Adresse des EtherCAT-Masters |
| <uiSlaveAddr> | UInt | Benutzer | Adresse des EtherCAT-Slaves |
| <stPrivate> | ST_Private | Funktionsbausteine | Datenstruktur zur internen Verwendung |

Die folgenden Darstellungen zeigen, wie in der Automatisierungssoftware TwinCAT die vergebenen Adressen des Masters und des Slaves überprüft werden können:

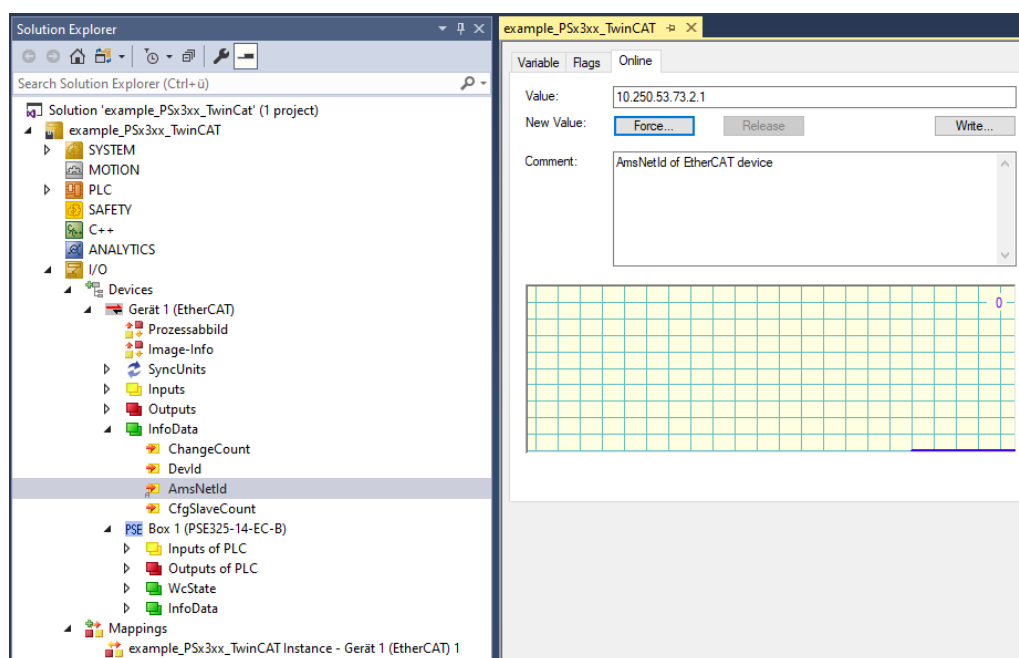


Abbildung 1: Adresse des EtherCAT-Masters

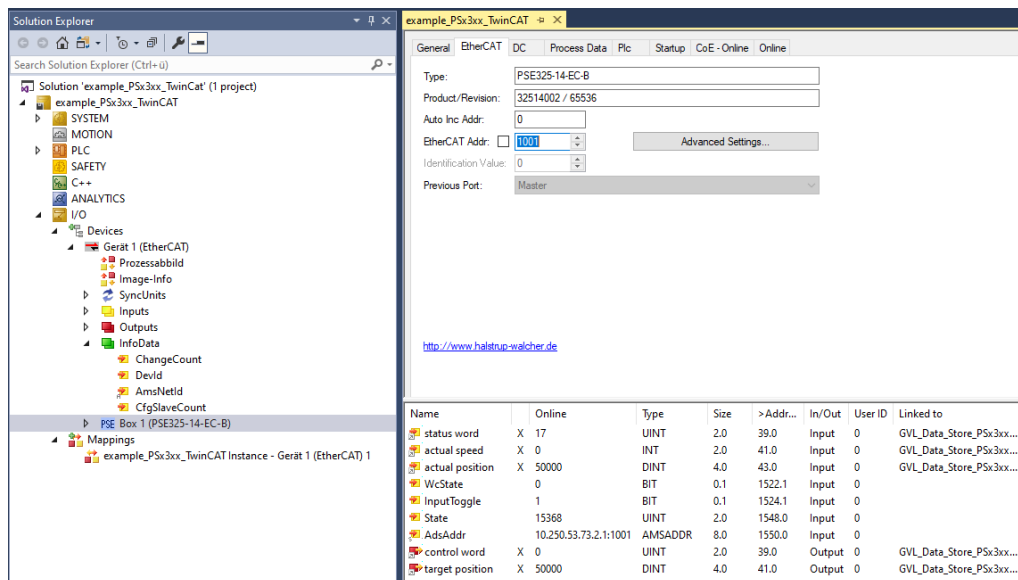


Abbildung 2: Adresse des EtherCAT-Slaves

2.2 <ST_FunctionBlocks_PSx3xx>

Diese Struktur enthält alle nötigen Variablen um die Ein- und Ausgänge der Funktionsbausteine zu belegen, mit Ausnahme von stDrive der ST_DriveData benötigt (beschrieben in 2.1). Diese Struktur hat mehrere Unterstrukturen, die in der Dokumentation nicht weiter ausgeführt werden. Es wird empfohlen für jeden Baustein eine globale Instanz dieser Struktur zu nutzen, alternativ können die notwendigen Variablen auch individuell angelegt werden. Bei Nichtbenutzung von Funktionsbausteinen ist es empfehlenswert, die Datenstruktur anzupassen, damit nicht unnötig Speicher in der SPS belegt wird.

