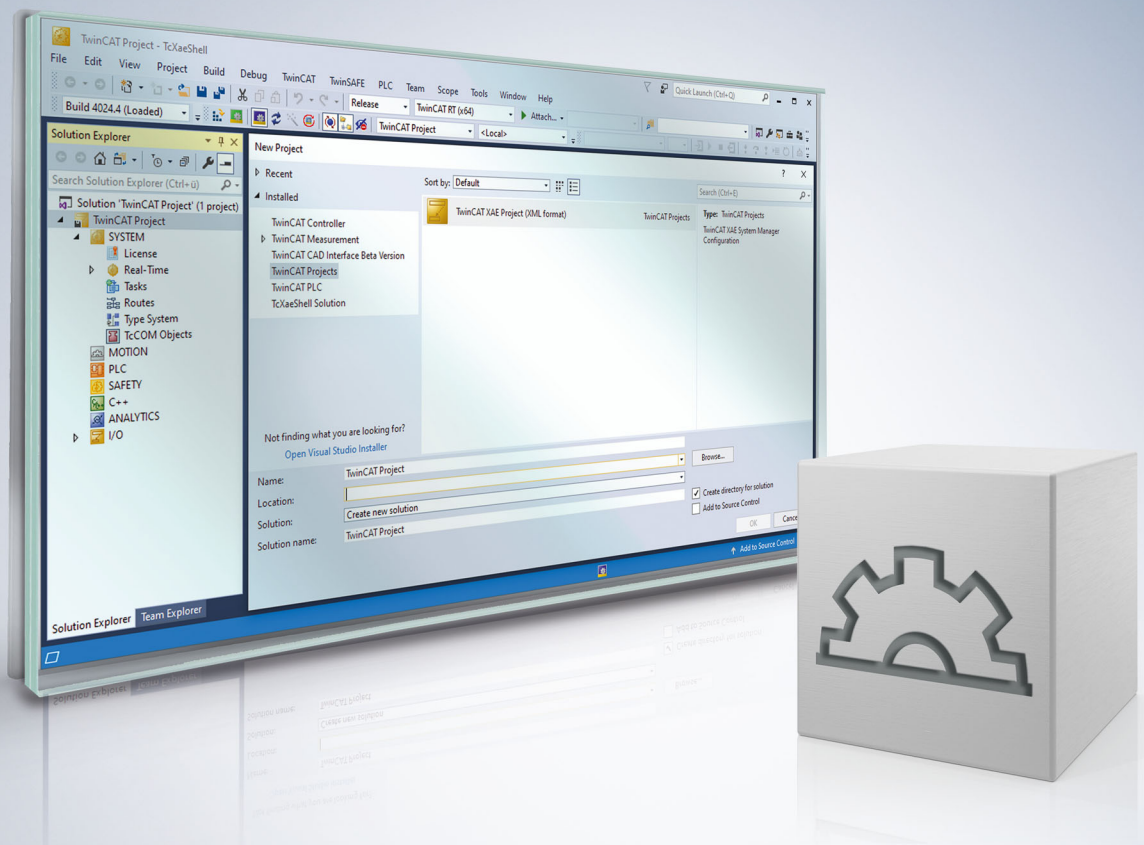
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc2_Drive



13.10.2022 | Version: 1.3.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Die Bibliothek Tc2_Drive sollte in neueren Projekten nicht mehr verwendet werden. Verwenden Sie stattdessen die Bibliothek Tc2_MC2_Drive (siehe Dokumentation TwinCAT 3 PLC Lib Tc2_MC2_Drive).

Die Bibliothek Tc2_Drive umfasst Funktionen und Funktionsbausteine für SoE-Antriebe, die über eine Drive-Referenz auf den Antrieb zugreifen.

Drive-Bibliotheken

Die drei Drive-Bibliotheken Tc2_Drive, Tc2_NcDrive und Tc2_MC2_Drive sind für unterschiedliche Funktionszwecke entwickelt worden, sind in ihrem Funktionsumfang aber nahezu identisch. Die Bausteine der Bibliotheken Tc2_NcDrive und Tc_MC2_Drive bilden Wrapper-Bausteine um die Bausteine der Bibliothek Tc2_Drive.

Drive-Bibliothek	Verwendung	Zugriff auf den Antrieb	Hinweis
Tc2_Drive Siehe: Dokumentation TwinCAT 3 PLC Lib: Tc2_Drive	Verwenden Sie die Bibliothek Tc2_Drive, wenn Sie den Antrieb komplett aus der SPS heraus verwenden (also ohne NC).	Auf den Antrieb wird über eine Drive-Referenz zugegriffen. Bibliotheksintern wird dafür die Struktur ST_DriveRef mit der NetID als String verwendet. Zu Verlinkungszwecken wird zusätzlich eine Struktur ST_PlcDriveRef mit der NetID als Bytearray zur Verfügung gestellt. (Siehe Drive-Referenz ST_DriveRef [▶ 11])	Wenn Sie auf Parameter im Antrieb zugreifen wollen, für die kein spezieller Baustein implementiert wurde, verwenden Sie die Bausteine FB_SoERead_ByDriveRef und FB_SoEWrite_ByDriveRef. Diese Funktionsbausteine sind in der PLC Lib Tc2_EtherCAT im Ordner SoE Interface implementiert.
Tc2_NcDrive Siehe: Dokumentation TwinCAT 3 PLC Lib: Tc2_NcDrive	Verwenden Sie die Bibliothek Tc2_NcDrive, wenn Sie den Antrieb über die NC mit der Bibliotheken Tc2_Nc verwenden.	Auf den Antrieb wird über die NC-Achsstruktur (NC_TO_PLC) zugegriffen. Die Bausteine der Bibliothek Tc2_NcDrive ermitteln eigenständig über die NC-AchsID aus der NC-Achsstruktur die Zugriffsdaten auf den Antrieb (NetID, Adresse und Kanalnummer).	Wenn Sie auf Parameter im Antrieb zugreifen wollen, für die kein spezieller Baustein implementiert wurde, verwenden Sie die Bausteine FB_SoERead und FB_SoEWrite.
Tc2_MC2_Drive Siehe: Dokumentation TwinCAT 3 PLC Lib Tc2_MC2_Drive	Verwenden Sie die Bibliothek Tc2_MC2_Drive, wenn Sie den Antrieb über NC mit der Bibliothek Tc2_MC2 verwenden.	Auf den Antrieb wird über die MC2-Achsreferenz (AXIS_REF) zugegriffen. Die Bausteine der Bibliothek Tc2_MC2_Drive ermitteln eigenständig über die NC-AchsID aus der MC2-Achsreferenz die Zugriffsdaten auf den Antrieb (NetID, Adresse und Kanalnummer).	Wenn Sie auf Parameter im Antrieb zugreifen wollen, für die kein spezieller Baustein implementiert wurde, verwenden Sie die Bausteine FB_SoERead und FB_SoEWrite.

Beachten Sie die Unterschiede bei der Verwendung der Drive-Bibliotheken mit AX5000 und Bosch Rexroth IndraDrive CS (siehe Beispiele [▶ 56])

Funktionen

Name	Beschreibung
F_GetVersionTcDrive [▶ 40]	Liest Versionsinformationen der SPS-Bibliothek aus. Die Funktion wird durch die globale Struktur stLibVersion_Tc2_Drive ersetzt.

Name	Beschreibung
F_ConvWordToSTAX5000C1D [▶ 32]	Konvertiert das C1D-Wort (S-0-0011) des AX5000 in eine Struktur ST_AX5000_C1D [▶ 49]
F_ConvWordToSTAX5000DriveStatus [▶ 32]	Konvertiert das Antriebsstatuswort (S-0-0135) des AX5000 in eine Struktur ST_AX5000DriveStatus [▶ 50]
F_ConvWordToSTIndraDriveCsC1D [▶ 39]	Konvertiert das C1D-Wort (S-0-0011) des IndraDrive Cs in eine Struktur ST_IndraDriveCs_C1D [▶ 51]
F_ConvWordToSTIndraDriveCsDriveStatus [▶ 40]	Konvertiert das Antriebsstatuswort (S-0-0135) des IndraDrive Cs in eine Struktur ST_IndraDriveCsDriveStatus [▶ 52]

Funktionsbausteine

Name	Beschreibung
FB_SoEReset_ByDriveRef [▶ 13]	Führt ein Reset des Antriebs aus (S-0-0099).
FB_SoEWritePassword_ByDriveRef [▶ 14]	Setzt das Antriebspasswort (S-0-0267).
FB_SoEExecuteCommand_ByDriveRef [▶ 15]	Führt ein Kommando aus.
FB_SoEReadDiagMessage_ByDriveRef [▶ 20]	Liest die Diagnosenachricht (S-0-0095).
FB_SoEReadDiagNumber_ByDriveRef [▶ 21]	Liest die Diagnosenummer (S-0-0390).
FB_SoEReadDiagNumberList_ByDriveRef [▶ 22]	Liest die Diagnosenummernliste (bis zu 30 Einträge) (S-0-0375).
FB_SoEReadClassXDiag_ByDriveRef [▶ 24]	Liest die Class-1-Diagnose (S-0-0011) ... Class-3-Diagnose (S-0-0013).
FB_SoEWriteCommandControl_ByDriveRef [▶ 17]	Setzt den Command Control.
FB_SoEReadCommandState_ByDriveRef [▶ 18]	Prüft den Kommandostatus.
FB_SoERead_ByDriveRef	Liest einen Parameter (siehe PLC Lib Tc2_EtherCAT).
FB_SoEWrite_ByDriveRef	Schreibt einen Parameter (siehe PLC Lib Tc2_EtherCAT).
FB_SoEReadAmplifierTemperature_ByDriveRef [▶ 26]	Liest die Antriebstemperatur (S-0-0384).
FB_SoEReadMotorTemperature_ByDriveRef [▶ 27]	Liest die Motortemperatur (S-0-0383).
FB_SoEReadDcBusCurrent_ByDriveRef [▶ 30]	Liest den Dc-Bus-Strom (S-0-0381).
FB_SoEReadDcBusVoltage_ByDriveRef [▶ 29]	Liest die Dc-Bus-Spannung (S-0-0380).
FB_SoEAX5000ReadActMainVoltage_ByDriveRef [▶ 33]	Liest die Netzspannung (P-0-0200).
FB_SoEAX5000SetMotorCtrlWord_ByDriveRef [▶ 34]	Setzt das Motor Control Word (P-0-0096).
FB_SoEAX5000FirmwareUpdate_ByDriveRef [▶ 36]	Führt ein automatisches Firmware-Update für den AX5000 aus.
FB_CoEDriveEnable [▶ 41]	Gibt einen CoE-Antrieb frei.
FB_CoEDriveMoveVelocity [▶ 42]	Erzeugt ein einfaches Dreiphasen-Geschwindigkeitsprofil, mit dem ein CoE-Antrieb direkt versorgt werden kann.
FB_SoEDriveEnable [▶ 43]	Gibt einen SoE-Antrieb frei.

Name	Beschreibung
FB_SoEDriveMoveVelocity [▶ 44]	Erzeugt ein einfaches Dreiphasen-Geschwindigkeitsprofil, mit dem ein SoE-Antrieb direkt versorgt werden kann.

Voraussetzungen

Komponente	Version
TwinCAT auf dem Entwicklungsrechner	3.1 Build 4016 oder höher
TwinCAT auf dem Windows CE-Image	3.1 Build 4016 oder höher
TwinCAT auf dem Windows XP-Image	3.1 Build 4016 oder höher

3 Drive-Referenz ST_DriveRef

Auf den Antrieb wird über eine Drive-Referenz zugegriffen. Bibliotheksintern wird dafür die Struktur ST_DriveRef aus der Tc2_EtherCAT Bibliothek mit der NetID als String verwendet. Da auf I/O-Ebene die NetID üblicherweise als Bytearray vorliegt, wird zusätzlich eine Struktur ST_PlcDriveRef, auch aus der Tc2_EtherCAT Bibliothek, mit der NetID als Bytearray zur Verfügung gestellt. Die beiden Strukturen müssen ineinander überführt werden.

