

Handbuch_PtP_Package_Festo

FESTO

**Codesys, TwinCAT
und Sysmac Studio
Bibliothek für Festo
Motorcontroller im
Punkt-zu-Punkt
Betrieb**

Datum
22.01.2024

Festo SE & Co. KG
Ruiter Straße 82
73734 Esslingen

www.festo.com/contact

Inhaltsverzeichnis

1	Wichtige Hinweise.....	4
1.1	Versionsübersicht	4
1.2	Urheberrechtshinweis	5
1.3	Rechtliche Hinweise	5
1.4	Bestimmungsgemäße Verwendung	5
1.5	Sicherheitshinweise	5
1.6	Zielgruppe	6
1.7	Service	6
2	Einführung	7
2.1	Installation des Bibliothek-Package	7
2.1.1	Codesys extension in Automation Suite	7
2.1.2	Codesys 3.5 (stand-alone)	7
2.1.3	Beckhoff TwinCAT 3	8
2.1.4	Omron Sysmac Studio	10
2.1.5	Lenze PLC Designer	12
2.2	Sachgemäße Verwendung der azyklischen Kommunikation (SDO-Zugriff)	13
3	PLCopen Funktionsbausteine	14
3.1	Allgemein	14
3.2	PLCopen Diagramm	15
3.3	MC_Power_Festo	16
3.4	MC_Home_Festo	17
3.5	MC_Stop_Festo	18
3.6	MC_Halt_Festo	19
3.7	MC_MoveAbsolute_Festo	20
3.8	MC_MoveRelative_Festo	21
3.9	MC_MoveAdditive_Festo	22
3.10	MC_MoveVelocity_Festo	23
3.11	MC_TorqueControl_Festo	24
3.12	MC_ReadParameter_Festo	25
3.13	MC_WriteParameter_Festo	26
3.14	MC_ReadStringParameter_Festo	27
3.15	MC_WriteStringParameter_Festo	28
3.16	MC_ReadActualPosition_Festo	29
3.17	MC_ReadActualVelocity_Festo	30
3.18	MC_ReadActualTorque_Festo	31
3.19	MC_ReadStatus_Festo	32
3.20	MC_ReadAxisError_Festo	33
3.21	MC_Reset_Festo	34
3.22	MC_Jog_Festo	35
3.23	MC_RecordTable_Festo	36
3.24	MC_ReadAxisInfo_Festo	37
3.25	MC_DeviceService_Festo	38
3.26	MC_ForceControl_Festo	39
4	Steuerung via Methodenaufruf	40
5	Diagnose.....	42
6	BufferMode.....	47

7	Direction	48
8	Visualisierung der Funktionsblöcke.....	49
9	Anwendungsbeispiele	50
9.1	Beispiel Funktionsbausteine	50
9.2	Beispiel Methodenaufruf	51

1 Wichtige Hinweise

1.1 Versionsübersicht

Bearbeiter	Datum	Kommentar
chmm	29.05.2018	Handbuch erstellt
chmm	23.10.2018	Handbuch erweitert (MC_ReadAxisInfo_Festo, AxisIsMoving, AxisWarning)
chmm	28.11.2018	CMMT-ST hinzugefügt
chmm	17.12.2018	Eigenschaften und Fehlernummern erweitert
chmm	22.01.2019	Bausteine um Ausgang „Active“ und Eingang „ContinuousUpdate“ erweitert, Beispiele
chmm	28.01.2019	Neue Fehlernummer, TwinCAT Bild erneuert
chmm	29.04.2019	Beschreibung der Diagnose erweitert, Hinweis für „MC_Home_Festo“ erweitert
chmm	06.06.2019	Anpassung der Beschreibung für „MC_ReadStatus_Festo“ und der ErrorID 2069
chmm	02.07.2019	Korrektur Kapitel 1.5
chmm	09.09.2019	Fehlerkorrektur in Kapitel 4 „Steuerung via Methodenaufruf“
chmm	12.09.2019	Fehlernummer 0x800 hinzugefügt
chmm	17.09.2019	Datentyp der FB-Ausgänge „ErrorID“ in WORD geändert Nummernkreis der ErrorID angepasst (+1000)
chmm	30.09.2019	Bausteineingang „Direction“ hinzugefügt, Kapitel Diagnose erweitert
chmm	20.11.2019	Kapitel „Omron Sysmac Studio“ hinzugefügt
chmm	22.01.2020	Init Fehler erweitert, Bugfix in MC_Power_Festo, Neuer Eingang „WcState_In“ für TC3
chmm	13.02.2020	Init Fehler erweitert, Aktualisierung Abbildung 8
chmm	17.02.2020	Rechtsform der Firma angepasst (Deckblatt)
chmm	27.02.2020	UpdateTime zum Baustein „MC_ReadAxisInfo_Festo“ hinzugefügt
chmm	05.03.2020	Baustein „MC_DeviceService_Festo“ hinzugefügt
chmm	11.03.2020	Bezeichnung der Unterkapitel in Kapitel 1 verbessert
chmm	30.06.2020	Kapitel 2.2 hinzugefügt
chmm	02.09.2020	Hubgrenze hinzugefügt, Fehlerliste aktualisiert (Kapitel 5)
chmm	04.05.2021	Beschreibung von Kapitel 7 (Direction) erweitert, Kapitel „Urheberrechtshinweis“ und „Rechtliche Hinweise“ hinzugefügt, Beispiel für Methodenaufruf erweitert
chmm	28.06.2021	Kapitel „Lenze PLC Designer“ hinzugefügt
chmm	21.12.2022	Kapitel „MC_ForceControl_Festo“ hinzugefügt
chmm	14.02.2023	Bugfixes
chmm	16.11.2023	Beschreibung für „MC_ForceControl_Festo“, Kapitel 2.2 verlinkt bei Bausteinen
chmm	22.01.2024	Beschreibung für den Baustein „MC_DeviceService_Festo“ erweitert
fxlt	06.02.2026	Buffermode mcBlendingPrevious hinzugefügt und Beschreibung aktualisiert (Kapitel 6) Zusätzliche Fehlermeldung aktualisiert (Kapitel 5)

1.2 Urheberrechtshinweis

Diese Unterlagen sind geistiges Eigentum der Festo SE & Co. KG, der auch das ausschließliche Urheberrecht daran zusteht. Eine inhaltliche Änderung, die Vervielfältigung oder der Nachdruck dieser Unterlagen sowie deren Weitergabe an Dritte ist nur mit der ausdrücklichen Erlaubnis der Festo SE & Co. KG gestattet. Festo SE & Co. KG behält sich das Recht vor, dieses Dokument vollständig oder teilweise zu ändern. Alle Marken- und Produktnamen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Titelhälter.

1.3 Rechtliche Hinweise

Hardware, Software, Betriebssysteme und Treiber dürfen nur für die beschriebenen Einsatzfälle und nur in Verbindung mit den von Festo SE & Co. KG empfohlenen Komponenten verwendet werden. Festo SE & Co. KG lehnt jede Haftung für Schäden ab, die durch die Anwendung von allenfalls falschen bzw. unzureichenden Informationen oder aufgrund fehlender Informationen in diesen Unterlagen entstehen. Defekte, die durch unsachgemäße Behandlung von Geräten und Baugruppen entstehen, sind von der Gewährleistung ausgeschlossen. Sicherheitsrelevante Funktionen, im Sinne von Personen- und Maschinenschutz, dürfen mit Angaben und Informationen aus diesem Dokument nicht realisiert werden. Für Folgeschäden, die durch einen Ausfall oder eine Funktionsstörung entstehen, wird dann jede Haftung abgelehnt. Im Übrigen gelten die Regelungen bzgl. Haftung aus den Liefer-, Zahlungs- und Softwarenutzungsbedingungen der Festo SE & Co. KG, welche Sie unter www.festo.com finden, welche wir Ihnen aber auch auf Anforderung gerne zukommen lassen. Alle in diesem Dokument angegebenen Daten sind keine zugesicherten Eigenschaften, insbesondere nicht für Funktionalität, Zustand oder Qualität im rechtlichen Sinn. Die Informationen dieses Dokuments gelten nur als einfache Hinweise für die Umsetzung einer ganz bestimmten, hypothetischen Anwendung, keinesfalls als Ersatz für die Bedienungsanleitung der jeweiligen Hersteller sowie der Konstruktion und Prüfung jeweils eigenen Anwendung durch den Benutzer. Die jeweiligen Bedienungsanleitungen der Festo Produkte sind unter www.festo.com/sp zu finden. Der Benutzer dieses Dokuments (Funktion und Anwendung) muss selbst sicherstellen, dass jede Funktion die hier beschrieben ist, auch in seiner Applikation ordnungsgemäß funktioniert. Der Benutzer bleibt auch durch das Studium dieses Dokuments sowie der Nutzung der darin genannten Angaben weiterhin allein verantwortlich für die eigene Anwendung.

1.4 Bestimmungsgemäße Verwendung

Mit den PLCopen basierten Funktionsbausteinen oder den direkten Methodenaufrufen können die vielfältigen Funktionen der Motorcontroller von Festo komfortabel in das SPS-Programm eingebunden werden. Die Funktionsbausteine werden für jeden in das Anwenderprogramm eingebundenen Motorcontroller (jede Achse) mit einer separaten Instanz zyklisch aufgerufen. Die gleichzeitige Verwendung anderer Funktionsbausteine zur Steuerung desselben Gerätes ist unzulässig. Die beschriebene Bibliothek dient zur Steuerung und Parametrierung folgender Motorcontroller:

- Servoantriebsregler CMMT-AS
- Servoantriebsregler CMMT-ST

Die "Sicherheitstechnischen Hinweise" sowie der bestimmungsgemäße Gebrauch der jeweiligen Geräte, Baugruppen und Module ist zu beachten. Beim Anschluss handelsüblicher Zusatzkomponenten, wie Sensoren und Aktuatoren, sind die angegebenen Grenzwerte für Drücke, Temperaturen, elektrische Daten, Momente usw. einzuhalten.

1.5 Sicherheitshinweise

Bei der Inbetriebnahme und Programmierung von Positioniersystemen sind unbedingt die in den Beschreibungen und Bedienungsanleitungen zu den eingesetzten Komponenten gegebenen Sicherheitsvorschriften zu beachten. Der Anwender hat dafür Sorge zu tragen, dass sich niemand im Einflussbereich der angeschlossenen Aktuatoren aufhält. Der mögliche Gefahrenbereich muss durch geeignete Maßnahmen wie Absperrungen oder Warnhinweise gesichert werden.

1. 6 Zielgruppe

Diese Beschreibung wendet sich ausschließlich an ausgebildete Fachleute der Steuerungs- und Automatisierungstechnik, die Erfahrungen mit der Installation, Inbetriebnahme, Programmierung und Diagnose von Positioniersystemen und den relevanten Feldbussen besitzen.

1. 7 Service

Bei einem technischen Problem melden sie sich bitte bei ihrem lokalen Servicepartner von Festo oder nutzen sie das Kontaktformular auf folgender Internetseite.

www.festo.com/contact

2 Einführung

2.1 Installation des Bibliothek-Package

2.1.1 Codesys extension in Automation Suite

Eine separate Installation des Point-to-Point Package muss nicht erfolgen, da dieses bei der Installation des CMMT Plug-ins automatisch aktualisiert wird. Bei der Konfiguration muss lediglich ausgewählt werden welche Betriebsart und welche Version des Packages benutzt werden soll. Diese Einstellungen sind im CMMT Plug-in im Reiter „Parametrierung“ in der Kategorie „Feldbus“ unter dem Punkt „Betriebsart“ zu finden (siehe Abbildung 1).

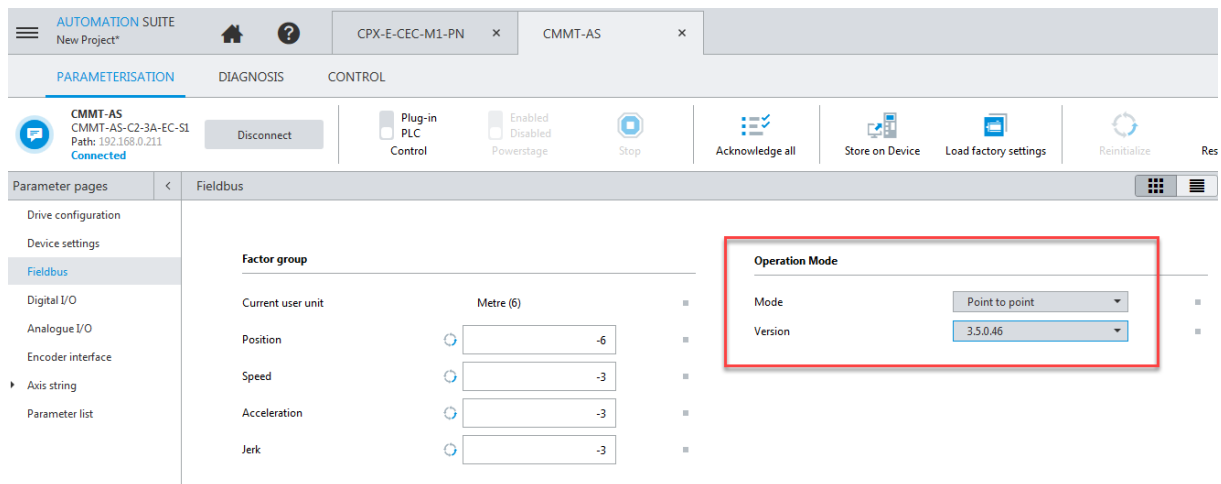


Abbildung 1: Automation Suite

2.1.2 Codesys 3.5 (stand-alone)

Wird Codesys 3.5 (stand-alone) verwendet, muss das Package durch den Codesys Package Manager installiert werden. Das Package ist im Support Portal der Festo Website zu finden. Mit einem Doppelklick auf die Package-Datei startet der Package Manager. Ist die Installation erfolgreich abgeschlossen muss zur Initialisierung der Daten ein Neustart von Codesys erfolgen. Im Anschluss kann der Motorcontroller von Festo an einen EtherCAT Master angehängt werden (siehe Abbildung 2). Der Motorcontroller ist anschließend sofort im Anwenderprogramm nutzbar. Eine separate Instanziierung oder Verknüpfung der Eingangs- und Ausgangsdaten ist nicht notwendig. Als Achsreferenz (AXIS_REF_FESTO) dient der Geräte name.

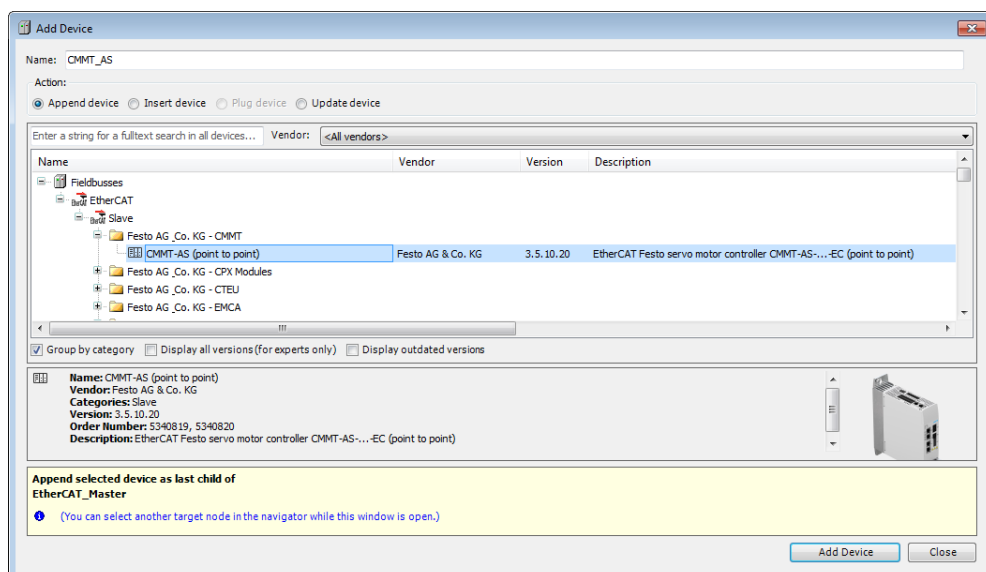


Abbildung 2: Codesys - Gerät anhängen

2. 1. 3 Beckhoff TwinCAT 3

Wird der CMMT-AS Motorcontroller von Festo in ein TwinCAT 3 Projekt von Beckhoff eingebunden, kann man das oben beschriebene Package nicht verwenden. In diesem Fall müssen die Gerätebeschreibungsdatei und die Bibliotheken manuell eingebunden werden. Die EtherCAT Slave Information (esi) ist in den hierfür vorgesehenen Installationspfad von TwinCAT zu kopieren (siehe Abbildung 3).

