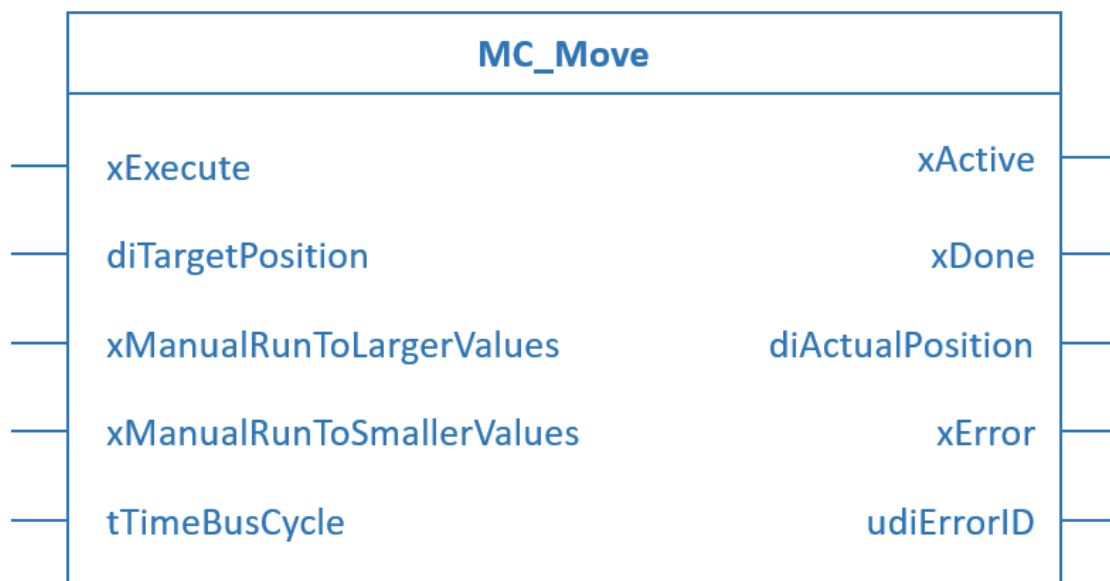


Funktionsbausteine PSx3xx für Studio5000



Bedeutung der Betriebsanleitung

Diese Betriebsanleitung beschreibt die Funktionsbausteine für die Positioniersysteme PSx3xx EIP (mit Ethernet/IP-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 | TS, FS

7100.007054A Version 2.1 Betriebsanleitung Funktionsbausteine PSx3xx EIP

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>.....	6
2.3	<ST_Variables_PSx3xx>.....	7
3	Fehlerbeschreibung (<udiErrorID>).....	8
4	Beschreibung der Funktionsbausteine.....	10
4.1	FB_MC_Move_PSx3xx.....	10
4.2	FB_MC_Error_PSx3xx.....	11
4.3	FC_MC_Error_ID_PSx3xx.....	11
4.4	FB_MC_ReadParameter_PSx3xx.....	11
4.5	FB_MC_WriteParameter_PSx3xx.....	11
4.6	FB_MC_Parametrization_PSx3xx.....	11
4.7	FB_MC_PosParametrization_PSx3xx.....	12
5	Parameterbeschreibung.....	14
5.1	<xActive>.....	14
5.2	<diActualPosition>.....	14
5.3	<xDeliveryState>.....	15
5.4	<uiDirection>.....	15
5.5	<xDone>.....	15
5.6	<stDrive>.....	16
5.7	<xError>.....	16
5.8	<sErrorDescription>.....	16
5.9	<udiErrorID>.....	17
5.10	<uiErrorParameter>.....	17
5.11	<xExecute>.....	18
5.12	<diLowerLimit>.....	18
5.13	<xManualRunToLargerValues>.....	18
5.14	<xManualRunToSmallerValues>.....	19
5.15	<stParameter>.....	19
5.16	<uiParameterNumber>.....	20
5.17	<xSaveSettings>.....	20
5.18	<diSetPoint>.....	20
5.19	<uiStepsPerTurn>.....	21
5.20	<usiSubIndex>.....	21
5.21	<diTargetPosition>.....	21
5.22	<tTimeBusCycle>.....	22