Struktur ST_PlcDriveRef

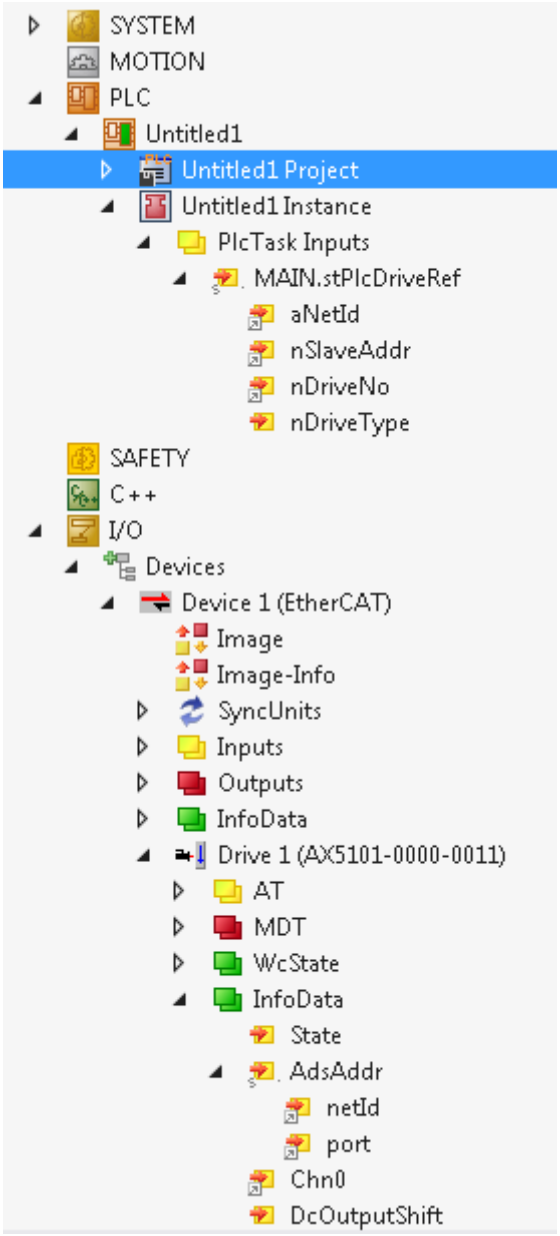
```
TYPE ST_PlcDriveRef :
STRUCT
    aNetId     : T_AmsNetIdArr; (* AmsNetId (array[0..5] of bytes) of the EtherCAT master device.*)
    nSlaveAddr : UINT; (* Address of the slave device.*)
    nDriveNo   : BYTE; (* Drive number*)
    nDriveType : BYTE; (* Drive type*)
END_STRUCT
END_TYPE
```

Struktur ST_DriveRef

```
TYPE ST_DriveRef :
STRUCT
    sNetId     : T_AmsNetId; (* AmsNetId (string(23)) of the EtherCAT master device.*)
    nSlaveAddr : UINT; (* Address of the slave device.*)
    nDriveNo   : BYTE; (* Drive number*)
    nDriveType : BYTE; (* Drive type*)
END_STRUCT
END_TYPE
```

Mapping der Antriebsreferenz in die SPS

Die Antriebsreferenz kann im System Manager in die SPS gemappt werden. Lokieren Sie dazu eine Instanz der Struktur ST_PlcDriveRef als AT %I*. Verlinken Sie anschließend aNetID auf netId, nSlaveAddr auf port und nDriveNo auf Chn0 (A) bzw. Chn1 (B). Bei mehrkanaligen Antrieben beziehen sich beide Kanäle auf dieselbe netId und port-Nummer, da es sich um einen EtherCAT-Slave handelt.



Überführung ST_PlcDriveRef und ST_DriveRef

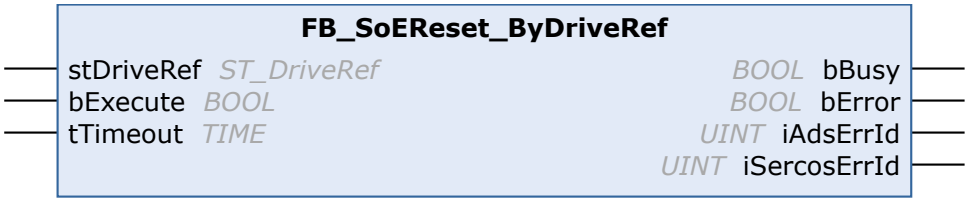
Die Bausteine der Bibliothek Tc2_Drive verwenden eine Instanz der Struktur ST_DriveRef. Im Unterschied zu der Struktur ST_PlcDriveRef wird die NetID als T_AmsNetId erwartet (also als STRING(23)). Zum Wandeln des Bytearrays verwenden Sie die Funktion F_CreateAmsNetId() der PLC Lib Tc2_System.

```
stDriveRef.sNetId     := F_CreateAmsNetId(stPlcDriveRef.aNetId);
stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
stDriveRef.nDriveNo   := stPlcDriveRef.nDriveNo;
stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
```

4 Funktionsbausteine

4.1 Allgemein SoE

4.1.1 FB_SoEReset_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReset_ByDriveRef kann ein Reset des Antriebs (S-0-0099) ausgeführt werden. Bei mehrkanaligen Geräten müssen ggf. beide Kanäle einen Reset ausführen. Die Timeout-Zeit muss 10 s betragen, da der Reset je nach Fehler bis zu 10 s dauern kann. Ein NC-Reset wird nicht ausgeführt.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := T#10s;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit (10 s), die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

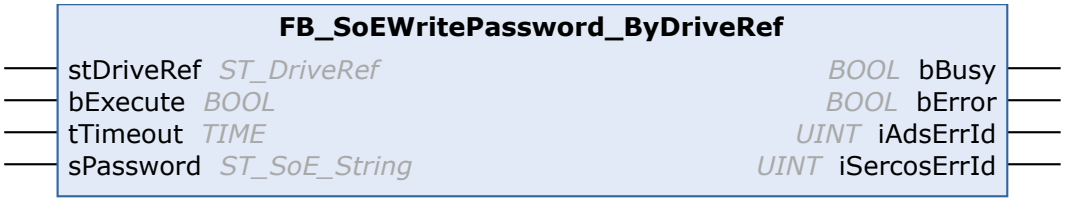
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.

Beispiel

```
fbSoEReset : FB_SoEReset_ByDriveRef;
bSoEReset : BOOL;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;
```

```
IF bInit THEN
   stDriveRef.sNetId  := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
     bInit := FALSE;
   END_IF
END_IF

IF bSoEReset AND NOT bInit THEN
  fbSoEReset(
    stDriveRef := stDriveRef,
    bExecute := TRUE,
    tTimeout := DEFAULT_ADS_TIMEOUT,
  );
  IF NOT fbSoEReset.bBusy THEN
    fbSoEReset(stDriveRef := stDriveRef, bExecute := FALSE);
    bSoEReset := FALSE;
   END_IF
END_IF
```

4.1.2 FB_SoEWritePassword_ByDriveRef



Mit dem Funktionsbaustein FB_SoEWritePassword_ByDriveRef kann das Antriebspasswort (S-0-0267) gesetzt werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;sPassword  : ST_SoE_String;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit (10 s), die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.
sPassword	ST_SoE_String	Passwort als Sercos-String

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
```

```
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.

Beispiel

```
fbWritePassword : FB_SoEWritePassword_ByDriveRef;
bWritePassword  : BOOL;
sPassword : ST_SoE_String;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bWritePassword AND NOT bInit THEN
   fbWritePassword(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     sPassword := sPassword
   );
   IF NOT fbWritePassword.bBusy THEN
     fbWritePassword(stDriveRef := stDriveRef, bExecute := FALSE);
     bWritePassword := FALSE;
   END_IF
END_IF
```

4.1.3 FB_SoEExecuteCommand_ByDriveRef

Mit dem Funktionsbaustein FB_SoEExecuteCommand_ByDriveRef kann ein Kommando ausgeführt werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    nIdn       : WORD;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
nIdn	WORD	Parameternummer, auf das sich das FB_SoEExecuteCommand_ByDriveRef bezieht, z. B. „P_0_IDN + 160“ für P-0-0160
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit (10 s), die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

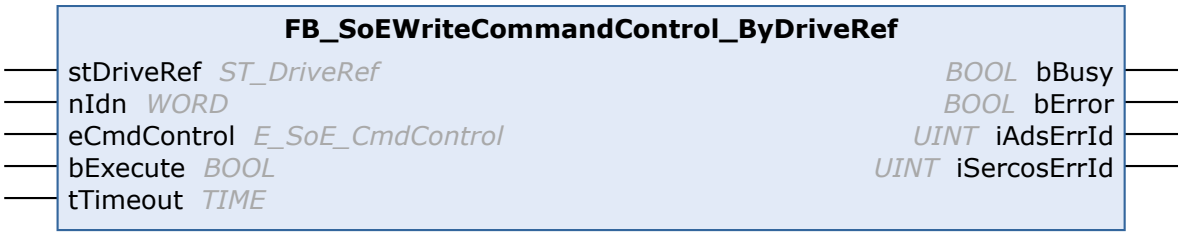
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.

Beispiel

```
fbExecuteCommand : FB_SoEExecuteCommand_ByDriveRef;
bExecuteCommand  : BOOL;
nIdn : WORD;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
     bInit := FALSE;
   END_IF
END_IF

IF bExecuteCommand AND NOT bInit THEN
  nIdn := P_0_IDN + 160;
  fbExecuteCommand(
    stDriveRef := stDriveRef,
    bExecute := TRUE,
    tTimeout := DEFAULT_ADS_TIMEOUT,
    nIdn := nIdn,
  );
  IF NOT fbExecuteCommand.bBusy THEN
    fbExecuteCommand(stDriveRef := stDriveRef, bExecute := FALSE);
    bExecuteCommand := FALSE;
  END_IF
END_IF
```

4.1.4 Funktionsbausteine für Kommandos

4.1.4.1 FB_SoEWriteCommandControl_ByDriveRef



Mit dem Funktionsbaustein FB_SoEWriteCommandControl_ByDriveRef kann ein Kommando vorbereitet, gestartet oder abgebrochen werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    nIdn       : WORD;
    eCmdControl: E_SoE_CmdControl;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
nIdn	WORD	Parameternummer, auf die sich FB_SoEReadCommandState_ByDriveRef bezieht, z. B. „P_0_IDN + 23“ für P-0-0023.
eCmdControl	E_SoE_CmdControl	Gibt an, ob das Kommando vorbereitet (eSoE_CmdControl_Set := 1), ausgeführt (eSoE_CmdControl_SetAndEnable := 3) oder abgebrochen (eSoE_CmdControl_Cancel := 0) werden soll.
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit (10 s), die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.

Name	Typ	Beschreibung
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.

Beispiel

```
fbWriteCommandControl : FB_SoEWriteCommandControl_ByDriveRef;
bWriteCommandControl  : BOOL;
nIdn                  : WORD;
eCmdControl           : E_SoE_CmdControl;

stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef           : ST_DriveRef;
IF bInit THEN
    stDriveRef.sNetId     := F_CreateAmsNetId(stPlcDriveRef.aNetId);
    stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
    stDriveRef.nDriveNo   := stPlcDriveRef.nDriveNo;
    stDriveRef.nDriveType := stPlcDriveRef.nDriveType;

    IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
        bInit := FALSE;
    END_IF
END_IF

IF bWriteCommandControl AND NOT bInit THEN
    nIdn := P_0_IDN + 160;
    fbWriteCommandControl(
        stDriveRef := stDriveRef,
        bExecute := TRUE,
        tTimeout := DEFAULT_ADS_TIMEOUT,
        nIdn := nIdn,
        eCmdControl := eCmdControl
    );
    IF NOT fbWriteCommandControl.bBusy THEN
        fbWriteCommandControl(stDriveRef := stDriveRef, bExecute := FALSE);
        bWriteCommandControl := FALSE;
    END_IF
END_IF
```

4.1.4.2 FB_SoEReadCommandState_ByDriveRef

Mit dem Funktionsbaustein FB_SoEReadCommandState_ByDriveRef kann die Kommandoausführung überprüft werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    Idn        : WORD;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])

Name	Typ	Beschreibung
nIdn	WORD	Parameternummer, auf die sich FB_SoEReadCommandState_ByDriveRef bezieht, z. B. „P_0_IDN + 160“ für P-0-0160
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit (10 s), die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    eCmdState    : E_SoE_CmdState;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

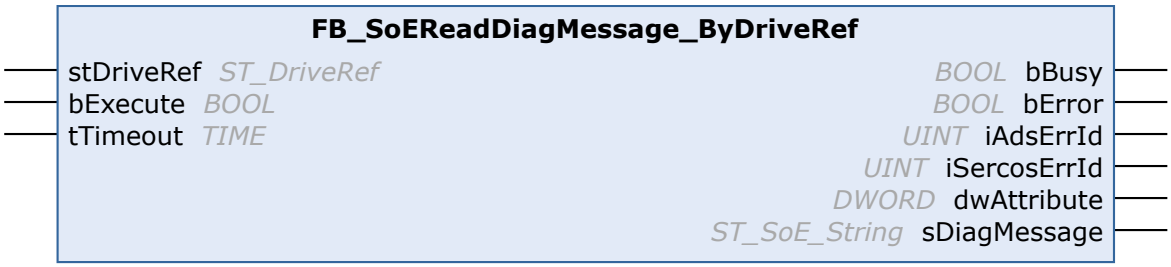
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
eCmdState	eSoE_CmdState	Liefert den Kommandostatus [▶ 53].
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.