| Parametername | Datentyp | geschrieben von | Beschreibung |
|-------------------------------|------------------------------|--------------------|---|
| <stMove_PSx3xx> | ST_Move_PSx3xx | Funktionsbausteine | Variablen für FB_MC_Move_PSx3xx |
| <stError_PSx3xx> | ST_Error_PSx3xx | Funktionsbausteine | Variablen für FB_MC_Error_PSx3xx |
| <stReadParameter_PSx3xx> | ST_ReadParameter_PSx3xx | Funktionsbausteine | Variablen für FB_MC_ReadParameter_PSx3xx |
| <stWriteParameter_PSx3xx> | ST_WriteParameter_PSx3xx | Funktionsbausteine | Variablen für FB_MC_WriteParameter_PSx3xx |
| <stParametrization_PSx3xx> | ST_Parametrization_PSx3xx | Funktionsbausteine | Variablen für FB_MC_Parametrization_PSx3xx |
| <stPosParametrization_PSx3xx> | ST_PosParametrization_PSx3xx | Funktionsbausteine | Variablen für FB_MC_PosParametrization_PSx3xx |

2.3 <ST_Variables_PSx3xx>

Diese Struktur beinhaltet die Strukturen <ST_FunctionBlocks_PSx3xx> und <ST_DriveData_PSx3xx>. Es wird empfohlen für jeden Antrieb eine Instanz dieser Struktur in <GVL_Data_Store_PSD4xx> hinzuzufügen.

3 <GVL_Data_Store_PSx3xx>

Die GVL (globale Variablenliste) <GVL_Data_Store_PSx3xx> 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 PSx3xx in einem Projekt muss die Anzahl der Variablen entsprechend multipliziert werden.

Beispiel

```

19 //FBMC_Move_PSx3xx_1 --> instance of FB_MC_Move_PSx3xx (1. PSx3xx)
20 fbMC_Move_PSx3xx_1(
21     xExecute:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xExecute,
22     diTargetPosition:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.diTargetPosition,
23     xManualRunToLargerValues:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xManualRunToLargerValues,
24     xManualRunToSmallerValues:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.xManualRunToSmallerValues,
25     tTimeBusCycle:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stInput_Move_PSx3xx.tTimeBusCycle,
26     xActive=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xActive,
27     xDone=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xDone,
28     diActualPosition:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.diActualPosition,
29     xError=> GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.xError,
30     udiErrorID:= GVL_Data_Store_PSx3xx.PSE_1.stFunctionBlocks_PSx3xx.stMove_PSx3xx.stOutput_Move_PSx3xx.udiErrorID,
31     stDrive:= GVL_Data_Store_PSx3xx.PSE_1.stDriveData_PSx3xx);

```

Abbildung 3: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Move_PSx3xx

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.

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

Abbildung 4: <GVL_Data_Store_PSx3xx> 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_PSx3xx |
| 16#2xxxx | Fehler in FB_MC_Error_PSx3xx |
| 16#3xxxx | Fehler in FB_MC_ReadParameter_PSx3xx |
| 16#4xxxx | Fehler in FB_MC_WriteParameter_PSx3xx |
| 16#5xxxx | Fehler in FB_MC_Parametrization_PSx3xx |
| 16#6xxxx | Fehler in FB_MC_PosParametrization_PSx3xx |
| 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: ungültiger 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 aktuellen Betriebszustand nicht bearbeitet werden |
| 16#xxAxx | Andere Fehler |
| 16#000F0 (Maske) | Antriebsfehler |
| 16#xxx0x | <u>Kein</u> Antriebsfehler |
| 16#xxx1x | Schleppfehler |
| 16#xxx2x | Unter- oder Überspannung der Motorversorgung (STO-Freigabe inaktiv*) |
| 16#xxx3x | Positionierung wurde abgebrochen |
| 16#xxx4x | Temperaturüberschreitung |
| 16#xxx5x | Messsystemfehler (Messsystem- oder STO-Hardwarefehler*) |
| 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 |

* Falls das PSx3xx ein Gerät mit STO ist, dann müssen die Fehlerbeschreibung in Klammern berücksichtigt werden! (<udiErrorID>: 16#xxx2x und 16#xxx5x)

Die Fehler „Antriebsfehler“ sind eine Abbildung der Fehlerbits im Statuswort des PSx3xx.

Beispiele

- Fahrauftrag (**FB_MC_Move_PSx3xx**) mit falscher Zielposition → <udiErrorID> = 16#10080
- Parameter schreiben (**FB_MC_WriteParameter_PSx3xx**) mit ungültiger Parameternummer → <udiErrorID> = 16#40200

5 Beschreibung der Funktionsbausteine

Die Kommunikation zum Antrieb findet über die Prozessdaten des Antriebs statt. Diese Prozessdaten sind direkt mit den Variablen in der globalen Variablenliste <GVL_Data_Store_PSx3xx> verknüpft.

