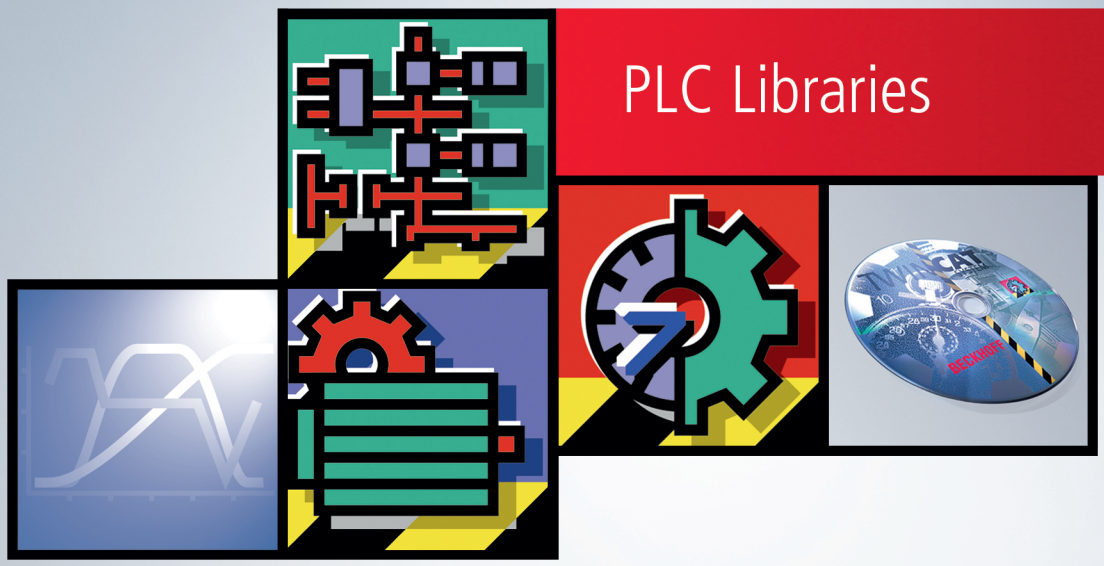
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcDrive



14.10.2022 | Version: 1.1

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 POUs der TcDrive.lib

In dieser Bibliothek sind Funktionen und Funktionsbausteine für SoE-Antriebe enthalten, die per Drive-Referenz auf den Antrieb zugreifen.

Es gibt Unterschiede bei der Verwendung der Drive Libs in Verbindung mit AX5000 und Bosch Rexroth IndraDriveCS. Siehe Beispiel.

Die TcDrive.lib sollte dann verwendet werden, wenn der Antrieb komplett aus der SPS (also ohne NC) verwendet wird. Hierzu wird auf den Antrieb über eine Drive-Referenz zugegriffen. Siehe auch ST_DriveRef [▶ 10]. Bibliotheksintern wird die ST_DriveRef mit der NetID als String verwendet. Zu Verlinkungszwecken wird aber auch eine ST_PlcDriveRef mit der NetID als ByteArray zur Verfügung gestellt. Siehe auch Beispiel bei den jeweiligen Funktionsbausteinen.

Zugriff auf Parameter

Um auf Parameter im Antrieb zuzugreifen, für die kein spezieller Baustein implementiert wurde, können die Bausteine FB_SoERead_ByDriveRef und FB_SoEWrite_ByDriveRef verwendet werden.

Abweichende Implementierung

Die Bausteine FB_SoERead_ByDriveRef und FB_SoEWrite_ByDriveRef sind abweichend in der TcEtherCAT.lib im Ordner SoE-Interface implementiert, da in der TcEtherCAT.lib auch die allgemeinen Bausteine für CoE und FoE implementiert sind.

Funktionen

Name	Beschreibung
F_GetVersionTcDrive [▶ 44]	Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.
F_ConvWordToSTAX5000C1D [▶ 36]	Konvertiert das C1D-Wort (S-0-0011) des AX5000 in eine Struktur ST_AX5000_C1D [▶ 12]
F_ConvWordToSTAX5000DriveStatus [▶ 36]	Konvertiert das Antriebsstatuswort (S-0-0135) des AX5000 in eine Struktur ST_AX5000DriveStatus [▶ 12]
F_ConvWordToSTIndraDriveCsC1D [▶ 43]	Konvertiert das C1D-Wort (S-0-0011) des IndraDrive Cs in eine Struktur ST_IndraDriveCs_C1D [▶ 18]
F_ConvWordToSTIndraDriveCsDriveStatus [▶ 43]	Konvertiert das Antriebsstatuswort (S-0-0135) des IndraDrive Cs in eine Struktur ST_IndraDriveCsDriveStatus [▶ 18]