Beispiel

```
fbReadCommandState : FB_SoEReadCommandState_ByDriveRef;
bReadCommandState  : BOOL;
nIdn : WORD;
eCmdState : E_SoE_CmdState;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
     bInit := FALSE;
   END_IF
END_IF

IF bReadCommandState AND NOT bInit THEN
   nIdn := P_0_IDN + 160;
   fbReadCommandState(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     nIdn := nIdn,
     eCmdState => eCmdState
   );
   IF NOT fbReadCommandState.bBusy THEN
      fbReadCommandState(stDriveRef := stDriveRef, bExecute := FALSE);
      bReadCommandState := FALSE;
   END_IF
END_IF
```

4.1.5 Funktionsbausteine für die Diagnose

4.1.5.1 FB_SoEReadDiagMessage_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadDiagMessage_ByDriveRef kann die Diagnosenachricht als Sercos-String (S-0-0095) ausgelesen werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;ND_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

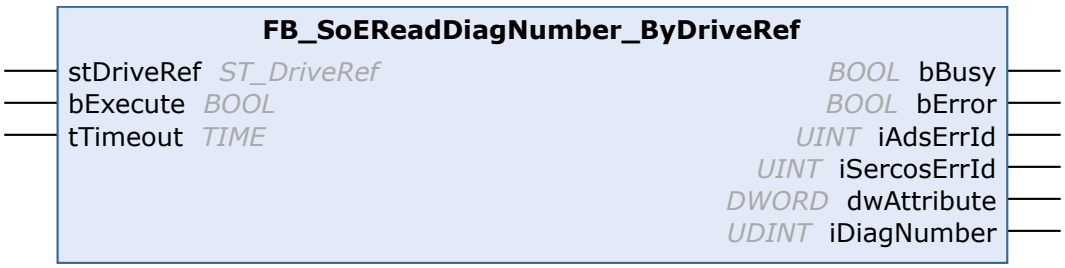
```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
    dwAttribute  : DWORD;
    sDiagMessage : ST_SoE_String;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.
sDiagMessage	ST_SoE_String	Liefert die Diagnosenachricht.

Beispiel

```
fbDiagMessage : FB_SoEReadDiagMessage_ByDriveRef;
bDiagMessage  : BOOL;
sDiagMessage  : ST_SoE_String;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bDiagMessage AND NOT bInit THEN
   fbDiagMessage(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     sDiagMessage => sDiagMessage
   );
   IF NOT fbDiagMessage.bBusy THEN
      fbDiagMessage(
        stDriveRef:= stDriveRef,
        bExecute := FALSE
      );
      bDiagMessage := FALSE;
   END_IF
END_IF
```

4.1.5.2 FB_SoEReadDiagNumber_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadDiagNumber_ByDriveRef kann die aktuelle Diagnosenummer als UDINT (S-0-0390) ausgelesen werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
    dwAttribute  : DWORD;
    iDiagNumber  : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.
iDiagNumber	UDINT	Liefert die aktuelle Diagnosenummer.

Beispiel

```
fbDiagNumber : FB_SoEReadDiagNumber_ByDriveRef;
bDiagNumber  : BOOL;
iDiagNumber  : UDINT;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bDiagNumber AND NOT bInit THEN
   fbDiagNumber(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     iDiagNumber => iDiagNumber
   );
   IF NOT fbDiagNumber.bBusy THEN
      fbDiagNumber(stDriveRef := stDriveRef, bExecute := FALSE);
      bDiagNumber := FALSE;
   END_IF
END_IF
```

4.1.5.3 FB_SoEReadDiagNumberList_ByDriveRef

Mit dem Funktionsbaustein FB_SoEReadDiagNumberList_ByDriveRef kann eine Historie der Diagnosenummern als Liste (S-0-0375) ausgelesen werden.

Eingänge

```
VAR_INPUT
    stDriveRef    : ST_DriveRef;
    bExecute      : BOOL;
    tTimeout      : TIME := DEFAULT_ADS_TIMEOUT;
    piDiagNumber : POINTER TO ST_SoE_DiagNumList;
    iSize         : UDINT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.
piDiagNumber	POINTER TO ST_SoE_DiagNumList	Zeiger auf die Liste der letzten max. 30 Fehlernummern. Die Liste besteht aus der aktuellen und maximalen Anzahl von Bytes in der Liste sowie den 30 Listeneinträgen.
iSize	UDINT	Größe der Liste in Bytes (als Sizeof())

Ausgänge

```
VAR_OUTPUT
    bBusy         : BOOL;
    bError        : BOOL;
    iAdsErrId     : UINT;
    iSercosErrId : UINT;
    dwAttribute   : DWORD;
END_VAR
```

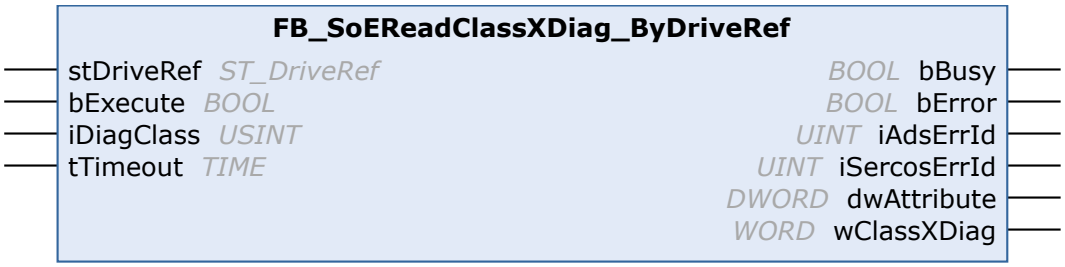
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.

Beispiel

```
fbDiagNumberList : FB_SoEReadDiagNumberList_ByDriveRef;
bDiagNumberList : BOOL;
stDiagNumberList : ST_SoE_DiagNumList;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
```

```
       bInit := FALSE;
   END_IF
END_IF

IF bDiagNumberList AND NOT bInit THEN
   fbDiagNumberList(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     piDiagNumber:= ADR(stDiagNumberList),
     iSize := SIZEOF(stDiagNumberList),
   );
   IF NOT fbDiagNumberList.bBusy THEN
      fbDiagNumberList(stDriveRef := stDriveRef, bExecute := FALSE);
      bDiagNumberList := FALSE;
   END_IF
END_IF
```

4.1.5.4 FB_SoEReadClassXDiag_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadClassXDiag_ByDriveRef kann die aktuelle Class-1-Diagnose (S-0-0011) ... Class-3-Diagnose (S-0-0013) als WORD ausgelesen werden. Für die Auswertung der Class-1-Diagnose als Struktur ST_AX5000_C1D [▶ 49] gibt es die Konvertierungsfunktion F_ConvWordToSTAX5000C1D [▶ 32].

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    iDiagClass : USINT:= 1; (* 1: C1D (S-0-0011) is default, 2: C2D (S-0-0012), 3: C3D (S-0-0013) *)
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
iDiagClass	USINT	Gibt an, welche Diagnose gelesen werden soll. Die Diagnoseparameter können sich von Hersteller zu Hersteller unterscheiden. Nicht immer sind alle Diagnoseparameter (C1D ... C3D) oder alle Bits darin implementiert. 1: Fehler: Class 1 Diag (S-0-0011) 2: Warnungen: Class 2 Diag (S-0-0012) 3: Informationen: Class 3 Diag (S-0-0013)
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
    dwAttribute  : DWORD;
    wClassXDiag  : WORD;
END_VAR
```

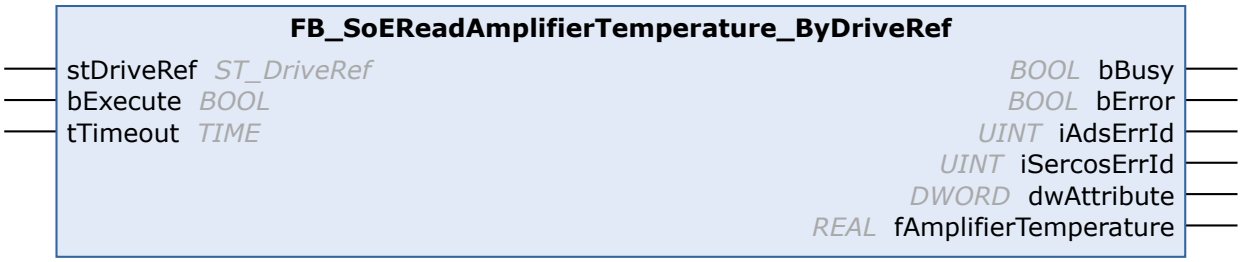
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.
wClassXDiag	WORD	Liefert die aktuelle Class-X-Diagnose.

Beispiel

```
fbClassXDiag : FB_SoEReadClassXDiag_ByDriveRef;
bClassXDiag  : BOOL;
iDiagClass   : USINT := 1;
wClass1Diag  : WORD;
stAX5000C1D  : ST_AX5000_C1D;
wClass2Diag  : WORD;
bInit : BOOL := TRUE;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bClassXDiag AND NOT bInit THEN
   fbClassXDiag(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     iDiagClass := iDiagClass,
     tTimeout   := DEFAULT_ADS_TIMEOUT
   );
   IF NOT fbClassXDiag.bBusy THEN
     fbClassXDiag(stDriveRef := stDriveRef, bExecute := FALSE);
     bClassXDiag := FALSE;
     CASE fbClassXDiag.iDiagClass OF
       1:
         wClass1Diag := fbClassXDiag.wClassXDiag;
         stAX5000C1D := F_ConvWordToSTAX5000C1D(wClass1Diag);
       2:
          wClass2Diag := fbClassXDiag.wClassXDiag;
     END_CASE
   END_IF
END_IF
```

4.1.6 Funktionsbausteine für die Ermittlung aktueller Werte

4.1.6.1 FB_SoEReadAmplifierTemperature_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadAmplifierTemperature_ByDriveRef kann die Temperatur des Antriebs (S-0-0384) eingelesen werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy                : BOOL;
    bError               : BOOL;
    iAdsErrId            : UINT;
    iSercosErrId         : UINT;
    dwAttribute          : DWORD;
    fAmplifierTemperature : REAL;
END_VAR
```

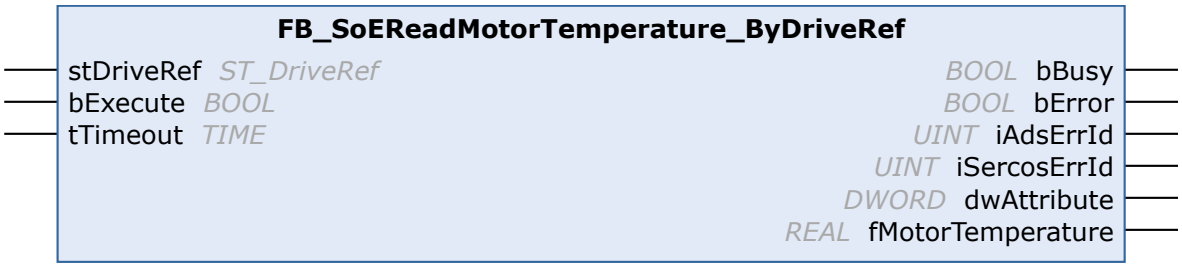
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.

Name	Typ	Beschreibung
fAmplifierTemperature	REAL	Liefert die Antriebstemperatur (z. B. 26.2 entspricht 26.2 °C).