5.23	<diUpperLimit>.....	23
5.24	<diValue>.....	23
5.25	<sValue>.....	23
6	Anwendung der Funktionsbausteine	24
6.1	Projekt.....	24
6.2	Bibliothek/AOIs	25
6.3	Sperrung zwischen den Funktionsbausteinen	26
7	Beispielprogramme.....	27
7.1	Beispiel 1: Positioniermodus	27
7.2	Beispiel 2: Handfahrmodus.....	28
7.3	Beispiel 3: Aktuelle Position lesen.....	28
7.4	Beispiel 4: Auslieferungszustand schreiben	29
7.5	Beispiel 5: Parameter parametrisieren.....	29
7.6	Beispiel 6: Positionsparameter parametrisieren	30
7.7	Beispiel 7: Fehler obere Endbegrenzung erreicht.....	30
	Abbildungsverzeichnis	31

1 Sicherheitshinweise




1.1 Bestimmungsgemäße Verwendung

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

Die PSx3xx EIP 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 Körpervverletzung eintreten.
 WARNUNG!	WARNUNG! Bei Nichtbeachtung dieses Sicherheitshinweises können Tod oder schwere Körpervverletzung eintreten.
 VORSICHT!	VORSICHT! Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körpervverletzung 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 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	Datenstruktur zur internen Verwendung

2.2 <ST_FunctionBlocks_PSx3xx>

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_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>

In dieser Datenstruktur werden die Strukturen <ST_FunctionBlocks_PSx3xx> und <ST_DriveData_PSx3xx> zusammengefasst. Es können alle Ein- bzw. Ausgänge der Funktionsbausteine und des Antriebs mit dieser Struktur verbunden werden.

Zur einfachen Gruppierung mehrerer Antriebe, wird empfohlen ein Array dieser Struktur mit einem Eintrag für jeden Antrieb anzulegen.

3 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

4 Beschreibung der Funktionsbausteine

Eine ausführliche Beschreibung aller Parameter ist im Kapitel **5. Parameterbeschreibung** zu finden.

4.1 FB_MC_Move_Ps3xx

Dieser Funktionsbaustein wird zur Positionierung des Antriebs verwendet.

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

Abbildung 1 Verknüpfung der Variablen aus ST_Variablen_Ps3xx mit den Ein- und Ausgängen von FB_MC_Move_Ps3xx

Falls der Antrieb mehrere Fehler meldet, wird die <udErrorID> 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):

<udErrorID>	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 Ps3xx ein Gerät mit STO ist, dann müssen die Fehlerbeschreibung in Klammer berücksichtigt werden! (<udErrorID>: 16#xxx2x und 16#xxx5x)

4.2 FB_MC_Error_PSx3xx

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

```
FB_MC_Error_PSx3xx(FB_MC_Error_PSE1,
    PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stInput_Error_PSx3xx.xExecute,
    PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stOutput_Error_PSx3xx.xError,
    PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stOutput_Error_PSx3xx.udiErrorID,
    PSE_1.stFunctionBlocks_PSx3xx.stError_PSx3xx.stOutput_Error_PSx3xx.sErrorDescription,
    PSE_1.stDriveDate_PSx3xx);
```

Abbildung 2 Verknüpfung der Variablen aus ST_Variablen_PSx3xx mit den Ein- und Ausgängen von FB_MC_Error_PSx3xx

4.3 FC_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.

4.4 FB_MC_ReadParameter_PSx3xx

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

```
FB_MC_ReadParameter_PSx3xx(fbMC_ReadParameter_PSE1,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stInput_ReadParameter_PSx3xx.xExecute,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stInput_ReadParameter_PSx3xx.uiParameterNumber,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stInput_ReadParameter_PSx3xx.usiSubindex,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.xActive,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.xDone,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.xError,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.udiErrorID,
    PSE_1.stFunctionBlocks_PSx3xx.stReadParameter_PSx3xx.stOutput_ReadParameter_PSx3xx.diValue,
    PSE_1.stDriveDate_PSx3xx);
```

Abbildung 3 Verknüpfung der Variablen aus ST_Variablen_PSx3xx mit den Ein- und Ausgängen von FB_MC_ReadParameter_PSx3xx

4.5 FB_MC_WriteParameter_PSx3xx

Mit diesem FB können Parameterwerte in den Antrieb geschrieben werden.

```
FB_MC_WriteParameter_PSx3xx(fbMC_WriteParameter_PSE1,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.xExecute,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.uiParameterNumber,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stInput_WriteParameter_PSx3xx.diValue,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.xActive,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.xDone,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.xError,
    PSE_1.stFunctionBlocks_PSx3xx.stWriteParameter_PSx3xx.stOutput_WriteParameter_PSx3xx.udiErrorID,
    PSE_1.stDriveDate_PSx3xx);
```