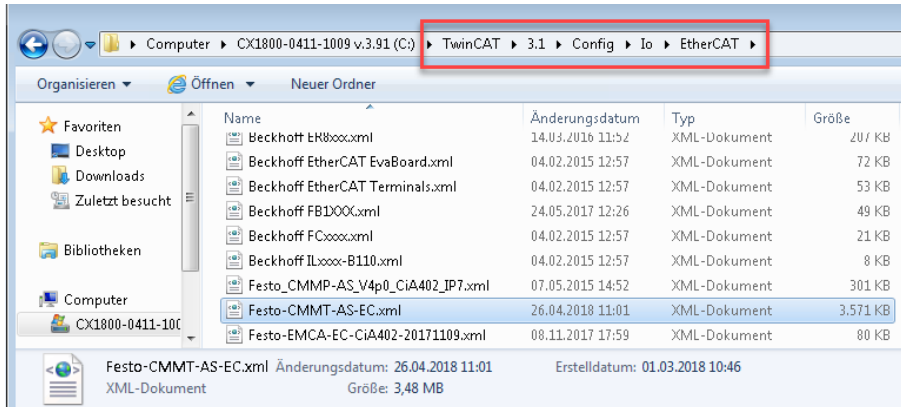


Abbildung 3: Esi Verzeichnis TwinCAT

Hinweis
Die Bibliotheken werden nicht von einer TwinCAT XAE Entwicklungsumgebung unterstützt die älter als Version 3.1.4020.0 ist (Grund: Compilerversion ist nicht kompatibel).

Wurde die Gerätebeschreibungsdatei eingefügt, kann der Programmierer anschließend TwinCAT starten. Danach müssen die Bibliotheken für den Antrieb im Bibliotheksrepository installiert werden. Wird beispielsweise der CMMT-AS Motorcontroller betrieben, muss der Anwender 5 Bibliotheken installieren (siehe Abbildung 4).

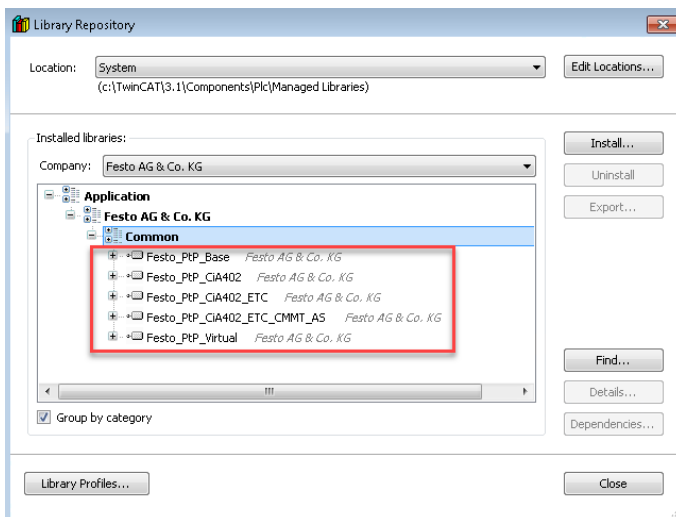


Abbildung 4: Bibliotheksrepository TwinCAT

In einem nächsten Schritt muss der Anwender die installierten Bibliotheken zum Projekt hinzufügen (siehe Abbildung 5 – lila Rahmen). Im Folgenden muss die Antriebsbibliothek instanziiert und zyklisch aufgerufen werden (Abbildung 5 – rote Rahmen). Nach der erfolgreichen Kompilierung des Anwenderprojekts können die Variablen der Bibliothek mit den Prozessdatenobjekten (PDOs) des Antriebs verknüpft werden. Hierzu muss eine Verlinkung der Bibliotheksvariablen mit den gleichnamigen Variablen des Motorcontrollers erfolgen (siehe Abbildung 5 orange und grüne Rahmen). Nun kann die angelegte Bausteininstanz als Achsreferenz im Anwenderprojekt verwendet. Zum einen kann diese als Axis-Variable vom Typ „AXIS_REF_FESTO“ für die unten beschriebenen Bausteine dienen. Zum anderen können Methoden direkt aus dieser Instanz heraus gestartet werden. Eine genauere Beschreibung zum Methodenaufruf ist in Kapitel 4 zu finden.



Hinweis

Die Instanzvariablen „WcState_In“, „SlaveState_In“ und „AdsAddr_In“ müssen zwingend mit dem Antrieb verknüpft werden. Nur durch diese Verlinkungen ist die SDO-Kommunikation (Service-Daten-Objekt Kommunikation) zwischen PLC und CMMT-AS bzw. CMMT-ST möglich.

The screenshot displays the Microsoft Visual Studio (Administrator) interface for the TwinCAT project 'Festo_PtP_Example_CMMT-AS_TC3'. The Solution Explorer on the left shows the project structure, with several components highlighted by colored boxes:

- Purple box:** References folder containing Festo_PtP_Base, Festo_PtP_CiA402, Festo_PtP_CiA402_ETC, Festo_PtP_CiA402_ETC_CMMT_AS, and Festo_PtP_Virtual.
- Orange box:** MAIN Instance folder containing PlcTask Inputs, MAIN.CMMT_AS.WcState_In, MAIN.CMMT_AS.SlaveState_In, and MAIN.CMMT_AS.AdsAddr_In.
- Green box:** MAIN Instance folder containing PlcTask Outputs, MAIN.CMMT_AS.Statusword_In, MAIN.CMMT_AS.ModesOfOperationDisplay_In, MAIN.CMMT_AS.PositionActualValue_In, MAIN.CMMT_AS.VelocityActualValue_In, MAIN.CMMT_AS.TorqueActualValue_In, MAIN.CMMT_AS.Controlword_Out, MAIN.CMMT_AS.ModesOfOperation_Out, MAIN.CMMT_AS.TargetPosition_Out, MAIN.CMMT_AS.ProfileVelocity_Out, MAIN.CMMT_AS.TargetVelocity_Out, and MAIN.CMMT_AS.TargetTorque_Out.
- Orange box:** I/O folder containing Device 1 (EtherCAT), which includes Image, Image-Info, SyncUnits, Inputs, Outputs, InfoData, Term 1 (EK1200), Drive 3 (CMMT-AS), Statusword, Modes of operation display, Position actual value, Velocity actual value, Torque actual value, Outputs, Controlword, Modes of operation, Target position, Profile velocity, Target velocity, Target torque, Velocity offset, Torque offset, WcState, WcState, InputToggle, InfoData, State, AdsAddr, and Chn0.

The MAIN program window on the right shows the ladder logic code for the CMMT_AS instance. The code includes variable declarations for ActPos, ActVelo, ActTorq, and CMMT_AS, followed by a list of function blocks and their parameters. The code is as follows:

```
PROGRAM MAIN
VAR
  ActPos          : REAL;
  ActVelo         : REAL;
  ActTorq         : REAL;
  CMMT_AS         : AXIS_REF_CiA402_ETC_CMMT_AS;

  Power_CMMT      : MC_Power_Festo;
  Reset_CMMT      : MC_Reset_Festo;
  Home_CMMT       : MC_Home_Festo;
  Halt_CMMT       : MC_Halt_Festo;
  MoveAbs_CMMT    : MC_MoveAbsolute_Festo;
  MoveRel_CMMT    : MC_MoveRelative_Festo;
  ReadParam_CMMT  : MC_ReadParameter_Festo;
  WriteParam_CMMT : MC_WriteParameter_Festo;
  ReadPos_CMMT    : MC_ReadActualPosition_Festo;
  ReadTorq_CMMT   : MC_ReadActualTorque_Festo;
  ReadVelo_CMMT   : MC_ReadActualVelocity_Festo;
  ReadStatus_CMMT : MC_ReadStatus_Festo;
  ReadAxisErr_CMMT : MC_ReadAxisError_Festo;
  MoveAdd_CMMT    : MC_MoveAdditive_Festo;
  Stop_CMMT       : MC_Stop_Festo;
  MoveVelo_CMMT   : MC_MoveVelocity_Festo;
  TorqControl_CMMT : MC_TorqueControl_Festo;
  Jog_CMMT        : MC_Jog_Festo;
  Record_CMMT     : MC_RecordTable_Festo;
  ReadStringParam_CMMT : MC_ReadStringParameter_Festo;
  WriteStringParam_CMMT : MC_WriteStringParameter_Festo;
  ReadAxisInfo_CMMT : MC_ReadAxisInfo_Festo;
END_VAR

// call axis reference cyclical
CMMT_AS();

// get actual values
ActPos := CMMT_AS.ActualPosition;
ActVelo := CMMT_AS.ActualVelocity;
ActTorq := CMMT_AS.ActualTorque;

// call function blocks
Power_CMMT(Axis:=CMMT_AS);
Reset_CMMT(Axis:=CMMT_AS);
Home_CMMT(Axis:=CMMT_AS);
Halt_CMMT(Axis:=CMMT_AS);
MoveAbs_CMMT(Axis:=CMMT_AS);
MoveRel_CMMT(Axis:=CMMT_AS);
ReadParam_CMMT(Axis:=CMMT_AS);
WriteParam_CMMT(Axis:=CMMT_AS);
ReadPos_CMMT(Axis:=CMMT_AS);
ReadTorq_CMMT(Axis:=CMMT_AS);
ReadVelo_CMMT(Axis:=CMMT_AS);
ReadStatus_CMMT(Axis:=CMMT_AS);
ReadAxisErr_CMMT(Axis:=CMMT_AS);
MoveAdd_CMMT(Axis:=CMMT_AS);
Stop_CMMT(Axis:=CMMT_AS);
MoveVelo_CMMT(Axis:=CMMT_AS);
TorqControl_CMMT(Axis:=CMMT_AS);
Jog_CMMT(Axis:=CMMT_AS);
Record_CMMT(Axis:=CMMT_AS);
ReadStringParam_CMMT(Axis:=CMMT_AS);
WriteStringParam_CMMT(Axis:=CMMT_AS);
ReadAxisInfo_CMMT(Axis:=CMMT_AS);
```

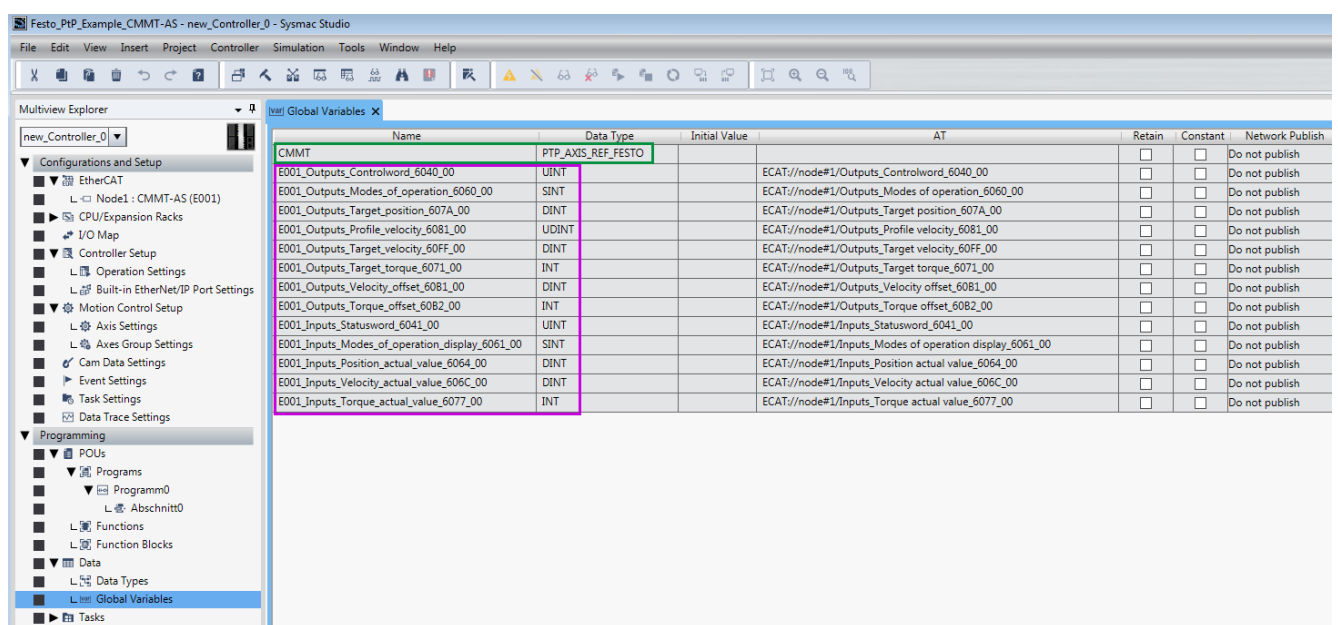
Abbildung 5: Geräteeinbindung TwinCAT

2. 1. 4 Omron Sysmac Studio

Um die PtP-Bibliothek im Sysmac Studio von Omron verwenden zu können, wird empfohlen sie in den dafür vorgesehenen Standardordner zu kopieren (C:\OMRON\Data\Lib). Anschließend kann die Bibliothek in Sysmac Studio unter „Projekt – Bibliothek – Zeige Referenzen“ in das Projekt eingebunden werden.

Nach erfolgreicher Konfiguration des Netzwerks, muss der Servoantriebsregler mit der Achsstruktur der Bibliothek verbunden werden (PTP_AXIS_REF_FESTO). Hierfür sind die Achsstruktur und die Verknüpfungsvariablen für die Ein- und Ausgangsdaten des Antriebs (PDOs) zu instanziiieren. Dies kann beispielsweise in der globalen Variablenliste erfolgen (siehe Abbildung 6: Globale Variablenliste Sysmac Studio).

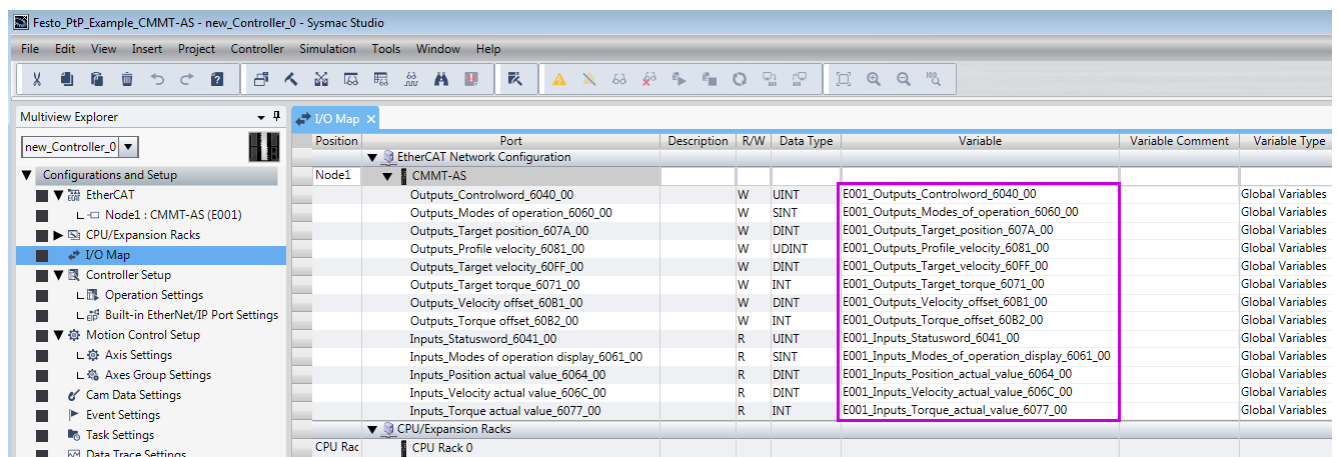
**Hinweis**
Für jeden Servoantriebsregler muss im PLC-Programm eine separate Achsstruktur vom Typ „PTP_AXIS_REF_FESTO“ instanziiert werden.