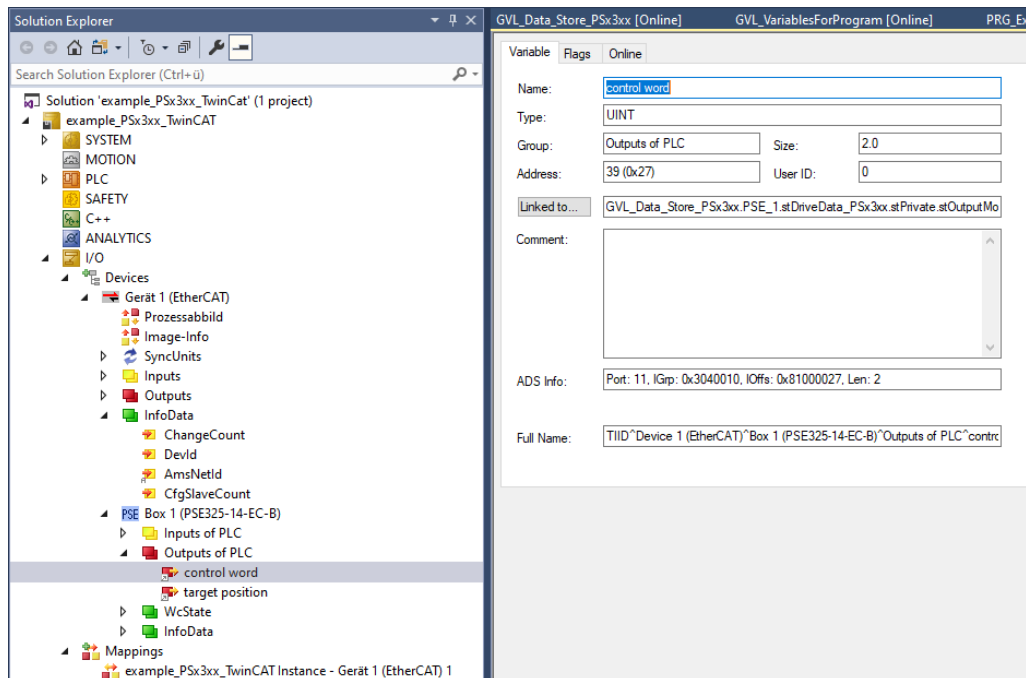


Abbildung 5: Verknüpfung des Steuerworts mit einer Variablen aus <GVL_Data_Store_PSx3xx>

5.1 FB_MC_Move_PSx3xx

Dieser Funktionsbaustein wird zur Positionierung des Antriebs verwendet.

```

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

```

Abbildung 6: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Move_PSx3xx

Falls der 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 Lesen 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 |

*Falls das PSx3xx ein Gerät mit STO ist, dann müssen die Fehlerbeschreibung in Klammer berücksichtigt werden! (<udiErrorID>: 16#xxx2x und 16#xxx5x)

5.2 FB_MC_Error_PSx3xx

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

```

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

```

Abbildung 7: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Error_PSx3xx

5.3 FUN_MC_Error_ID_PSx3xx

Diese Funktion wird intern als Unterfunktion der FBs **FB_MC_Move_PSx3xx**, **FB_MC_ReadParameter_PSx3xx**, **FB_MC_WriteParameter_PSx3xx** und **FB_MC_Error_PSx3xx** eingesetzt. Er ist lediglich aus der Bibliothek in das Anwenderprogramm zu kopieren. Die Beschaltung findet schon intern statt.

5.4 FB_MC_ReadParameter_PSx3xx

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

```

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

```

Abbildung 8: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_ReadParameter_PSx3xx

5.5 FB_MC_WriteParameter_PSx3xx

Mit diesem FB können Parameterwerte in den Antrieb geschrieben werden.

```

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

```

Abbildung 9: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum
FB_MC_WriteParameter_PSx3xx

5.6 FB_MC_Parametrization_PSx3xx

Mit diesem FB können sämtliche Parameter des Antriebs auf einmal geschrieben werden. Das ist vorallem für die Erstinbetriebnahme von Nutzen. Dabei wurde eine übersichtliche Struktur gewählt, da sämtliche Parameter als Block mit dem benutzerspezifische Datentypen <ST_Parameter_PSx3xx> übergeben werden.

```

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

```

Abbildung 10: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum
FB_MC_Parametrization_PSx3xx

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

```

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

```

Abbildung 11: Parameter „Schleifenlänge“ mit dem Wert von 0

- 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 16#202C, 16#2010, 16#2011, 16#2003...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.

5.7 FB_MC_PosParametrization_PSx3xx

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

```

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

```

Abbildung 12: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_PosParametrization_PSx3xx

Folgendes ist bei der Nutzung des FBs zu beachten:

- Es müssen alle Werte gesetzt werden und die Werte müssen in einem sinnvollen Bezug zueinanderstehen. Alle Bedingungen sind in der unten aufgeführten Tabelle beschrieben. Alle Werte werden verarbeitet, danach werden die folgenden Parameter in der angegebenen Reihenfolge geschrieben:
 - Drehsinn (Parameter 16#202C) → <uiDirection>
 - Istwertbewertung Zähler (Parameter 16#2010) → 400
 - Istwertbewertung Nenner (Parameter 16#2011) → <uiStepsPerTurn>
 - Istwert (Parameter 16#2003) → <diSetPoint>
 - Falls (<diSetPoint> > <diUpperLimit>):
Oberes Mapping-Ende (Parameter 16#2028) = <diSetPoint> + (3 x <uiStepsPerTurn>)
sonst:
Oberes Mapping-Ende (Parameter 16#2028) = <diUpperLimit> + (3 x <uiStepsPerTurn>)
 - Obere Endbegrenzung (Parameter 16#2016) → <diUpperLimit>
 - Untere Endbegrenzung (Parameter 16#2017) → <diLowerLimit>

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

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

- 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.