Beispiel

```
fbExecuteCommand : FB_SoEExecuteCommand_ByDriveRef;
bExecuteCommand  : BOOL;
nIdn             : WORD;

stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef           : ST_DriveRef;

IF bInit THEN
    stDriveRef.sNetId     := F_CreateAmsNetId(stPlcDriveRef.aNetId);
    stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
    stDriveRef.nDriveNo   := stPlcDriveRef.nDriveNo;
    stDriveRef.nDriveType := stPlcDriveRef.nDriveType;

    IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
        bInit := FALSE;
    END_IF
END_IF

IF bExecuteCommand AND NOT bInit THEN
    nIdn := P_0_IDN + 160;
    fbExecuteCommand(
        stDriveRef := stDriveRef,
        bExecute := TRUE,
        tTimeout := DEFAULT_ADS_TIMEOUT,
        nIdn := nIdn,
    );
    IF NOT fbExecuteCommand.bBusy THEN
        fbExecuteCommand(stDriveRef := stDriveRef, bExecute := FALSE);
        bExecuteCommand := FALSE;
    END_IF
END_IF
```

4.1.6.2 FB_SoEReadMotorTemperature_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadMotorTemperature_ByDriveRef kann die Temperatur des Motors (S-0-0383) eingelesen werden. Wenn der Motor keinen Temperatursensor enthält, steht hier 0.0, d. h. 0.0 °C.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])

Name	Typ	Beschreibung
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

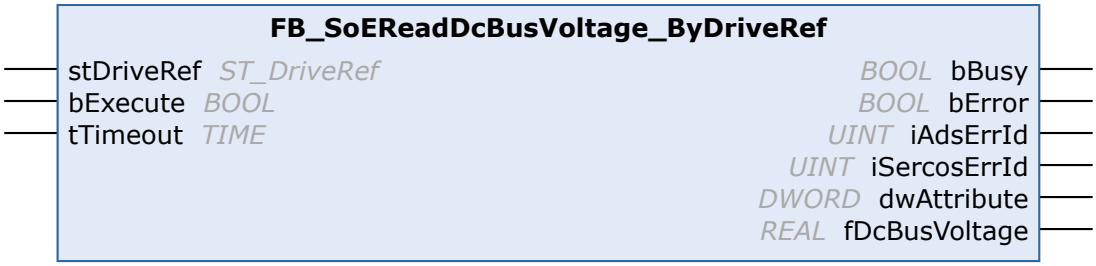
```
VAR_OUTPUT
    bBusy             : BOOL;
    bError            : BOOL;
    iAdsErrId         : UINT;
    iSercosErrId      : UINT;
    dwAttribute       : DWORD;
    fMotorTemperature : REAL;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.
fMotorTemperature	REAL	Liefert die Motortemperatur (z. B. 30.5 entspricht 30.5 °C). Wenn der Motor keinen Temperatursensor enthält, steht hier 0.0, d. h. 0.0 °C.

Beispiel

```
fbReadMotorTemp : FB_SoEReadMotorTemperature_ByDriveRef;
bReadMotorTemp : BOOL;
fMotorTemperature : REAL;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bReadMotorTemp AND NOT bInit THEN
   fbReadMotorTemp(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     fMotorTemperature=>fMotorTemperature
   );
   IF NOT fbReadMotorTemp.bBusy THEN
     fbReadMotorTemp(stDriveRef := stDriveRef, bExecute := FALSE);
     bReadMotorTemp := FALSE;
   END_IF
END_IF
```

4.1.6.3 FB_SoEReadDcBusVoltage_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadDcBusVoltage_ByDriveRef kann die DC-Bus-Spannung des Antriebs (S-0-0380) eingelesen werden.

Eingänge

```
VAR_INPUT
   stDriveRef : ST_DriveRef;
   bExecute   : BOOL;
   tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
   bBusy         : BOOL;
   bError        : BOOL;
   iAdsErrId     : UINT;
   iSercosErrId  : UINT;
   dwAttribute   : DWORD;
   fDcBusVoltage : REAL;
END_VAR
```

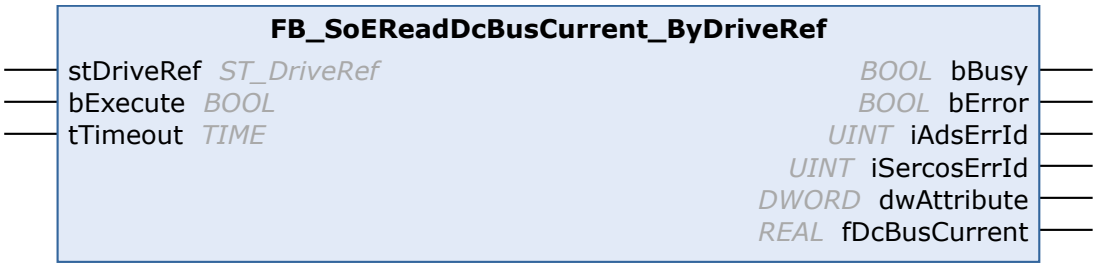
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.
fDcBusVoltage	REAL	Liefert die DC-Bus-Spannung des Antriebs.

Beispiel

```
VAR
   bInit                : BOOL;
   fbReadDcBusVoltage   : FB_SoEReadDcBusVoltage_ByDriveRef;
   bReadDcBusVoltage    : BOOL;
   fDcBusVoltage        : REAL;
   stPlcDriveRef AT %I* : ST_PlcDriveRef;
   stDriveRef           : ST_DriveRef;
END_VAR
```

```
IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
      bInit := FALSE;
   END_IF
END_IF

IF bReadDcBusVoltage AND NOT bInit THEN
   fbReadDcBusVoltage(stDriveRef := stDriveRef,
                      bExecute := TRUE,
                      tTimeout := DEFAULT_ADS_TIMEOUT,
                      fDcBusVoltage => fDcBusVoltage );

   IF NOT fbReadDcBusVoltage.bBusy THEN
      fbReadDcBusVoltage(stDriveRef := stDriveRef, bExecute := FALSE);
      bReadDcBusVoltage := FALSE;
   END_IF
END_IF
```

4.1.6.4 FB_SoEReadDcBusCurrent_ByDriveRef



Mit dem Funktionsbaustein FB_SoEAX5000ReadDcBusCurrent_ByDriveRef kann der DC-Bus-Strom (S-0-0381) eingelesen werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
    dwAttribute  : DWORD;
    fDcBusCurrent: REAL;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.
fDcBusCurrent	REAL	Liefert den DC-Bus-Strom (z. B. 2.040 entspricht 2.040 A).

Beispiel

```
fbReadDcBusCurrent : FB_SoEReadDcBusCurrent_ByDriveRef;
bReadDcBusCurrent : BOOL;
fDcBusCurrent : REAL;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bReadDcBusCurrent AND NOT bInit THEN
   fbReadDcBusCurrent(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     fDcBusCurrent=>fDcBusCurrent
   );
   IF NOT fbReadDcBusCurrent.bBusy THEN
      fbReadDcBusCurrent(stDriveRef := stDriveRef, bExecute := FALSE);
      bReadDcBusCurrent := FALSE;
   END_IF
END_IF
```

4.2 AX5000 SoE

4.2.1 Konvertierungsfunktionen

4.2.1.1 F_ConvWordToSTAX5000C1D

Mit dieser Funktion kann die Class-1-Diagnose FB_SoEReadClassXDiag_ByDriveRef [▶ 24] (S-0-0011) in eine Struktur ST_AX5000_C1D [▶ 49] gewandelt werden.

Eingänge

```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

Name	Typ	Beschreibung
wClass1Diag	WORD	Class-1-Diagnose-Wort aus S-0-0011 (siehe FB_SoEReadClassXDiag_ByDriveRef [▶ 24])

Rückgabewert

```
FUNCTION F_ConvWordToSTAX5000C1D : ST_AX5000_C1D
```

Name	Typ	Beschreibung
F_ConvWordToSTAX5000C1D	ST_AX5000_C1D [▶ 49]	Rückgabewert der Funktion. Class-1-Diagnose als Struktur.

4.2.1.2 F_ConvWordToSTAX5000DriveStatus

Mit dieser Funktion kann das Antriebsstatuswort (S-0-0135) in eine Struktur ST_AX5000DriveStatus [▶ 50] gewandelt werden.

Eingänge

```
VAR_INPUT
    wDriveStatus : WORD;
END_VAR
```

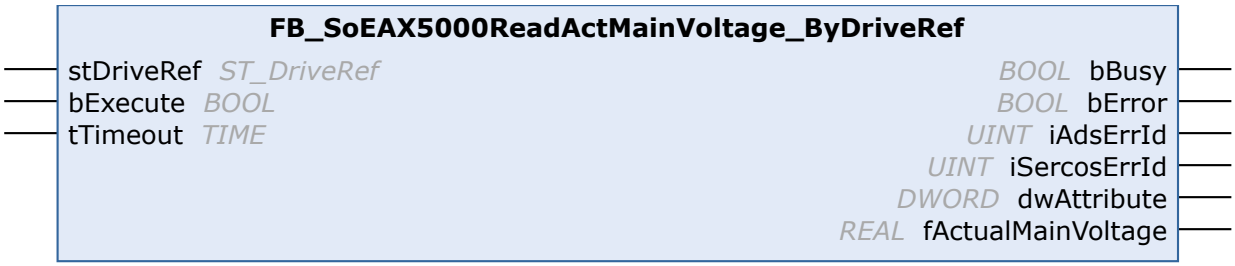
Name	Typ	Beschreibung
wDriveStatus	WORD	Antriebsstatuswort aus S-0-0135 (kann mit FB_SoE_Read_ByDriveRef gelesen werden und ggf. gemappt werden).

Rückgabewert

```
FUNCTION F_ConvWordToSTAX5000DriveStatus : ST_AX5000DriveStatus
```

Name	Typ	Beschreibung
F_ConvWordToSTAX5000DriveStatus	ST_AX5000DriveStatus	Rückgabewert der Funktion. Achszustand als Struktur.

4.2.2 FB_SoEAX5000ReadActMainVoltage_ByDriveRef



Mit dem Funktionsbaustein FB_SoEAX5000ReadActMainVoltage_ByDriveRef kann der aktuelle Scheitelwert der Netzspannung des AX5000 (P-0-0200) eingelesen werden.

Eingänge

```
VAR_INPUT
    stDriveRef : ST_DriveRef;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
    bBusy              : BOOL;
    bError             : BOOL;
    iAdsErrId          : UINT;
    iSercosErrId       : UINT;
    dwAttribute        : DWORD;
    fActualMainVoltage : REAL;
END_VAR
```

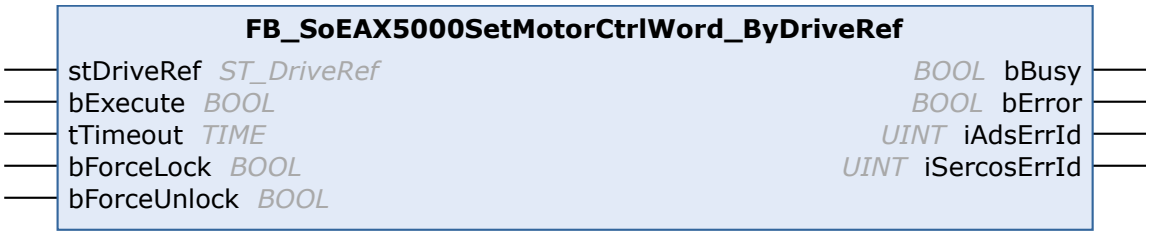
Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
dwAttribute	DWORD	Liefert das Attribut des Sercos-Parameters.

Name	Typ	Beschreibung
fActualMainVoltage	REAL	Liefert den Scheitelwert der aktuellen Netzspannung des AX5000 (z. B. 303.0 entspricht 303.0 V).

Beispiel

```
fbReadActMainVoltage : FB_SoEAX5000ReadActMainVoltage_ByDriveRef;
bReadActMainVoltage  : BOOL;
fActualMainVoltage   : REAL;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef : ST_DriveRef;

IF bInit THEN
    stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
    stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
    stDriveRef.nDriveNo   := stPlcDriveRef.nDriveNo;
    stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
    IF (stDriveRef.sNetId <> '') AND(stDriveRef.nSlaveAddr <> 0)THEN
        bInit := FALSE;
    END_IF
END_IF

IF bReadActMainVoltage AND NOT bInit
   THEN fbReadActMainVoltage(
        stDriveRef := stDriveRef,
        bExecute := TRUE,
        tTimeout := DEFAULT_ADS_TIMEOUT,
        fActualMainVoltage=>fActualMainVoltage
        );
   IF NOT fbReadActMainVoltage.bBusy THEN
       fbReadActMainVoltage(stDriveRef := stDriveRef, bExecute := FALSE);
       bReadActMainVoltage := FALSE;
   END_IF
END_IF
```

4.2.3 FB_SoEAX5000SetMotorCtrlWord_ByDriveRef



Mit dem Funktionsbaustein FB_SoEAX5000SetMotorCtrlWord_ByDriveRef kann das ForceLock-Bit (Bit 0) bzw. das ForceUnlock-Bit im Motor Control Word (P-0-0096) gesetzt werden, um die Bremse zu aktivieren oder zu lösen. Im Normalfall wird die Bremse automatisch über das Enable des Antriebs gesteuert.

Mit dem ForceLock-Bit kann die Bremse unabhängig vom Enable aktiviert werden, mit dem ForceUnlock-Bit kann die Bremse unabhängig vom Enable gelöst werden. Bei gleichzeitig gesetztem ForceLock und ForceUnlock hat das ForceLock (Bremse aktiviert) die höhere Priorität.