Name	Data Type	Initial Value	AT	Retain	Constant	Network Publish
CMMT	PTP_AXIS_REF_FESTO					
E001_Outputs_Controlword_6040_00	UINT		ECAT://node#1/Outputs_Controlword_6040_00	☐	☐	Do not publish
E001_Outputs_Modes_of_operation_6060_00	SINT		ECAT://node#1/Outputs_Modes of operation_6060_00	☐	☐	Do not publish
E001_Outputs_Target_position_607A_00	DINT		ECAT://node#1/Outputs_Target position_607A_00	☐	☐	Do not publish
E001_Outputs_Profile_velocity_6081_00	UDINT		ECAT://node#1/Outputs_Profile velocity_6081_00	☐	☐	Do not publish
E001_Outputs_Target_velocity_60FF_00	DINT		ECAT://node#1/Outputs_Target velocity_60FF_00	☐	☐	Do not publish
E001_Outputs_Target_torque_6071_00	INT		ECAT://node#1/Outputs_Target torque_6071_00	☐	☐	Do not publish
E001_Outputs_Velocity_offset_6081_00	DINT		ECAT://node#1/Outputs_Velocity offset_6081_00	☐	☐	Do not publish
E001_Outputs_Torque_offset_6082_00	INT		ECAT://node#1/Outputs_Torque offset_6082_00	☐	☐	Do not publish
E001_Inputs_Statusword_6041_00	UINT		ECAT://node#1/Inputs_Statusword_6041_00	☐	☐	Do not publish
E001_Inputs_Modes_of_operation_display_6061_00	SINT		ECAT://node#1/Inputs_Modes of operation display_6061_00	☐	☐	Do not publish
E001_Inputs_Position_actual_value_6064_00	DINT		ECAT://node#1/Inputs_Position actual value_6064_00	☐	☐	Do not publish
E001_Inputs_Velocity_actual_value_606C_00	DINT		ECAT://node#1/Inputs_Velocity actual value_606C_00	☐	☐	Do not publish
E001_Inputs_Torque_actual_value_6077_00	INT		ECAT://node#1/Inputs_Torque actual value_6077_00	☐	☐	Do not publish

Abbildung 6: Globale Variablenliste Sysmac Studio

Nach erfolgreicher Instanziierung müssen die zyklischen Prozessdaten des Motorcontrollers mit den angelegten Variablen verknüpft werden (PDO Mapping), siehe Abbildung 7: PDO Mapping Sysmac Studio.



Position	Port	Description	R/W	Data Type	Variable	Variable Comment	Variable Type
Node1	EtherCAT Network Configuration						
	CMMT-AS						
		Outputs_Controlword_6040_00	W	UINT	E001_Outputs_Controlword_6040_00		Global Variables
		Outputs_Modes of operation_6060_00	W	SINT	E001_Outputs_Modes_of_operation_6060_00		Global Variables
		Outputs_Target position_607A_00	W	DINT	E001_Outputs_Target_position_607A_00		Global Variables
		Outputs_Profile velocity_6081_00	W	UDINT	E001_Outputs_Profile_velocity_6081_00		Global Variables
		Outputs_Target velocity_60FF_00	W	DINT	E001_Outputs_Target_velocity_60FF_00		Global Variables
		Outputs_Target torque_6071_00	W	INT	E001_Outputs_Target_torque_6071_00		Global Variables
		Outputs_Velocity offset_6081_00	W	DINT	E001_Outputs_Velocity_offset_6081_00		Global Variables
		Outputs_Torque offset_6082_00	W	INT	E001_Outputs_Torque_offset_6082_00		Global Variables
		Inputs_Statusword_6041_00	R	UINT	E001_Inputs_Statusword_6041_00		Global Variables
		Inputs_Modes of operation display_6061_00	R	SINT	E001_Inputs_Modes_of_operation_display_6061_00		Global Variables
		Inputs_Position actual value_6064_00	R	DINT	E001_Inputs_Position_actual_value_6064_00		Global Variables
		Inputs_Velocity actual value_606C_00	R	DINT	E001_Inputs_Velocity_actual_value_606C_00		Global Variables
		Inputs_Torque actual value_6077_00	R	INT	E001_Inputs_Torque_actual_value_6077_00		Global Variables

Abbildung 7: PDO Mapping Sysmac Studio

Abschließend muss im PLC-Programm der Achsbaustein instanziiert und zyklisch aufgerufen werden (rote Rahmen). Als Ein- und Ausgangsvariablen dienen die bereits angelegte Achsstruktur (grüne Rahmen) und die Verknüpfungsvariablen (lila Rahmen) der Ein- und Ausgänge des Antriebs (siehe Abbildung 8). Die Eingangsvariable „EC_PDActive“ muss für die Abarbeitung des Achsbausteins (PTP_CIA402_CMMT_AS) aktiv sein, sie kann beispielweise mit der Systemvariablen „_EC_PDActive“ belegt werden (gelber Rahmen). Um die zwingend erforderliche SDO-Kommunikation (Service-Daten-Objekt Kommunikation) zwischen PLC und Antrieb zu gewährleisten, muss dem Achsbaustein zusätzlich noch die „SlaveNodeInfo“ übergeben werden (blauer Pfeil).

➔

Hinweis

Der Achsbaustein vom Typ „PTP_CIA402_CMMT_FESTO“ muss immer zyklisch aufgerufen werden.

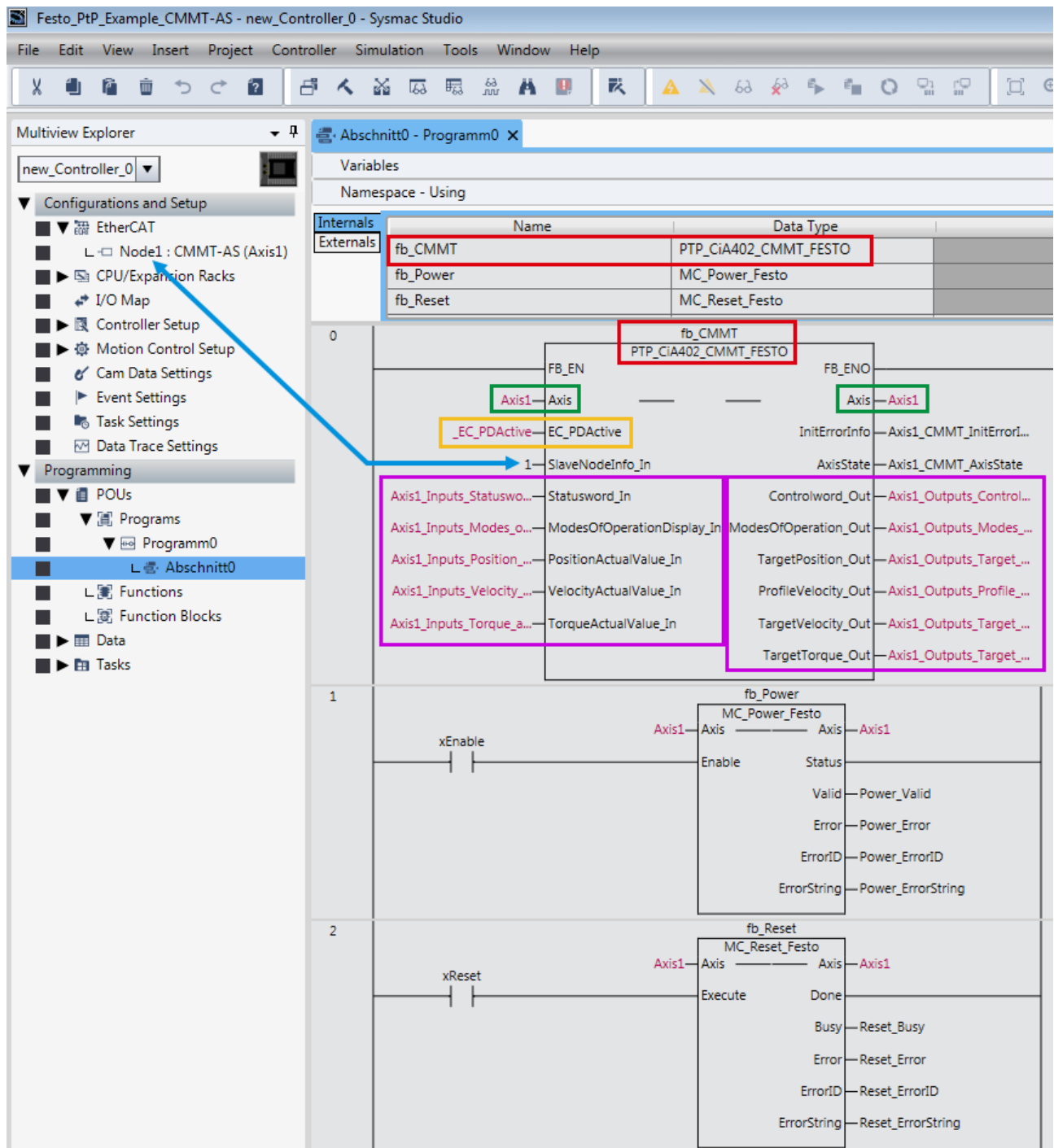


Abbildung 8: Zyklischer Aufruf des Achsbausteins

2. 1. 5 Lenze PLC Designer

Zuerst muss das jeweilige Package des Motorcontrollers für Lenze PLCs unter „Tools“ mit dem „CODESYS Package Manager“ installiert werden. Anschließend müssen die Bibliotheken und der Antrieb dem PLC Designer Projekt hinzugefügt werden (siehe Abbildung 9).

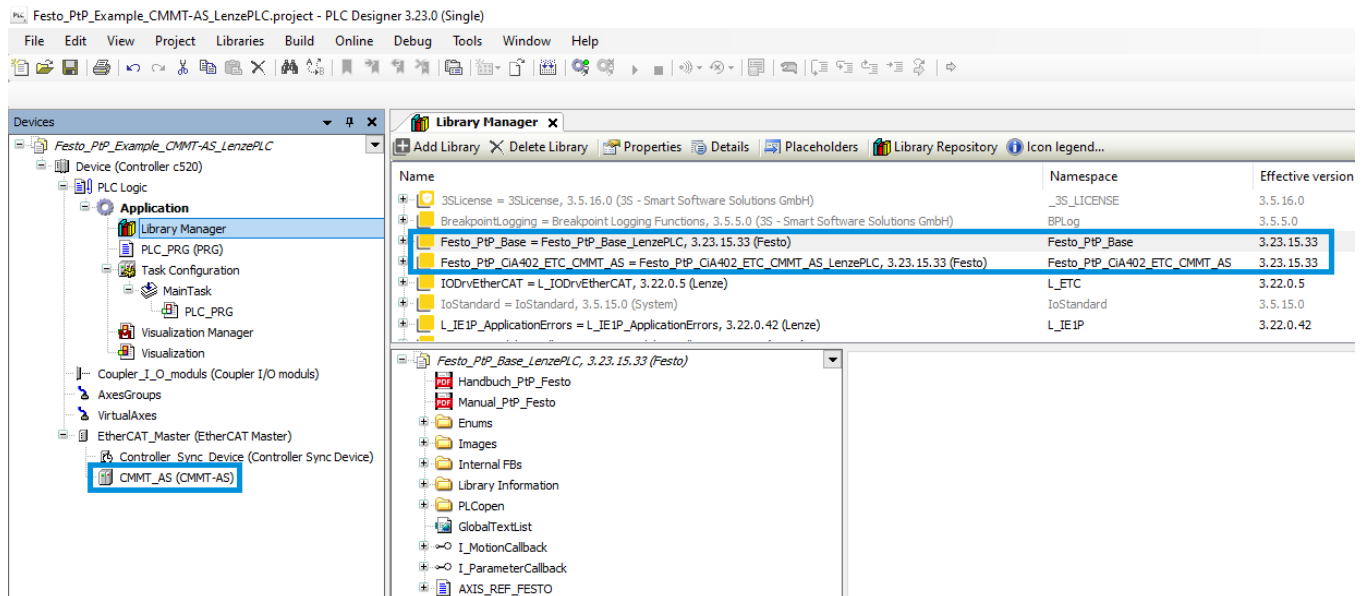


Abbildung 9: Hinzufügen von Bibliotheken und Antrieb im Lenze PLC Designer

Im nächsten Schritt kann der Achstreiberbaustein (AXIS_REF_CiA402_ETC_CMMT_xx) deklariert und zyklisch aufgerufen werden. Hierbei muss die Instanz des EtherCAT Masters und die EtherCAT Slave Adresse des Motorcontrollers angegeben werden (siehe Abbildung 10).

➔

Hinweis

Der Achsbaustein vom Typ „AXIS_REF_CiA402_ETC_CMMT_xx“ muss immer zyklisch aufgerufen werden.

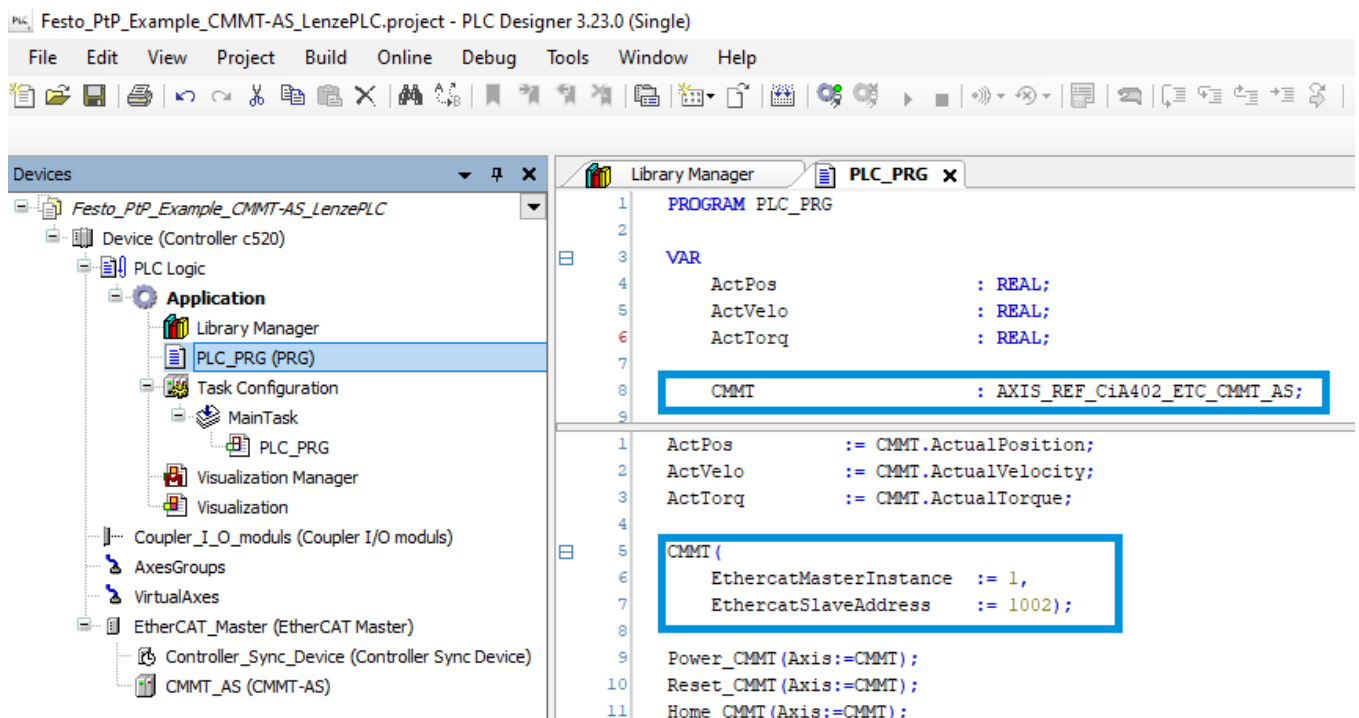


Abbildung 10: Zyklischer Aufruf des Achstreibers für Lenze PLCs

Abschließend müssen die Prozessdatenobjekte (PDOs) noch mit dem Achstreiberbaustein verknüpft werden (siehe Abbildung 11).