6 Parameterbeschreibung

6.1 <xActive>

| | |
|---|---|
| Baustein | FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| FBs ist aktiv oder Fahrauftrag bzw. Fahrt ist aktiv | |
| Datentyp | Bool |
| Default | - |
| Art | Output |
| Beschreibung | |
| <u>FB MC Move PSx3xx</u> <p>Dieser Ausgang wird gesetzt, wenn:</p> <ul style="list-style-type: none"> 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) <p>Dieser Ausgang wird zurückgesetzt, wenn:</p> <ul style="list-style-type: none"> am Ende einer Fahrt das Bit „Antrieb läuft“ im Status des Antriebs nicht mehr gesetzt ist und die Zeit des Parameters Buszyklus (siehe 6.24 <tTimeBusCycle>) abgelaufen ist ein Kommunikationsfehler auftritt <p><u>Alle anderen Bausteine</u></p> <p>Das Bit wird gesetzt, solange der jeweilige Funktionsbaustein läuft. Sobald der Vorgang abgeschlossen wurde oder ein Fehler aufgetreten ist, wird das Bit zurückgesetzt.</p> | |

6.2 <diActualPosition>

| | |
|---|-------------------|
| Baustein | FB_MC_Move_PSx3xx |
| 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. | |

6.3 <xDeliveryState>

| | |
|---|-------------------------------------|
| Baustein | FB_MC_Parametrization_PSx3xx |
| Laden der Werkseinstellungen (zunächst ohne Speichern) | |
| Datentyp | Bool |
| Default | FALSE |
| Art | Input |
| Beschreibung | |
| Der Stationsname und die IP-Adresse bleiben jedoch unbeeinflusst. | |

6.4 <uiDirection>

| | |
|---|--|
| Baustein | FB_MC_PosParametrization_PSx3xx |
| Drehrichtung der Abtriebswelle | |
| Datentyp | UInt |
| Default | 0 |
| Art | Input |
| Beschreibung | |
| <p>Dieser Parameter entspricht dem Parameter Nummer 16#202C „Drehsinn“.</p> <p>Richtung, in der der Antrieb bei größeren Werten drehen soll (bei Sicht auf die Abtriebswelle):</p> <p>(0 → im Uhrzeigersinn; 1 → gegen Uhrzeigersinn)</p> | |

6.5 <xDone>

| | |
|---|--|
| Baustein | FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| Zielposition erreicht oder FBs hat Vorgang abgeschlossen | |
| Datentyp | Bool |
| Default | - |
| Art | Output |
| Beschreibung | |
| <p><u>FB MC Move PSx3xx</u></p> <p>Dieser Ausgang ist eine Abbildung des Statusbits „Sollposition erreicht“. Zusätzlich wird der Ausgang für die Dauer des Parameters „Buszyklus“ (siehe 6.24 <tTimeBusCycle>) nach dem Start durch <xExecute> auf 0 gesetzt. Falls ein Kommunikationsfehler auftritt, wird er zurückgesetzt.</p> <p><u>Alle anderen Bausteine</u></p> <p>Das Bit wird gesetzt, sobald der Vorgang erfolgreich abgeschlossen wurde. Es wird beim Start des jeweiligen Vorgangs zurückgesetzt.</p> | |

6.6 <stDrive>

| | |
|---|---|
| Baustein | FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| Datenstruktur zur Kommunikation | |
| Datentyp | ST_DriveData_PSx3xx |
| 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. | |

6.7 <xError>

| | |
|---|---|
| Baustein | FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| Fehler bei der Ausführung des FBs oder Fehler im Antrieb | |
| Datentyp | Bool |
| Default | FALSE |
| Art | Output |
| Beschreibung | |
| <u>FB MC Move PSx3xx</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 PSx3xx</u> Der Ausgang <xError> wird jeden Zyklus aktualisiert, solange <xExecute> gesetzt ist. Wird <xExecute> zurückgesetzt, so nimmt dieser Ausgang den angegebenen Defaultwert an. | |
| <u>Alle anderen Bausteine</u> Das Fehlerbit wird jeden Zyklus aktualisiert und wird gesetzt, wenn während der Ausführung des FBs ein Fehler aufgetreten ist. | |

6.8 <sErrorDescription>

| | |
|---|--------------------|
| Baustein | FB_MC_Error_PSx3xx |
| Fehlerbeschreibung als Textausgabe | |
| Datentyp | String[254] |
| Default | „ " |
| Art | Output |
| Beschreibung | |
| Eine Fehlerbeschreibung als Textausgabe | |

6.9 <udiErrorID>

| | |
|--|---|
| Baustein | FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| Fehlererkennung (siehe Kapitel 4 Fehlerbeschreibung (<udiErrorID>)) | |
| Datentyp | UDint |
| Default | 0 |
| Art | Output |
| Beschreibung | |
| Die Fehlererkennung kann auch gesetzt sein, während der Antrieb fährt (z.B. Schleppfehler) | |
| <u>FB MC Move PSx3xx</u> | |
| Die Fehlererkennung 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 PSx3xx</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 Fehlererkennung 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_PSx3xx** stets aktualisiert – auch dann, wenn der Eingang <xExecute> nicht gesetzt ist.