Abbildung 4 Verknüpfung der Variablen aus ST_Variablen_PSx3xx mit den Ein- und Ausgängen von FB_MC_WriteParameter_PSx3xx

4.6 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.

```
FB_MC_Parametrization_PSx3xx(fbMC_Parametrization_PSE1,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.xExecute,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.xDeliveryState,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.stParameter,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stInput_Parametrization_PSx3xx.xSaveSettings,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.xActive,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.xDone,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.xError,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.udtErrorID,
    PSE_1.stFunctionBlocks_PSx3xx.stParametrization_PSx3xx.stOutput_Parametrization_PSx3xx.uiErrorParameter,
    PSE_1.stDriveDate_PSx3xx);
```

Abbildung 5 Verknüpfung der Variablen aus ST_Variablen_PSx3xx mit den Ein- und Ausgängen von 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

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

Abbildung 6 Parameter „Solldrehzahl Posi“ mit dem Wert von 20 parametrieren

- Durch das Setzen des <xEnable> = 0, wird der jeweilige Parameter nicht geschrieben.
- Wahlweise kann vor dem Setzen einzelner Parameter ein Auslieferungszustand angefordert werden. Dazu muss der Eingang <xDeliveryState> auf 1 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 1 gesetzt werden.
- Die Reihenfolge der Schreibzugriffe ist wie folgt:
<xDeliveryState>, Parameter 26, 28, 30, 10, 32, 34, ...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.

4.7 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.

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

Abbildung 7 Verknüpfung der Variablen aus ST_Variablen_PSx3xx mit den Ein- und Ausgängen von 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:
 1. Drehsinn (Parameter 26) → <uiDirection>
 2. Istwertbewertung Zähler (Parameter 28) → 400
 3. Istwertbewertung Nenner (Parameter 30) → <uiStepsPerTurn>

4. Istwert (Parameter 10) → $\langle diSetPoint \rangle$
5. Falls $\langle diSetPoint \rangle > \langle diUpperLimit \rangle$:
Oberes Mapping-Ende (Parameter 34) = $\langle diSetPoint \rangle + (3 \times \langle uiStepsPerTurn \rangle)$
sonst:
Oberes Mapping-Ende (Parameter 34) = $\langle diUpperLimit \rangle + (3 \times \langle uiStepsPerTurn \rangle)$
6. Obere Endbegrenzung (Parameter 36) → $\langle diUpperLimit \rangle$
7. Untere Endbegrenzung (Parameter 38) → $\langle diLowerLimit \rangle$

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

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

- Wahlweise können die geschriebenen Werte am Ende auch gespeichert werden. Dazu muss der Eingang $\langle xSaveSettings \rangle$ auf 1 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 $\langle xSaveSettings \rangle$ gesetzt ist.

5 Parameterbeschreibung

5.1 <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 0 auf 1 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 5.22 <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>	

5.2 <diActualPosition>

Baustein	FB_MC_Move_PSx3xx
Istwert der Position	
Datentyp	DInt
Default	-
Art	Output
Beschreibung	
<p>Dieser Wert ist eine Abbildung der Istposition. Falls ein Kommunikationsfehler auftritt, wird der Wert auf 0 gesetzt.</p>	

5.3 <xDeliveryState>

Baustein	FB_MC_Parametrization_PSx3xx
Laden der Werkseinstellungen (zunächst ohne Speichern)	
Datentyp	Bool
Default	0
Art	Input
Beschreibung	
Der Stationsname und die IP-Adresse bleiben jedoch unbeeinflusst.	

5.4 <uiDirection>

Baustein	FB_MC_PosParametrization_PSx3xx
Drehrichtung der Abtriebswelle	
Datentyp	UInt
Default	0
Art	Input
Beschreibung	
<p>Dieser Parameter entspricht dem Parameter Nummer 26 „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>	

5.5 <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 5.22 <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>	

5.6 <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.	