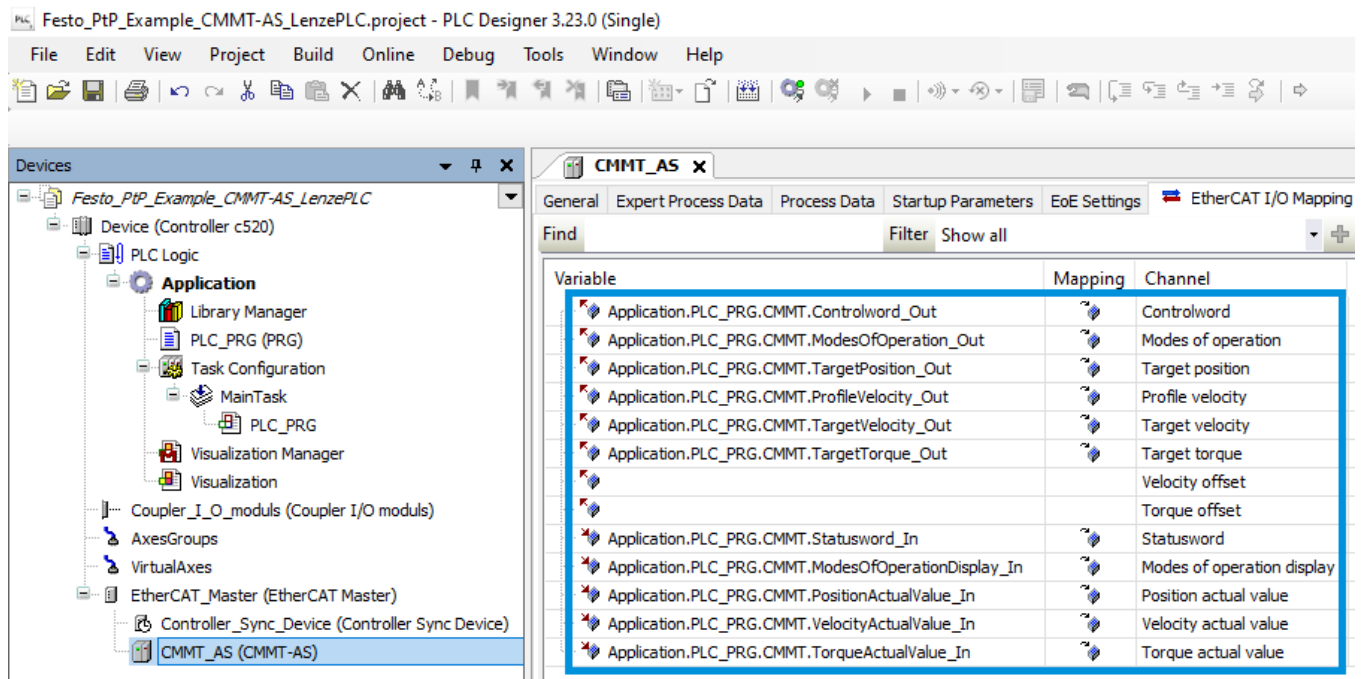


Abbildung 11: Verknüpfung PDOs mit Achstreiberstruktur im Lenze PLC Designer

2.2 Sachgemäße Verwendung der azyklischen Kommunikation (SDO-Zugriff)

Über den Service-Daten-Objekt (SDO) Schreib- bzw. Lesebefehl kann die Steuerung azyklisch auf das Objektverzeichnis des Antriebs zugreifen. Dadurch werden Daten eines Parameters beschreiben oder gelesen. Nach Abarbeitung des SDO-Befehls sendet der Antrieb eine Quittierung an die Steuerung. Diese Kommunikation erzeugt azyklische Buslast. Abhängig von der Anzahl der Antriebe und der Anzahl der gleichzeitig abgesetzten SDO-Befehle kann die Steuerung das Limit der maximal gleichzeitig zulässigen SDO-Kommunikationsbefehle erreichen. Daher wird empfohlen, die Anzahl der azyklischen Lese- und Schreibzugriffe auf ein Minimum zu beschränken.

Auch die Bausteine der PtP-Bibliothek von Festo erzeugen implizit SDO-Befehle, wenn sich Eingangsdaten am Baustein im Vergleich zum vorherigen Aufruf verändern (z. B. „Acceleration“, „Deceleration“, „Jerk“, „Direction“, „BufferMode“, „HomingMethod“, usw.). Bleiben die Werte unverändert zum vorherigen Aufruf, erzeugt der jeweilige Baustein nur beim allerersten Aufruf jeweils ein SDO-Schreibbefehl pro Eingangsparameter. Auch dieser erste Schreibbefehl kann verhindert werden, wenn der Eingang „Acceleration“, „Deceleration“, „Jerk“, „Direction“ oder „BufferMode“ zum Zeitpunkt des Bewegungsstarts = 0 ist. In diesem Fall wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Wert verwendet. Es wird kein azyklischer Schreibbefehl generiert.

Zusätzlich erzeugt die PtP-Bibliothek in der Initialisierungsphase pro Antrieb mehrere SDO-Befehle, um eingestellte Parameter wie z. B. die Faktorengruppe auszulesen. Diese azyklischen Lesebefehle können nicht deaktiviert werden.

➔

Hinweis

Bei Steuerungen von Omron können maximal 32 Anweisungen gleichzeitig ausgeführt werden. Wird die Anzahl von 32 überschritten, erscheint der Fehler „Communication Resource Overflow“. In diesem Fall wird empfohlen die obigen Hinweise zur Minimierung der azyklischen Befehle umzusetzen.

Folgende Bausteine von Omron erzeugen eine Anweisung: EC_CoESDOWrite, EC_CoESDOWrite, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj, and IOL_WriteObj.

3 PLCopen Funktionsbausteine

3.1 Allgemein

Die hier beschriebenen Bausteine wurden anhand der technischen PLCopen Spezifikation „Function blocks for motion control - Version 2.0“ entwickelt. Es wurde ein Großteil der „Single-Axis“ Bausteine umgesetzt. Um den Funktionsumfang für Motorcontroller von Festo zu erweitern, wurden zusätzlich weitere Bausteine im gleichen Design und mit gleichem Verhalten implementiert. Weitere Informationen, wie zum Beispiel Zeitdiagramme, können in der PLCopen Spezifikation nachgelesen werden. Folgende Bausteine beinhaltet diese Bibliothek:

- MC_Power_Festo
- MC_Home_Festo
- MC_Stop_Festo
- MC_Halt_Festo
- MC_MoveAbsolute_Festo
- MC_MoveRelative_Festo
- MC_MoveAdditive_Festo
- MC_MoveVelocity_Festo
- MC_TorqueControl_Festo
- MC_ReadParameter_Festo
- MC_WriteParameter_Festo
- MC_ReadStringParameter_Festo
- MC_WriteStringParameter_Festo
- MC_ReadActualPosition_Festo
- MC_ReadActualVelocity_Festo
- MC_ReadActualTorque_Festo
- MC_ReadStatus_Festo
- MC_ReadAxisError_Festo
- MC_Reset_Festo
- MC_Jog_Festo
- MC_RecordTable_Festo
- MC_ReadAxisInfo_Festo
- MC_DeviceService_Festo
- MC_ForceControl_Festo



Hinweis

Bausteineingänge müssen zum Teil in Benutzereinheiten vorgegeben werden (z. B. Position, Distanz, Geschwindigkeit und Dynamikwerte). Hierbei handelt es sich immer um die im Antrieb im Reiter „Feldbus“ parametrisierte Benutzereinheit (z. B. Umdrehungen, Grad, Inch, usw.).

Besonderheit:

Wurde im Antrieb die Benutzereinheit „Meter“ eingestellt, müssen die Bausteineingänge in mm, mm/s, mm/s² oder mm/s³ vorgegeben werden. Auch die Bausteinausgänge, zum Beispiel der aktuellen Position, sind in diesem Fall in mm angegeben.

3.2 PLCopen Diagramm

Das nachfolgende Diagramm zeigt die einzelnen Zustände einer Achse und die jeweils möglichen Übergänge. Der aktuelle Zustand der Achse kann über die Variable „Gertätename.AxisState“ abgefragt werden. Auf eine detaillierte Beschreibung der einzelnen Zustände wird an dieser Stelle verzichtet, da diese ebenfalls in der PLCopen Spezifikation eingesehen werden kann.

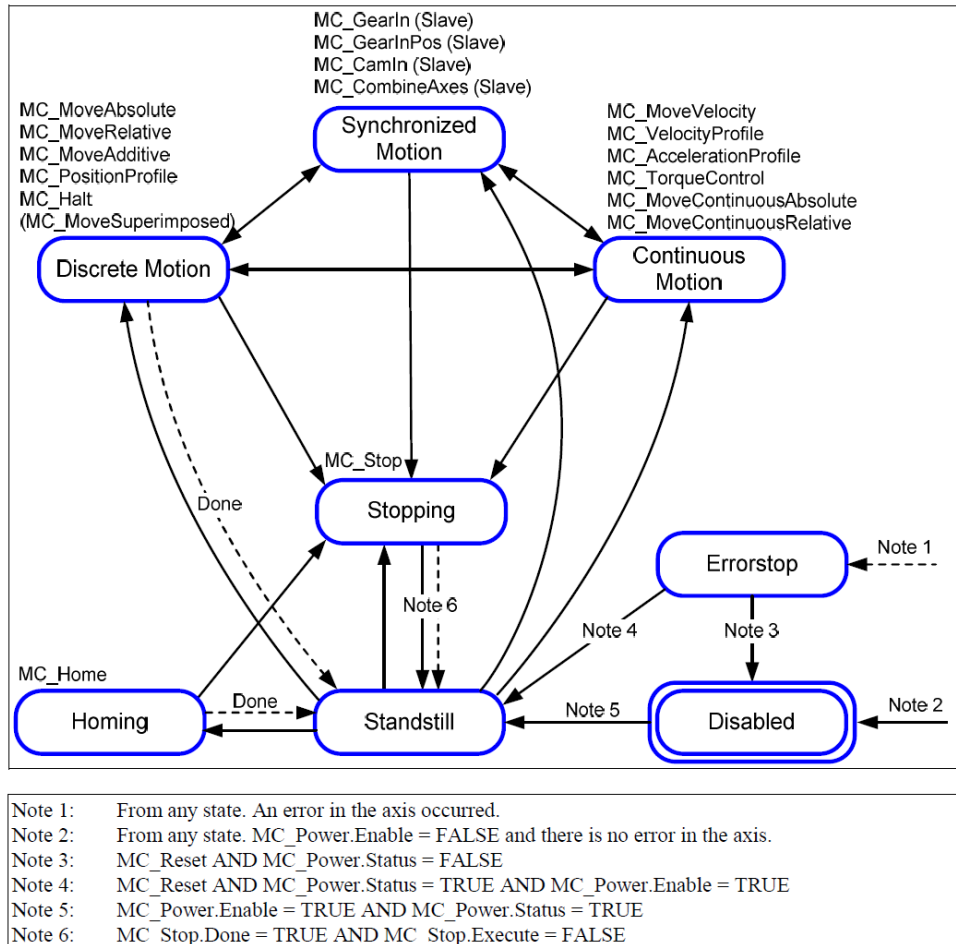


Abbildung 12: PLCopen Diagramm (Quelle: Technical Specification PLCopen - FB for motion control)

3.3 MC_Power_Festo

Dieser Funktionsblock steuert die Leistungsstufe (Freigeben oder Sperren).

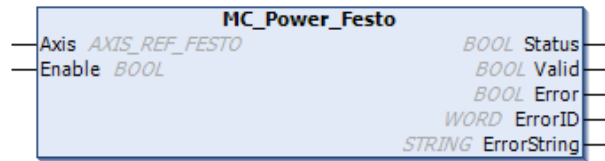


Abbildung 13: MC_Power_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Antrieb freigeben • TRUE = Leistungsstufe freigeben • FALSE = Leistungsstufe sperren
VAR_OUTPUT		
Status	BOOL	Status der Leistungsstufe • TRUE = Freigegeben • FALSE = Gesperrt
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.4 MC_Home_Festo

Dieser Funktionsblock startet eine Referenzfahrt. Wird das Referenzsignal detektiert, wird der Wert des Eingangs „Position“ als absolute Position gesetzt. Die Methode der Referenzfahrt kann über den Eingang „HomingMethod“ ausgewählt werden.



Abbildung 14: MC_Home_Festo

→ Hinweis

Ist der Eingang „HomingMethod“ = 0, wird die im Gerät parametrisierte Referenzfahrtmethode und der im Gerät parametrisierte Achsnulldpunkt verwendet. Der im Baustein angegebene Achsnulldpunkt (Eingang „Position“) wird ignoriert.

Ist der Eingang „HomingMethod“ $\neq$ 0, wird die im Gerät parametrisierte Referenzfahrtmethode und der im Gerät parametrisierte Achsnulldpunkt überschrieben.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Referenzfahrt starten FALSE --> True = Starte Referenzfahrt
Position	REAL	Absolute Position, wenn Referenzschalter detektiert wurde (Achsnulldpunkt in Benutzereinheit).
HomingMethod	DINT	Referenzfahrtmethode nach CiA402 Spezifikation. Die vom Gerät unterstützten Referenzfahrtmethoden sind im Gerätehandbuch zu finden.
VAR_OUTPUT		
Done	BOOL	Referenzfahrt erfolgreich abschlossen
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Referenzfahrt wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.5 MC_Stop_Festo

Dieser Funktionsblock hält eine Achse mit der in der Automation Suite parametrierten Stopprampe an und setzt die Achse in den Status „Stopping“. Alle weiteren Bewegungskommandos werden abgebrochen. Solange die Achse sich im Status „Stopping“ befindet, kann kein anderer Baustein diese Achse bewegen. Nachdem die aktuelle Geschwindigkeit der Achse null ist, wird der Ausgang „Done“ gesetzt. Die Achse bleibt im Zustand „Stopping“ solange „Execute“ aktiv (TRUE) ist oder die aktuelle Geschwindigkeit ungleich null ist. Sobald „Done“ gesetzt wurde und „Execute“ inaktiv (FALSE) ist wechselt der Achszustand zu „Standstill“.



Abbildung 15: MC_Stop_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Achsbewegung stoppen FALSE --> True = Stoppe Achsbewegung
VAR_OUTPUT		
Done	BOOL	Aktuelle Geschwindigkeit der Achse ist null (Achse steht still)
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
CommandAborted	BOOL	Auftrag wurde abgebrochen, da Leistungsstufe ausgeschaltet wurde (einzige Abbruchmöglichkeit)
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.6 MC_Halt_Festo

Dieser Funktionsblock hält eine Achse mit der eingestellten Bremsrampe der aktuellen Bewegung an und setzt sie in den Zustand „DiscreteMotion“. Nach dem die aktuelle Geschwindigkeit der Achse null ist, wird der Ausgang „Done“ gesetzt und die Achse nimmt den Zustand „Standstill“ an.



Abbildung 16: MC_Halt_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Achsbewegung anhalten FALSE --> True = Achsbewegung anhalten
VAR_OUTPUT		
Done	BOOL	Aktuelle Geschwindigkeit der Achse ist null (Achse steht still)
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Anhalten der Achse wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.7 MC_MoveAbsolute_Festo

Dieser Funktionsblock starte anhand der eingestellten Dynamikparameter eine Bewegung zu einer absoluten Position.

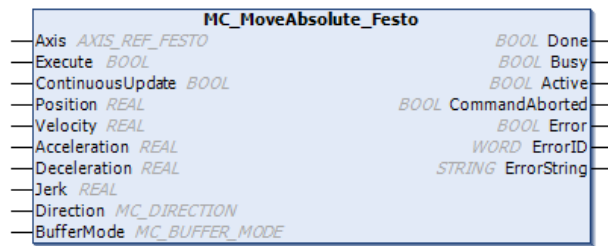


Abbildung 17: MC_MoveAbsolute_Festo

➔

Hinweis

Ist einer der Eingänge „Acceleration“, „Deceleration“ bzw. „Jerk“ zum Zeitpunkt des Bewegungsstarts = 0, wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Dynamikwert verwendet (siehe auch Kapitel 2. 2).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Verfahrauftrag starten FALSE --> True = Verfahrauftrag starten
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
Position	REAL	Absolute Zielposition (in Benutzereinheit)
Velocity	REAL	Maximale Geschwindigkeit (in Benutzereinheit)
Acceleration	REAL	Beschleunigung (in Benutzereinheit)
Deceleration	REAL	Verzögerung (in Benutzereinheit)
Jerk	REAL	Ruck (in Benutzereinheit)
Direction	MC_DIRECTION	Definiert die Positionierrichtung, wenn „Modulo“ im Antrieb aktiviert wurde (siehe Kapitel 7 – Direction)
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Verfahrauftrag erfolgreich abschlossen
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abgeschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.8 MC_MoveRelative_Festo

Dieser Funktionsblock startet anhand der eingestellten Dynamikparameter eine Bewegung um eine Distanz relativ zur Sollposition der Achse zum Zeitpunkt der Bausteinfreigabe.