6.10 <uiErrorParameter>

| | |
|--|---|
| Baustein | FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx |
| 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. | |

6.11 <xExecute>

| | | |
|---|--|--|
| Baustein | FB_MC_Error_PSx3xx, FB_MC_Move_PSx3xx, FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx, FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx | |
| Freigabe des Antriebs | | |
| Datentyp | Bool | |
| Default | FALSE | |
| Art | Input | |
| Beschreibung | | |
| <u>FB MC Move PSx3xx</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. | | |

6.12 <diLowerLimit>

| | | |
|---|---------------------------------|--|
| Baustein | FB_MC_PosParametrization_PSx3xx | |
| Untere Endbegrenzung | | |
| Datentyp | DInt | |
| Default | 0 | |
| Art | Input | |
| Beschreibung | | |
| Dieser Parameter entspricht dem Parameter 16#2017 „untere Endbegrenzung“. | | |

6.13 <xManualRunToLargerValues>

| | | |
|--|-------------------|--|
| Baustein | FB_MC_Move_PSx3xx | |
| 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.

6.14 <xManualRunToSmallerValues>

| | |
|--|--------------------------|
| Baustein | FB_MC_Move_PSx3xx |
| Handfahrt zu kleineren 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 <xManualRunToSmallerValues> muss auch <xExecute> zurückgesetzt werden, da der Antrieb ansonsten die Zielposition <diTargetPosition> anfährt.

6.15 <sNetId>

| | |
|---|--|
| Baustein | FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| Adresse des EtherCAT-Masters | |
| Datentyp | T_AmsNetID |
| Default | 10.250.53.73.2.1 |
| Art | - |
| Beschreibung | |
| Auf die Adresse des EtherCAT-Masters wird intern im FB_MC_ReadParameter_PSx3xx und FB_MC_WriteParameter_PSx3xx zugegriffen. | |

6.16 <stParameter>

| | |
|---|-------------------------------------|
| Baustein | FB_MC_Parametrization_PSx3xx |
| Übergabe des Parametersatzes | |
| Datentyp | ST_Parameter_PSx3xx |
| Default | - |
| Art | Input |
| Beschreibung | |
| Die Parameternummer ist hinter dem Parameternamen angegeben. Der Datentyp, eine Beschreibung sowie der Wertebereich können der Betriebsanleitung des PSx-3xx-EC entnommen werden. | |

6.17 <uiParameterNumber>

| | |
|--|--|
| Baustein | FB_MC_WriteParameter_PSx3xx, FB_MC_ReadParameter_PSx3xx |
| Parameternummer des zu lesenden bzw. schreibenden Parameters | |
| Datentyp | UInt |
| Default | 0 |
| Art | Input |
| Beschreibung | |
| Die Parameternummer des zu schreibenden bzw. zu lesenden Parameters (z.B. 16#2003 für den Istwert) | |

6.18 <xSaveSettings>

| | |
|---|--|
| Baustein | FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx |
| Speichern der Parametereinstellungen | |
| Datentyp | Bool |
| Default | FALSE |
| Art | Input |
| Beschreibung | |
| Dieser Parameter entspricht dem Parameter 16#204F „Auslieferungszustand“. (Funktion <Schreiben einer „1“>) | |

6.19 <diSetPoint>

| | |
|--|--|
| Baustein | FB_MC_PosParametrization_PSx3xx |
| Istposition des Messsystems | |
| Datentyp | DInt |
| Default | 0 |
| Art | Input |
| Beschreibung | |
| Dieser Parameter entspricht dem Parameter 16#2003 „Istwert“. | |

6.20 <uiSlaveAddr>

| | | |
|--|---|--|
| Baustein | FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx | |
| Adresse des EtherCAT-Slave (PSx3xx) | | |
| Datentyp | UInt | |
| Default | 1001 | |
| Art | - | |
| Beschreibung | | |
| Auf die Adresse des EtherCAT-Slaves wird intern im FB_MC_ReadParameter_PSx3xx und FB_MC_WriteParameter_PSx3xx zugegriffen. | | |

6.21 <uiStepsPerTurn>

| | | |
|--|---------------------------------|--|
| Baustein | FB_MC_PosParametrization_PSx3xx | |
| 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 Parameters „Istwertbewertung Nenner“ (Parameter 16#2011). Dabei wird angenommen, dass der Wert von „Istwertbewertung Zähler“ (Parameter 16#2010) im Auslieferungszustand (400) ist. | | |

6.22 <usiSubIndex>

| | | |
|---|---|--|
| Baustein | FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx | |
| 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. | | |

6.23 <diTargetPosition>

| | |
|--|--------------------------|
| Baustein | FB_MC_Move_PSx3xx |
| Anzufahrende Zielposition | |
| Datentyp | DInt |
| Default | 0 |
| Art | Input |
| Beschreibung | |
| <p>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.</p> | |

