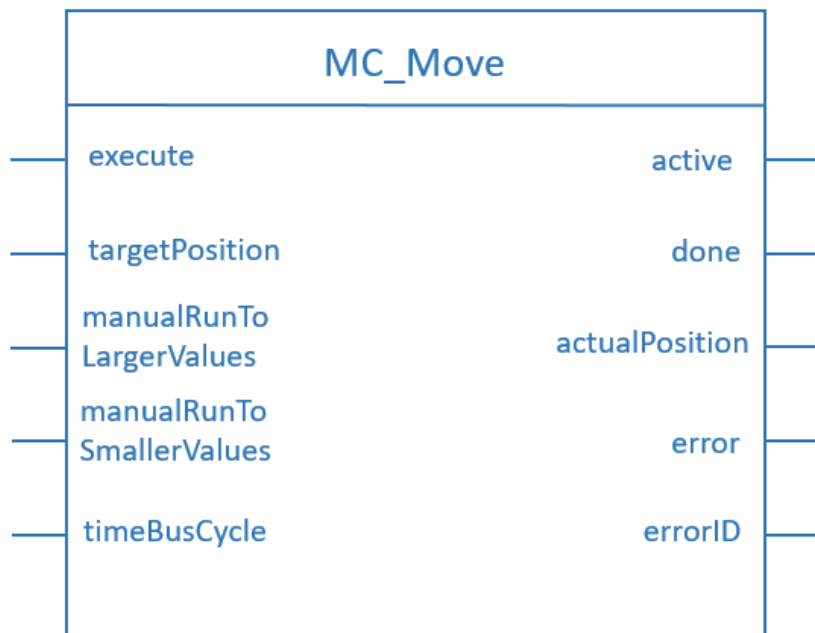


Funktionsbausteine PSD4xx für TIA-Portal

- SIMATIC S7 - 300
- SIMATIC S7 - 1200 und SIMATIC S7 - 1500



Bedeutung der Betriebsanleitung

Diese Betriebsanleitung beschreibt die Funktionsbausteine für die Positioniersysteme PSD4xx PN (mit PROFINET-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

© 2023

11.04.2023, TS & JB

7100.006634B Version 2.0 Betriebsanleitung Funktionsbausteine PSD4xx PN

Inhaltsverzeichnis

1	Sicherheitshinweise	5
1.1	Bestimmungsgemäße Verwendung.....	5
1.2	Warnsymbole.....	5
2	PLC-Datentypen.....	6
2.1	driveData_PSD4xx	6
2.2	Int_type_PSD4xx.....	7
2.3	DInt_Type_PSD4xx.....	7
2.4	parameter_PSD4xx.....	8
3	Datenbaustein Data_Store (DB100)	9
4	Fehlerbeschreibung (<errorID>)	10
5	Beschreibung der Funktionsbausteine	12
5.1	MC_Communication_PSD4xx_S7-1200_1500 (FC101), MC_Communication_PSD4xx_S7-300 (FC101).....	12
5.2	MC_Move_PSD4xx (FB110).....	12
5.3	MC_Error_PSD4xx (FB111)	13
5.4	MC_Error_ID_PSD4xx (FC100)	13
5.5	MC_ReadParameter_PSD4xx (FB112)	14
5.6	MC_WriteParameter_PSD4xx (FB113).....	14
5.7	MC_Parametrization_PSD4xx (FB114)	14
5.8	MC_PosParametrization_PSD4xx (FB115).....	15
6	Parameterbeschreibung	17
6.1	<active>	17
6.2	<actualPosition>	17
6.3	<communicationError>	18
6.4	<deliveryState>.....	18
6.5	<direction>.....	18
6.6	<done>	19
6.7	<drive>	19
6.8	<error>	20
6.9	<errorDescription>	20
6.10	<errorID>	21
6.11	<errorParameter>	21
6.12	<execute>.....	22
6.13	<inputAddress>	22
6.14	<lowerLimit>.....	22
6.15	<manuelRunToLargerValues>	23

6.16	<manualRunToSmallerValues>	23
6.17	<outputAddress>	23
6.18	<parameter>	24
6.19	<parameterNumber>	24
6.20	<saveSettings>	24
6.21	<setPoint>	25
6.22	<stepsPerTurn>	25
6.23	<subIndex>	25
6.24	<targetPosition>	26
6.25	<timeBusCycle>	26
6.26	<upperLimit>	27
6.27	<value>	27
7	Anwendung der Funktionsbausteine	28
7.1	Projekt	28
7.2	Bibliothek	28
7.3	Sperrung zwischen den Funktionsbausteinen	30
7.4	Mehrere PSD4xx in einem Projekt	30
8	Beispielprogramme	32
8.1	Beispiel 1: Positioniermodus	32
8.2	Beispiel 2: Handfahrmodus	33
8.3	Beispiel 3: Aktuelle Position lesen	33
8.4	Beispiel 4: Drehsinn schreiben	34
8.5	Beispiel 5: Parameter parametrisieren	35
8.6	Beispiel 6: Positionsdaten parametrisieren	35
8.7	Beispiel 7: Fehler obere Endbegrenzung erreicht	36
	Abbildungsverzeichnis	37

1 Sicherheitshinweise

1.1 Bestimmungsgemäße Verwendung

Die Positioniersysteme PSD4xx PN eignen sich besonders zur automatischen Einstellung von Werkzeugen, Anschlägen oder Spindeln bei Holzverarbeitungsmaschinen, Verpackungsmaschinen, Druckmaschinen, Abfüllanlagen und bei Sondermaschinen.

Die PSD4xx PN 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 darauf folgend beschriebenen Gefahren bei der Handhabung der Anlage hingewiesen:



WARNUNG!

Sie werden auf eine Gefährdung hingewiesen, die zu Körperverletzungen bis hin zum Tod führen kann, wenn Sie die gegebenen Anweisungen missachten.



ACHTUNG!

Sie werden auf eine Gefährdung hingewiesen, die zu einem erheblichen Sachschaden führen kann, wenn Sie die gegebenen Anweisungen missachten.



INFORMATION!

Sie erhalten wichtige Informationen zum sachgemäßen Betrieb.

2 PLC-Datentypen

2.1 driveData_PSD4xx

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

Parametername	Datentyp	geschrieben von	Beschreibung
<pdAddressIn>	Int (S7-1200 und S7-1500) DWord (S7-300)	Benutzer	Adresse Prozessdaten (Device → Controller)
<pdAddressOut>	Int (S7-1200 und S7-1500) DWord (S7-300)	Benutzer	Adresse Prozessdaten (Controller → Device)
<axisName>	String[16]	Benutzer (optional)	Name der Achse
<axisDescription>	String[32]	Benutzer (optional)	Beschreibung (z.B. Funktion, Aufgabe dieser Achse)
<state>	DInt	Funktionsbausteine	Aktueller Status
<communicationError>	Bool	Funktionsbausteine	Kommunikationsfehler zum IO-Device
<private>	Struct	Funktionsbausteine	Datenstruktur zur internen Verwendung

Die folgende Darstellung zeigt, wie im TIA-Portal die vergebenen Adressen der Prozessdaten überprüft werden können:

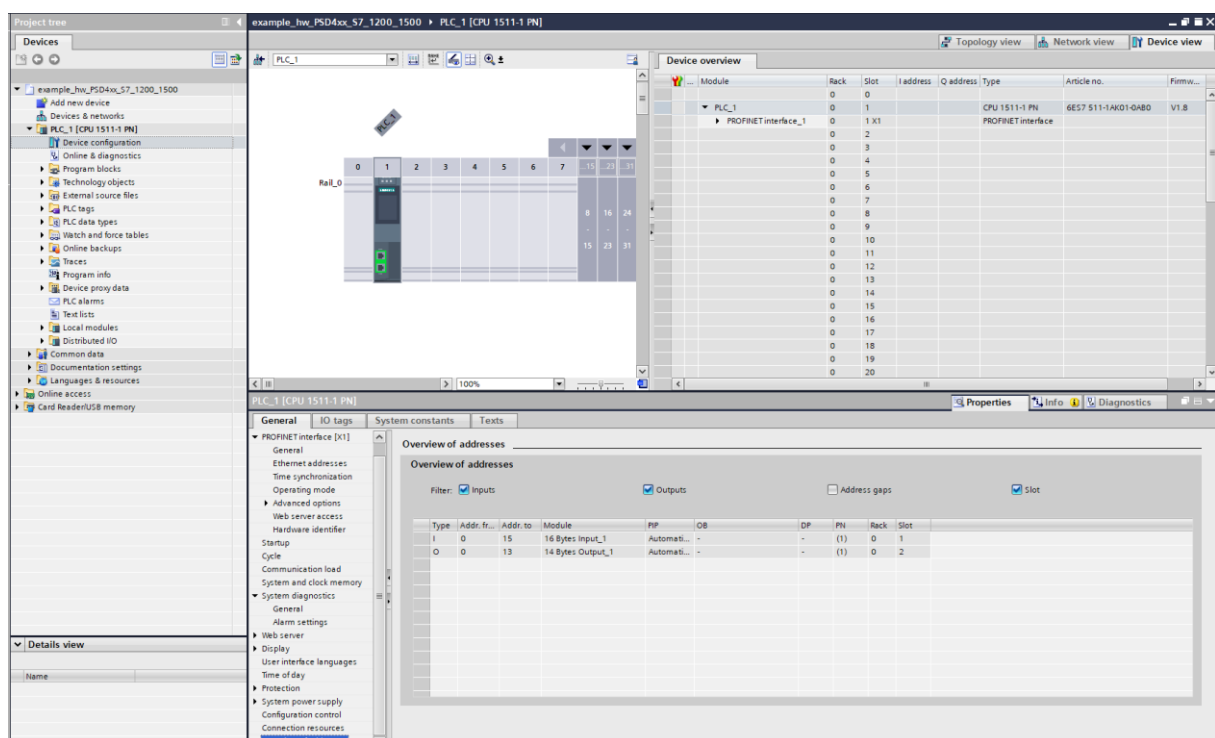


Abbildung 1: Adressen der Prozessdaten aus Sicht der PLC

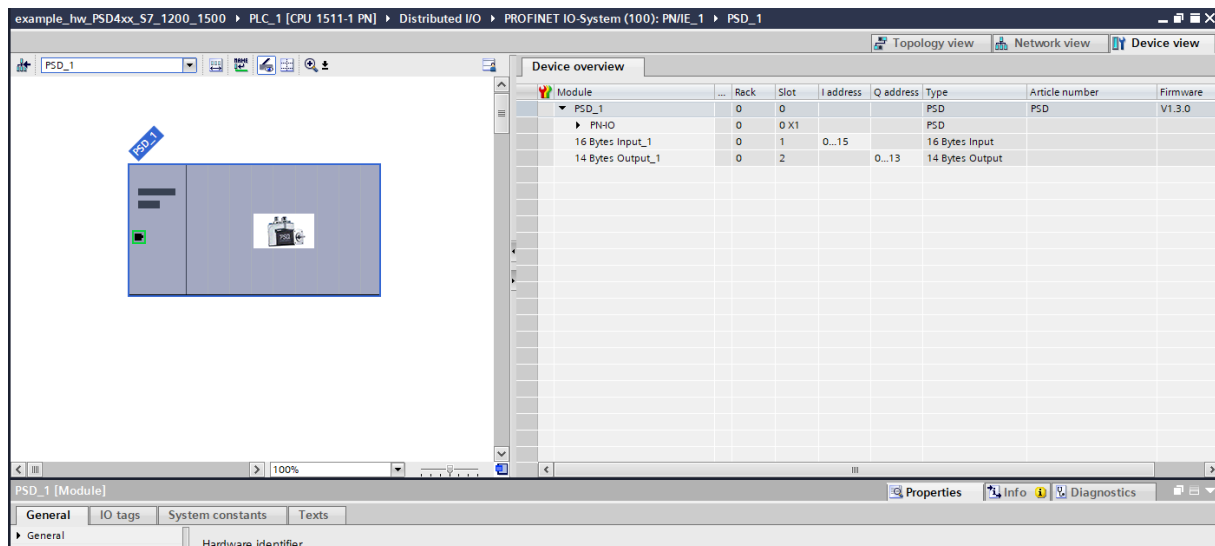


Abbildung 2: Einzustellende Adressen aus Sicht des Geräts

2.2 Int_type_PSD4xx

example_hw_PSD4xx_S7_1200_1500 ▶ PLC_1 [CPU 1511-1 PN] ▶ PLC data types ▶ Int_type_PSD4xx							
Int_type_PSD4xx							
	Name	Data type	Default value	Accessible f...	Visible in ...	Setpoint	Comment
1	enable	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	enable
2	value	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	value

Abbildung 3: Datentyp <Int_type_PSD4xx>

Dieser Datentyp wird nur zusammen mit dem FB MC_Parametrization_PSD4xx benötigt.

Der Datentyp ist als Unterstruktur im Datentyp <parameter_PSD4xx> enthalten. Er dient für alle Parameter des Typs <Integer> (16 Bit).

2.3 DInt_Type_PSD4xx

example_hw_PSD4xx_S7_1200_1500 ▶ PLC_1 [CPU 1511-1 PN] ▶ PLC data types ▶ DInt_type_PSD4xx							
DInt_type_PSD4xx							
	Name	Data type	Default value	Accessible f...	Visible in ...	Setpoint	Comment
1	enable	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	enable
2	value	DInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	value

Abbildung 4: Datentyp <DInt_type_PSD4xx>

Dieser Datentyp wird nur zusammen mit dem FB MC_Parametrization_PSD4xx benötigt.

Der Datentyp ist als Unterstruktur im Datentyp <parameter_PSD4xx> enthalten. Er dient für alle Parameter des Typs <Double Integer> (32 Bit).

2.4 parameter_PSD4xx

example_hw_PSD4xx_S7_1200_1500 ▶ PLC_1 [CPU 1511-1 PN] ▶ PLC data types ▶ parameter_PSD4xx

parameter_PSD4xx							
	Name	Data type	Default value	Accessible f...	Visible in ...	Setpoint	Comment
1	actualValue_3	"DInt_type_PSD..."		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	actual value
2	enable	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	enable
3	value	DInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	value
4	dirRotation_37	"Int_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	direction of rotation
5	enable	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	enable
6	value	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	value
7	posScaleNumerator_38	"Int_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	position scaling numerator
8	enable	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	enable
9	value	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	value
10	posScaleDenominator_39	"Int_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	position scaling denominator
11	referencingValue_40	"DInt_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	referencing value
12	upperMappingEnd_41	"DInt_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	upper mapping end
13	upperLimit_42	"DInt_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	upper limit
14	lowerLimit_43	"DInt_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	lower limit
15	positioningWindow_44	"Int_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	positioning window
16	loopLength_45	"DInt_type_PSD4xx"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	length of loop

Abbildung 5: Ausschnitt aus Datentyp <parameter_PSD4xx>

Dieser Datentyp wird nur zusammen mit dem FB MC_Parametrization_PSD4xx benötigt.

Er dient als Vorlage, um die einzelnen Parameter zu setzen und ist in dem Datenbaustein <Data_Store> enthalten.

Mit <enable> auf TRUE wird der jeweilige Parameter zum Schreiben freigegeben.

<value> enthält den zu schreibenden Wert.



INFORMATION!

Vergessen Sie nicht das <enable> wieder auf FALSE zu setzen, nachdem Sie das <execute> auf FALSE gesetzt haben!

3 Datenbaustein Data_Store (DB100)

Dieser Datenbaustein (DB) dient als Vorlage für die Zuweisung an die Funktionsbausteine. Der DB stellt die notwendigen Variablen bereit, um alle Ein- und Ausgänge aller zur Verfügung stehender Funktionsbausteine setzen zu können.

Jede Variable ist aktuell nur einfach vorhanden. Bei mehreren PSD4xx in einem Projekt muss die Anzahl der Variablen kopiert werden.

Beispiel

```

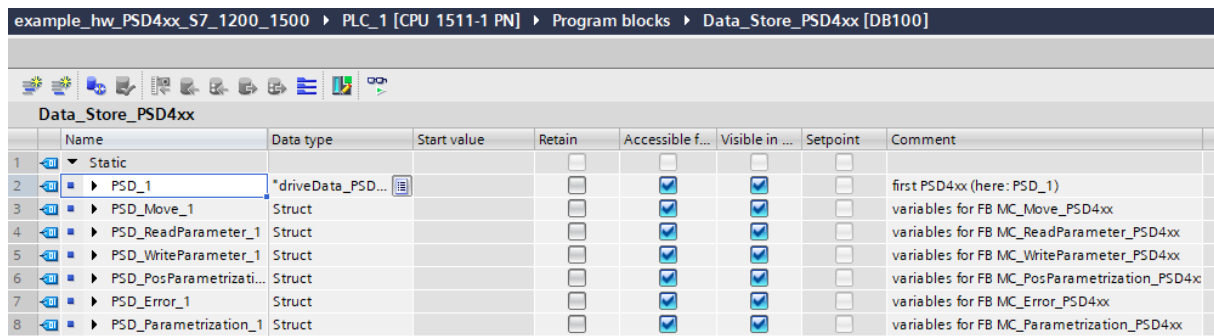
26 // "MC_Move_PSD4xx_DB" --> instance of "MC_Move_PSD4xx"
27 "MC_Move_PSD4xx_DB" (execute := "Data_Store_PSD4xx".PSD_Move_1.input.execute,
28     targetPosition := "Data_Store_PSD4xx".PSD_Move_1.input.targetPosition,
29     manualRunToLargerValues := "Data_Store_PSD4xx".PSD_Move_1.input.manualRunToLargerValues,
30     manualRunToSmallerValues := "Data_Store_PSD4xx".PSD_Move_1.input.manualRunToSmallerValues,
31     timeBusCycle := "Data_Store_PSD4xx".PSD_Move_1.input.timeBusCycle,
32     active => "Data_Store_PSD4xx".PSD_Move_1.output.active,
33     done => "Data_Store_PSD4xx".PSD_Move_1.output.done,
34     actualPosition => "Data_Store_PSD4xx".PSD_Move_1.output.actualPosition,
35     error => "Data_Store_PSD4xx".PSD_Move_1.output.error,
36     errorID => "Data_Store_PSD4xx".PSD_Move_1.output.errorID,
37     drive := "Data_Store_PSD4xx".PSD_1);
--