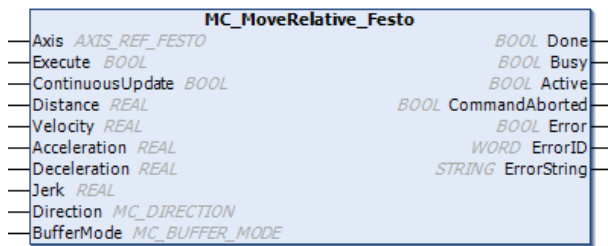


Abbildung 18: MC_MoveRelative_Festo

➔

Hinweis

Ist einer der Eingänge „Acceleration“, „Deceleration“ bzw. „Jerk“ zum Zeitpunkt des Bewegungsstarts = 0, wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Dynamikwert verwendet (siehe auch Kapitel 2. 2).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Verfahrauftrag starten FALSE --> True = Verfahrauftrag starten
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
Distance	REAL	Relative Distanz der Bewegung (in Benutzereinheit)
Velocity	REAL	Maximale Geschwindigkeit (in Benutzereinheit)
Acceleration	REAL	Beschleunigung (in Benutzereinheit)
Deceleration	REAL	Verzögerung (in Benutzereinheit)
Jerk	REAL	Ruck (in Benutzereinheit)
Direction	MC_DIRECTION	Definiert die Positionierrichtung, wenn „Modulo“ im Antrieb aktiviert wurde (siehe Kapitel 7 – Direction)
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Verfahrauftrag erfolgreich abschlossen
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.9 MC_MoveAdditive_Festo

Dieser Funktionsblock startet anhand der eingestellten Dynamikparameter eine Bewegung um eine Distanz relativ zur aktuell geplanten Zielposition im Achsstatus „DiscreteMotion“. Diese kann ein Resultat eines vorhergehenden Bausteins desselben Typs sein, welcher abgebrochen wurde. Wird er im Achsstatus „ContinuousMotion“ gestartet, führt er eine Bewegung um eine Distanz relativ zur Achsposition zum Zeitpunkt der Bausteinfreigabe aus.



Abbildung 19: MC_MoveAdditive_Festo

➔

Hinweis

Ist einer der Eingänge „Acceleration“, „Deceleration“ bzw. „Jerk“ zum Zeitpunkt des Bewegungsstarts = 0, wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Dynamikwert verwendet (siehe auch Kapitel 2. 2).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Verfahrauftrag starten FALSE --> True = Verfahrauftrag starten
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
Distance	REAL	Relative Distanz zur Zielposition (in Benutzereinheit)
Velocity	REAL	Maximale Geschwindigkeit (in Benutzereinheit)
Acceleration	REAL	Beschleunigung (in Benutzereinheit)
Deceleration	REAL	Verzögerung (in Benutzereinheit)
Jerk	REAL	Ruck (in Benutzereinheit)
Direction	MC_DIRECTION	Definiert die Positionierrichtung, wenn „Modulo“ im Antrieb aktiviert wurde (siehe Kapitel 7 – Direction)
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Verfahrauftrag erfolgreich abschlossen
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 10 MC_MoveVelocity_Festo

Dieser Funktionsblock startet eine kontrollierte Bewegung mit einer festgelegten Geschwindigkeit. Hierbei handelt es sich um den Geschwindigkeitsbetrieb (ProfileVelocityMode).



Abbildung 20: MC_MoveVelocity_Festo

Hinweis

Ist einer der Eingänge „Acceleration“, „Deceleration“, „Jerk“, „NegativeStrokeLimit“ bzw. „PositiveStrokeLimit“ zum Zeitpunkt des Bewegungsstarts = 0, wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Dynamikwert bzw. Grenzwert verwendet (siehe auch Kapitel 2. 2).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Geschwindigkeitsbetrieb starten FALSE --> True = Geschwindigkeitsbetrieb mit (eingestellten Dynamikparameter starten)
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
Velocity	REAL	Maximale Geschwindigkeit (in Benutzereinheit)
Acceleration	REAL	Beschleunigung (in Benutzereinheit)
Deceleration	REAL	Verzögerung (in Benutzereinheit)
Jerk	REAL	Ruck (in Benutzereinheit)
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
StrokeLimitation	BOOL	Aktivierung der Hubgrenze TRUE = Hubgrenze ist für den Bewegungsauftrag aktiv
NegativeStrokeLimit	REAL	Negative Hubgrenze (in Benutzereinheit)
PositiveStrokeLimit	REAL	Positive Hubgrenze (in Benutzereinheit)
VAR_OUTPUT		
InVelocity	BOOL	Zielgeschwindigkeit erreicht
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrenauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 11 MC_TorqueControl_Festo

Dieser Funktionsblock startet eine Bewegung mit einem kontinuierlichen Drehmoment gemessen an der Getriebewelle. Die angegebene Geschwindigkeit wird nicht überschritten. Hierbei handelt es sich um den Kraftbetrieb (ProfileTorqueMode).



Abbildung 21: MC_TorqueControl_Festo

➔ Hinweis

Ist einer der Eingänge „NegativeStrokeLimit“ bzw. „PositiveStrokeLimit“ zum Zeitpunkt des Bewegungsstarts = 0, wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Grenzwert verwendet (siehe auch Kapitel 2. 2).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Kraftbetrieb starten FALSE --> True = Starte Kraftbetrieb
Torque	REAL	Solldrehmoment an der Getriebewelle (in Nm)
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
TorqueRamp	REAL	Drehmomentrampe (in Nm/s)
Velocity	REAL	Maximale Geschwindigkeit (in Benutzereinheit)
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
StrokeLimitation	BOOL	Aktivierung der Hubgrenze TRUE = Hubgrenze ist für den Bewegungsauftrag aktiv
NegativeStrokeLimit	REAL	Negative Hubgrenze (in Benutzereinheit)
PositiveStrokeLimit	REAL	Positive Hubgrenze (in Benutzereinheit)
VAR_OUTPUT		
InTorque	BOOL	Zieldrehmoment erreicht
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrenauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 12 MC_ReadParameter_Festo

Dieser Funktionsblock liest den Wert eines Parameters via Servicedatenobjekt-Zugriff (SDO-Zugriff) aus.



Abbildung 22: MC_ReadParameter_Festo



Hinweis

Dieser Funktionsbaustein bewirkt eine ständige SDO-Kommunikation zwischen Antrieb und PLC. Bei kontinuierlichem Lesen belastet er den Feldbus unnötig stark. Diesen Baustein bitte nur situativ einsetzen.



Hinweis

Baustein nicht zum Lesen von Parametern des Datentyps „STRING“ geeignet. Hierfür ist der Baustein „MC_ReadStringParameter_Festo“ zu verwenden.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	SDO Wert lesen • TRUE = SDO kontinuierlich lesen solange TRUE • FALSE = Lesevorgang deaktivieren
ParameterNumber	UINT	Nummer des zu lesenden Parameters
SubindexNumber	USINT	Subindex des zu lesenden Parameters
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
Busy	BOOL	Auftrag ist noch nicht abgeschlossen • TRUE = Auftrag aktiv • FALSE = Auftrag abschlossen oder nicht gestartet
Value	LREAL	Wert des gelesenen Parameters
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.13 MC_WriteParameter_Festo

Dieser Funktionsblock modifiziert den Wert eines Parameters via Servicedatenobjekt-Zugriff (SDO-Zugriff).



Abbildung 23: MC_WriteParameter_Festo

→ Hinweis

Baustein nicht zum Schreiben von Parametern des Datentyps „STRING“ geeignet. Hierfür ist der Baustein „MC_WriteStringParameter_Festo“ zu verwenden.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	SDO Wert modifizieren FALSE --> TRUE = Schreibe neuen Wert des SDOs
ParameterNumber	UINT	Nummer des zu schreibenden Parameters
SubindexNumber	USINT	Subindex des zu schreibenden Parameters
Value	LREAL	Neuer Wert des zu schreibenden Parameters
VAR_OUTPUT		
Done	BOOL	Parameter erfolgreich geschrieben
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag aktiv - FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 14 MC_ReadStringParameter_Festo

Dieser Funktionsblock liest den Wert eines Servicedatenobjekts (SDO) des Datentyps „STRING“ aus. Es wird eine Zeichenkette von maximal 35 Zeichen ausgelesen.



Abbildung 24: MC_ReadStringParameter_Festo

Hinweis
Dieser Funktionsbaustein bewirkt eine ständige SDO-Kommunikation zwischen Antrieb und PLC. Bei kontinuierlichem Lesen belastet er den Feldbus unnötig stark. Diesen Baustein bitte nur situativ einsetzen.

Hinweis
Baustein nur zum Lesen von Parametern des Datentyps „STRING“ geeignet. Für Parameter mit anderen Datentypen ist der Baustein „MC_ReadParameter_Festo“ zu verwenden.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	SDO Wert lesen • TRUE = SDO kontinuierlich lesen solange TRUE • FALSE = Lesevorgang deaktivieren
ParameterNumber	UINT	Nummer des zu lesenden Parameters
SubindexNumber	USINT	Subindex des zu lesenden Parameters
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
Busy	BOOL	Auftrag ist noch nicht abgeschlossen • TRUE = Auftrag aktiv • FALSE = Auftrag abschlossen oder nicht gestartet
Value	STRING(35)	Zeichenkette des gelesenen Parameters
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 15 MC_WriteStringParameter_Festo

Dieser Funktionsblock modifiziert die Zeichenkette eines Servicedatenobjekts (SDO) des Datentyps „STRING“. Es wird eine Zeichenkette von maximal 35 Zeichen geschrieben.



Abbildung 25: MC_WriteStringParameter_Festo

➔

Hinweis

Baustein nur zum Schreiben von Parametern des Datentyps „STRING“ geeignet. Für Parameter mit anderen Datentypen ist der Baustein „MC_WriteParameter_Festo“ zu verwenden.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	SDO Wert modifizieren FALSE --> TRUE = Schreibe neuen Wert des SDOs
ParameterNumber	UINT	Nummer des zu schreibenden Parameters
SubindexNumber	USINT	Subindex des zu schreibenden Parameters
Value	STRING(35)	Neue Zeichenkette des zu schreibenden Parameters
VAR_OUTPUT		
Done	BOOL	Parameter erfolgreich geschrieben
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag aktiv - FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 16 MC_ReadActualPosition_Festo

Dieser Funktionsblock liest die aktuelle Position der Achse.

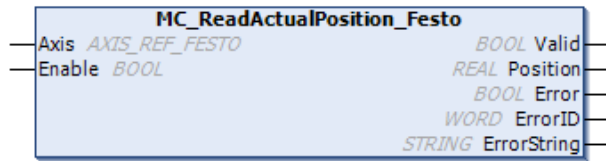


Abbildung 26: MC_ReadActualPosition_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Aktuelle Position der Achse (ActPos) lesen • TRUE = ActPos kontinuierlich lesen solange TRUE • FALSE = Lesevorgang deaktivieren
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
Position	REAL	Aktuelle Position der Achse (in Benutzereinheit)
Busy	BOOL	Auftrag ist noch nicht abgeschlossen • TRUE = Auftrag aktiv • FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 17 MC_ReadActualVelocity_Festo

Dieser Funktionsblock liest die aktuelle Geschwindigkeit der Achse.

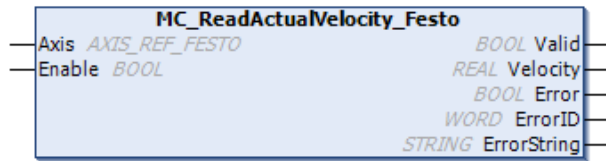


Abbildung 27: MC_ReadActualVelocity_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Aktuelle Geschwindigkeit der Achse (ActVelo) lesen • TRUE = ActVelo kontinuierlich lesen solange TRUE • FALSE = Lesevorgang deaktivieren
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
Velocity	REAL	Aktuelle Geschwindigkeit der Achse (in Benutzereinheit)
Busy	BOOL	Auftrag ist noch nicht abgeschlossen • TRUE = Auftrag aktiv • FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.18 MC_ReadActualTorque_Festo

Dieser Funktionsblock liest das aktuelle Drehmoment der Achse an der Getriebewelle.

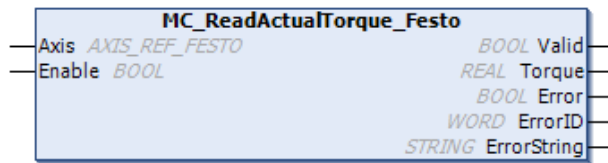


Abbildung 28: MC_ReadActualTorque_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Aktuelles Drehmoment der Achse (ActTorq) lesen • TRUE = ActTorq kontinuierlich lesen so lange TRUE • FALSE = Lesevorgang deaktivieren
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
Torque	REAL	Aktuelles Drehmoment der Achse (in Benutzereinheit)
Busy	BOOL	Auftrag ist noch nicht abgeschlossen • TRUE = Auftrag aktiv • FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 19 MC_ReadStatus_Festo

Dieser Funktionsblock liest den aktuellen Status des Motorcontrollers laut „PLCopen State Diagram“ (siehe Kapitel 3. 2 – PLCopen Diagramm).

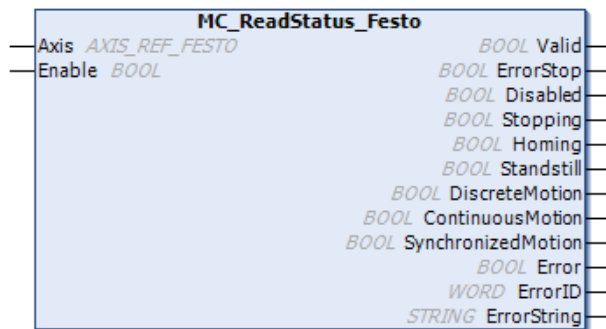


Abbildung 29: MC_ReadStatus_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Aktueller Status der Achse lesen • TRUE = Status kontinuierlich lesen solange TRUE • FALSE = Lesevorgang deaktivieren
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
ErrorStop	BOOL	Antrieb befindet sich im Zustand „ErrorStop“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
Disabled	BOOL	Antrieb befindet sich im Zustand „Disabled“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
Stopping	BOOL	Antrieb befindet sich im Zustand „Stopping“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
Homing	BOOL	Antrieb befindet sich im Zustand „Homing“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
Standstill	BOOL	Antrieb befindet sich im Zustand „Standstill“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
DiscreteMotion	BOOL	Antrieb befindet sich im Zustand „DiscreteMotion“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
ContinuousMotion	BOOL	Antrieb befindet sich im Zustand „ContinuousMotion“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
SynchronizedMotion	BOOL	Antrieb befindet sich im Zustand „SynchronizedMotion“ (siehe Kapitel 3. 2 – PLCopen Diagramm)
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 20 MC_ReadAxisError_Festo

Dieser Funktionsblock liest kontinuierlich den aktuellen Fehler des Motorcontrollers.

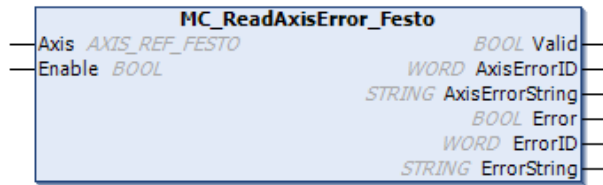


Abbildung 30: MC_ReadAxisError_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Aktuelle Fehler der Achse lesen • TRUE = Fehler kontinuierlich lesen solange TRUE • FALSE = Lesevorgang deaktivieren
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins • TRUE = Status der Ausgänge sind gültig • FALSE = Status der Ausgänge sind nicht gültig
AxisErrorID	WORD	Aktuelle Fehlernummer der Achse
AxisErrorString	STRING	Aktuelle Fehlerbeschreibung der Achse
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 21 MC_Reset_Festo

Dieser Funktionsblock löst einen Reset der Fehler des Motorcontrollers mit Hilfe des Steuerworts des Antriebsprofils aus. Zusätzlich werden die Fehler des EtherCAT Slaves im PLC Programm durch die Methode „ClearEmergency“ zurückgesetzt.

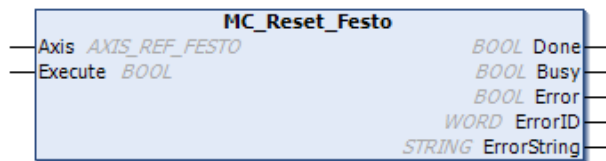


Abbildung 31: MC_Reset_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Setze Fehler zurück FALSE --> TRUE = Setze Fehler der Achse zurück
VAR_OUTPUT		
Done	BOOL	Fehler erfolgreich zurückgesetzt (kein Achsfehler)
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.22 MC_Jog_Festo

Dieser Funktionsblock startet eine Bewegung des Motorcontrollers im ProfileJogMode.