HINWEIS

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

6.24 <tTimeBusCycle>

| | |
|--|--------------------------|
| Baustein | FB_MC_Move_PSx3xx |
| Aktualisierungszeit des EtherCAT Buszyklus | |
| Datentyp | Time |
| Default | 40ms |
| Art | Input |
| Beschreibung | |
| <p>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.</p> <p>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.</p> <p>Bei längeren Buszykluszeiten empfiehlt es sich, anstatt dem Anfangswert die tatsächliche Dauer des Zyklus multipliziert mit 2 einzugeben.</p> <p>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.</p> | |

6.25 <diUpperLimit>

| | |
|--|--|
| Baustein | FB_MC_PosParametrization_PSx3xx |
| Obere Endbegrenzung | |
| Datentyp | DInt |
| Default | 0 |
| Art | Input |
| Beschreibung | |
| Dieser Parameter entspricht dem Parameter 16#2016 „obere Endbegrenzung“. | |

6.26 <diValue>

| | |
|--|--|
| Baustein | FB_MC_ReadParameter_PSx3xx, FB_MC_WriteParameter_PSx3xx |
| zu schreibender Wert oder lesenden Wert eines Parameters | |
| Datentyp | DInt |
| Default | 0 |
| Art | Input für FB_MC_WriteParameter_PSx3xx Output für FB_MC_ReadParameter_PSx3xx |
| Beschreibung | |
| FB_MC_ReadParameter: Wenn xDone gesetzt ist, repräsentiert dies den Wert eines numerischen Parameters. | |
| FB_MC_WriteParameter: Bei einer steigenden Flanke auf xExecute wird dieser Wert auf das Gerät geschrieben. | |

6.27 <sValue>

| | |
|--|-----------------------------------|
| Baustein | FB_MC_ReadParameter_PSx3xx |
| zu lesenden Wert eines Parameters | |
| Datentyp | String[31] |
| Default | 0 |
| Art | Output |
| Beschreibung | |
| Wenn xDone gesetzt ist, repräsentiert dies den Wert eines String Parameters. (Derzeit nur Softwarebezeichnung (16#100A) und Gerätetyp (16#20AD)) | |

7 Anwendung der Funktionsbausteine

Die Funktionsbausteine sind in TwinCAT Version v3.1.4024 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

Öffnen und gegebenenfalls upgraden des Projekts.

7.2 Bibliothek

Zuerst muss die Bibliothek in dem Bibliotheksrepository installiert werden.

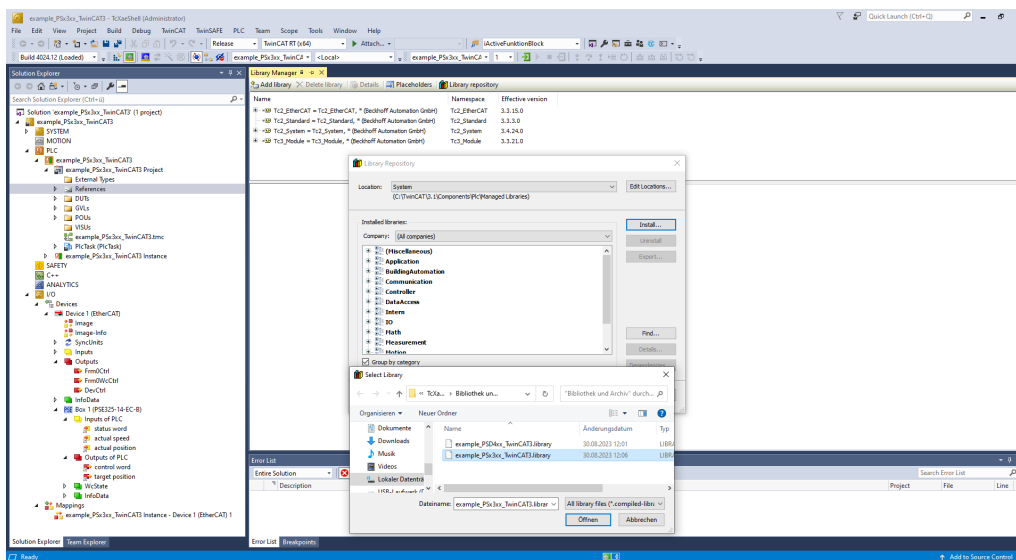


Abbildung 13: Installation der Bibliothek im Bibliotheksrepository

Danach kann die Bibliothek ausgewählt und hinzugefügt werden.