```

Abbildung 6: Zuweisung der Variablen von DB <Data_Store> zum FB **MC_Move_PSD4xx**

Falls der DB in das Projekt übernommen wird, ist 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_hw_PSD4xx_S7_1200_1500 ▶ PLC_1 [CPU 1511-1 PN] ▶ Program blocks ▶ Data_Store_PSD4xx [DB100]



	Name	Data type	Start value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Static			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2	PSD_1	*driveData_PSD...		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	first PSD4xx (here: PSD_1)
3	PSD_Move_1	Struct		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	variables for FB MC_Move_PSD4xx
4	PSD_ReadParameter_1	Struct		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	variables for FB MC_ReadParameter_PSD4xx
5	PSD_WriteParameter_1	Struct		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	variables for FB MC_WriteParameter_PSD4xx
6	PSD_PosParametrizati...	Struct		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	variables for FB MC_PosParametrization_PSD4x
7	PSD_Error_1	Struct		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	variables for FB MC_Error_PSD4xx
8	PSD_Parametrization_1	Struct		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	variables for FB MC_Parametrization_PSD4xx

Abbildung 7: DB <Data_Store> mit den Variablen für einen Antrieb

4 Fehlerbeschreibung (<errorID>)

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

<errorID> (hex)	Beschreibung
16#F000 (Maske)	Funktionsbaustein
16#0xxx	<u>Kein</u> Fehlercode
16#1xxx	Fehler in MC_Move_PSD4xx
16#2xxx	Fehler in MC_Error_PSD4xx
16#3xxx	Fehler in MC_ReadParameter_PSD4xx
16#4xxx	Fehler in MC_WriteParameter_PSD4xx
16#5xxx	Fehler in MC_Parametrization_PSD4xx
16#6xxx	Fehler in MC_PosParametrization_PSD4xx
16#0F00 (Maske)	Interne Fehler in den Funktionsbausteinen und Prozessdatenfehler
16#x0xx	<u>Kein</u> interner Fehler in den Funktionsbausteinen und Prozessdatenfehler
16#x1xx	Fehler in der Zustandsmaschine (Verriegelung)
16#x2xx	Ungültige Eingangsadresse der Prozessdaten
16#x3xx	Ungültige Ausgangsadresse der Prozessdaten
16#x4xx	Kommunikationsfehler beim Lesen der Prozessdaten
16#x5xx	Kommunikationsfehler beim Schreiben der Prozessdaten
16#x6xx	Ungültiger Schreibbefehl
16#00F0 (Maske)	Fehler in den Parametern
16#xx0x	<u>Kein</u> Fehlverhalten in den Parametern
16#xx1x	Parameter: Kommunikationsauszeit (1000 ms)
16#xx2x	Parameter: ungültiger Parameternummer
16#xx3x	Parameter: Parameternummer ist nur lesbar
16#xx4x	Parameter: untere Endbegrenzung wurde unterschritten oder obere Endbegrenzung wurde überschritten
16#xx5x	Parameter: ungültiger Subindex
16#xx6x	Parameter: kein Array
16#xx7x	Parameter: ungültiger Datentyp
16#xx8x	Parameter: kein Schreibzugriff
16#xx9x	Parameter: Anfrage kann wegen aktuellen Betriebszustand nicht bearbeitet werden
16#xxAx	Andere Fehler
16#000F (Maske)	Antriebsfehler
16#xxx0	<u>Kein</u> Antriebsfehler
16#xxx1	Schleppfehler
16#xxx2	Unter- oder Überspannung der Motorversorgung (STO-Freigabe inaktiv*)
16#xxx3	Positionierung wurde abgebrochen
16#xxx4	Temperaturüberschreitung
16#xxx5	Messsystemfehler (Messsystem- oder STO-Hardwarefehler*)
16#xxx6	Positionierfehler (Blockieren)
16#xxx7	Manuelles Verdrehen
16#xxx8	Zielposition falsch
16#xxx9	Unter- oder Überspannung der Motorspannung/ Ausfall der Spannungsüberwachung
16#xxxA	Untere Endbegrenzung unterschritten
16#xxxB	Obere Endbegrenzung überschritten

* Falls das PSD4xx ein Gerät mit STO ist, dann müssen die Fehlerbeschreibung in Klammern berücksichtigt werden! (<errorID>: 16#xxx2 und 16#xxx5)

Die Fehler „Antriebsfehler“ sind eine Abbildung der Fehlerbits im Statuswort des PSD4xx.

Beispiele

- Fahrauftrag (**MC_Move_PSD4xx**) mit falscher Zielposition → <errorID> = 16#1008
- Parameter schreiben (**MC_WriteParameter_PSD4xx**) mit ungültiger Parameternummer → <errorID> = 16#4020

5 Beschreibung der Funktionsbausteine

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

5.1 MC_Communication_PSD4xx_S7-1200_1500 (FC101), MC_Communication_PSD4xx_S7-300 (FC101)

Verwenden Sie für nachfolgende Steuerung die entsprechende Funktion:

Steuerung	Funktion
S7-300-Steuerung	MC_Communication_PSD4xx_S7-300
S7-1200-Steuerung S7-1500-Steuerung	MC_Communication_PSD4xx_S7-1200_1500

Die Funktionalität ist in beiden Fällen identisch. Im Folgenden wird die Funktion **MC_Communication_PSD4xx_S7-1200_1500** beschrieben.

Diese Funktion (FC) dient der Kommunikation mit dem Antrieb (IO-Device). Er führt die Kommunikation zentral für die folgenden Bausteine durch:

- MC_Move_PSD4xx
- MC_ReadParameter_PSD4xx
- MC_WriteParameter_PSD4xx
- MC_Error_PSD4xx
- MC_Parametrization_PSD4xx
- MC_PosParametrization_PSD4xx

Das Ein- und Ausgangsmodul wird im Datentyp <driveData_PSD4xx> abgelegt. Über diese Schnittstelle können alle andere Bausteine darauf zugreifen.

```

19 //function MC_Communication -> communication to PSDs
20 //central communication for all MCs (Motion Controls)
21 MC_Communication_PSD4xx_S7-1200_1500(inputAddress := "Data_Store_PSD4xx".PSD_1.pdAddressIn,
22                                     outputAddress := "Data_Store_PSD4xx".PSD_1.pdAddressOut,
23                                     communicationError := "Data_Store_PSD4xx".PSD_1.communicationError,
24                                     drive := "Data_Store_PSD4xx".PSD_1);
--

```

Abbildung 8: Zuweisung der Variablen von DB <Data_Store> zu der FC
MC_Communication_PSD4xx_1200_1500

5.2 MC_Move_PSD4xx (FB110)

Dieser Funktionsbaustein wird zur Positionierung des Antriebs verwendet.

```

26 //MC_Move_PSD4xx_DB --> instance of "MC_Move_PSD4xx"
27 MC_Move_PSD4xx_DB(execute := "Data_Store_PSD4xx".PSD_Move_1.input.execute,
28                  targetPosition := "Data_Store_PSD4xx".PSD_Move_1.input.targetPosition,
29                  manualRunToLargerValues := "Data_Store_PSD4xx".PSD_Move_1.input.manualRunToLargerValues,
30                  manualRunToSmallerValues := "Data_Store_PSD4xx".PSD_Move_1.input.manualRunToSmallerValues,
31                  timeBusCycle := "Data_Store_PSD4xx".PSD_Move_1.input.timeBusCycle,
32                  active => "Data_Store_PSD4xx".PSD_Move_1.output.active,
33                  done => "Data_Store_PSD4xx".PSD_Move_1.output.done,
34                  actualPosition => "Data_Store_PSD4xx".PSD_Move_1.output.actualPosition,
35                  error => "Data_Store_PSD4xx".PSD_Move_1.output.error,
36                  errorID => "Data_Store_PSD4xx".PSD_Move_1.output.errorID,
37                  drive := "Data_Store_PSD4xx".PSD_1);

```

Abbildung 9: Zuweisung der Variablen von DB <Data_Store> zum FB **MC_Move_PSD4xx**

Falls der Antrieb mehrere Fehler meldet, wird die <errorID> 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#x1xx):