Funktionsbausteine

Name	Beschreibung
FB_SoEReset_ByDriveRef [▶ 19]	Antriebsreset ausführen (S-0-0099)
FB_SoEWritePassword_ByDriveRef [▶ 20]	Setzen des Antriebspassworts (S-0-0267)
FB_SoEReadDiagMessage_ByDriveRef [▶ 25]	Lesen der Diagnosenachricht (S-0-0095)
FB_SoEReadDiagNumber_ByDriveRef [▶ 27]	Lesen der Diagnosenummer (S-0-0390)
FB_SoEReadDiagNumberList_ByDriveRef [▶ 28]	Lesen der Diagnosenummernliste (bis zu 30 Einträge) (S-0-0375)
FB_SoEReadClassXDiag_ByDriveRef [▶ 29]	Lesen der Class 1 Diagnose (S-0-0011) ... Class 3 Diagnose (S-0-0013)

Name	Beschreibung
FB_SoEExecuteCommand_ByDriveRef [▶ 21]	Ausführen eines Kommandos
FB_SoEWriteCommandControl_ByDriveRef [▶ 22]	Setzen des Command Control
FB_SoEReadCommandState_ByDriveRef [▶ 24]	Prüfen des Kommandostatus
FB_SoERead_ByDriveRef	Lesen eines Parameters, siehe TcEtherCAT.lib (Ordner SoE Interface)
FB_SoEWrite_ByDriveRef	Schreiben eines Parameters, siehe TcEtherCAT.lib (Ordner SoE Interface)
https://infosys.beckhoff.com/content/1031/tcplclibdrive/Resources/10850019723.htm	Lesen der Antriebstemperatur (S-0-0384)
FB_SoEReadMotorTemperature_ByDriveRef [▶ 32]	Lesen der Motortemperatur (S-0-0383)
FB_SoEReadDcBusCurrent_ByDriveRef [▶ 33]	Lesen des Dc-Bus-Stroms (S-0-0381)
https://infosys.beckhoff.com/content/1031/tcplclibdrive/Resources/10850022667.htm	Lesen der Dc-Bus-Spannung (S-0-0380)
FB_SoEAX5000ReadActMainVoltage_ByDriveRef [▶ 37]	Lesen der Netzspannung (P-0-0200)
FB_SoEAX5000SetMotorCtrlWord_ByDriveRef [▶ 38]	Setzen des Motor Control Words (P-0-0096)
FB_SoEAX5000FirmwareUpdate_ByDriveRef [▶ 39]	Automatischer Firmwareupdate für den AX5000

Antriebsreferenz

Siehe ST_DriveRef [▶ 10].

Beispielprojekt und Beispielkonfiguration für AX5000 Diagnose

Siehe https://infosys.beckhoff.com/content/1031/tcplclibdrive/Resources/10850025611.zip,

Beispielprojekt und Beispielkonfiguration für IndraDrive Cs Diagnose

Siehe https://infosys.beckhoff.com/content/1031/tcplclibdrive/Resources/10850027019.zip, (TcDrive.lib ab v0.0.25)

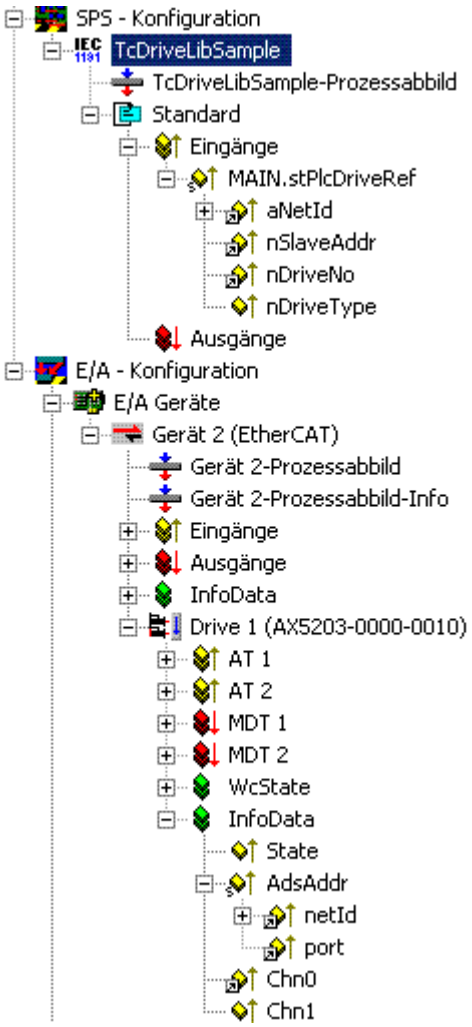
Voraussetzungen

Komponente	Version
TwinCAT auf dem Entwicklungsrechner	2.10 Build 1335 oder höher (IndraDrive Cs: 2.10 Build >1340, 2.11 > Build 1541)
TwinCAT auf dem Windows CE-Image	2.10 Build 1333 oder höher (IndraDrive Cs: 2.10 Build >1340, 2.11 > Build 1541)
TwinCAT auf dem Windows XP-Image	2.10 Build 1333 oder höher (IndraDrive Cs: 2.10 Build >1340, 2.11 > Build 1541)