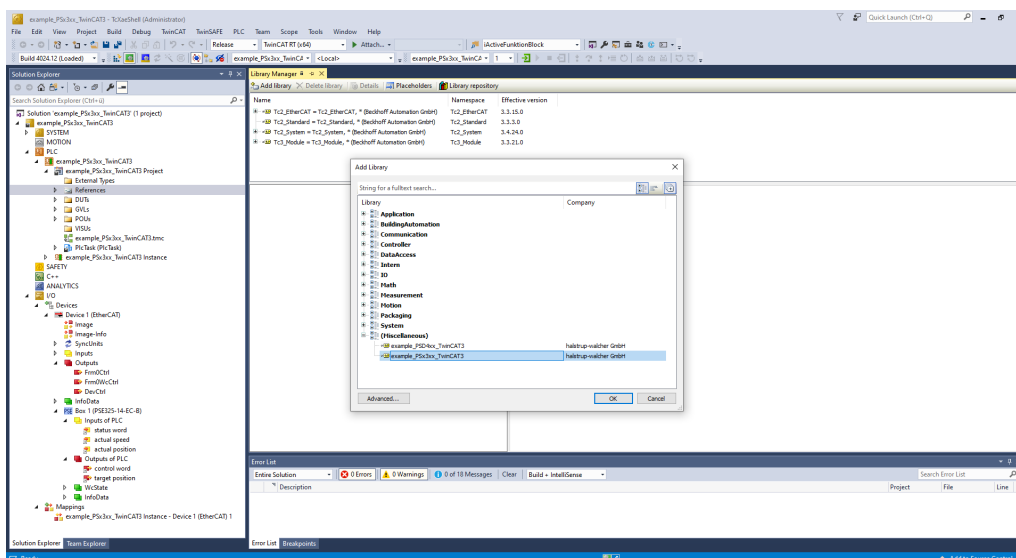


Abbildung 14: Einfügen der Bibliothek

Zuletzt können die Funktionsbausteine und Beispielprogramme ausgewählt werden.

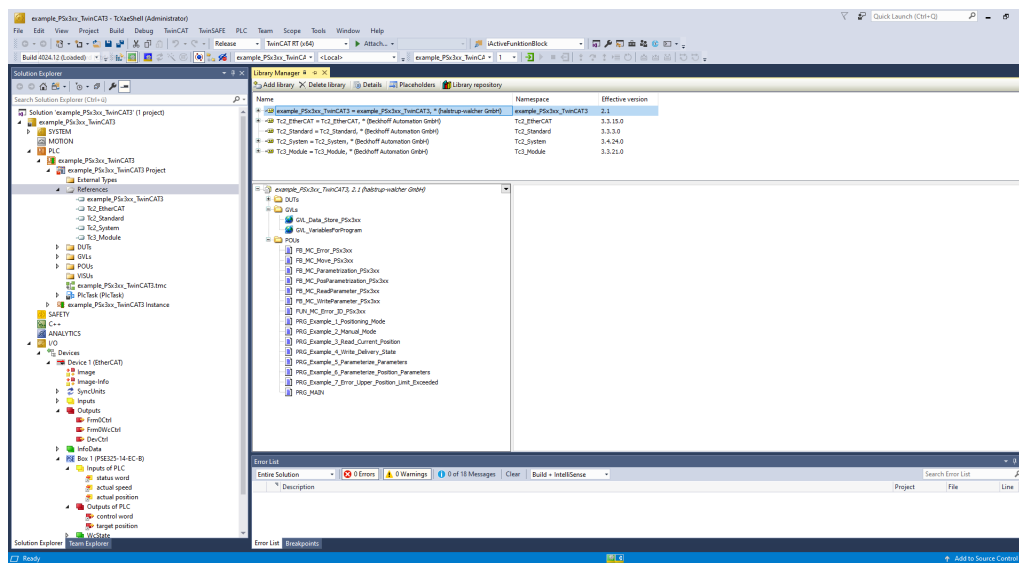


Abbildung 15: Enthaltene Dateien in der Bibliothek

Folgende Funktion aus der Bibliothek sind (unabhängig von den tatsächlich verwendeten Funktionsbausteinen **FB_MC...PSx3xx** und der Anzahl der Antriebe) in jedem Fall in das Projekt des Anwenders zu kopieren:

- **FB_MC_Error_ID_PSi3xx**

Diese Funktion dient der Fehlererkennung mit dem Antrieb.

Daneben sind die gewünschten Funktionsbausteine **FB_MC...PSx3xx** in das Anwenderprojekt zu kopieren. Es können alle oder eine Auswahl der nachfolgenden Bausteine verwendet werden:

- **FB_MC_Move_PSi3xx**
- **FB_MC_Error_PSi3xx**
- **FB_MC_ReadParameter_PSi3xx**
- **FB_MC_WriteParameter_PSi3xx**
- **FB_MC_Parametrization_PSi3xx**
- **FB_MC_PosParametrization_PSi3xx**

Zusätzlich kann die globale Variablenliste <GVL_Data_Store_PSi3xx> 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:

- Wenn der Eingang <xExecute> von **FB_MC_Move_PSi3xx** gesetzt ist, können die Bausteine **FB_MC_Parametrization_PSi3xx** und **FB_MC_PosParametrization_PSi3xx** nicht aktiviert werden (<udiErrorID>: 16#x1000).
- Dagegen ist es möglich, während einer Bewegung **FB_MC_ReadParameter_PSi3xx** oder **FB_MC_WriteParameter_PSi3xx** aufzurufen (z.B. um das aktuelle Drehmoment auszulesen oder während der Fahrt die Solldrehzahl zu ändern).

- Wenn der Eingang <xExecute> von **FB_MC_Parametrization_PSx3xx** oder **FB_MC_PosParametrization_PSx3xx** gesetzt ist, kann der Baustein **FB_MC_Move_PSx3xx** nicht aktiviert werden (<udiErrorID>: 16#01000).
- Es kann stets nur einer der Bausteine **FB_MC_ReadParameter_PSx3xx**, **FB_MC_WriteParameter_PSx3xx**, **FB_MC_Parametrization_PSx3xx** oder **FB_MC_PosParametrization_PSx3xx** 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#x1000 aus.
- Der Baustein **FB_MC_Error_PSx3xx** kann unabhängig von den anderen Bausteinen stets aktiviert sein.