<errorID>	Beschreibung
16#x1xx	Fehler in der Zustandsmaschine (Verriegelung)
16#x2xx	Ungültige Eingangsadresse der Prozessdaten
16#x3xx	Ungültige Ausgangsadresse der Prozessdaten
16#x4xx	Kommunikationsfehler beim Lesen der Prozessdaten
16#x5xx	Kommunikationsfehler beim Lesen der Prozessdaten
16#xxx2	Unter- oder Überspannung der Motorversorgung (STO-Freigabe inaktiv*)
16#xxx4	Temperaturüberschreitung
16#xxx5	Messsystemfehler (Messsystem- oder STO-Hardwarefehler*)
16#xxx8	Zielposition falsch
16#xxx9	Unter- oder Überspannung der Motorspannung/ Ausfall der Spannungsüberwachung
16#xxx6	Positionierfehler (Blockieren)
16#xxx7	Manuelles Verdrehen
16#xxxA	Untere Endbegrenzung unterschritten
16#xxxB	Obere Endbegrenzung überschritten
16#xxx3	Positionierung wurde abgebrochen
16#xxx1	Schleppfehler

*Falls das PSD4xx ein Gerät mit STO ist, dann müssen die Fehlerbeschreibung in Klammer berücksichtigt werden! (<errorID>: 16#xxx2 und 16#xxx5)

5.3 MC_Error_PSD4xx (FB111)

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

```

60 // "MC_Error_PSD4xx_DB" --> instance of "MC_Error_PSD4xx"
61 □ "MC_Error_PSD4xx_DB" (execute := "Data_Store_PSD4xx".PSD_Error_1.input.execute,
62   error => "Data_Store_PSD4xx".PSD_Error_1.output.error,
63   errorID => "Data_Store_PSD4xx".PSD_Error_1.output.errorID,
64   errorDescription => "Data_Store_PSD4xx".PSD_Error_1.output.errorDescription,
65   drive := "Data_Store_PSD4xx".PSD_1);

```

Abbildung 10: Zuweisung der Variablen von DB <Data_Store> zum FB **MC_Error_PSD4xx**

5.4 MC_Error_ID_PSD4xx (FC100)

Dieser FC wird intern als Unterfunktion der FBs **MC_Move_PSD4xx**, **MC_ReadParameter_PSD4xx**, **MC_WriteParameter_PSD4xx** und **MC_Error_PSD4xx** eingesetzt. Er ist lediglich aus der Bibliothek in das Anwenderprogramm zu kopieren. Die Beschaltung findet schon intern statt.

5.5 MC_ReadParameter_PSD4xx (FB112)

Mit diesem FB können Werte von Parametern aus dem Antrieb ausgelesen werden. Alle Parameter außer Parameter 23 („Gerätetyp als String“) können gelesen werden.

```

39 // "MC_ReadParameter_PSD4xx_DB" --> instance of "MC_ReadParameter_PSD4xx"
40 □ "MC_ReadParameter_PSD4xx_DB" (execute := "Data_Store_PSD4xx".PSD_ReadParameter_1.input.execute,
41     parameterNumber := "Data_Store_PSD4xx".PSD_ReadParameter_1.input.parameterNumber,
42     subindex := "Data_Store_PSD4xx".PSD_ReadParameter_1.input.subindex,
43     active => "Data_Store_PSD4xx".PSD_ReadParameter_1.output.active,
44     done => "Data_Store_PSD4xx".PSD_ReadParameter_1.output.done,
45     error => "Data_Store_PSD4xx".PSD_ReadParameter_1.output.error,
46     errorID => "Data_Store_PSD4xx".PSD_ReadParameter_1.output.errorID,
47     value => "Data_Store_PSD4xx".PSD_ReadParameter_1.output.value,
48     drive := "Data_Store_PSD4xx".PSD_1);

```

Abbildung 11: Zuweisung der Variablen von DB <Data_Store> zum FB **MC_ReadParameter_PSD4xx**

5.6 MC_WriteParameter_PSD4xx (FB113)

Mit diesem FB können Parameterwerte in den Antrieb geschrieben werden.

```

50 // "MC_WriteParameter_PSD4xx_DB" --> instance of "MC_WriteParameter_PSD4xx"
51 □ "MC_WriteParameter_PSD4xx_DB" (execute := "Data_Store_PSD4xx".PSD_WriteParameter_1.input.execute,
52     parameterNumber := "Data_Store_PSD4xx".PSD_WriteParameter_1.input.parameterNumber,
53     value := "Data_Store_PSD4xx".PSD_WriteParameter_1.input.value,
54     active => "Data_Store_PSD4xx".PSD_WriteParameter_1.output.active,
55     done => "Data_Store_PSD4xx".PSD_WriteParameter_1.output.done,
56     error => "Data_Store_PSD4xx".PSD_WriteParameter_1.output.error,
57     errorID => "Data_Store_PSD4xx".PSD_WriteParameter_1.output.errorID,
58     drive := "Data_Store_PSD4xx".PSD_1);
59

```

Abbildung 12: Zuweisung der Variablen von DB <Data_Store> zum FB **MC_WriteParameter_PSD4xx**

5.7 MC_Parametrization_PSD4xx (FB114)

Mit diesem FB können sämtliche Parameter des Antriebs auf einmal geschrieben werden. Das ist vor allem für die Erstinbetriebnahme von Nutzen. Dabei wurde eine übersichtliche Struktur gewählt, da sämtliche Parameter als Block mit dem Datentyp <parameter_PSD4xx> übergeben werden.

```

68 // "MC_Parametrization_PSD4xx_DB" --> instance of "MC_Parametrization_PSD4xx"
69 □ "MC_Parametrization_PSD4xx_DB" (execute := "Data_Store_PSD4xx".PSD_Parametrization_1.input.execute,
70     deliveryState := "Data_Store_PSD4xx".PSD_Parametrization_1.input.deliveryState,
71     parameter := "Data_Store_PSD4xx".PSD_Parametrization_1.input.parameter,
72     saveSettings := "Data_Store_PSD4xx".PSD_Parametrization_1.input.saveSettings,
73     active => "Data_Store_PSD4xx".PSD_Parametrization_1.output.active,
74     done => "Data_Store_PSD4xx".PSD_Parametrization_1.output.done,
75     error => "Data_Store_PSD4xx".PSD_Parametrization_1.output.error,
76     errorID => "Data_Store_PSD4xx".PSD_Parametrization_1.output.errorID,
77     errorParameter => "Data_Store_PSD4xx".PSD_Parametrization_1.output.errorParameter,
78     drive := "Data_Store_PSD4xx".PSD_1);
79

```

Abbildung 13: Zuweisung der Variablen von DB <Data_Store> zum FB **MC_Parametrization_PSD4xx**

Folgendes ist bei der Nutzung des FBs zu beachten:

Zu jedem Parameter gibt es eine Variable <value> und eine Variable <enable>. Siehe hierzu auch Kapitel 2.4, parameter_PSD4x.

Beispiel

```
"Data_Store".PSD_Parametrization_1.input.parameter.dirRotation_37.enable = TRUE;
```

```
"Data_Store".PSD_Parametrization_1.input.parameter.dirRotation_37.value = 1;
```

→ Parameter 37 (Drehsinn) wird auf linksdrehend geschrieben.

- Durch das Setzen des <enable> = FALSE, wird der jeweilige Parameter nicht geschrieben.

- Wahlweise kann vor dem Setzen einzelner Parameter ein Auslieferungszustand angefordert werden. Dazu muss der Eingang <deliveryState> 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 <saveSettings> auf TRUE gesetzt werden.
- Die Reihenfolge der Schreibzugriffe ist wie folgt:
<deliveryState>, Parameter 37, 38, 39, 3, 40, 41, 42...und <saveSettings>.
- Bei einem Schreibfehler eines Parameters werden die nachfolgenden Parameter nicht mehr geschrieben und es erfolgt auch kein Speichern der Werte, falls der Eingang <saveSettings> gesetzt ist.



INFORMATION!

Im Falle der S7-300 muss bei dem Baustein **MC_Parametrization_PSD4xx** die Variable vom Datentyp < driveData_PSD4xx > an den Eingang <driveIn> und an den Ausgang <driveOut> zugewiesen werden (bei den S7-1200 und S7-1500 im TIA-Portal ist die Variable nur einmal an den Ein- und Ausgang <Drive> zugewiesen).