5.7 <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	0
Art	Output
Beschreibung	
<p><u>FB MC Move PSx3xx</u></p> <p>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).</p> <p><u>FB MC Error PSx3xx</u></p> <p>Der Ausgang <xError> wird jeden Zyklus aktualisiert, solange <xExecute> gesetzt ist. Wird <xExecute> zurückgesetzt, so nimmt dieser Ausgang den angegebenen Defaultwert an.</p> <p><u>Alle anderen Bausteine</u></p> <p>Das Fehlerbit wird jeden Zyklus aktualisiert und wird gesetzt, wenn während der Ausführung des FBs ein Fehler aufgetreten ist.</p>	

5.8 <sErrorDescription>

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

5.9 <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.

5.10 <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.	

5.11 <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	0	
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.		

5.12 <diLowerLimit>

Baustein	FB_MC_PosParametrization_PSx3xx	
Untere Endbegrenzung		
Datentyp	DInt	
Default	0	
Art	Input	
Beschreibung		
Dieser Parameter entspricht dem Parameter 38 „untere Endbegrenzung“.		

5.13 <xManualRunToLargerValues>

Baustein	FB_MC_Move_PSx3xx	
Handfahrt zu größeren Werten		
Datentyp	Bool	
Default	0	
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.

5.14 <xManualRunToSmallerValues>

Baustein	FB_MC_Move_PSx3xx
Handfahrt zu kleineren Werten	
Datentyp	Bool
Default	0
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.

5.15 <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-EIP entnommen werden.	

5.16 <uiParameterNumber>

Baustein	FB_MC_WriteParameter_PSx3xx, FB_MC_ReadParameter_PSx3xx	
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.		

5.17 <xSaveSettings>

Baustein	FB_MC_Parametrization_PSx3xx, FB_MC_PosParametrization_PSx3xx	
Speichern der Parametereinstellungen		
Datentyp	Bool	
Default	0	
Art	Input	
Beschreibung		
Dieser Parameter entspricht dem Parameter 113 „Auslieferungszustand“. (Funktion <Schreiben einer „1“>)		

5.18 <diSetPoint>

Baustein	FB_MC_PosParametrization_PSx3xx	
Istposition des Messsystems		
Datentyp	DInt	
Default	0	
Art	Input	
Beschreibung		
Dieser Parameter entspricht dem Parameter 10 „Istwert“.		

5.19 <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 30). Dabei wird angenommen, dass der Wert von „Istwertbewertung Zähler“ (Parameter 28) im Auslieferungszustand (400) ist.	

5.20 <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.	

5.21 <diTargetPosition>

Baustein	FB_MC_Move_PSx3xx
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.

5.22 <tTimeBusCycle>

Baustein	FB_MC_Move_PSx3xx
Aktualisierungszeit des EtherNet/IP Buszyklus	
Datentyp	Time
Default	40ms
Art	Input
Beschreibung	
<p>Bei langen Zykluszeiten auf dem EtherNet/IP-Bus kann es passieren, dass die SPS keinen Wechsel des Bits „Antrieb läuft“ empfängt. Dies ist dann der Fall, wenn die Fahrtdauer 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>	

5.23 <diUpperLimit>

Baustein	FB_MC_PosParametrization_PSx3xx
Obere Endbegrenzung	
Datentyp	DInt
Default	0
Art	Input
Beschreibung	
Dieser Parameter entspricht dem Parameter 36 „obere Endbegrenzung“.	

5.24 <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 numerischen Wert des gelesenen Parameters. FB_MC_WriteParameter: Bei einer steigenden Flanke auf xExecute wird dieser Wert auf das Gerät geschrieben.	

5.25 <sValue>

Baustein	FB_MC_ReadParameter_PSD4xx
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 Gerätetyp (23))	

6 Anwendung der Funktionsbausteine

Die Funktionsbausteine sind in Studio5000 v33 programmiert. Ein Upgrade auf höhere Versionen kann problemlos durchgeführt werden.

Dabei gibt es zwei Möglichkeiten die Funktionsbausteine zu nutzen.

- als Projekt oder
- als eingebundene Add On Instruction (AOI)

6.1 Projekt

Das Beispielprojekt ControlLogix_PSo3xx_2_2_AOI.L5K enthält bereits alle Funktionsbausteine, einen Antrieb und einige Beispielanwendungen.