7.4 Mehrere PSx3xx in einem Projekt

In den Beispielprojekten befindet sich nur ein PSx3xx. Wenn mehrere Antriebe genutzt werden, muss das Projekt entsprechend angepasst werden.

Beispielweise für drei PSx3xx:

1. Instanzen initialisieren
 - drei Instanzen des FBs **FB_MC_Move_PSx3xx**, z.B.
 - fbMC_Move_PSx3xx_1
 - fbMC_Move_PSx3xx_2
 - fbMC_Move_PSx3xx_3
 - drei Instanzen des FBs **FB_MC_Error_PSx3xx**, z.B.
 - fbMC_Error_PSx3xx_1
 - fbMC_Error_PSx3xx_2
 - fbMC_Error_PSx3xx_3
 - weitere benötigte Instanzen
2. Anschließend wird die globale Variablenliste <GVL_Data_Store_PSx3xx> angepasst. Darin werden drei Variablen vom Typ <ST_Variables_PSx3xx> angelegt, z.B. mit den folgenden Bezeichnungen:
 - „PSE_1“
 - „PSE_2“
 - „PSE_3“
3. Zum Schluss müssen die Variablen von <GVL_Data_Store_PSx3xx> den Funktionsbausteinen zugewiesen werden. (siehe Kapitel 5 Beschreibung der Funktionsbausteine)

8 Beispielprogramme

Um den Einsatz der Funktionsbausteine besser kennen zu lernen, können die Beispielprogramme verwendet werden. Die Beispiele sind im Strukturierten Text programmiert und das jeweilige Programm muss in das Hauptprogramm eingefügt werden, um das Programm ausführen zu können.

```

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

```

Abbildung 16: Aufrufendes Programm in dem 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_PSx3xx** vorgestellt. Der Antrieb 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 16#2006 „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 16#2006 „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_PSx3xx** vorgestellt. Der Antrieb 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 16#2006 „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_PSx3xx** vorgestellt. Der Antrieb 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 16#2006 „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 16#2006 „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_PSx3xx** vorgestellt. Der Parameter „Auslieferungszustand“ wird in diesem Beispiel geschrieben und der Antrieb 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 16#2006 „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 parametrisieren

In diesem Beispielprogramm wird der FB **FB_MC_Parametrization_PSx3xx** vorgestellt. Die Parameter „Schleifenlänge“ und „Solldrehzahl“ werden in diesem Beispiel parametrisiert 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 16#2006 „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 parametrisiert. Dabei müssen die <xEnable> der Parameter gesetzt sein. Wenn der Parametrisierungsbefehl erfolgreich durchgeführt wurde, dann wird der Parametrisierungsbefehl 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 44 „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 parametrisieren

In diesem Beispielprogramm wird der FB **FB_MC_PosParametrization_PSx3xx** vorgestellt. Es wird parametrisiert 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 16#2006 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine um eins erhöht. |
| 2 | Alle Positionsparameter werden parametrisiert. 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 Parametrisierbefehl der Positionsparameter erfolgreich durchgeführt wurde, dann wird der Parametrisierbefehl der Positionsparameter zurückgesetzt und die State Machine um eins erhöht. |
| 3 | Der Parameter „aktuelle Position“ wird gelesen. Wenn der Wert bei 0 ± 2 liegt, dann 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_PSx3xx** vorgestellt. Es 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 16#2006 „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_PSx3xx 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_PSx3xx (<udiErrorID>) zu erhalten. Die <udiErrorID> wäre dann 0x100B0.) |

Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 1: Adresse des EtherCAT-Masters..... | 6 |
| Abbildung 2: Adresse des EtherCAT-Slaves | 7 |
| Abbildung 3: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Move_PSx3xx..... | 9 |
| Abbildung 4: <GVL_Data_Store_PSx3xx> mit den Variablen für einen Antrieb..... | 9 |
| Abbildung 5: Verknüpfung des Steuerworts mit einer Variablen aus <GVL_Data_Store_PSx3xx> | 12 |
| Abbildung 6: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Move_PSx3xx..... | 12 |
| Abbildung 7: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Error_PSx3xx | 13 |
| Abbildung 8: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_ReadParameter_PSx3xx..... | 13 |
| Abbildung 9: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_WriteParameter_PSx3xx | 14 |
| Abbildung 10: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_Parametrization_PSx3xx..... | 14 |
| Abbildung 11: Parameter „Schleifenlänge“ mit dem Wert von 0..... | 14 |
| Abbildung 12: Zuweisung der Variablen von <GVL_Data_Store_PSx3xx> zum FB_MC_PosParametrization_PSx3xx..... | 15 |
| Abbildung 13: Installation der Bibliothek im Bibliotheksrepository..... | 26 |
| Abbildung 14: Einfügen der Bibliothek..... | 26 |
| Abbildung 15: Enthaltene Dateien in der Bibliothek | 27 |
| Abbildung 16: Aufrufendes Programm in dem Hauptprogramm | 29 |