5.8 MC_PosParametrization_PSD4xx (FB115)

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

```

80 //MC_PosParametrization_PSD4xx_DB --> instance of "MC_PosParametrization_PSD4xx"
81 MC_PosParametrization_PSD4xx_DB"(execute := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.execute,|
82     direction := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.direction,
83     stepsPerTurn := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.stepsPerTurn,
84     lowerLimit := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.lowerLimit,
85     upperLimit := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.upperLimit,
86     setPoint := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.setPoint,
87     saveSettings := "Data_Store_PSD4xx".PSD_PosParametrization_1.input.saveSettings,
88     active => "Data_Store_PSD4xx".PSD_PosParametrization_1.output.active,
89     done => "Data_Store_PSD4xx".PSD_PosParametrization_1.output.done,
90     error => "Data_Store_PSD4xx".PSD_PosParametrization_1.output.error,
91     errorID => "Data_Store_PSD4xx".PSD_PosParametrization_1.output.errorID,
92     errorParameter => "Data_Store_PSD4xx".PSD_PosParametrization_1.output.errorParameter,
93     drive := "Data_Store_PSD4xx".PSD_1);

```

Abbildung 14: Zuweisung der Variablen von DB <Data_Store> zum FB
MC_PosParametrization_PSD4xx

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 Werte werden verarbeitet, danach werden die folgenden Parameter in der angegebenen Reihenfolge geschrieben:
 1. Drehsinn (Parameter 37) → <direction>
 2. Istwertbewertung Zähler (Parameter 38) → 400
 3. Istwertbewertung Nenner (Parameter 39) → <stepsPerTurn>
 4. Istwert (Parameter 3) → <setPoint>
 5. Falls (setPoint > upperLimit):
Oberes Mapping-Ende (Parameter 41) = <setPoint> + (3 x <stepsPerTurn>)
sonst:
Oberes Mapping-Ende (Parameter 41) = <upperLimit> + (3 x <stepsPerTurn>)
 6. Obere Endbegrenzung (Parameter 42) → <upperLimit>
 7. Untere Endbegrenzung (Parameter 43) → <lowerLimit>



INFORMATION

Im Falle der S7-300 muss bei dem Baustein **MC_PosParametrization_PSD4xx** die Variable vom Datentyp < driveData_PSD4xx > an den Eingang <driveIn> und an

den Ausgang <driveOut> zugewiesen werden (bei den S7-1200 und S7-1500 im TIA-Portal ist die Variable nur einmal an den Ein- und Ausgang <Drive> zugewiesen).

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

Bedingung	<errorID>	<errorParameter>
<stepsPerTurn> \geq 1	16#6140	39
<stepsPerTurn> \leq 10000	16#6140	39
<lowerLimit> \leq <upperLimit>	16#6140	42
(<upperLimit> – <lowerLimit>) / <stepsPerTurn> \leq 250	16#6140	43
Falls <setPoint> < <lowerLimit>: (<upperLimit> – <setPoint>) / <stepsPerTurn> \leq 250	16#6140	3
Falls <setPoint> > <upperLimit>: (<setPoint> – <lowerLimit>) / <stepsPerTurn> \leq 250	16#6140	3

- Wahlweise können die geschriebenen Werte am Ende auch gespeichert werden. Dazu muss der Eingang <saveSettings> 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 <saveSettings> gesetzt ist.

6 Parameterbeschreibung

6.1 <active>

Baustein	MC_Move_PSD4xx, MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx, MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx		
FBs ist aktiv oder Fahrauftrag bzw. Fahrt ist aktiv			
Datentyp	Bool		
Default	-		
Art	Output		
Beschreibung			
<u>MC_Move_PSD4xx</u>			
Dieser Ausgang wird gesetzt, wenn:			
<ul style="list-style-type: none">die Freigabe <execute> von FALSE auf TRUE gesetzt wirddie Freigabe <execute> schon vorhanden ist und sich die Zielposition ändert, das Bit „Antrieb läuft“ im Status des Antriebs gesetzt ist (z.B. beim Nachregeln des Antriebs)			
Dieser Ausgang wird zurückgesetzt, wenn:			
<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 6.25 <<timeBusCycle>) abgelaufen istein Kommunikationsfehler auftritt			
<u>Alle anderen Bausteine</u>			
Das Bit wird gesetzt, solange der jeweilige Funktionsbaustein läuft. Sobald der Vorgang abgeschlossen wurde oder ein Fehler aufgetreten ist, wird das Bit zurückgesetzt.			

6.2 <actualPosition>

Baustein	MC_Move_PSD4xx	
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 <communicationError>

Baustein	MC_Communication_PSD4xx...
Kommunikationsfehler zu dem IO-Device (Antrieb)	
Datentyp	Bool
Art	Input
Beschreibung	
<ul style="list-style-type: none"> Die Diagnose der Baugruppen-Zustände der IO-Devices wird in der Regel zentral im SPS-Programm verwaltet, z.B. im OB86 oder mit der Anweisung „DeviceStates“. Beim Ausfall des entsprechenden IO-Devices (d.h. hier des Antriebs) ist dieser Eingang auf TRUE zu setzen. Solange die Kommunikation nicht beeinträchtigt ist, ist der Eingang mit FALSE zu belegen. Bei Kommunikationsfehlern werden mit Hilfe von MC_Move_PSD4xx gestartete Fahraufträge abgebrochen. Ebenso werden laufende Parameterzugriffe (MC_ReadParameter_PSD4xx und MC_WriteParameter_PSD4xx) abgebrochen. 	

6.4 <deliveryState>

Baustein	MC_Parametrization_PSD4xx
Laden der Werkseinstellungen (zunächst ohne Speichern)	
Datentyp	Bool
Default	FALSE
Art	Input
Beschreibung	
Der Stationsname und die IP-Adresse bleiben jedoch unbeeinflusst.	

6.5 <direction>

Baustein	MC_PosParametrization_PSD4xx
Drehrichtung der Abtriebswelle	
Datentyp	Int
Default	0
Art	Input
Beschreibung	
<p>Dieser Parameter entspricht dem Parameter Nummer 37 „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>	

6.6 <done>

Baustein	MC_Move_PSD4xx, MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx, MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx	
Zielposition erreicht oder FBs hat Vorgang abgeschlossen		
Datentyp	Bool	
Default	-	
Art	Output	
Beschreibung		
<u>MC Move PSD4xx</u> Dieser Ausgang ist eine Abbildung des Statusbits „Sollposition erreicht“. Zusätzlich wird der Ausgang für die Dauer des Parameters „Buszyklus“ (siehe 6.25 <timeBusCycle>) nach dem Start durch <execute> auf 0 gesetzt. Falls ein Kommunikationsfehler auftritt, wird er zurückgesetzt.		
<u>Alle anderen Bausteine</u> Das Bit wird gesetzt, sobald der Vorgang erfolgreich abgeschlossen wurde. Es wird beim Start des jeweiligen Vorgangs zurückgesetzt.		

6.7 <drive>

Baustein	MC_Communication_PSD4xx..., MC_Error_PSD4xx, MC_Move_PSD4xx, MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx, MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx	
Datenstruktur zur Kommunikation		
Datentyp	driveData_PSD4xx	
Art	Input & Output	
Beschreibung		
Für jeden Antrieb wird eine globale Instanz dieser Struktur benötigt. Diese Instanz wird jedem Funktionsbaustein (FB) übergeben, der auf den betriebenen Antrieb wirkt.		

6.8 <error>

Baustein	MC_Communication_PSD4xx_..., MC_Error_PSD4xx, MC_Move_PSD4xx, MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx, MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx	
Fehler bei der Ausführung des FBs oder Fehler im Antrieb		
Datentyp	Bool	
Default	FALSE	
Art	Output	
Beschreibung		
<u>MC Move PSD4xx</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>MC Error PSD4xx</u>		
Der Ausgang <error> wird jeden Zyklus aktualisiert, solange <execute> gesetzt ist. Wird <execute> 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.		

6.9 <errorDescription>

Baustein	MC_Error_PSD4xx		
Fehlerbeschreibung als Textausgabe			
Datentyp	String		
Default	"		
Art	Output		
Beschreibung			
Eine Fehlerbeschreibung als Textausgabe			

6.10 <errorID>

Baustein	MC_Communication_PSD4xx..., MC_Error_PSD4xx, MC_Move_PSD4xx, MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx, MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx
Fehlererkennung (siehe Kapitel 4 Fehlerbeschreibung (<errorID>))	
Datentyp	Word
Default	0
Art	Output
Beschreibung	
Die Fehlererkennung kann auch gesetzt sein, während der Antrieb fährt (z.B. Schleppfehler)	
<u>MC Move PSD4xx</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>MC Error PSD4xx</u>	
Der Ausgang <errorID> wird jeden Zyklus aktualisiert, solange <execute> gesetzt ist. Wird <execute> 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.	



ACHTUNG!

Die Ausgänge <error> und <errorID> der Funktionsbausteine werden stets aktualisiert – auch dann, wenn der Eingang <execute> nicht gesetzt ist, außer bei **MC_Error_PSD4xx**.

6.11 <errorParameter>

Baustein	MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx
Parameternummer, bei dem im Fall eines Fehlers der Fehler aufgetreten ist	
Datentyp	Int
Default	0
Art	Output
Beschreibung	
Falls es keinen Fehler gab, wird der Wert 0 ausgegeben.	

6.12 <execute>

Baustein	MC_Error_PSD4xx, MC_Move_PSD4xx, MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx, MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx	
Freigabe des Antriebs		
Datentyp	Bool	
Default	FALSE	
Art	Input	
Beschreibung		
<u>MC Move PSD4xx</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.		