Eingänge

```
VAR_INPUT
    stDriveRef   : ST_DriveRef;
    bExecute     : BOOL;
    tTimeout     : TIME := DEFAULT_ADS_TIMEOUT;bForceLock   : BOOL;
    bForceUnlock : BOOL
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])

Name	Typ	Beschreibung
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.
bForceLock	BOOL	Aktiviert die Bremse unabhängig vom Enable.
bForceUnlock	BOOL	Löst die Bremse unabhängig vom Enable.

Ausgänge

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.

Beispiel

```
fbSetMotorCtrlWord : FB_SoEAX5000SetMotorCtrlWord_ByDriveRef;
bSetMotorCtrlWord  : BOOL;
bForceLock         : BOOL;
bForceUnlock       : BOOL;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef         : ST_DriveRef;

IF bInit THEN
   stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
   stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
   stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
   stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
   IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
       bInit := FALSE;
   END_IF
END_IF

IF bSetMotorCtrlWord AND NOT bInit THEN
   fbSetMotorCtrlWord(
      stDriveRef := stDriveRef,
      bExecute := TRUE,
      tTimeout := DEFAULT_ADS_TIMEOUT,
      bForceLock := bForceLock,
      bForceUnlock:= bForceUnlock
   );
   IF NOT fbSetMotorCtrlWord.bBusy THEN
      fbSetMotorCtrlWord(stDriveRef := stDriveRef, bExecute := FALSE);
      bSetMotorCtrlWord := FALSE;
   END_IF
END_IF
```

4.2.4 FB_SoEAX5000FirmwareUpdate_ByDriveRef

Mit dem Funktionsbaustein FB_SoEAX5000FirmwareUpdate_ByDriveRef kann die Firmware des AX5000 überprüft und automatisch auf eine bestimme Version (Revision und Build) oder auf das aktuellste Build der konfigurierten Revision geändert werden.

Für das Update wird:

- Der konfigurierte Slave-Typ ermittelt, z. B. AX5103-0000-0010.
- Der aktuelle Slave mit der vorgegebenen Slave-Adresse ermittelt, z. B. AX5103-0000-0009.
- Die aktuelle Slave-Firmware ermittelt, z. B. v1.05_b0009.
- Ein Vergleich der Konfiguration und des gefundenen Slaves auf Anzahl der Kanäle, Strom, Revision und Firmware ausgeführt.
- Der Name des erforderlichen Firmware-Files ermittelt und die Datei gesucht.
- Der Firmware-Update (falls erforderlich) ausgeführt.
- Der aktuelle Slave mit der vorgegebenen Slave-Adresse erneut ermittelt.
- Der Slave in den vorgegebenen EtherCAT-Status geschaltet.

Ein erfolgreiches Update endet mit eFwUpdateState = eFwU_FwUpdateDone.

Wenn das Update nicht erforderlich ist, wird dies über eFwUpdateState = eFwU_NoFwUpdateRequired signalisiert.

Das Firmware-Update erfolgt über den angegebenen Kanal (A = 0 oder B = 1) aus stDriveRef. Bei zweikanaligen Geräten kann nur einer der beiden Kanäle verwendet werden. Der andere Kanal signalisiert eFwUpdateState = eFwU_UpdateViaOtherChannelActive bzw.
eFwUpdateState = eFwU_UpdateViaOtherChannel.

Während des Firmware-Updates (eFwUpdateState = eFwU_FwUpdateInProgress) signalisiert iLoadProgress den Fortschritt in Prozent.

HINWEIS

Fehlerhaftes Update durch Unterbrechungen

Unterbrechungen während des Updates können dazu führen, dass dieses nicht oder fehlerhaft ausgeführt wird. Der Antriebsverstärker kann danach ohne die passende Firmware möglicherweise nicht mehr verwendet werden.

Während des Updates gilt:

- Die SPS und TwinCAT dürfen nicht gestoppt werden.
- Die EtherCAT-Verbindung darf nicht unterbrochen werden.
- Der AX5000 darf nicht ausgeschaltet werden.

Eingänge

```
VAR_INPUT
   stDriveRef       : ST_DriveRef;
   bExecute         : BOOL;
   tTimeout         : TIME := DEFAULT_ADS_TIMEOUT;
   sFirmwareVersion : STRING(20);  (* version string vx_yy_bnnnn, e.g. "v1.05_b0009" for v1.05 Build
 0009*)
   sFirmwarePath    : T_MaxString; (* drive:\path, e.g. "C:
```

```
\TwinCAT\Io\TcDriveManager\FirmwarePool" *)
   sNetIdIPC        : T_AmsNetId;
   iReqEcState      : UINT := EC_DEVICE_STATE_OP;
END_VAR
```

Name	Typ	Beschreibung
stDriveRef	ST_DriveRef	Referenz auf den Antrieb. Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der Struktur ST_PlcDriveRef lokiert und die NetID vom Bytearray in einen String konvertiert werden. (Typ: ST_DriveRef [▶ 11])
bExecute	BOOL	Der Baustein wird über eine positive Flanke an diesem Eingang aktiviert.
tTimeout	TIME	Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.
sFirmwareVersion	STRING(20)	Gibt die gewünschte Firmware-Version in Form von vx.yy_bnnnn an, z. B. „v1.05_b0009“ für Version v1.05 Build 0009. **Release-Builds:** • "v1.05_b**00**09" für ein spezifisches Build, z. B. v1.05 Build 0009 • "v1.05_b**00**??" aktuellstes Build einer vorgegebenen Version, z. B. v1.05 • "v1.??_b**00**??" aktuellstes Build einer vorgegebenen Hauptversion, z. B. v1 • " " aktuellstes Build der aktuellsten Version **Kundenspezifische Firmware-Builds:** • "v1.05_b**10**09" für ein spezifisches Build, z. B. v1.05 Build 0009 • "v1.05_b**10**??" aktuellstes Build einer vorgegebenen Version, z. B. v1.05 • "v1.??_b**10**??" aktuellstes Build einer vorgegebenen Hauptversion, z. B. v1 • "v?.??_b**10**??" aktuellstes Build der aktuellsten Version • "v1.05_b**89**09" für ein spezifisches Build, z. B. v1.05 Build 8909 • "v1.05_b**89**??" aktuellstes Build einer vorgegebenen Version, z. B. v1.05 • "v1.??_b**89**??" aktuellstes Build einer vorgegebenen Hauptversion, z. B. v1 • "v?.??_b**89**??" aktuellstes Build der aktuellsten Version **Debug-Builds:** • "v1.05_b**90**09" für ein spezifisches Build, z. B. v1.05 Build 9009 • "v1.05_b**90**??" aktuellstes Build einer vorgegebenen Version, z. B. v1.05 • "v1.??_b**90**??" aktuellstes Build einer vorgegebenen Hauptversion, z. B. v1 • "v?.??_b**90**??" aktuellstes Build der aktuellsten Version
sFirmwarePath	T_MaxString	Gibt den Pfad für den Firmware-Pool an, in dem sich die Firmware-Dateien befinden, z. B. *C:\TwinCAT\Io\TcDriveManager\FirmwarePool*.
sNetIdIPC	T_AmsNetId	AMS-NetID der Steuerung (IPC)
iReqEcState	UINT	Gewünschter EtherCAT-Status nach dem Update, nur wenn tatsächlich ein Update ausgeführt wird. Die Status sind in der PLC Lib Tc2_EtherCAT als globale Konstanten definiert.

Ausgänge

```
VAR_OUTPUT
   bBusy                : BOOL;
   bError               : BOOL;
   iAdsErrId            : UINT;
   iSercosErrId         : UINT;
   iDiagNumber          : UDINT;
   eFwUpdateState       : E_FwUpdateState;
   iLoadProgress        : INT;
   sSelectedFirmwareFile : STRING(MAX_STRING_LENGTH); (* found firmware file, e.g. "AX5yxx_xxxx_0010
_v1_05_b0009.efw" *)
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bError	BOOL	Dieser Ausgang wird gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde, wenn ein Fehler bei der Übertragung des Kommandos auftritt.
iAdsErrId	UINT	Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehls.
iSercosErrId	UINT	Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehls.
iDiagNumber	UDINT	Liefert bei gesetztem bError-Ausgang den Antriebsverstärkerfehler des letzten Firmware-Updates.
eFwUpdateState	E_FwUpdateState	Liefert den Status der Firmware-Updates (Siehe E_FwUpdateState [▶ 48]).
iLoadProgress	INT	Liefert den Fortschritt des eigentlichen Firmware-Updates in Prozent.
sSelectedFirmwareFile	STRING(MAX_STRING_LENGTH)	Zeigt den Namen der gesuchten Firmware-Datei an.

Beispiel

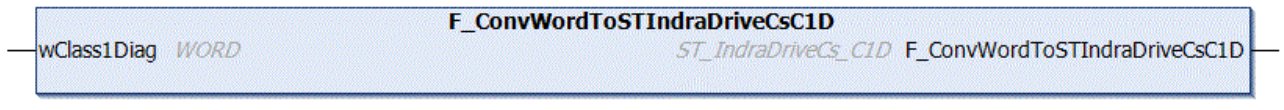
```
VAR CONSTANT
    iNumOfDrives : INT := 2;
END_VAR

VAR
    bInit : ARRAY [1..iNumOfDrives] OF BOOL := 2(TRUE);
    fbFirmwareUpdate : ARRAY[1..iNumOfDrives] OF FB_SoEAX5000FirmwareUpdate_ByDriveRef;
    stPlcDriveRef AT %I* : ARRAY[1..iNumOfDrives]OF ST_PlcDriveRef;
    stDriveRef : ARRAY [1..iNumOfDrives] OF ST_DriveRef;
    sFirmwareVersion : ARRAY[1..iNumOfDrives] OF STRING(20) := 2('v1.05_b0009');
    eFwUpdateState : ARRAY[1..iNumOfDrives] OF E_FwUpdateState;
    sSelectedFirmwareFile: ARRAY [1..iNumOfDrives] OF STRING(MAX_STRING_LENGTH);
    iUpdateState : INT;
    bExecute : BOOL;
    sNetIdIPC : T_AmsNetId := '';
    sFirmwarePath  : T_MaxString := 'C:\TwinCAT\Io\TcDriveManager\FirmwarePool';
    I : INT;
    bAnyInit : BOOL;
    bAnyBusy : BOOL;
    bAnyError : BOOL;
END_VAR

CASE iUpdateState OF
0:
  bAnyInit := FALSE;
  FOR I := 1 TO iNumOfDrives DO
    IF bInit[I] THEN
       bAnyInit := TRUE;
       stDriveRef[I].sNetId := F_CreateAmsNetId(stPlcDriveRef[I].aNetId);
       stDriveRef[I].nSlaveAddr := stPlcDriveRef[I].nSlaveAddr;
       stDriveRef[I].nDriveNo   := stPlcDriveRef[I].nDriveNo;
       stDriveRef[I].nDriveType := stPlcDriveRef[I].nDriveType;
       IF (stDriveRef[I].sNetId <> '') AND (stDriveRef[I].nSlaveAddr <> 0)
           THEN bInit[I] := FALSE;
```

```
      END_IF
    END_IF
  END_FOR
  IF NOT bAnyInit AND bExecute THEN
    iUpdateState := 1;
  END_IF
1:
    FOR I := 1 TO iNumOfDrives DO
      fbFirmwareUpdate[I](
        stDriveRef := stDriveRef[I],
        bExecute := TRUE,
        tTimeout := T#15s,
        sFirmwareVersion := sFirmwareVersion[I],
        sFirmwarePath := sFirmwarePath,
        sNetIdIPC := sNetIdIPC,
        iReqEcState := EC_DEVICE_STATE_OP,
        eFwUpdateState => eFwUpdateState[I],
      );
    END_FOR
    iUpdateState := 2;
2:
    bAnyBusy := FALSE;
    bAnyError:= FALSE;
    FOR I := 1 TO iNumOfDrives DO
        fbFirmwareUpdate[I](
          eFwUpdateState => eFwUpdateState[I],
          sSelectedFirmwareFile => sSelectedFirmwareFile[I],
        );
      IF NOT fbFirmwareUpdate[I].bBusy THEN
        fbFirmwareUpdate[I](bExecute := FALSE);
        IF fbFirmwareUpdate[I].bError THEN
          bAnyError := TRUE;
        END_IF
      ELSE
        bAnyBusy := TRUE;
      END_IF
    END_FOR
    IF NOT bAnyBusy THEN
      bExecute := FALSE;
      IF NOT bAnyError THEN
        iUpdateState := 0; (* OK *)
      ELSE
      iUpdateState := 0; (* Error *)
      END_IF
    END_IF
END_CASE
```

4.3 IndraDrive Cs

4.3.1 Konvertierungsfunktionen

4.3.1.1 F_ConvWordToSTIndraDriveCsC1D



Mit dieser Funktion kann die Class-1-Diagnose FB_SoEReadClassXDiag_ByDriveRef [▶ 24] (S-0-0011) in eine Struktur ST_IndraDriveCs_C1D [▶ 51] umgewandelt werden.