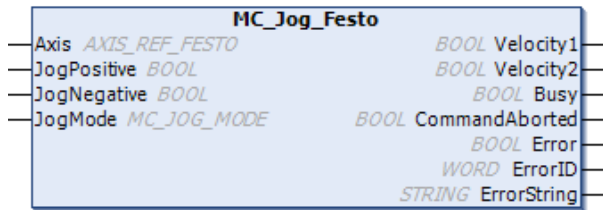


Abbildung 32: MC_Jog_Festo

Hinweis

Achse bleibt stehen, wenn die Eingänge „JogPositive“ und „JogNegative“ denselben Status haben (beide TRUE oder beide FALSE).

Hinweis

Der Eingang „JogMode“ hat 3 zulässige Eingangswerte (Modi). Wird der Eingangswert von 0 angegeben, joggt der Antrieb erst in Geschwindigkeit 1. Nach einer parametrierbaren Zeit beschleunigt/verzögert der Antrieb auf Geschwindigkeit 2. Die jeweilige Geschwindigkeit und die Zeit können im Motorcontroller eingestellt werden (weitere Infos sind im Gerätehandbuch zu finden).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
JogPositive	BOOL	joggen in positive Richtung TRUE = Achse joggt in positive Richtung
JogNegative	BOOL	joggen in negative Richtung TRUE = Achse joggt in negative Richtung
JogMode	MC_JOG_MODE	Jog Modus 0 (mcDefault) = Geschwindigkeit 1 und 2 1 (mcOnlyVelocity1) = nur Geschwindigkeit 1 2 (mcOnlyVelocity2) = nur Geschwindigkeit 2
VAR_OUTPUT		
Velocity1	BOOL	Antrieb joggt mit Geschwindigkeit 1
Velocity2	BOOL	Antrieb joggt mit Geschwindigkeit 2
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag aktiv - FALSE = Auftrag abschlossen oder nicht gestartet
CommandAborted	BOOL	Verfahrenauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 23 MC_RecordTable_Festo

Dieser Funktionsblock startet einen im Motorcontroller parametrierten Verfahrssatz. Es handelt sich hierbei um den RecordTableMode.



Abbildung 33: MC_RecordTableMode_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Verfahrssatz starten FALSE --> True = Verfahrssatz starten
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
RecordNumber	DINT	Nummer des zu startenden Verfahrssatz
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
VAR_OUTPUT		
Done	BOOL	Verfahrssatz erfolgreich ausgeführt
ActualRecordNumber	DINT	Aktive Verfahrssatznummer
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 24 MC_ReadAxisInfo_Festo

Dieser Funktionsblock liest interne Statusinformationen der Achse aus (z: B. Status der End- und Referenzschalter).

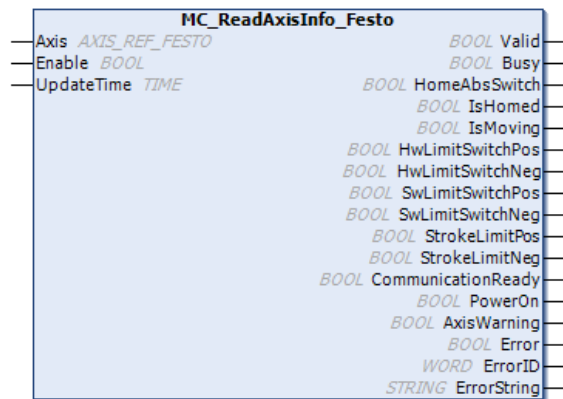


Abbildung 34: MC_ReadAxisInfo_Festo

➔

Hinweis

Dieser Funktionsbaustein bewirkt eine ständige SDO-Kommunikation zwischen Antrieb und PLC. Bei kontinuierlichem Lesen belastet er den Feldbus unnötig stark. Diesen Baustein bitte nur situativ einsetzen. Informationen sind auch über „Eigenschaften“ (siehe Kapitel 4 – Steuerung via Methodenaufruf) abrufbar. Sie belasten den Feldbus nicht zusätzlich.

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Enable	BOOL	Aktueller Status der Achse lesen - TRUE = Status kontinuierlich lesen solange TRUE - FALSE = Lesevorgang deaktivieren
UpdateTime	TIME	Verzögerungszeit zwischen zwei Lesebefehlen des Status-SDOs (Voreinstellung = T#1S = 1 Sekunde)
VAR_OUTPUT		
Valid	BOOL	Status der Ausgänge des Bausteins - TRUE = Status der Ausgänge sind gültig - FALSE = Status der Ausgänge sind nicht gültig
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag aktiv - FALSE = Auftrag abschliessen oder nicht gestartet
HomeAbsSwitch	BOOL	Status des digitalen Referenzschalters
IsHomed	BOOL	Absolute Referenzposition der Achse ist bekannt (referenziert)
IsMoving	BOOL	Achse bewegt sich
HwLimitSwitchPos	BOOL	Positiver Hardware-Endschalter ist belegt
HwLimitSwitchNeg	BOOL	Negativer Hardware-Endschalter ist belegt
SwLimitSwitchPos	BOOL	Positiver Software-Endschalter ist belegt
SwLimitSwitchNeg	BOOL	Negativer Software-Endschalter ist belegt
StrokeLimitPos	BOOL	Positive Hubgrenze ist erreicht
StrokeLimitNeg	BOOL	Negativer Hubgrenze ist erreicht
CommunicationReady	BOOL	Netzwerk ist initialisiert und Kommunikation zur PLC besteht
PowerOn	BOOL	Endstufe ist bestromt
AxisWarning	BOOL	Warnung ist am Motorcontroller vorhanden
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3.25 MC_DeviceService_Festo

Dieser Funktionsblock führt den ausgewählten Gerätedienst des Servoantriebsregler aus.

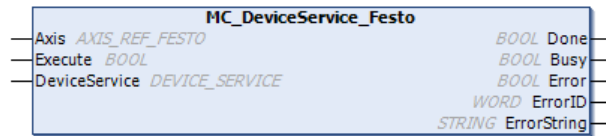


Abbildung 35: MC_DeviceService_Festo

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Gerätedienst anfordern FALSE --> True = Gerätedienst anfordern
DeviceService	DEVICE_SERVICE	Auswahl des Gerätedienst • 00 (none) = keine Auswahl • 01 (SaveZeroPointOffset) = Nullpunktverschiebung im Encoder speichern • 02 (SaveParameterset) = Parametersatz speichern • 03 (ReinitDrive) = Reinitialisierung des Antriebs • 04 (ResetDrive) = Neustart des Antriebs • 05 (OpenHoldingBrake) = Öffnen der Haltebremse • 06 (CloseHoldingBrake) = Schließen der Haltebremse • 07 (StartEventTable) = Eventtabelle starten • 08 (StopEventTable) = Eventtabelle stoppen • 09 (SetLedDeviceIdentification) = LED-Geräteidentifizierung starten • 10 (ResetLedDeviceIdentification) = LED-Geräteidentifizierung stoppen • 11 (ActivateFirmwareUpdate) = Firmware Update starten • 12 (ResetReferencingStatus) = Referenzierungsstatus zurücksetzen • 13 (ActivateFactoryParameter) = Werksparametersatz laden
VAR_OUTPUT		
Done	BOOL	Gerätedienst erfolgreich ausgeführt
Busy	BOOL	Gerätedienst wird angefordert • TRUE = Auftrag gestartet aber noch nicht abgeschlossen • FALSE = Auftrag abschlossen oder nicht gestartet
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten • TRUE = Störung aktiv (siehe Ausgang ErrorID) • FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

3. 26 MC_ForceControl_Festo

Dieser Funktionsblock startet die Kraftregelung mit der Option einer begrenzten Geschwindigkeit. Die Sollkraft wird in Prozent der Nennkraft angegeben. Als Nennkraft ist die Kraft definiert, die der Motor bei Nennmoment erzeugt. Das Getriebe und die Vorschubkonstante werden berücksichtigt, die Reibung wird vernachlässigt. Der Antrieb befindet sich im Kraftbetrieb (ProfileTorqueMode).



Abbildung 36: MC_ForceControl_Festo

➔

Hinweis

Ist einer der Eingänge „NegativeStrokeLimit“ bzw. „PositiveStrokeLimit“ zum Zeitpunkt des Bewegungsstarts = 0, wird für diesen Bewegungsparameter der im Motorcontroller eingestellte Grenzwert verwendet (siehe auch Kapitel 2. 2).

VAR_IN_OUT		
Axis	AXIS_REF_FESTO	Referenzstruktur zur Achse
VAR_INPUT		
Execute	BOOL	Kraftbetrieb starten FALSE --> True = Starte Kraftbetrieb
Force	REAL	Sollkraft in Prozent der Nennkraft (100 = 100 % = Nennkraft).
ContinuousUpdate	BOOL	Kontinuierliche Bewegungsaktualisierung Wenn „ContinuousUpdate“ bei steigender Flanke von „Execute“ TRUE ist, wird bei Änderung der Eingangsparameter unmittelbar ein neuer Bewegungsauftrag gestartet
ForceRamp	REAL	Kraftrampe in Prozent der Nennkraft pro Sekunde (%/s)
Velocity	REAL	Maximale Geschwindigkeit (in Benutzereinheit)
BufferMode	MC_BUFFER_MODE	Definiert die chronische Abfolge des Bausteins (siehe Kapitel 6 – BufferMode)
StrokeLimitation	BOOL	Aktivierung der Hubgrenze TRUE = Hubgrenze ist für den Bewegungsauftrag aktiv
NegativeStrokeLimit	REAL	Negative Hubgrenze (in Benutzereinheit)
PositiveStrokeLimit	REAL	Positive Hubgrenze (in Benutzereinheit)
VAR_OUTPUT		
InForce	BOOL	Zielkraft erreicht
Busy	BOOL	Auftrag ist noch nicht abgeschlossen - TRUE = Auftrag gestartet aber noch nicht abgeschlossen - FALSE = Auftrag abschlossen oder nicht gestartet
Active	BOOL	Baustein besitzt Steuerhoheit über die Achse - TRUE = Steuerhoheit über die Achse vorhanden - FALSE = Keine Steuerhoheit über die Achse
CommandAborted	BOOL	Verfahrenauftrag wurde während der Bearbeitung durch einen anderen Auftrag abgebrochen
Error	BOOL	Während der Bearbeitung ist eine Störung aufgetreten - TRUE = Störung aktiv (siehe Ausgang ErrorID) - FALSE = keine Störung
ErrorID	WORD	Fehlernummer (siehe Kapitel 5 – Diagnose)
ErrorString	STRING	Fehlermeldung anhand der Fehlernummer

4 Steuerung via Methodenaufruf

Zusätzlich zum zyklischen Aufruf der Funktionsbausteine, kann eine Funktion alternativ ebenfalls mittels eines Aufrufs einer Methode ausgeführt werden. Im Gegensatz zur Funktionsausführung durch einen Baustein muss bei einem Methodenaufruf keinerlei Instanziierung erfolgen. Wird der Gerätenamen eingetippt und am Ende ein Punkt gesetzt, erscheinen die zur Verfügung stehenden Methoden in einem Dropdown Fenster (blaues „M“). Eine Methode signalisiert die erfolgreiche Ausführung durch einen boolschen Rückgabewert. Dieser wird im selben SPS-Zyklus gesetzt, in dem die Methode aufgerufen wird. Zusätzlich besitzt jede Methode die Ausgänge (ErrorID und ErrorString). Durch diese können eventuell auftretende Fehler ausgegeben werden (siehe Kapitel 5 – Diagnose). Es muss sichergestellt werden, dass eine Methode nicht zyklisch, sondern nur **einmalig aufgerufen** wird. Weitere Informationen, wie Methoden zu verwenden sind können im Kapitel „Beispiel Methodenaufruf“ oder in der Online Hilfe von Codesys nachgelesen werden. Nachfolgende Tabelle zeigt die zur Verfügung stehenden Methoden inklusive der jeweiligen Übergabeparameter. Eine genaue Erläuterung der einzelnen Übergabeparameter (Datentypen und Bedeutung) ist in der Beschreibung des dazugehörigen Funktionsbausteins zu finden (siehe oben).

Methode(<i>Übergabeparameter</i>)	Beschreibung
DisableDrive()	Endstufe deaktivieren (OFF)
EnableDrive()	Endstufe aktivieren (ON)
Halt()	Aktiven Verfahrtauftrag anhalten (Haltrampe)
ResetHalt()	Aktiven Verfahrtauftrag fortsetzen
Home(<i>Position, HomingMethod</i>)	Referenzfahrt starten
Jog(<i>JogPositive, JogNegative, JogMode</i>)	Bewegung im ProfileJogMode starten (Joggen)
MoveAbsolute(<i>Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode</i>)	Absolute Positionierung starten
MoveAdditive(<i>Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode</i>)	Positionierung relativ zur aktuellen Zielposition starten
MoveRelative(<i>Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode</i>)	Positionierung relativ zur aktuellen Position starten
MoveVelocity(<i>Velocity, Acceleration, Deceleration, Jerk, BufferMode</i>)	Bewegung mit konstanter Geschwindigkeit starten (ProfileVelocityMode)
RecordTable(<i>RecordNumber, BufferMode</i>)	Parametrierter Verfahrssatz starten
Reset()	Fehler des Motorcontrollers zurücksetzen
Stop()	Aktiven Verfahrtauftrag stoppen (Stopprampe)
ResetStop()	Zustandsübergang von „Stopping“ in „Standstill“ (Deaktivierung des Stoppbefehls)
TorqueControl(<i>Torque, TorqueRamp, Velocity, BufferMode</i>)	Bewegung mit konstanter Kraft starten (ProfileTorqueMode)
SaveZeroPointOffset()	Nullpunktverschiebung im Encoder speichern
SaveParameterset()	Parametersatz sichern
ReinitDrive()	Reinitialisierung des Antriebs
ResetDrive()	Reset des Antriebs anfordern
OpenHoldingBrake()	Haltebremse öffnen
CloseHoldingBrake()	Haltebremse schließen

Bei den oben beschriebenen Bausteinen wird eine erfolgreich durchgeführte Aktion meist mit der Ausgangsvariablen „Done“ signalisiert. Im Gegensatz dazu bedeutet der Rückgabewert einer Methode nur, dass die sie erfolgreich gestartet werden konnte. Eine Bewegungsüberwachung, wie zum Beispiel „Ziel erreicht“, muss im Anwenderprogramm erfolgen. Hierzu stehen dem Anwender Eigenschaften zur Verfügung. Diese sind ebenfalls im Dropdown Fenster zu erkennen, wenn der Gerätenamen mit einem anschließenden Punkt eingetippt wird (rotes „E“). Eigenschaften werden vor jedem Aufruf aktualisiert. Weitere allgemeine Informationen zu Eigenschaften sind in der Online Hilfe von Codesys zu finden. Wie eine Eigenschaft verwendet werden kann, wird im Kapitel Beispiel Methodenaufruf gezeigt. Nachfolgende Tabelle zeigt die für Motorcontroller von Festo zur Verfügung stehenden Eigenschaften.

Eigenschaft	Datentyp	Beschreibung
ActualPosition	REAL	Aktuelle Position der Achse (in Benutzereinheiten)
ActualTorque	REAL	Aktuelle Moment der Achse an der Welle (in Nm)
ActualVelocity	REAL	Aktuelle Geschwindigkeit der Achse (in Benutzereinheiten)
HomingValid	BOOL	Achse ist referenziert
TargetReached	BOOL	Aktueller Verfahrauftrag erfolgreich abgeschlossen oder kein Auftrag aktiv
AxisError	BOOL	Fehler ist im Motorcontroller aufgetreten
AxisErrorID	WORD	Fehlernummer
AxisErrorString	STRING	Fehlerbeschreibung anhand der Fehlernummer
DeviceControl	DEVICE_CONTROL	Instanz welche Steuerhoheit besitzt (Feldbus oder Parametriersoftware)
Enabled	BOOL	Endstufe ist bestromt (ON)
SDOsInitialized	BOOL	Antrieb wurde erfolgreich initialisiert
SlaveConnectorFlags	DWORD	Verbindungsparameter des EtherCAT Slaves
AxisWarning	BOOL	Warnung ist am Motorcontroller vorhanden
AxisIsMoving	BOOL	Achse bewegt sich
SetpointAcknowledge	BOOL	Quittierung des Bewegungsauftrags vom Servoantriebsregler

5 Diagnose

Es wird prinzipiell in drei verschiedenen Fehlerkategorien unterschieden: Initialisierungsfehler, Fehler des Servoantriebsregler und Bausteinfehler. Zu unterscheiden ist dies am Prefix in der Ausgangsvariable „ErrorString“ eines Funktionsblocks:

- **Prefix „Init:“ (ErrorID < 10)** = Fehler bei der Initialisierung des EtherCAT Slaves oder der Bibliothek (Diagnose ist in nachfolgender Tabelle aufgelistet.)
- **Prefix „Axis:“ (ErrorID > 10 und < 1000)** = Fehler des Controllers. (Diagnose und Störungsbeseitigung müssen im jeweiligen Gerätehandbuch nachgeschlagen werden.)
- **Prefix „FB:“ (ErrorID > 1000)** = Bausteinfehler (Diagnose und Störungsbeseitigung sind in nachfolgender Tabelle aufgelistet.)