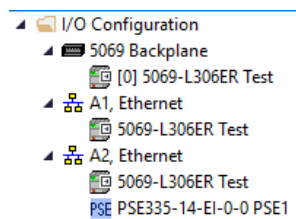


Abbildung 8 Antrieb

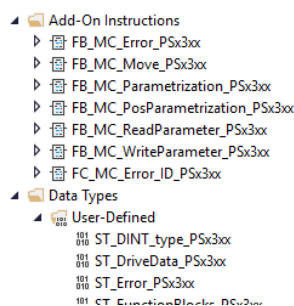


Abbildung 9
Funktionsbausteine und
Datentypen

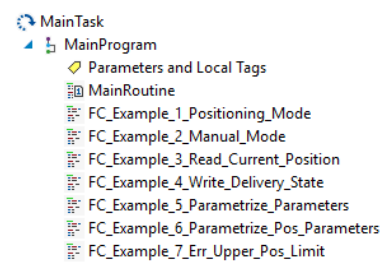


Abbildung 10
Beispielanwendungen

Bei Verwendung einer anderen als der dargestellten PSx3xx Variante, sollte der vorkonfigurierte Antrieb ersetzt und für den neu hinzugefügten Antrieb, ebenfalls die Bezeichnung „PSE1“ gewählt werden. Andernfalls müssen die Zuweisung in MainRoutine angepasst werden. Durch ein- bzw. auskommentieren, kann das gewünschte Beispielprogramm ausgewählt werden.

```

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

```

Abbildung 11 Auswahl des Beispielprogramms

Zum Start des Programms muss die Variable *xStartProgram* auf 1 gesetzt werden.

Über die Ansicht „Monitor Tags“ der MainRoutine, können zudem alle Eingänge der Funktionsbausteine beschrieben und gelesen werden. So kann die Funktionsweise der Bausteine bei laufender SPS getestet werden.

▲ PSE_1	Local	{...}
▶ PSE_1.stDriveDate_P5x3xx		{...}
▲ PSE_1.stFunctionBlocks_P5x3xx		{...}
▲ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx		{...}
▲ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx		{...}
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx.xExecute		1
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx.uiParameterNumber		10
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stInput_ReadParameter_P5x3xx.xSubindex		0
▲ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx		{...}
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.xActive		0
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.xDone		1
PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.xError		0
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.xdiErrorID		0
▶ PSE_1.stFunctionBlocks_P5x3xx.stReadParameter_P5x3xx.stOutput_ReadParameter_P5x3xx.xdiValue		25000
▶ PSE_1.stFunctionBlocks_P5x3xx.stWriteParameter_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stError_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stMove_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stParametrization_P5x3xx		{...}
▶ PSE_1.stFunctionBlocks_P5x3xx.stPosParametrization_P5x3xx		{...}

Abbildung 12 Auslesen von Parameter 10 (Istwert) über den FB_MC_ReadParameter_P5x3xx Baustein

6.2 Bibliothek/AOIs

Alternativ können die AOIs in ein bestehendes Projekt importiert werden. Dazu muss die Datei AOI_P5x3xx.L5X importiert werden. Zusätzlich kann die Datei AOI_P5x3xx_Datatypes.L5X importiert werden um den Typ „ST_DriveData_P5x3xx“ zu nutzen.

Für jeden verwendeten Antrieb muss ein Element vom Typ „ST_DriveData_P5x3xx“ angelegt und mit den Ein- und Ausgangsdaten des Antriebs verknüpft werden.

```
//copy input data from drive into struct
PSE_1.stDriveDate_P5x3xx.xCommunicationError := PSE1:I.ConnectionFaulted;
PSE_1.stDriveDate_P5x3xx.stPrivate.stInputModule.luiStatusWord := PSE1:I.status_word;
PSE_1.stDriveDate_P5x3xx.stPrivate.stInputModule.liCurrentSpeed := PSE1:I.actual_speed;
PSE_1.stDriveDate_P5x3xx.stPrivate.stInputModule.lidiCurrentPosition := PSE1:I.actual_position;
PSE_1.stDriveDate_P5x3xx.stPrivate.stInputModule.IuiPKE := PSE1:I.Parameter_ID;
PSE_1.stDriveDate_P5x3xx.stPrivate.stInputModule.IuiIND := PSE1:I.Subindex;
PSE_1.stDriveDate_P5x3xx.stPrivate.stInputModule.IudiPWE := PSE1:I.Parameter_value;
//copy output data from struct to output
PSE1:O.control_word := PSE_1.stDriveDate_P5x3xx.stPrivate.stOutputModule.OuiControlWord;
PSE1:O.target_position := PSE_1.stDriveDate_P5x3xx.stPrivate.stOutputModule.OdiTargetPosition;
PSE1:O.Parameter_ID := PSE_1.stDriveDate_P5x3xx.stPrivate.stOutputModule.OuiPKE;
PSE1:O.Subindex:= PSE_1.stDriveDate_P5x3xx.stPrivate.stOutputModule.OuiIND;
PSE1:O.Parameter_value:= PSE_1.stDriveDate_P5x3xx.stPrivate.stOutputModule.OudiPWE;
```