6.13 <inputAddress>

Baustein	MC_Communication_PSD4xx_...	
Adresse des Eingangsmoduls des Antriebs		
Datentyp bei S7-1200 S7-1500	Int	
Datentyp bei S7-300	DWord	
Art	Input	
Beschreibung		
Eine genaue Beschreibung der Adressvergabe finden Sie im Kapitel 2.1 <driveData_PSD4xx>		

6.14 <lowerLimit>

Baustein	MC_PosParametrization_PSD4xx	
Untere Endbegrenzung		
Datentyp	DInt	
Default	0	
Art	Input	
Beschreibung		
Dieser Parameter entspricht dem Parameter 43 „untere Endbegrenzung“.		

6.15 <manuelRunToLargerValues>

Baustein	MC_Move_PSD4xx
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 <execute> muss zusätzlich gesetzt werden.	



ACHTUNG!

Beim Zurücksetzen des Eingangs <manuelRunToLargerValues> muss auch <execute> zurückgesetzt werden, da der Antrieb ansonsten die Zielposition <targetPosition> anfährt.

6.16 <manualRunToSmallerValues>

Baustein	MC_Move_PSD4xx
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 <execute> muss zusätzlich gesetzt werden.	

6.17 <outputAddress>

Baustein	MC_Communication_PSD4xx_...
Adresse des Ausgangsmoduls des Antriebs	
Datentyp bei S7-1200 S7-1500	Int
Datentyp bei S7-300	DWord
Art	Input
Beschreibung	
Eine genaue Beschreibung der Adressvergabe finden Sie im Kapitel 2.1 <driveData_PSD4xx>.	

6.18 <parameter>

Baustein	MC_Parametrization_PSD4xx
Übergabe des Parametersatzes mit dem Datentyp <parameter_PSD4xx> (siehe Kapitel 2.4 <parameter_PSD4xx>)	
Datentyp	parameter_PSD4xx
Default	-
Art	Input
Beschreibung	
Die Parameternummer ist hinter dem Parameternamen angegeben. Der Datentyp, eine Beschreibung sowie der Wertebereich kann der Betriebsanleitung des PSD4xx PN entnommen werden.	

6.19 <parameterNumber>

Baustein	MC_WriteParameter_PSD4xx, MC_ReadParameter_PSD4xx
Parameternummer des zu lesenden bzw. schreibenden Parameters	
Datentyp	Int
Default	0
Art	Input
Beschreibung	
Bei einer steigenden Flanke wird ein Lese- bzw. Schreibvorgang gestartet. Für einen neuen Vorgang muss erneut eine steigende Flanke generiert werden. Wird das Bit zurückgesetzt, so nehmen die Ausgänge die angegebenen Anfangswerte an.	

6.20 <saveSettings>

Baustein	MC_Parametrization_PSD4xx, MC_PosParametrization_PSD4xx
Speichern der Parametereinstellungen	
Datentyp	Bool
Default	FALSE
Art	Input
Beschreibung	
Dieser Parameter entspricht dem Parameter 96 „Auslieferungszustand“. (Funktion <Schreiben einer „1“>)	

6.21 <setPoint>

Baustein	MC_PosParametrization_PSD4xx
Istposition des Messsystems	
Datentyp	DInt
Default	0
Art	Input
Beschreibung	
Dieser Parameter entspricht dem Parameter 3 „Istwert“.	

6.22 <stepsPerTurn>

Baustein	MC_PosParametrization_PSD4xx
Schritte pro Umdrehung an der Abtriebswelle (Auflösung)	
Datentyp	Int
Default	0
Art	Input
Beschreibung	
Die Anzahl der Schritte pro Umdrehung <stepsPerTurn> ergibt unmittelbar den Wert des Parameters „Istwertbewertung Nenner“ (Parameter 39). Dabei wird angenommen, dass der Wert von „Istwertbewertung Zähler“ (Parameter 38) im Auslieferungszustand (400) ist.	

6.23 <subIndex>

Baustein	MC_ReadParameter_PSD4xx
Subindex des Parameters	
Datentyp	Int
Default	0
Art	Input
Beschreibung	
Der Wert des Parameters <subIndex> kann auf den Defaultwert belassen werden. Andere Wertezuweisungen sind aktuell nicht vorgesehen.	

6.24 <targetPosition>

Baustein	MC_Move_PSD4xx
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 <execute> noch gesetzt und wird die Zielposition geändert, so fährt der Antrieb diesen sofort an.</p>	



INFORMATION!

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

6.25 <timeBusCycle>

Baustein	MC_Move_PSD4xx
Aktualisierungszeit des Profinet Buszyklus	
Datentyp	Time
Default	40ms
Art	Input
Beschreibung	
<p>Bei langen Zykluszeiten auf dem Profinet-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 <timeBusCycle> eingeführt. Als Anfangswert werden 40 ms für einen Buszyklus angesetzt. Für diese Dauer wird nach einem Fahrauftrag der Ausgang <active> auf jeden Fall gesetzt und der Ausgang <done> 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 <timeBusCycle> auch verringert werden.</p>	

6.26 <upperLimit>

Baustein	MC_PosParametrization_PSD4xx
Obere Endbegrenzung	
Datentyp	DInt
Default	0
Art	Input
Beschreibung	
Dieser Parameter entspricht dem Parameter 42 „obere Endbegrenzung“.	

6.27 <value>

Baustein	MC_ReadParameter_PSD4xx, MC_WriteParameter_PSD4xx
zu schreibender Wert oder lesenden Wert eines Parameters	
Datentyp	DInt
Default	0
Art	Input für MC_WriteParameter_PSD4xx Output für MC_ReadParameter_PSD4xx
Beschreibung	
Bei einer steigenden Flanke wird ein Lese- bzw. Schreibvorgang gestartet. Für einen neuen Vorgang muss erneut eine steigende Flanke generiert werden.	

7 Anwendung der Funktionsbausteine

Die Funktionsbausteine sind in TIA V13 programmiert. Ein Upgrade auf höhere TIA Versionen kann problemlos durchgeführt werden.

Dabei gibt es zwei Möglichkeiten die Funktionsbausteine zu nutzen.

- Projekt
- Bibliothek

7.1 Projekt

Öffnen und upgraden des Projekts abhängig von der CPU:

- example_hw_PSD4xx_1200_1500_xxxxxxx.zap13
→ für SIMATIC S7 - 1200 und SIMATIC S7 – 1500
- example_hw_PSD4xx_300_xxxxxxx.zap13
→ für SIMATIC S7 - 300

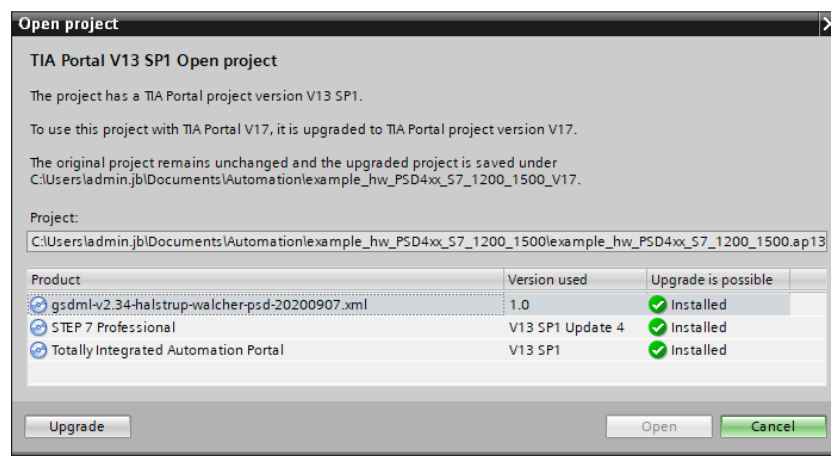


Abbildung 15: Upgrade eines Projekts in TIA V13 auf TIA V17

7.2 Bibliothek

Öffnen und upgraden der Bibliothek abhängig von der CPU:

- library_PSD4xx_1200_1500_xxxxxxx.zal13
→ für SIMATIC S7 - 1200 und SIMATIC S7 – 1500
- library_PSD4xx_300_xxxxxxx.zal13
→ für SIMATIC S7 – 300

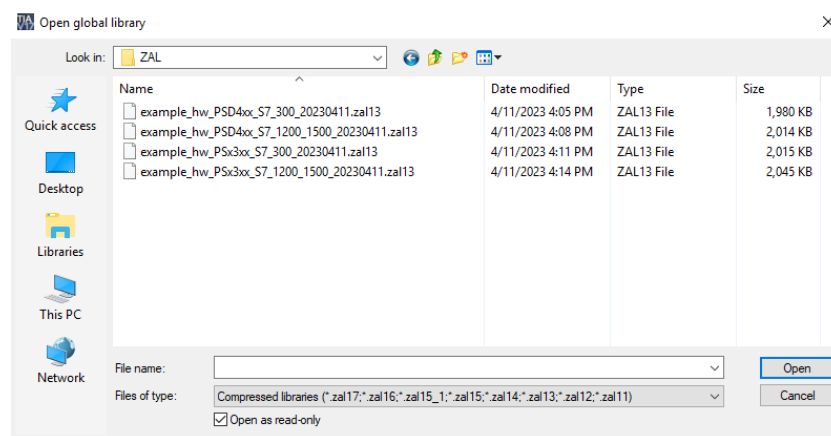


Abbildung 16: Öffnen einer komprimierten Bibliothek

Nach dem Öffnen und dem Upgrade der Bibliothek präsentieren sich die Funktionsbausteine folgendermaßen:

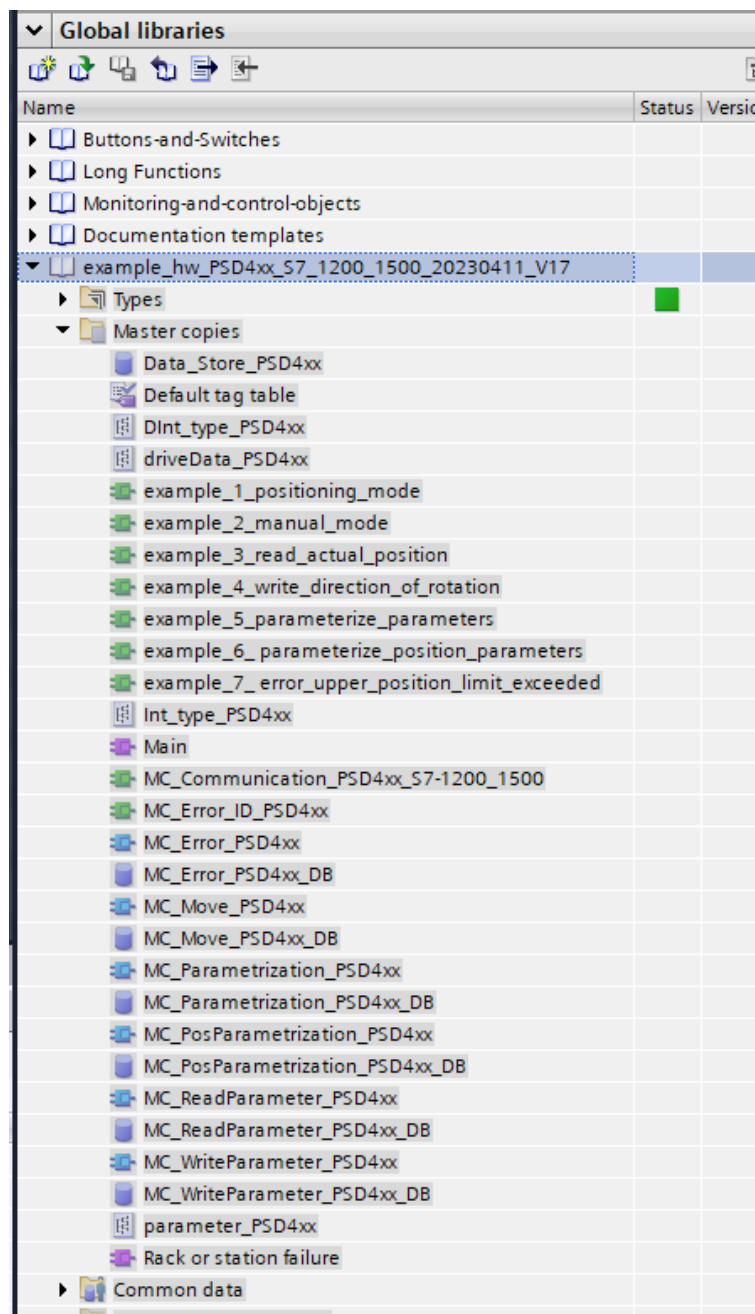


Abbildung 17: Globale Bibliothek in TIA V17

Folgende Funktionen aus der Bibliothek sind (unabhängig von den tatsächlich verwendeten Funktionsbausteinen **MC_..._PSD4xx** und der Anzahl der Antriebe) in jedem Fall in das Projekt des Anwenders zu kopieren:

- für SIMATIC S7 - 1200 und S7 - 1500: **MC_Communication_PSD4xx_S7-1200_1500**
für SIMATIC S7 - 300: **MC_Communication_PSD4xx_S7-300**
- **MC_Error_ID_PSD4xx**

Diese Funktionen dienen der Kommunikation und der Fehlererkennung mit dem Antrieb.

Daneben sind die gewünschten Funktionsbausteine **MC_..._PSD4xx** in das Anwenderprojekt zu kopieren. Es können alle oder eine Auswahl der nachfolgenden Bausteine verwendet werden:

- **MC_Move_PSD4xx**
- **MC_Error_PSD4xx**
- **MC_ReadParameter_PSD4xx**
- **MC_WriteParameter_PSD4xx**
- **MC_Parametrization_PSD4xx**
- **MC_PosParametrization_PSD4xx**
- Datentyp <Int_type_PSD4xx> (für **MC_Parametrization_PSD4xx**)
- Datentyp <DInt_type_PSD4xx> (für **MC_Parametrization_PSD4xx**)
- Datentyp <parameter_PSD4xx> (für **MC_Parametrization_PSD4xx**)

Zusätzlich kann der Datenbaustein <Data_Store> als Vorlage für die Anbindung an die Funktionsbausteine übernommen werden. In dessen Standardausführung müssen dann auch die Datentypen <Int_type_PSD4xx>, <DInt_type_PSD4xx> und <parameter_PSD4xx> (für **MC_Parametrization_PSD4xx**) 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 <execute> von **MC_Move_PSD4xx** gesetzt ist, können die Bausteine **MC_Parametrization_PSD4xx** und **MC_PosParametrization_PSD4xx** nicht aktiviert werden (<errorID>: 16#x100).
- Dagegen ist es möglich, während einer Bewegung **MC_ReadParameter_PSD4xx** oder **MC_WriteParameter_PSD4xx** aufzurufen (z.B. um das aktuelle Drehmoment auszulesen oder während der Fahrt die Solldrehzahl zu ändern).
- Wenn der Eingang <execute> von **MC_Parametrization_PSD4xx** oder **MC_PosParametrization_PSD4xx** gesetzt ist, kann der Baustein **MC_Move_PSD4xx** nicht aktiviert werden (<errorID>: 16#0100).
- Es kann stets nur einer der Bausteine **MC_ReadParameter_PSD4xx**, **MC_WriteParameter_PSD4xx**, **MC_Parametrization_PSD4xx** oder **MC_PosParametrization_PSD4xx** aktiv sein. Wenn der Eingang <execute> einer dieser Bausteine gesetzt ist und der Eingang <execute> eines weiteren Bausteins gesetzt wird, gibt dieser den Fehler 16#x100 aus.
- Der Baustein **MC_Error_PSD4xx** kann unabhängig von den anderen Bausteinen stets aktiviert sein.

7.4 Mehrere PSD4xx in einem Projekt

In den Beispielprojekten befindet sich nur ein PSD4xx. Wenn mehrere Antriebe genutzt werden, muss das Projekt entsprechend angepasst werden.

Beispielweise für drei PSD4xx:

1. Instanzdatenbausteine initialisieren
 - drei Instanzdatenbausteine vom Typ FB110 (**MC_Move_PSD4xx**), z.B.
 - DB1101, symbolischer Name = MC_Move_PSD4xx_DB
 - DB1102, symbolischer Name = MC_Move_PSD4xx_DB
 - DB1103, symbolischer Name = MC_Move_PSD4xx_DB
 - drei Instanzdatenbausteine vom Typ FB111 (**MC_Error_PSD4xx**), z.B.
 - DB1111, symbolischer Name = MC_Error_PSD4xx_DB
 - DB1112, symbolischer Name = MC_Error_PSD4xx_DB
 - DB1113, symbolischer Name = MC_Error_PSD4xx_DB
 - weitere benötigte Instanzdatenbausteine

2. Anschließend wird der Datenbaustein vom Typ „Global“ mit Namen DB100 (<Data_Store_PSD4xx>) angepasst. Darin legen wir drei Variablen vom Typ <driveData_PSD4xx> an, z.B. mit den folgenden Bezeichnungen:
 - „PSE_1“
 - „PSE_2“
 - „PSE_3“
3. Zusätzlich müssen die Variablen im <Data_Store_PSD4xx> zur Steuerung der einzelnen Bausteine angelegt werden:
 - drei Variablen für den Typ FB110 (**MC_Move_PSD4xx**), z.B.
 - „PSE_Move_1“
 - „PSE_Move_2“
 - „PSE_Move_3“
 - drei Variablen für den Typ FB111 (**MC_Error_PSD4xx**), z.B.
 - „PSE_Error_1“
 - „PSE_Error_2“
 - „PSE_Error_3“
 - Weitere Variablen für die Steuerung
4. Zum Schluss müssen die Variablen von DB <Data_Store_PSD4xx> den Funktionsbausteinen zugewiesen werden. (siehe Kapitel 0)

8 Beispielprogramme

In diesem Verzeichnis sind Beispielprogramme zu finden, um zu veranschaulichen, wie die Funktionsbausteine einzusetzen sind. Die Funktionen sind in SCL programmiert und die jeweilige Funktion muss in das Hauptprogramm eingefügt werden, um die Funktion aufrufen zu können.