Eingänge

```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

Name	Typ	Beschreibung
wClass1Diag	WORD	Class-1-Diagnose-Wort aus S-0-0011 (siehe FB_SoEReadClassXDiag_ByDriveRef [▶ 24]).

Rückgabewert

```
FUNCTION F_ConvWordToSTIndraDriveCsC1D : ST_IndraDriveCs_C1D
```

Name	Typ	Beschreibung
F_ConvWordToSTIndraDriveCsC1D	ST_IndraDriveCs_C1D [▶ 51]	Rückgabewert der Funktion Class-1-Diagnose als ST_IndraDrivesCs_C1D-Struktur.

4.3.1.2 F_ConvWordToSTIndraDriveCsDriveStatus

Mit dieser Funktion kann das Antriebsstatuswort (S-0-0135) in eine Struktur ST_IndraDriveCsDriveStatus [▶ 52] gewandelt werden.

Eingänge

```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

Name	Typ	Beschreibung
wClass1Diag	WORD	Antriebsstatuswort aus S-0-0135 Kann mit FB_SoE_Read_ByDriveRef gelesen werden und kann ggf. gemappt werden.

Rückgabewert

```
FUNCTION F_ConvWordToSTIndraDriveCsDriveStatus : ST_IndraDriveCsDriveStatus
```

Name	Typ	Beschreibung
F_ConvWordToSTIndraDriveCsDriveStatus	ST_IndraDriveCsDriveStatus [▶ 52]	Rückgabewert der Funktion. Antriebsstatuswort als ST_IndraDriveCsDriveStatus-Struktur.

4.4 F_GetVersionTcDrive

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

Eingänge

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

Name	Typ	Beschreibung
nVersionElement	INT	**nVersionElement**: Versionselement, das gelesen werden soll. Mögliche Parameter: • 1 : major number; • 2 : minor number; • 3 : revision number;

Rückgabewert

```
FUNCTION F_GetVersionTcDrive : UINT
```

Name	Typ	Beschreibung
F_GetVersionTcDrive	UINT	Rückgabewert der Funktion. Versionselement als UINT.

4.5 SimplePlcMotion

Simple PLC Motion-Funktionsbausteine ermöglichen den einfachen Betrieb eines Antriebs direkt aus der SPS.

4.5.1 FB_CoEDriveEnable

Der Funktionsbaustein FB_CoEDriveEnable gibt einen CoE-Antrieb frei, um ihn anschließend mit dem Funktionsbaustein FB_CoEDriveMoveVelocity [▶ 42] mit Sollwerten versorgen zu können.

Eingänge

```
VAR_INPUT
    bEnable : BOOL;
    bReset  : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEnable	BOOL	Aktiviert den CoE-Antrieb.
bReset	BOOL	Führt einen Antriebsreset im Fehlerfall aus. Im Antriebs-Control-Wort wird das „Bit 7“ gesetzt.

/ Ein-/Ausgänge

```
VAR_IN_OUT
    stCoeDriveIoInterface : ST_CoeDriveIoInterface;
END_VAR
```

Name	Typ	Beschreibung
stCoeDriveIoInterface	ST_CoeDriveIoInterface	Datenstruktur, mit der das Prozessabbild des CoE-Antriebs verlinkt werden muss.

Ausgänge

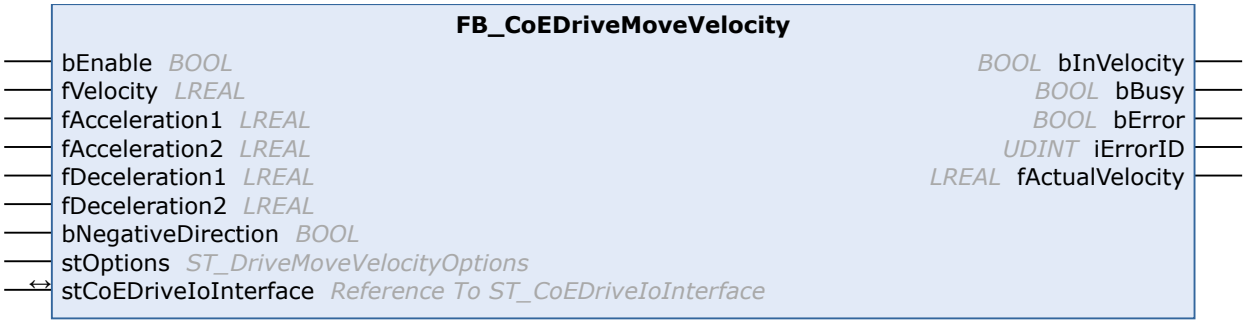
```
VAR_OUTPUT
    bStatus     : BOOL;
    bDriveError : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bStatus	BOOL	Wenn bStatus=TRUE ist, ist der Antrieb betriebsbereit und folgt den Sollwerten.
bDriveError	BOOL	Der Antrieb ist im Fehlerzustand.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

4.5.2 FB_CoEDriveMoveVelocity



Der Funktionsbaustein FB_CoEDriveMoveVelocity erzeugt ein einfaches Dreiphasen-Geschwindigkeitsprofil (ohne Ruckbegrenzung), mit dem ein CoE-Antrieb direkt versorgt werden kann. Unterhalb und oberhalb einer parametrierbaren Geschwindigkeitsschwelle können unterschiedliche Beschleunigungen bzw. Verzögerungen verwendet werden. Die Zielgeschwindigkeit kann im Betrieb geändert werden.

Vorab muss der CoE-Antrieb über den Funktionsbaustein FB_CoEDriveEnable [▶ 41] freigegeben werden.

Eingänge

```
VAR_INPUT
    bEnable            : BOOL;
    fVelocity          : LREAL;
    fAcceleration1     : LREAL;
    fAccelaration2     : LREAL;
    fDeceleration1     : LREAL;
    fDeceleration2     : LREAL;
    bNegativeDirection : BOOL;
    stOptions          : ST_DriveMoveVelocityOptions;
END_VAR
```

Name	Typ	Beschreibung
bEnable	BOOL	Aktiviert die Sollwertgenerierung.
fVelocity	LREAL	Zielgeschwindigkeit. fVelocity kann im Betrieb geändert werden.
fAcceleration1	LREAL	Beschleunigung 1 wird unterhalb des parametrierten Geschwindigkeitsschwellwertes stOptions. fVelocityThreshold verwendet.
fAcceleration2	LREAL	Beschleunigung 2 wird oberhalb des parametrierten Geschwindigkeitsschwellwertes stOptions. fVelocityThreshold verwendet.

Name	Typ	Beschreibung
fDeceleration1	LREAL	Verzögerung 1 wird unterhalb des parametrierten Geschwindigkeitsschwellwertes stOptions.fVelocityThreshold verwendet.
fDeceleration2	LREAL	Verzögerung 2 wird oberhalb des parametrierten Geschwindigkeitsschwellwertes stOptions.fVelocityThreshold verwendet.
bNegativeDirection	BOOL	bNegativeDirection kehrt die Fahrtrichtung um.
stOptions	ST_DriveMoveVelocityOptions	Datenstruktur mit weiteren Parametern.

/ Ein-/Ausgänge

```
VAR_IN_OUT
    stCoEDriveIoInterface : ST_CoeDriveIoInterface;
END_VAR
```

Name	Typ	Beschreibung
stCoEDriveIoInterface	ST_CoeDriveIoInterface	Prozessabbild des CoE-Antriebs

Ausgänge

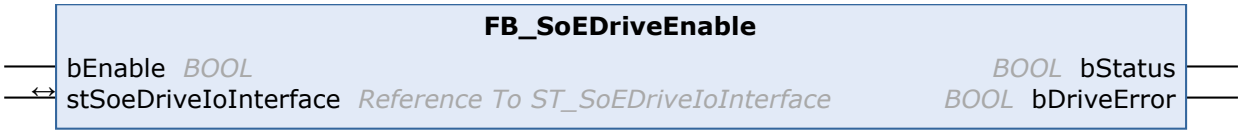
```
VAR_OUTPUT
    bInVelocity     : BOOL;
    bBusy           : BOOL;
    bError          : BOOL;
    iErrorID        : UDINT;
    fActualVelocity : LREAL;
END_VAR
```

Name	Typ	Beschreibung
bInVelocity	BOOL	Zielgeschwindigkeit ist erreicht.
bBusy	BOOL	bBusy ist TRUE, solange der Funktionsbaustein aktiv ist und ein Sollwertprofil berechnet wird.
bError	BOOL	bError wird im Fehlerfall TRUE.
iErrorID	UDINT	Fehlernummer
fActualVelocity	LREAL	Aktuell erreichte Geschwindigkeit des Antriebs.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

4.5.3 FB_SoEDriveEnable



Der Funktionsbaustein FB_SoEDriveEnable gibt einen SoE-Antrieb frei, um ihn anschließend mit dem Funktionsbaustein FB_SoEDriveMoveVelocity [▶ 44] mit Sollwerten versorgen zu können.

Eingänge

```
VAR_INPUT
    bEnable : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEnable	BOOL	Aktiviert den SoE-Antrieb.

/ Ein-/Ausgänge

```
VAR_IN_OUT
    stSoeDriveIoInterface : ST_SoEDriveIoInterface;
END_VAR
```

Name	Typ	Beschreibung
stSoeDriveIoInterface	ST_SoEDriveIoInterface	Datenstruktur, mit der das Prozessabbild des SoE-Antriebs verlinkt werden muss.

Ausgänge

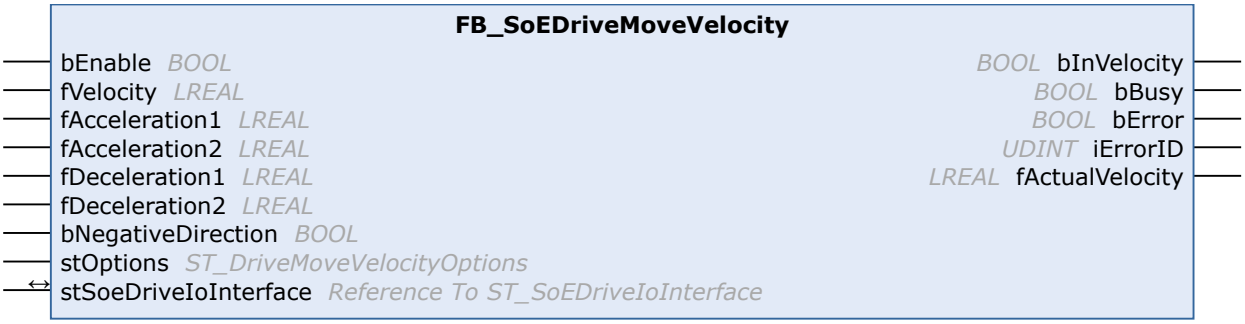
```
VAR_OUTPUT
    bStatus     : BOOL;
    bDriveError : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bStatus	BOOL	Wenn bStatus=TRUE ist, ist der Antrieb betriebsbereit und folgt den Sollwerten.
bDriveError	BOOL	Der Antrieb ist im Fehlerzustand.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

4.5.4 FB_SoEDriveMoveVelocity



Der Funktionsbaustein FB_SoEDriveMoveVelocity erzeugt ein einfaches Dreiphasen-Geschwindigkeitsprofil (ohne Ruckbegrenzung), mit dem ein SoE-Antrieb direkt versorgt werden kann. Unterhalb und oberhalb einer parametrierbaren Geschwindigkeitsschwelle können unterschiedliche Beschleunigungen bzw. Verzögerungen verwendet werden. Die Zielgeschwindigkeit kann im Betrieb geändert werden.

Vorab muss der SoE-Antrieb über den Funktionsbaustein FB_SoEDriveEnable [▶ 43] freigegeben werden.

Eingänge

```
VAR_INPUT
    bEnable           : BOOL;
    fVelocity         : LREAL;
```

```
    fAcceleration1     : LREAL;
    fAccelaration2     : LREAL;
    fDeceleration1     : LREAL;
    fDeceleration2     : LREAL;
    bNegativeDirection : BOOL;
    stOptions          : ST_DriveMoveVelocityOptions;
END_VAR
```

Name	Typ	Beschreibung
bEnable	BOOL	Aktiviert die Sollwertgenerierung.
fVelocity	LREAL	Zielgeschwindigkeit. fVelocity kann im Betrieb geändert werden.
fAcceleration1	LREAL	Beschleunigung 1 wird unterhalb des parametrierten Geschwindigkeitsschwellwertes stOptions. fVelocityThreshold verwendet.
fAcceleration2	LREAL	Beschleunigung 2 wird oberhalb des parametrierten Geschwindigkeitsschwellwertes stOptions. fVelocityThreshold verwendet.
fDeceleration1	LREAL	Verzögerung 1 wird unterhalb des parametrierten Geschwindigkeitsschwellwertes stOptions. fVelocityThreshold verwendet.
fDeceleration2	LREAL	Verzögerung 2 wird oberhalb des parametrierten Geschwindigkeitsschwellwertes stOptions. fVelocityThreshold verwendet.
bNegativeDirection	BOOL	bNegativeDirection kehrt die Fahrtrichtung um.
stOptions	ST_DriveMoveVelocityOptions	Datenstruktur mit weiteren Parametern.