ID (dez)	ID (hex)	Beschreibung
Allgemeine Fehler		
0	0x00	Kein Fehler
Initialisierungsfehler		
1	0x1	Fehler in der Profibibliothek (z. B. CiA402) – ein oder mehrere PDO Pointer konnten nicht dereferenziert werden
2	0x2	Fehler in der Gerätebibliothek (z. B. CMMT-AS) – der richtige EtherCAT Master wurde nicht gefunden
3	0x3	Fehler in der Gerätebibliothek (z. B. CMMT-AS) – der richtige EtherCAT Slave wurde nicht gefunden
4	0x4	Fehler in der Feldbusbibliothek (z. B. EtherCAT) – Lesefehler der SDO Liste des EtherCAT Slaves
5	0x5	Fehler in der Feldbusbibliothek (z. B. EtherCAT) – Lesefehler der Länge der SDO Liste des EtherCAT Slaves (nur bei Beckhoff PLCs)
6	0x6	Feldbusfehler – EtherCAT Slave ist nicht im Status “operational”
7	0x7	Feldbusfehler – Kommunikationsdaten des EtherCAT Slaves sind nicht gültig
8	0x8	Eingang „EC_PDActive“ des Achsbausteins nicht aktiv (Systemvariable „_EC_PDActive“)
Fehler EtherCAT Bibliothek		
1001	0x03E9	ServiceDatenObjekt (SDO) Befehl ist fehlgeschlagen - Fehler bei Schreibzugriff: Prüfen ob ServiceDatenObjekt (SDO) schreibbar ist
1002	0x03EA	Anderer Fehler der EtherCAT Bibliothek
1003	0x03EB	Datenüberlauf in EtherCAT Bibliothek
1004	0x03EC	Zeitüberschreitung in EtherCAT Bibliothek - Gerätestatus des Antriebs prüfen - Kommunikation zwischen PLC und Antrieb prüfen

Fehler in der Kommunikation zwischen PLC und Antrieb		
1050	0x041A	Kommunikationsfehler zwischen PLC und Antrieb – Kommunikation ist nicht aktiv - Zustand des Masters und des Slaves prüfen
1051	0x041B	Kommunikationsfehler zwischen PLC und Antrieb – Busfehler - Prüfen ob Verbindungsleitung gesteckt ist
1052	0x041C	Kommunikationsfehler zwischen PLC und Antrieb – Allgemeiner Fehler
1053	0x041D	Kommunikationsfehler zwischen PLC und Antrieb – Erweiterter Fehler
1054	0x041E	Kommunikationsfehler zwischen PLC und Antrieb – Antrieb befindet sich nicht im Kommunikationsstatus „Operational“
Fehler der PtP Bibliothek		
1100	0x044C	Achse befindet sich für die angeforderte Bewegung im falschen Zustand - Überprüfen Sie den aktuellen Achszustand mit Hilfe der Variablen „AxisState“ - Um die Endstufe freizugeben sollte sich Achse im Zustand 1 (Disabled) befinden - Für Bewegungen sollte sich die Achse im Zustand 2 (Standstill) befinden
1101	0x044D	Achse hat einen Fehler - Fehlerursache beseitigen und mit Hilfe des Bausteins „MC_Reset_Festo“ quittieren
1102	0x044E	Zeitüberschreitung - Angeforderte Funktion konnte nicht in der vorgegebenen Zeit umgesetzt werden - Timeout bei „MC_Power_Festo“ = Prüfen ob Reinitialisierung des Antriebs notwendig ist bzw. prüfen des Achszustands - Timeout bei „MC_Reset_Festo“ = Achsfehler konnte nicht quittiert werden
1103	0x044F	EtherCAT Slave hat keine Verbindung zum Master - Prüfung der Kommunikationsverbindung
1104	0x0450	PLC (Feldbus) hat keine Steuerhoheit - Prüfen ob Parametrierung Steuerhoheit besitzt - Kommunikationsverbindung und Status des Antriebs prüfen
1105	0x0451	Initialisierungsfehler der Bibliothek
1106	0x0452	Angeforderte Aktion wird nicht unterstützt - Funktion wird vom Gerät nicht unterstützt - Bibliotheksreferenzierung fehlerhaft
1107	0x0453	Aktion kann nicht ausgeführt werden, da falsche Startbedingungen vorherrschen
1108	0x0454	Angeforderte Referenzfahrtmethode wird nicht unterstützt - Antrieb unterstützt die angewählte Referenzfahrtmethode nicht
1109	0x0455	Endstufe ist nicht mehr bestromt - Angeforderte Funktion konnte nicht ausgeführt werden oder wurde unterbrochen, da Antrieb nicht mehr bestromt ist
1110	0x0456	Fehler während der Referenzfahrt ist aufgetreten / Referenzfahrt wurde abgebrochen - Diagnosemeldung des Antriebs überprüfen
1111	0x0457	Fehler während des Löschvorgangs eines Bewegungsjobs - Aktiver Bewegungsauftrag oder Bewegungsauftrag in der Warteschlange konnte nicht gelöscht werden

1112	0x0458	Ein zusätzlicher „MC_Halt_Festo“ Funktionsblock hat Steuerhoheit übernommen - Es greifen mindestens zwei unterschiedliche Instanzen des „MC_Halt_Festo“ gleichzeitig auf den Antrieb zu
1113	0x0459	Zu viele Bewegungsaufträge gleichzeitig aktiv - Es wurden zu viele gepufferte Bewegungsaufträge eingelastet
1114	0x045A	Es wurde kein Gerätedienst ausgewählt - Vorhandener Gerätedienst muss ausgewählt werden (siehe Kapitel 3. 25 – MC_DeviceService_Festo)
1115	0x045B	Unbekannter Gerätedienst wurde ausgewählt - Vorhandener Gerätedienst muss ausgewählt werden (siehe Kapitel 3. 25 – MC_DeviceService_Festo)
1116	0x045C	Keine Steuerhoheit über die Bremse möglich - Antrieb ist bestromt, daher keine Steuerhoheit über die Bremse durch PLC Programm vorhanden
1117	0x045D	Fehler konnte nicht quittiert werden - Fehler konnte nicht quittiert werden, bitte Fehlerursache beseitigen
1118	0x045E	SDO nicht schreibbar
1119	0x045F	Buffer Mode „mcBlendingPrevious“ wird nicht unterstützt - BufferMode wechseln, da Blending nur im ProfilePosition Mode (PP) unterstützt wird
Fehler Parameterzugriff (SDO-Zugriff)		
1120	0x0460	Dynamikwerte für eine Bewegung konnten nicht geschrieben - Beschleunigung (Objekt 0x6083), Verzögerung (Objekt 0x6084) oder Ruck (Objekt 0x60A4) konnte nicht geschrieben werden
1121	0x0461	Beschleunigung (Objekt 0x6083) konnte nicht geschrieben werden
1122	0x0462	Verzögerung (Objekt 0x6084) konnte nicht geschrieben werden
1123	0x0463	Ruck (Objekt 0x60A4) konnte nicht geschrieben werden
1124	0x0464	Optionscode für Positionierung (Objekt 0x60F2) konnte nicht geschrieben werden
1125	0x0465	Krafttrampe im Kraftbetrieb (Objekt 0x6087) konnte nicht geschrieben werden
1126	0x0466	Geschwindigkeitsgrenze im Kraftbetrieb (Objekt 0x2060) konnte nicht geschrieben werden
1127	0x0467	Ungültige Parameternummer - Parameternummer 0 kann nicht gelesen/geschrieben werden
1128	0x0468	Fehler während Lesezugriff auf die „Referenzfahrtmethode“ (Objekt 0x6098)
1129	0x0469	Fehler während Schreibzugriff auf die „Referenzfahrtmethode“ (Objekt 0x6098)
1130	0x046A	Fehler während Lesezugriff auf den „Referenz-Offset“ (Objekt 0x607C)
1131	0x046B	Fehler während Schreibzugriff auf den „Referenz-Offset“ (Objekt 0x607C)
1132	0x046C	Fehler während Schreibzugriff auf die "Aktivierung der Hubgrenze" (Objekt 0x216F.15)
1133	0x046D	Fehler während Schreibzugriff auf die "Negative Hubgrenze" (Objekt 0x216F.11)

1134	0x046E	Fehler während Schreibzugriff auf die "Positive Hubgrenze" (Objekt 0x216F.10)
Fehler interner Parameterhandler		
1200	0x04B0	Interne Warteschlange für SDO Befehle ist voll - Es wurden zu viele Lese-/Schreibbefehle angefordert
1201	0x04B1	Datentyp des SDOs nicht identifizierbar - Antrieb konnte keinen Datentyp zurückmelden/finden
1202	0x04B2	Angeforderte Parameternummer nicht vorhanden - Angeforderter Parameternummer befindet sich nicht Objektverzeichnis des Antriebs
1203	0x04B3	Angeforderter Subindex nicht vorhanden - Angeforderter Subindex befindet sich nicht Objektverzeichnis des Antriebs (angeforderter Subindex ist zu hoch)
1204	0x04B4	Fehler beim identifizieren des SDOs - Datentyp konnte vom Antrieb nicht identifiziert werden (einfach)
1205	0x04B5	Angeforderter Subindex nicht vorhanden - Angeforderter Subindex befindet sich nicht Objektverzeichnis des Antriebs (angeforderter Subindex existiert nicht)
1206	0x04B6	Fehler beim identifizieren des SDOs - Datentyp konnte vom Antrieb nicht identifiziert werden (erweitert)
1207	0x04B7	Datentyp des SDOs nicht identifizierbar - Unbekannter Datentyp wurde vom Antrieb zurückgemeldet
1208	0x04B8	Datentyp des Parameters ist kein „STRING“ - Beim zurückgemeldeten Datentyp handelt es sich nicht um ein „STRING“
Fehler im Satzbetrieb		
1300	0x0514	Nächste Verfahrssatznummer konnte nicht geschrieben werden (Objekt 0x216F.20)
Fehler beim Lesen der Parameterdatei		
1400	0x0578	Fehler beim Öffnen der Parameterdatei
1401	0x0579	Fehler beim Lesen der Parameterdatei
1402	0x057A	Gelesener Parameterdateieintrag ist ungültig (SDO Beschreibung ungültig)
1403	0x057B	Gelesener Parameterdateieintrag ist ungültig (Parameternummer = 0)
1404	0x057C	Gelesener Parameterdateieintrag ist ungültig (unbekannter Datentyp)

Hardware Fehler

2048	0x0800	Fehlerbeschreibung wird momentan ausgelesen. Es existiert ein Fehler im Gerät, die dazugehörige Fehlerursache wird momentan ausgelesen. Anmerkung: Es handelt sich um ein azyklisches Servicedatenobjekt (SDO-Befehl). Die Abarbeitung ist typischerweise länger als bei den zyklischen Prozessdaten (PDOs) und kann bis zu 300 ms in Anspruch nehmen. Zum Zeitpunkt zu diesem der Fehler auftrat (Fault-Bit im Statuswort) ist die Fehlerursache noch nicht bekannt und muss ausgelesen werden.
2069	0x0815	Allgemeiner Fehler – Antrieb ist nicht bereit - Der Antrieb hat einen Fehler, es existiert jedoch keine „Emergency Message“ - Fehlerursache beseitigen und Fehler quittieren Anmerkung: Bei der „Emergency Message“ handelt es sich um ein azyklisches Servicedatenobjekt (SDO-Befehl). Die Abarbeitung ist typischerweise länger als bei den zyklischen Prozessdaten (PDOs) und kann bis zu 300 ms in Anspruch nehmen. Eventuell war zum Zeitpunkt zu diesem der Fehler auftrat (Fault-Bit im Statuswort) noch keine dazugehörige „Emergency Message“ vorhanden. Lesen Sie die aktuelle ErrorID und den ErrorString nochmals über den Baustein „MC_ReadAxisError_Festo“ aus. Gibt auch dieser Baustein den gleichen Fehler aus, existiert keine „Emergency Message“.

6 BufferMode

Einige Bausteine haben die Eingangsvariable „BufferMode“. Mit dieser Variablen kann ausgewählt werden, ob ein Baustein entweder gepuffert oder nicht gepuffert gestartet wird. Im nicht gepufferten Modus wird der Baustein unmittelbar gestartet. Ist ein anderer Baustein aktiv, wird dieser abgebrochen. Standardmäßig wird jeder Baustein im nicht gepufferten Modus ausgeführt. Im gepufferten Modus unterbricht der gestartete Baustein einen eventuell aktiven Baustein nicht, er wartet bis dieser abgeschlossen wurde und startet anschließend. Mit dieser Option kann beispielsweise eine Bewegung an eine aktive Bewegung angehängt werden. Folgende Modis wurden umgesetzt:

Wert	MC_BUFFER_MODE	Beschreibung
0	mcAborting	Baustein wird unmittelbar gestartet (Standardeinstellung)
1	mcBuffered	Baustein wird gestartet nachdem aktuelle Bewegung beendet wurde. Es findet kein Verschleifen/Blending statt.
3	mcBlendingPrevious	Baustein wird gestartet, nachdem die aktuelle Bewegung beendet wurde. Es wird mit der Geschwindigkeit der vorangegangenen Bewegung verschliffen.

7 Direction

Der Modulobetrieb vereinfacht die Realisierung getakteter Endlos-Bewegungen, z. B. den Betrieb von Rundschalttischen und Förderbändern. Der Modulobereich wird durch einen Minimalwert und einen Maximalwert beschrieben. Falls ein Sollwert vorgegeben wird, der außerhalb des definierten Modulobereichs liegt, wird nur der Restweg verfahren, der sich aus der Modulodivision ergibt. Zur Aktivierung über das Antriebsprofil müssen folgende Parameter geschrieben werden (beide ungleich 0):

- Oberer Grenzwert Modulo
- Unterer Grenzwert Modulo

Zur Deaktivierung des Modulobetriebs über das Antriebsprofil müssen beide Parameter mit dem Wert 0 beschrieben werden. Bei aktivem Modulobetrieb kann zwischen dem Profil "Position Mode PP" und dem Profil "Velocity Mode PV" gewechselt werden, ohne dass der Modulobetrieb deaktiviert wird. Ist im Antrieb der Modulobetrieb aktiviert, kann die Richtung der Positionierung über den Eingang „Direction“ vorgegeben werden. Dieser ist vom Typ „MC_DIRECTION“ und kann alle Werte der nachfolgenden Tabelle annehmen. Der Eingang ist in den Funktionsbausteinen „MC_MoveAbsolute_Festo“, „MC_MoveRelative_Festo“ und „MC_MoveAdditive_Festo“ verfügbar.



Hinweis
Eine genauere Beschreibung des Modulobetriebs kann im jeweiligen Gerätehandbuch nachgelesen werden.

Wert	MC_DIRECTION	Beschreibung
0	mcNormal	Antrieb positioniert wie bei einer linearen Achse üblich. Achse fährt nicht über den Grenzen der parametrisierten Achse. (Standardeinstellung)
64	mcNegativeDirection	Antrieb positioniert mit Fahrt in negativer Richtung
128	mcPositiveDirection	Antrieb positioniert mit Fahrt in positiver Richtung
192	mcShortestWay	Der Antrieb fährt den kürzesten Weg auf die Zielposition, die sich aus der Modulodivision ergibt. Die Modulogrenzen werden nicht berücksichtigt

8 Visualisierung der Funktionsblöcke

Für jeden Funktionsbaustein steht ein Visualisierungselement zur Verfügung. Dieses soll hauptsächlich die Erstinbetriebnahme des Motorcontrollers unterstützen. Das Visualisierungselement muss mit der jeweiligen zyklisch aufgerufenen Bausteininstanz verknüpft werden. Nachfolgendes Bild zeigt dies beispielhaft am „MC_Power_Festo“.