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
(* MAIN *)

//call of function
"example_1_positioning_mode"();

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

Abbildung 18: Aufrufende Funktion in dem Hauptprogramm

Gemeinsamkeiten aller Beispielprogramme

- Bei jedem Beispielprogramm wird die Funktion **MC_Communication_PSD4xx_S7-1200_1500** am Anfang zyklisch aufgerufen.
- Bei allen Funktionen werden am Anfang die benötigten Instanzen definiert und aufgerufen.
- Bei allen Funktionen muss die Variable <startProgram> 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 <stopProgram> kann das Programm beendet werden.

8.1 Beispiel 1: Positioniermodus

In diesem Beispielprogramm wird der Positioniermodus des FBs **MC_Move_PSD4xx** vorgestellt. Der Antrieb fährt in diesem Beispiel eine Endlosschleife zwischen den Werten 60000 und 50000.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und der Index wird um eins hochgesetzt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Zielposition von 60000 und der Fahrbefehl werden gesetzt. Wenn die aktuelle Position bei 60000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Die Zielposition von 50000 und der Fahrbefehl werden gesetzt. Wenn die aktuelle Position bei 50000 ± 2 (siehe Parameter 44 „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index auf 1 gesetzt.

8.2 Beispiel 2: Handfahrmodus

In diesem Beispielprogramm wird der Handmodus des FBs **MC_Move_PSD4xx** 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 der Index wird um eins hochgesetzt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 55000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 55000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Mit Handmodus wird das Ziel von 60000 angefahren. Erst wenn die aktuelle Position 60000 entspricht, wird der Fahrbefehl für die Handfahrt zurückgesetzt und der Index auf 100 gesetzt.

8.3 Beispiel 3: Aktuelle Position lesen

In diesem Beispielprogramm wird der FB **MC_ReadParameter_PSD4xx** 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 der Index wird um eins hochgesetzt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 55000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 55000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Der Parameter „aktuelle Position“ wird gelesen. Wenn der Wert bei 55000 ± 2 liegt (siehe Parameter 44 „Positionierfenster“), dann wird der Lesebefehl zurückgesetzt und der Index auf 100 gesetzt.

8.4 Beispiel 4: Drehsinn schreiben

In diesem Beispielpogramm wird der FB **MC_WriteParameter_PSD4xx** vorgestellt. Der Parameter „Drehsinn“ wird in diesem Beispiel geschrieben und gelesen und es werden Positionierfahrten gemacht.

Programmablauf

Case	Beschreibung
0	Alle relevanten Variablen werden für die Initialisierung auf FALSE gesetzt und der Index wird um eins hochgesetzt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 60000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 60000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Der Wert des Parameters „Drehsinns“ wird auf 1 geschrieben. Wenn der Schreibbefehl erfolgreich durchgeführt wurde, dann wird der Schreibbefehl zurückgesetzt und der Index um eins hochgesetzt.
3	Der Wert des Parameters „Drehsinns“ wird gelesen. Es wird überprüft, ob der Wert des Parameters „Drehsinns“ auf 1 ist. Wenn das der Fall ist und der Lesevorgang erfolgreich durchgeführt wurde, dann wird der Lesebefehl zurückgesetzt und der Index um eins hochgesetzt.
4	Durch den Drehsinnwechsel liegt die aktuelle Position bei 42400 ($102400 - 60000 = 42400$). Die Zielposition von 60000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 60000 ± 2 (siehe Parameter 44 „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
5	Der Wert des Parameters „Drehsinns“ wird auf 0 geschrieben. Wenn der Schreibbefehl erfolgreich durchgeführt wurde, dann wird der Schreibbefehl zurückgesetzt und der Index um eins hochgesetzt.
6	Der Wert des Parameters „Drehsinns“ wird gelesen. Es wird überprüft, ob der Wert des Parameters „Drehsinns“ auf 0 ist. Wenn das der Fall ist und der Lesevorgang erfolgreich durchgeführt wurde, dann wird der Lesebefehl zurückgesetzt und der Index um eins hochgesetzt.
7	Durch den Drehsinnwechsel liegt die aktuelle Position bei 42400 ($102400 - 60000 = 42400$). Die Zielposition von 60000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 60000 ± 2 (siehe Parameter 44 „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index auf 100 gesetzt.

8.5 Beispiel 5: Parameter parametrisieren

In diesem Beispielpogramm wird der FB **MC_Parametrization_PSD4xx** 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 der Index wird um eins hochgesetzt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 60000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 60000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Die Parameter „Schleifenlänge“ und „Solldrehzahl“ werden parametrisiert. Dabei müssen die <enable> der Parameter gesetzt sein. Wenn der Parametrisierbefehl erfolgreich durchgeführt wurde, dann wird der Parametrisierbefehl zurückgesetzt und der Index um eins hochgesetzt.
3	Die Zielposition von 50000 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 50000 ± 2 (siehe Parameter 44 „Positionierfenster“) ist und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index auf 100 gesetzt.

8.6 Beispiel 6: Positionsdaten parametrisieren

In diesem Beispielpogramm wird der FB **MC_PosParametrization_PSD4xx** 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 der Index wird um eins hochgesetzt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 50000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 50000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Alle Parameter, die die Position beeinflussen 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 der Index um eins hochgesetzt.
3	Der Parameter „aktuelle Position“ wird gelesen. Wenn der Wert bei 0 liegt (in diesem Fall muss die aktuelle Position genau 0 betragen), dann wird der Lesebefehl zurückgesetzt und der Index auf 100 gesetzt.

8.7 Beispiel 7: Fehler obere Endbegrenzung erreicht

In diesem Beispielprogramm wird der FB **MC_Error_PSD4xx** 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 der Index um eins hochgezählt. (Es ist immer sinnvoll, am Anfang eines Programms alle relevanten Variablen auf den Anfangswert zu setzen.)
1	Die Startposition von 65000 und der Fahrbefehl werden gesetzt. Wenn der aktuelle Wert bei 65000 ± 2 ist (siehe Parameter 44 „Positionierfenster“) und die Fahrt erfolgreich durchgeführt wurde, dann wird der Fahrbefehl zurückgesetzt und der Index um eins hochgesetzt.
2	Der Parameter 42 obere Endbegrenzung wird auf den Wert 70000 geschrieben. Wenn der Schreibbefehl erfolgreich durchgeführt wurde, dann wird der Schreibbefehl zurückgesetzt und der Index um eins hochgesetzt.
3	Der Funktionsbaustein MC_Error_PSD4xx wird aktiviert, um Fehler zu erkennen.
4	Mit Handmodus wird die obere Endbegrenzung angefahren. Erst wenn die aktuelle Position 70000 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 Ausgang <errorID> von dem FB MC_Move_PSD4xx zu erhalten. Die <errorID> wäre dann 0x100B.)

Abbildungsverzeichnis

Abbildung 1: Adressen der Prozessdaten aus Sicht der PLC.....	6
Abbildung 2: Einzustellende Adressen aus Sicht des Geräts.....	7
Abbildung 3: Datentyp <Int_type_PSD4xx>.....	7
Abbildung 4: Datentyp <DInt_type_PSD4xx>.....	7
Abbildung 5: Ausschnitt aus Datentyp <parameter_PSD4xx>.....	8
Abbildung 6: Zuweisung der Variablen von DB <Data_Store> zum FB MC_Move_PSD4xx	9
Abbildung 7: DB <Data_Store> mit den Variablen für einen Antrieb.....	9
Abbildung 8: Zuweisung der Variablen von DB <Data_Store> zu der FC MC_Communication_PSD4xx_1200_1500	12
Abbildung 9: Zuweisung der Variablen von DB <Data_Store> zum FB MC_Move_PSD4xx	12
Abbildung 10: Zuweisung der Variablen von DB <Data_Store> zum FB MC_Error_PSD4xx	13
Abbildung 11: Zuweisung der Variablen von DB <Data_Store> zum FB MC_ReadParameter_PSD4xx	14
Abbildung 12: Zuweisung der Variablen von DB <Data_Store> zum FB MC_WriteParameter_PSD4xx	14
Abbildung 13: Zuweisung der Variablen von DB <Data_Store> zum FB MC_Parametrization_PSD4xx	14
Abbildung 14: Zuweisung der Variablen von DB <Data_Store> zum FB MC_PosParametrization_PSD4xx	15
Abbildung 15: Upgrade eines Projekts in TIA V13 auf TIA V17.....	28
Abbildung 16: Öffnen einer komprimierten Bibliothek.....	28
Abbildung 17: Globale Bibliothek in TIA V17.....	29
Abbildung 18: Aufrufende Funktion in dem Hauptprogramm.....	32