/ Ein-/Ausgänge

```
VAR_IN_OUT
    stSoEDriveIoInterface : ST_SoeDriveIoInterface;
END_VAR
```

Name	Typ	Beschreibung
stSoEDriveIoInterface	ST_SoeDriveIoInterface	Prozessabbild des CoE-Antriebs

Ausgänge

```
VAR_OUTPUT
    bInVelocity     : BOOL;
    bBusy           : BOOL;
    bError          : BOOL;
    iErrorID        : UDINT;
    fActualVelocity : LREAL;
END_VAR
```

Name	Typ	Beschreibung
bInVelocity	BOOL	Zielgeschwindigkeit ist erreicht.
bBusy	BOOL	bBusy ist TRUE, solange der Funktionsbaustein aktiv ist und ein Sollwertprofil berechnet wird.
bError	BOOL	bError wird im Fehlerfall TRUE.
iErrorID	UDINT	Fehlernummer
fActualVelocity	LREAL	Aktuell erreichte Geschwindigkeit des Antriebs.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

5 Datentypen

5.1 Allgemein SoE

5.1.1 ST_SoE_String

Die Struktur ST_SoE_String beschreibt einen String, wie er bei SoE-Zugriffen verwendet werden kann.

```
TYPE ST_SoE_String :
STRUCT
    iActualSize  : UINT;
    iMaxSize     : UINT;
    strData      : STRING(MAX_STRING_LENGTH);
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
iActualSize	UINT	Aktuelle Länge des Strings (ohne abschließende \0)
iMaxSize	UINT	Maximale Länge des Strings (ohne abschließende \0)
strData	STRING(MAX_STRING_LENGTH)	String

5.1.2 ST_SoE_StringEx

Die Struktur ST_SoE_StringEx beschreibt einen String, wie er bei SoE-Zugriffen verwendet werden kann, inklusive vorangestelltem Parameterattribut.

```
TYPE ST_SoE_StringEx :
STRUCT
    dwAttribute  : DWORD;
    iActualSize  : UINT;
    iMaxSize     : UINT;
    strData      : STRING(MAX_STRING_LENGTH);
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
dwAttribute	DWORD	Parameterattribut
iActualSize	UINT	Aktuelle Länge des Strings (ohne abschließende \0)
iMaxSize	UINT	Maximale Länge des Strings (ohne abschließende \0)
strData	STRING(MAX_STRING_LENGTH)	String

5.1.3 Listentypen

5.1.3.1 ST_SoE_DiagNumList

Die Struktur ST_SoE_DiagNumList enthält die Listenlänge (Minimum, Maximum) in Bytes sowie die Historie der Diagnosenummern.

```
TYPE ST_SoE_DiagNumList :
STRUCT
    iActualSize     : UINT;
    iMaxSize        : UINT;
```

```
    arrDiagNumbers  : ARRAY [0..29] OF UDINT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
iActualSize	UINT	Aktuelle Länge des Strings (ohne abschließende \0)
iMaxSize	UINT	Maximale Länge des Strings (ohne abschließende \0)
arrDiagNumbers	ARRAY [0..29] OF UDINT	Liste der maximal 30 letzten Fehlernummern (als UDINT)

5.2 AX5000 SoE

5.2.1 E_FwUpdateState

Die Enumeration E_FwUpdateState beschreibt den Zustand eines Firmware-Updates.

```
TYPE E_SoE_CmdState :(
    (* update states *)
    eFwU_NoError := 0,
    eFwU_CheckCfgIdentity,
    eFwU_CheckSlaveCount,
    eFwU_CheckFindSlavePos,
    eFwU_WaitForScan,
    eFwU_ScanningSlaves,
    eFwU_CheckScannedIdentity,
    eFwU_CheckScannedFirmware,
    eFwU_FindFirmwareFile,
    eFwU_WaitForUpdate,
    eFwU_WaitForSlaveState,
    eFwU_StartFwUpdate,
    eFwU_FwUpdateInProgress,
    eFwU_FwUpdateDone,
    eFwU_NoFwUpdateRequired,

    (* not updating via this channel *)
    eFwU_UpdateViaOtherChannelActive,
    eFwU_UpdatedViaOtherChannel,

    (* error states *)
    eFwU_GetSlaveIdentityError         := -1,
    eFwU_GetSlaveCountError            := -2,
    eFwU_GetSlaveAddrError             := -3,
    eFwU_StartScanError                := -4,
    eFwU_ScanStateError                := -5,
    eFwU_ScanIdentityError             := -6,
    eFwU_GetSlaveStateError            := -7,
    eFwU_ScanFirmwareError             := -8,
    eFwU_FindFileError                 := -9,
    eFwU_CfgTypeInNoAX5xxx             := -10,
    eFwU_ScannedTypeInNoAX5xxx         := -11,
    eFwU_ChannelMismatch               := -12,
    eFwU_ChannelMismatch_1Cfg_2Scanned := -13,
    eFwU_ChannelMismatch_2Cfg_1Scanned := -14,
    eFwU_CurrentMismatch               := -15,
    eFwU_FwUpdateError                 := -16,
    eFwU_ReqSlaveStateError            := -17
);
END_TYPE
```

Update-Status

eFwU_NoError	Initialzustand
eFwU_CheckCfgIdentity	Einlesen des konfigurierten Slavetypen (Anzahl Kanäle, Strom, Revision)
eFwU_CheckSlaveCount	Ermitteln der konfigurierten Slaveanzahl
eFwU_CheckFindSlavePos	Suchen der Slaveadresse im Master-Objektverzeichnis
eFwU_WaitForScan	Warten auf Online-Scan

eFwU_ScanningSlaves	Online-Scan der Slaves
eFwU_CheckScannedIdentity	Einlesen des gescannten Slavetypen (Anzahl Kanäle, Strom, Revision)
eFwU_CheckScannedFirmware	Einlesen der Firmware-Version
eFwU_FindFirmwareFile	Suchen nach der gewählten Firmware-Datei
eFwU_WaitForUpdate	Warten auf Status des Updates
eFwU_WaitForSlaveState	Ermitteln des EtherCAT-Slave-Status
eFwU_StartFwUpdate	Starten des Firmware-Updates
eFwU_FwUpdateInProgress	Firmware-Update aktiv
eFwU_FwUpdateDone	Firmware-Update erfolgreich beendet
eFwU_NoFwUpdateRequired	Kein Firmware-Update erforderlich
eFwU_UpdateViaOtherChannelActive	Update erfolgt über den anderen Achskanal
eFwU_UpdatedViaOtherChannel	Update erfolgte über den anderen Achskanal

Update-Fehler

eFwU_GetSlaveIdentityError	Einlesen des konfigurierten Slavetypen schlug fehl (siehe iAdsErrId)-
eFwU_GetSlaveCountError	Ermitteln der konfigurierten Slaveanzahl schlug fehl (siehe iAdsErrId)-
eFwU_GetSlaveAddrError	Suchen der Slaveadresse im Master-Objektverzeichnis schlug fehl (siehe iAdsErrId)-
eFwU_StartScanError	Starten des Online-Scan schlug fehl (siehe iAdsErrId)-
eFwU_ScanStateError	Online-Scan schlug fehl (siehe iAdsErrId)-
eFwU_ScanIdentityError	Einlesen des gescannten Slavetypen (Anzahl Kanäle, Strom, Revision) schlug fehl (siehe iAdsErrId)-
eFwU_GetSlaveStateError	Ermitteln des EtherCAT-Slave-Status schlug fehl (siehe iAdsErrId)-
eFwU_ScanFirmwareError	Einlesen der Firmware-Version schlug fehl (siehe iAdsErrId + iSercosErrId).
eFwU_FindFileError	Suchen nach der gewählten Firmware-Datei schlug fehl (siehe iAdsErrId).
eFwU_CfgTypeInNoAX5xxx	Der konfigurierte Slave ist kein AX5000.
eFwU_ScannedTypeInNoAX5xxx	Der gescannte Slave ist kein AX5000.
eFwU_ChannelMismatch	Die Anzahl der konfigurierten und gefundenen Kanäle des AX5000 passen nicht zusammen.
eFwU_ChannelMismatch_1Cfg_2Scanned	Einkanaliges Gerät konfiguriert, aber zweikanaliges Gerät gefunden.
eFwU_ChannelMismatch_2Cfg_1Scanned	Zweikanaliges Gerät konfiguriert, aber einkanaliges Gerät gefunden.
eFwU_CurrentMismatch	AX5000-Typ passt vom Strom her nicht, z. B. AX5103 (3A) konfiguriert aber AX5106 (6A) gefunden.
eFwU_FwUpdateError	Allgemeiner Updatefehler (siehe iAdsErrId)
eFwU_ReqSlaveStateError	Umschalten in den gewünschten EtherCAT-Status schlug fehl.

5.2.2 ST_AX5000_C1D für Class 1 Diagnose

```
TYPE ST_AX5000_C1D :
STRUCT
    bOverloadShutdown           : BOOL; (* C1D Bit 0 *)
    bAmplifierOverTempShutdown  : BOOL; (* C1D Bit 1 *)
    bMotorOverTempShutdown      : BOOL; (* C1D Bit 2 *)
    bCoolingErrorShutdown       : BOOL; (* C1D Bit 3 *)
    bControlVoltageError        : BOOL; (* C1D Bit 4 *)
    bFeedbackError              : BOOL; (* C1D Bit 5 *)
    bCommunicationError         : BOOL; (* C1D Bit 6 *)
    bOverCurrentError           : BOOL; (* C1D Bit 7 *)
    bOverVoltageError           : BOOL; (* C1D Bit 8 *)
```

```
    bUnderVoltageError          : BOOL; (* C1D Bit 9 *)
    bPowerSupplyPhaseError      : BOOL; (* C1D Bit 10 *)
    bExcessivePosDiviationError : BOOL; (* C1D Bit 11 *)
    bCommunicationErrorBit      : BOOL; (* C1D Bit 12 *)
    bOvertravelLimitExceeeded   : BOOL; (* C1D Bit 13 *)
    bReserved                   : BOOL; (* C1D Bit 14 *)
    bManufacturerSpecificError  : BOOL; (* C1D Bit 15 *)
END_STRUCT
END_TYPE
```

5.2.3 ST_AX5000DriveStatus

```
TYPE ST_AX5000DriveStatus :
STRUCT
    bStatusCmdValProcessing : BOOL;
    bRealTimeStatusBit1     : BOOL;
    bRealTimeStatusBit2     : BOOL;
    bDrvShutdownBitC1D      : BOOL;
    bChangeBitC2D           : BOOL;
    bChangeBitC3D           : BOOL;
    bNotReadyToPowerUp      : BOOL;
    bReadyForPower          : BOOL;
    bReadyForEnable         : BOOL;
    bEnabled                : BOOL;
    iActOpModeParNum        : UINT;
    eActOpMode              : E_AX5000_DriveOpMode;
    iReserved               : UINT;
END_STRUCT
END_TYPE
```

5.2.4 E_AX5000_DriveOpMode

```
TYPE E_AX5000_DriveOpMode : (
    eOPM_NoModeOfOperation  := 0,
    eOPM_TorqueCtrl         := 1,
    eOPM_VeloCtrl           := 2,
    eOPM_PosCtrlFbk1        := 3,
    eOPM_PosCtrlFbk2        := 4,
    eOPM_PosCtrlFbk1LagLess := 11,
    eOPM_PosCtrlFbk2LagLess := 12
);
END_TYPE
```

5.3 IndraDrive Cs

5.3.1 E_IndraDriveCs_DriveOpMode

```
TYPE E_IndraDriveCs_DriveOpMode : (
    eIDC_NoModeOfOperation := 0,
    eIDC_TorqueCtrl        := 1,
    eIDC_VeloCtrl          := 2,

    eIDC_PosCtrlFbk1        := 3,
    eIDC_PosCtrlFbk2        := 4,
    eIDC_PosCtrlFbk1LagLess := 11,
    eIDC_PosCtrlFbk2LagLess := 12,

    eIDC_DrvInternInterpolFbk1        := 19,
    eIDC_DrvInternInterpolFbk2        := 20,
    eIDC_DrvInternInterpolFbk1LagLess := 27,
    eIDC_DrvInternInterpolFbk2LagLess := 28,

    eIDC_PosBlockModeFbk1        := 51,
    eIDC_PosBlockModeFbk2        := 52,
    eIDC_PosBlockModeFbk1LagLess := 59,
    eIDC_PosBlockModeFbk2LagLess := 60,

    eIDC_PosCtrlDrvCtrldFbk1        := 259,
    eIDC_PosCtrlDrvCtrldFbk2        := 260,
    eIDC_PosCtrlDrvCtrldFbk1LagLess := 267,
    eIDC_PosCtrlDrvCtrldFbk2LagLess := 268,
```