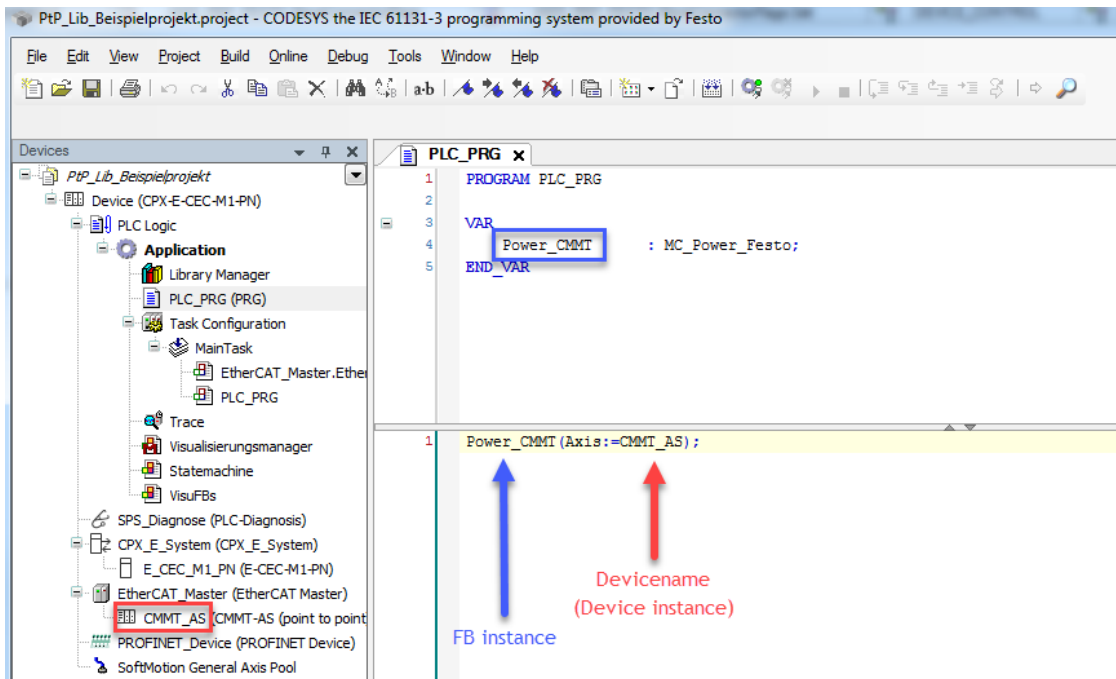


Abbildung 37: Zyklischer Bausteinaufruf

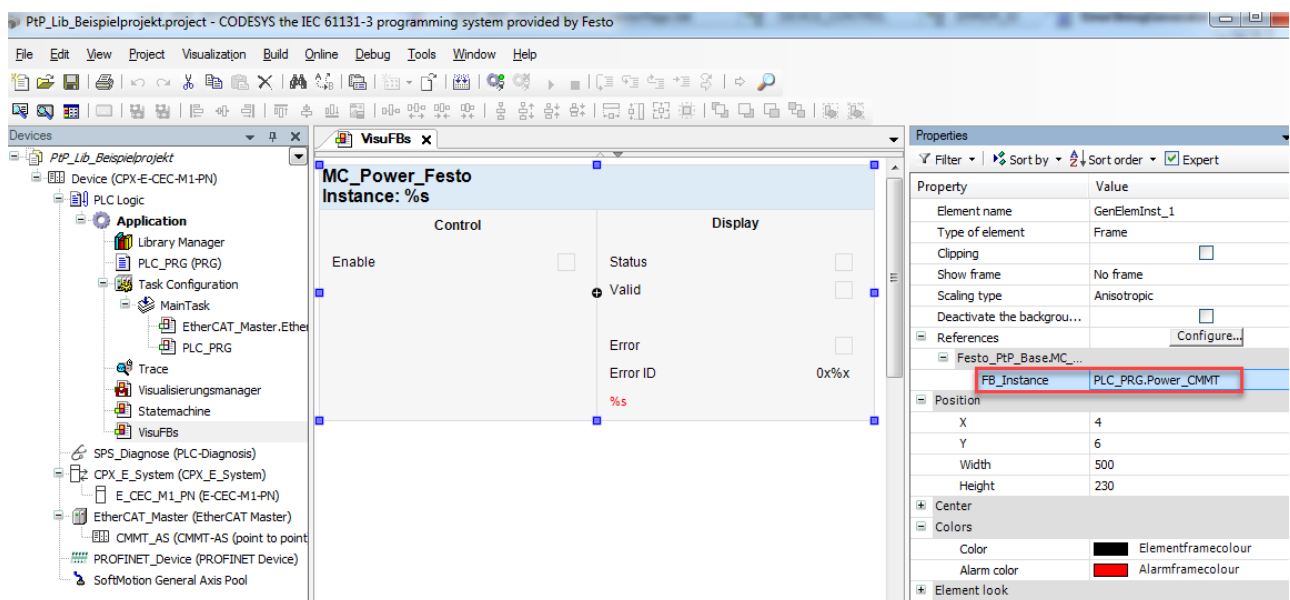


Abbildung 38: Verknüpfung eines Visualisierungselements

9 Anwendungsbeispiele

Wie bereits beschrieben, bestehen zwei Möglichkeiten der Funktionsausführung im Anwenderprogramm. Nachfolgend wird in beiden Varianten beispielhaft gezeigt, wie eine Antriebsfreigabe mit einer anschließenden relativen Bewegung um 100 mm aussehen könnte. Der Antrieb muss dafür referenziert sein.

9.1 Beispiel Funktionsbausteine

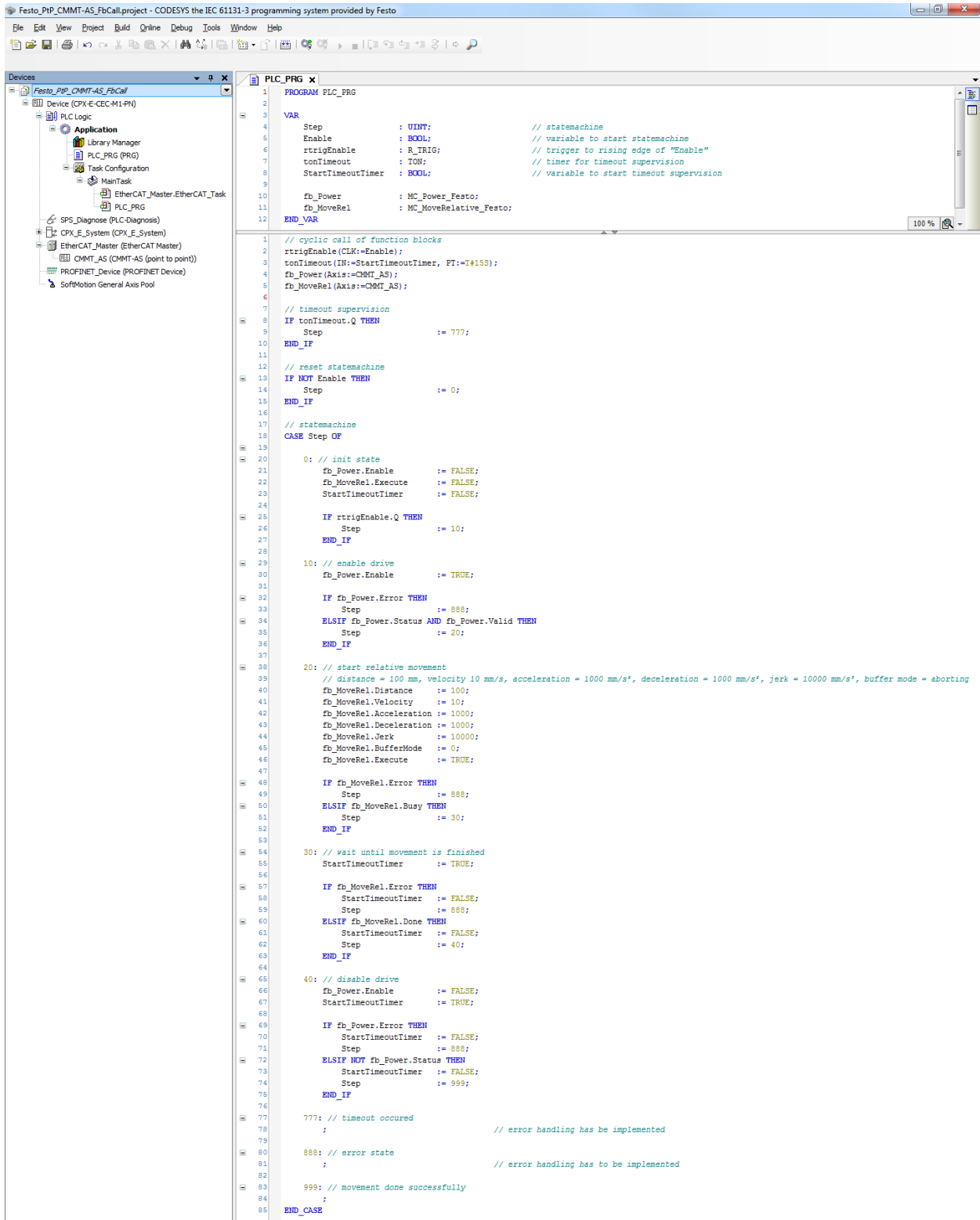


Abbildung 39: Beispiel Bausteinaufruf

9.2 Beispiel Methodenaufruf

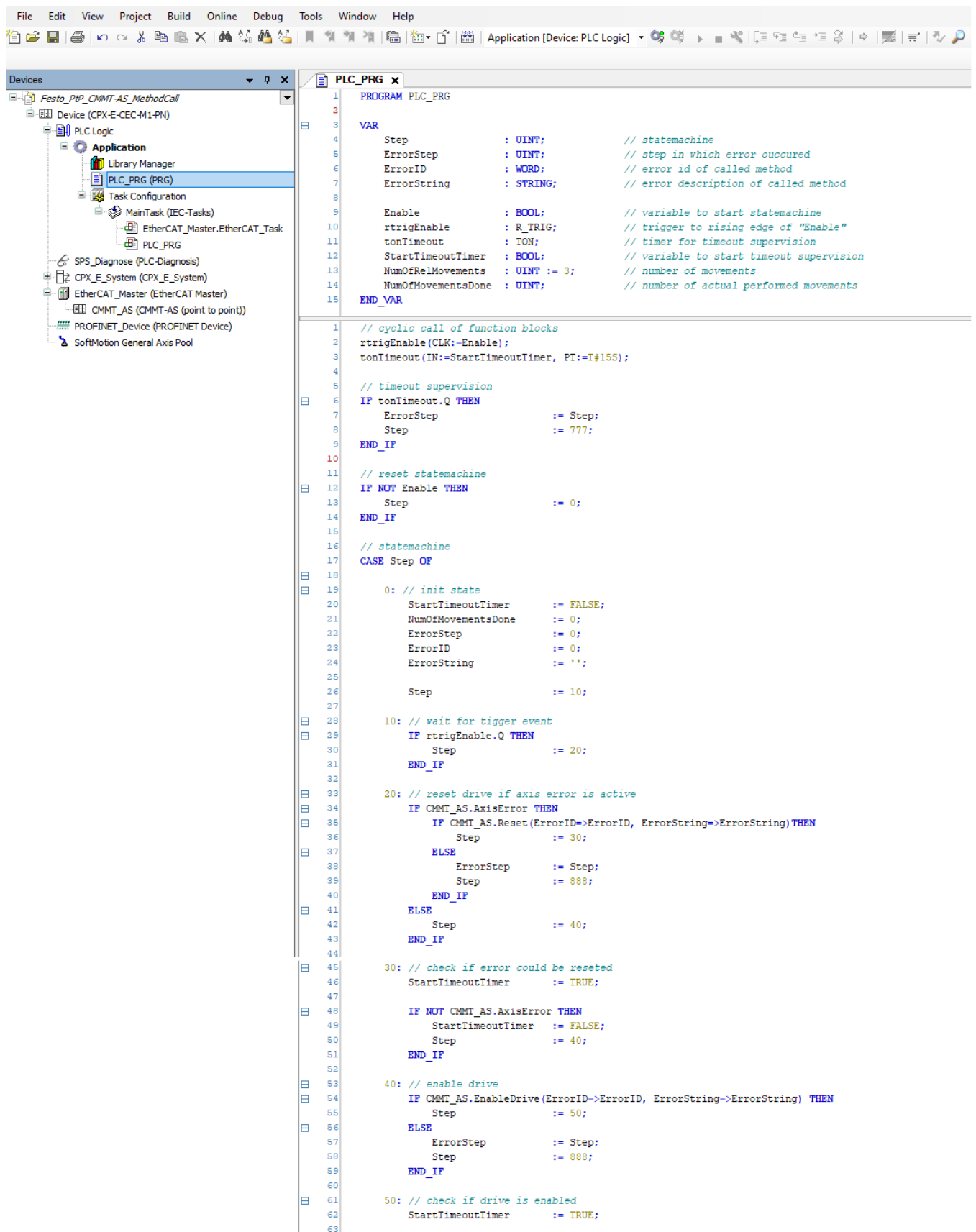


Abbildung 40: Beispiel Methodenaufruf Teil 1

```

64     IF CMMT_AS.Enabled AND NOT CMMT_AS.HomingValid THEN
65         StartTimeoutTimer := FALSE;
66         Step               := 60;
67     ELIF CMMT_AS.Enabled AND CMMT_AS.HomingValid THEN
68         StartTimeoutTimer := FALSE;
69         Step               := 80;
70     END_IF
71
72 60: // start homing as configured in Festo Automation Suite
73     IF CMMT_AS.Home(0, 0, ErrorID=>ErrorID, ErrorString=>ErrorString) THEN
74         Step               := 70;
75     ELSE
76         ErrorStep          := Step;
77         Step               := 888;
78     END_IF
79
80 70: // check if homing is valid
81     StartTimeoutTimer     := TRUE;
82
83     IF CMMT_AS.HomingValid THEN
84         StartTimeoutTimer := FALSE;
85         Step               := 80;
86     END_IF
87
88 80: // start relative movement
89     // distance = 10 mm, velocity 100 mm/s, acceleration = 10000 mm/s², deceleration = 1000 mm/s², jerk = 10000 mm/s³, buffer mode = aborting
90     IF CMMT_AS.MoveRelative(10, 100, 1000, 1000, 10000, 0, 0, ErrorID=>ErrorID, ErrorString=>ErrorString) THEN
91         Step               := 90;
92     ELSE
93         ErrorStep          := Step;
94         Step               := 888;
95     END_IF
96
97 90: // wait until start of movement is acknowledged from drive
98     StartTimeoutTimer     := TRUE;
99
100     IF CMMT_AS.SetpointAcknowledge AND NOT CMMT_AS.TargetReached THEN
101         NumOfMovementsDone := NumOfMovementsDone + 1;
102         StartTimeoutTimer  := FALSE;
103         Step               := 100;
104     END_IF
105
106 100: // wait until movement is finished
107     StartTimeoutTimer     := TRUE;
108
109     IF CMMT_AS.TargetReached THEN
110         StartTimeoutTimer := FALSE;
111         Step               := 110;
112     END_IF
113
114 110: // check number of movements
115     IF NumOfMovementsDone >= NumOfRelMovements THEN
116         Step               := 120;
117     ELSE
118         Step               := 80;
119     END_IF
120
121 120: // disable drive
122     IF CMMT_AS.DisableDrive(ErrorID=>ErrorID, ErrorString=>ErrorString) THEN
123         Step               := 130;
124     ELSE
125         ErrorStep          := Step;
126         Step               := 888;
127     END_IF
128
129 130: // wait until drive is disabled
130     StartTimeoutTimer     := TRUE;
131
132     IF NOT CMMT_AS.Enabled THEN
133         StartTimeoutTimer := FALSE;
134         Step               := 999;
135     END_IF
136
137 777: // timeout occurred
138     ; // error handling has be implemented
139
140 888: // error state
141     ; // error handling has to be implemented
142
143 999: // movements done successfully
144     ;
145 END_CASE

```

Abbildung 4136: Beispiel Methodenaufruf Teil 2