Abbildung 13 Verknüpfung der Prozessdaten des Antriebs mit den Variablen der Steuerung

Dieses Element muss mit den *stDrive* Eingängen der verwendeten Funktionbausteine des jeweiligen Antriebs verbunden werden. Es wird empfohlen die restlichen Ein- und Ausgänge, wie in Kapitel 4 dargestellt, mit den entsprechenden Gegenstücken in ST_FunctionBlocks_P5x3xx zu verknüpfen.

Ebenfalls müssen Instanzen der zu verwendenden Bausteine *FB_MC_*_P5x3xx* angelegt werden. Es können auch nur einzelne Bausteine verwendet werden, falls ein eingeschränkter Funktionsumfang ausreichend ist.

6.3 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_PSx3xx** gesetzt ist, können die Bausteine **FB_MC_Parametrization_PSx3xx** und **FB_MC_PosParametrization_PSx3xx** nicht aktiviert werden (<udiErrorID>: 16#x1000).
- Dagegen ist es möglich, während einer Bewegung **FB_MC_ReadParameter_PSx3xx** oder **FB_MC_WriteParameter_PSx3xx** 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 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 der Aufruf der jeweiligen Subroutine muss in das Hauptprogramm eingefügt werden, um das Programm auszuführen (siehe Abbildung 11).

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 1 gesetzt werden.
- Bei jedem Beispielprogramm ist der erste Schritt nach dem Start des Programms, dass alle relevanten Variablen für die Initialisierung auf 0 gesetzt werden.
- Mit <xStopProgram> können die Programme beendet werden.

7.1 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 0 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 40 „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 40 „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, wird der Fahrbefehl zurückgesetzt und die State Machine auf 1 gesetzt.

7.2 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 0 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 40 „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.

7.3 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 0 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 40 „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 40 „Positionierfenster“), dann wird der Lesebefehl zurückgesetzt und die State Machine auf 100 gesetzt.

7.4 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 0 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 40 „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.)

7.5 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 0 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 40 „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 <code><xEnable></code> der Parameter gesetzt sein. Wenn der Parametrisierbefehl erfolgreich durchgeführt wurde, dann wird der Parametrisierbefehl 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 40 „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und die State Machine auf 100 gesetzt.

7.6 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 0 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 40 „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.

7.7 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 0 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 40 „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 Verknüpfung der Variablen aus ST_ Variables_PSt3xx mit den Ein- und Ausgängen von FB_MC_Move_PSt3xx	10
Abbildung 2 Verknüpfung der Variablen aus ST_ Variables_PSt3xx mit den Ein- und Ausgängen von FB_MC_Error_PSt3xx	11
Abbildung 3 Verknüpfung der Variablen aus ST_ Variables_PSt3xx mit den Ein- und Ausgängen von FB_MC_ReadParameter_PSt3xx.....	11
Abbildung 4 Verknüpfung der Variablen aus ST_ Variables_PSt3xx mit den Ein- und Ausgängen von FB_MC_WriteParameter_PSt3xx	11
Abbildung 5 Verknüpfung der Variablen aus ST_ Variables_PSt3xx mit den Ein- und Ausgängen von FB_MC_Parametrization_PSt3xx.....	12
Abbildung 6 Parameter „Solldrehzahl Posi“ mit dem Wert von 20 parametrieren.....	12
Abbildung 7 Verknüpfung der Variablen aus ST_ Variables_PSt3xx mit den Ein- und Ausgängen von FB_MC_PosParametrization_PSt3xx.....	12
Abbildung 8 Antrieb.....	24
Abbildung 9 Funktionsbausteine und Datentypen	24
Abbildung 10 Beispielanwendungen.....	24
Abbildung 11 Auswahl des Beispielprogramms.....	24
Abbildung 12 Auslesen von Parameter 10 (Istwert) über den FB_MC_ReadParameter_PSt3xx Baustein.....	25
Abbildung 13 Verknüpfung der Prozessdaten des Antriebs mit den Variablen der Steuerung	25