```
    eIDC_DrvCtrldPositioningFbk1        := 531,
    eIDC_DrvCtrldPositioningFbk2        := 532,
    eIDC_DrvCtrldPositioningFbk1LagLess := 539,
    eIDC_DrvCtrldPositioningFbk2LagLess := 540,

    eIDC_CamFbk1VirtMaster        := -30717,
    eIDC_CamFbk2VirtMaster        := -30716,
    eIDC_CamFbk1VirtMasterLagLess := -30709,
    eIDC_CamFbk2VirtMasterLagLess := -30708,

    eIDC_CamFbk1RealMaster        := -30701,
    eIDC_CamFbk2RealMaster        := -30700,
    eIDC_CamFbk1RealMasterLagLess := -30693,
    eIDC_CamFbk2RealMasterLagLess := -30692,

    eIDC_PhaseSyncFbk1VirtMaster        := -28669,
    eIDC_PhaseSyncFbk2VirtMaster        := -28668,
    eIDC_PhaseSyncFbk1VirtMasterLagLess := -28661,
    eIDC_PhaseSyncFbk2VirtMasterLagLess := -28660,

    eIDC_PhaseSyncFbk1RealMaster        := -28653,
    eIDC_PhaseSyncFbk2RealMaster        := -28652,
    eIDC_PhaseSyncFbk1RealMasterLagLess := -28645,
    eIDC_PhaseSyncFbk2RealMasterLagLess := -28644,

    eIDC_VeloSyncVirtMaster := -24574,
    eIDC_VeloSyncRealMaster := -24558,

    eIDC_MotionProfileFbk1VirtMaster        := -26621,
    eIDC_MotionProfileFbk2VirtMaster        := -26620,
    eIDC_MotionProfileLagLessFbk1VirtMaster := -26613,
    eIDC_MotionProfileLagLessFbk2VirtMaster := -26612,

    eIDC_MotionProfileFbk1RealMaster        := -26605,
    eIDC_MotionProfileFbk2RealMaster        := -26604,
    eIDC_MotionProfileLagLessFbk1RealMaster := -26597,
    eIDC_MotionProfileLagLessFbk2RealMaster := -26596,

    eIDC_PosCtrlDrvCtrld     := 773,
    eIDC_DrvCtrldPositioning := 533,
    eIDC_PosBlockMode        := 565,
    eIDC_VeloSynchronization := 66,

    eIDC_PosSynchronization := 581

);

END_TYPE
```

5.3.2 ST_IndraDriveCs_C1D für Class 1 Diagnose

```
TYPE ST_IndraDriveCs_C1D :
STRUCT
    bOverloadShutdown           : BOOL; (* C1D Bit 0 *)
    bAmplifierOverTempShutdown  : BOOL; (* C1D Bit 1 *)
    bMotorOverTempShutdown      : BOOL; (* C1D Bit 2 *)
    bReserved_3                 : BOOL; (* C1D Bit 3 *)
    bControlVoltageError        : BOOL; (* C1D Bit 4 *)
    bFeedbackError              : BOOL; (* C1D Bit 5 *)
    bReserved_6                 : BOOL; (* C1D Bit 6 *)
    bOverCurrentError           : BOOL; (* C1D Bit 7 *)
    bOverVoltageError           : BOOL; (* C1D Bit 8 *)
    bUnderVoltageError          : BOOL; (* C1D Bit 9 *)
    bReserved_10                : BOOL; (* C1D Bit 10 *)
    bExcessivePosDiviationError : BOOL; (* C1D Bit 11 *)
    bCommunicationErrorBit      : BOOL; (* C1D Bit 12 *)
    bOvertravelLimitExceeeded   : BOOL; (* C1D Bit 13 *)
    bReserved_14                : BOOL; (* C1D Bit 14 *)
    bManufacturerSpecificError  : BOOL; (* C1D Bit 15 *)
END_STRUCT
END_TYPE
```

5.3.3 ST_IndraDriveCsDriveStatus

```
TYPE ST_IndraDriveCsDriveStatus :
STRUCT
    bStatusCmdValProcessing : BOOL;
    bRealTimeStatusBit1     : BOOL;
    bRealTimeStatusBit2     : BOOL;
    bDrvShutdownBitC1D      : BOOL;
    bChangeBitC2D           : BOOL;
    bChangeBitC3D           : BOOL;
    bNotReadyToPowerUp      : BOOL;
    bReadyForPower          : BOOL;
    bReadyForEnable         : BOOL;
    bEnabled                : BOOL;
    iActOpModeParNum        : UINT;
    eActOpMode              : E_IndraDriveCs_DriveOpMode;
    iReserved               : UINT;
END_STRUCT
END_TYPE
```

5.4 SERCOS

5.4.1 E_SoE_AttribLen

Die Enumeration E_SoE_AttribLen im Attribut eines Parameters gibt an, ob der Wert des Parameters ein 2-, 4- oder 8-Byte-Datentyp ist (Einzelwert), oder ob es sich um eine Liste bestehend aus 1-, 2-, 4- oder 8-Byte-Datentypen handelt. Listentypen (mit eSoE_LEN_V...) haben erst die aktuelle Listenlänge in Bytes (in einem 16-bit-Wert), dann die maximale Listenlänge in Bytes (in einem 16-bit-Wert) und dann die eigentliche Liste im angegebenen Datentyp.

Beispiel: Siehe ST_SoE_String [▶ 47] vom Typ eSoE_LEN_V1BYTE.

```
TYPE E_SoE_AttribLen : (
    eSoE_LEN_2BYTE  := 1,
    eSoE_LEN_4BYTE  := 2,
    eSoE_LEN_8BYTE  := 3,
    eSoE_LEN_V1BYTE := 4,
    eSoE_LEN_V2BYTE := 5,
    eSoE_LEN_V4BYTE := 6,
    eSoE_LEN_V8BYTE := 7
    );
END_TYPE
```

Name	Beschreibung
eSoE_LEN_2BYTE	2-Byte-Datentyp (z. B. UINT, INT, WORD, IDN)
eSoE_LEN_4BYTE	4-Byte-Datentyp (z. B. UDINT, DINT, DWORD, REAL)
eSoE_LEN_8BYTE	8-Byte-Datentyp (z. B. ULINT, LINT, LREAL)
eSoE_LEN_V1BYTE	Liste von 1-Byte-Datentypen (z. B. String)
eSoE_LEN_V2BYTE	Liste von 2-Byte-Datentypen (z. B. IDN-Liste)
eSoE_LEN_V4BYTE	Liste von 4-Byte-Datentypen
eSoE_LEN_V8BYTE	Liste von 8-Byte-Datentypen

5.4.2 E_SoE_CmdControl

Die Enumeration E_SoECmdControl bestimmt, ob das Kommando abgebrochen, gesetzt oder gestartet werden soll.

```
TYPE E_SoE_CmdControl :(
    eSoE_CmdControl_Cancel       := 0,
    eSoE_CmdControl_Set          := 1,
    eSoE_CmdControl_SetAndEnable := 3
    );
END_TYPE
```

Name	Beschreibung
eSoE_CmdControl_Cancel	Kommando abbrechen.
eSoE_CmdControl_Set	Kommando setzen.
eSoE_CmdControl_SetAndEnable	Kommando setzen und ausführen.

5.4.3 E_SoE_CmdState

Die Enumeration E_SoE_CmdState beschreibt den Zustand eines SoE-Kommandos.

```
TYPE E_SoE_CmdState :(
    eSoE_CmdState_NotSet                := 0,
    eSoE_CmdState_Set                   := 1,
    eSoE_CmdState_Executed              := 2,
    eSoE_CmdState_SetEnabledExecuted    := 3,
    eSoE_CmdState_SetAndInterrupted     := 5,
    eSoE_CmdState_SetEnabledNotExecuted := 7,
    eSoE_CmdState_Error                 := 15
    );
END_TYPE


eSoE_CmdState_NotSet = 0
- kein Kommando aktiv

eSoE_CmdState_Set = 1
- Kommando gesetzt (vorbereitet) aber (noch) nicht ausgeführt

eSoE_CmdState_Executed = 2
- Kommando wurde ausgeführt

eSoE_CmdState_SetEnabledExecuted = 3
- Kommando gesetzt (vorbereitet) und ausgeführt

eSoE_CmdState_SetAndInterrupted = 5
- Kommando wurde gesetzt aber unterbrochen

eSoE_CmdState_SetEnabledNotExecuted = 7
- Kommandoausführung ist noch aktiv

eSoE_CmdState_Error = 15
- Fehler bei der Kommandoausführung, es wurde in den Fehlerstate
gewechselt
```

5.4.4 E_SoE_Type

Die Enumeration E_SoE_Type beschreibt die Darstellung des Parameterwerts im Attribut des Parameters.

```
TYPE E_SoE_Type : (
    eSoE_Type_BIN       := 0,
    eSoE_Type_UNSIGNED  := 1,
    eSoE_Type_SIGNED    := 2,
    eSoE_Type_HEX       := 3,
    eSoE_Type_TEXT      := 4,
    eSoE_Type_IDN       := 5,
    eSoE_Type_FLOAT     := 6
    );
END_TYPE
```

Über die Enumeration E_SoE_Type wird festgelegt, wie die Daten interpretiert werden können:

Name	Beschreibung
eSoE_Type_BIN	Binär
eSoE_Type_UNSIGNED	Integer ohne Vorzeichen
eSoE_Type_SIGNED	Integer mit Vorzeichen
eSoE_Type_HEX	Hexadezimalzahl
eSoE_Type_TEXT	Text
eSoE_Type_IDN	Parameternummer
eSoE_Type_FLOAT	Fließkommazahl

5.5 SimplePlcMotion

5.5.1 E_CoEDriveEnableState

```
TYPE E_CoEDriveEnableState : (
    eCoEDriveEnableState_ReadyToSwitchOn := 0,
    eCoEDriveEnableState_SwitchedOn := 1,
    eCoEDriveEnableState_OperationEnabled := 2,
    eCoEDriveEnableState_Fault := 3,
    eCoEDriveEnableState_FaultReactionActive := 4,
    eCoEDriveEnableState_NotReadySwitchedOn := 5,
    eCoEDriveEnableState_SwitchedOnDisabled := 6,
    eCoEDriveEnableState_DriveFollows := 7
);
END_TYPE
```

5.5.2 E_DriveMoveVelocityError

```
TYPE E_CoEDriveEnableState : (
    DRIVEPLCERROR_DRIVENOTREADY := 16#4BF0,
    DRIVEPLCERROR_INVALIDDYNAMICSPARAMETER,
    DRIVEPLCERROR_INVALIDVELOCITYSCALING,
    DRIVEPLCERROR_FUNCTIONBLOCKSUDDENLYDISABLED
);
END_TYPE
```

5.5.3 ST_CoEDriveIoInterface

Datenstruktur zur Abbildung des Prozessabbildes eines CoE-Antriebs zur Verwendung der Funktionsbausteine FB_CoEDriveEnable [▶ 41] und FB_CoEDriveMoveVelocity [▶ 42].

```
TYPE ST_CoEDriveIoInterface :
STRUCT
    iControl      : UINT;
    iStatus       : UINT;
    iCmdVelo      : DINT;
    iActVelo      : DINT;
    stAdsAddr     : ST_AmsAddr;
    iChannel      : BYTE;
    eStateMachine : E_CoEDriveEnableState;
END_STRUCT
END_TYPE
```

5.5.4 ST_DriveMoveVelocityOptions

Die Struktur ST_DriveMoveVelocityOptions beschreibt zusätzliche Parameter der Funktionsbausteine FB_CoEDriveMoveVelocity [▶ 42] und FB_SoEDriveMoveVelocity [▶ 44].

```
TYPE ST_DriveMoveVelocityOptions :
STRUCT
    bVelocityUnitRPM          : BOOL;
    fVelocityThreshold        : LREAL;
    fVelocityScalingFactor    : LREAL;
    fFilterTimeActualVelocity : LREAL;
    bInvertDirection          : BOOL;
END_STRUCT
END_TYPE
```

5.5.5 ST_SoEDriveIoInterface

Datenstruktur zur Abbildung des Prozessabbildes eines SoE-Antriebs zur Verwendung der Funktionsbausteine FB_SoEDriveEnable [▶ 43] und FB_SoEDriveMoveVelocity [▶ 44].

```
TYPE ST_SoEDriveIoInterface :
STRUCT
    iMasterControlWord      : WORD;
```

```
    iVelocityCommandValue   : DINT;
    iDriveStatusWord        : WORD;
    iVelocityFeedbackValue  : DINT;
    iState                  : UINT;
    stAdsAddr               : ST_AmsAddr;
    iChannel                : BYTE;
END_STRUCT
END_TYPE
```

6 Beispiele

Beispielprojekt und Beispielkonfiguration für die Diagnose von AX5000

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_Drive/Resources/2307584011.zip

Beispielprojekt und Beispielkonfiguration für die Diagnose von IndraDrive Cs

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_Drive/Resources/2307586955.zip

Mehr Informationen:
www.beckhoff.de/te1000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.de
www.beckhoff.de