3 ST_DriveRef für Verwenden mit den Bibliotheksbausteinen

```
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
```

Die Antriebsreferenz kann im System Manager in die SPS gemappt werden. Hierzu muss eine Instanz der ST_PlcDriveRef als "AT %I*" lokiert werden. Anschließend können 'aNetID' auf 'netId', 'nSaveAddr' auf 'port' und 'nDriveNo' auf 'Chn0' (A) bzw. 'Chn1' (B) verlinkt werden. Bei mehrkanaligen Antrieben beziehen sich beide Kanäle auf dieselbe 'netId' und 'port'-Nummer, da es sich um einen EtherCAT-Slave handelt.




```
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
```

Die Bibliotheksbausteine der TcDrive.lib verwenden eine Instanz der ST_DriveRef. Im Unterschied zu der ST_PlcDriveRef wird die NetID als T_AmsNetId (also als STRING(23)) erwartet. Zum Wandeln des Bytearrays kann die Funktion F_CreateAmsNetId() der TcSystem.lib verwendet werden.

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

4 Datentypen

4.1 General SoE DTs

4.1.1 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
```

iActualSize: enthält die aktuelle Länge des Strings (ohne abschließende \0)
iMaxSize: enthält die maximale Länge des Strings (ohne abschließende \0)
strData: enthält den String

4.1.2 ST_SoE_StringEx

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

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

dwAttribute: enthält das Parameter-Attribut
iActualSize: enthält die aktuelle Länge des Strings (ohne abschließende \0)
iMaxSize: enthält die maximale Länge des Strings (ohne abschließende \0)
strData: enthält den String

4.1.3 ST_SoE_DiagNumList

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

```
TYPE ST_SoE_DiagNumList :
STRUCT
    iActualSize     : UINT;
    iMaxSize        : UINT;
    arrDiagNumbers  : ARRAY [0..29] OF UDINT;
END_STRUCT
END_TYPE
```

iActualSize: aktuelle Listenlänge in Bytes
iMaxSize: maximale Listenlänge in Bytes
arrDiagNumbers: Liste der maximal 30 letzen Fehlernummern (als UDINT).

4.2 AX5000 SoE DTs

4.2.1 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; (* C1DBit 13 *)
   bReserved                    : BOOL; (* C1D Bit 14 *)
    bManufacturerSpecificError  : BOOL; (* C1D Bit 15 *)
END_STRUCT
END_TYPE
```

4.2.2 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
```

4.2.3 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
```

4.2.4 E_FwUpdateState

Der 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 Slave-Adresse 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 State des Updates


eFwU_WaitForSlaveState
: Ermitteln des EtherCAT Slave-States


eFwU_StartFwUpdate
: Starten des Firmware-Updates


eFwU_FwUpdateInProgress
: Firmwareupdate aktiv


eFwU_FwUpdateDone
: Firmwareupdate erfolgreich beendet


eFwU_NoFwUpdateRequired
: Kein Firmwareupdate 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 Slave-Adresse 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-States 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
: Anzahl der konfigurierten bzw. 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-Type 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-State schlug fehl
```

4.3 SERCOS

4.3.1 E_SoE_AttribLen

Die 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 Datentypen.

Beispiel siehe ST_SoE_String [▶ 11], der vom Typ eSoE_LEN_V1BYTE ist.

```
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
```

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

4.3.2 E_SoE_CmdControl

Das 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
```

eSoE_CmdControl_Cancel : Kommando abbrechen
eSoE_CmdControl_Set : Kommando setzen
eSoE_CmdControl_SetAndEnable : Kommando setzen und ausführen

4.3.3 E_SoE_CmdState

Der 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
```

4.3.4 E_SoE_Type

Der E_SoE_Type beschreibt die Darstellung des Parameterwertes 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 E_SoE_Type wird festgelegt, wie die Daten interpretiert werden können:

eSoE_Type_BIN : binär
eSoE_Type_UNSIGNED : Integer ohne Vorzeichen
eSoE_Type_SIGNED : Integer ohne Vorzeichen
eSoE_Type_HEX : Hexadezimalzahl
eSoE_Type_TEXT : Text
eSoE_Type_IDN : Parameternummer
eSoE_Type_FLOAT : Fließkommazahl

4.4 IndraDriveCs

4.4.1 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
```

4.4.2 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
```

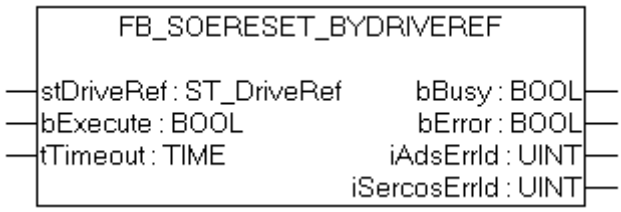
4.4.3 ST_IndraDriveCsDriveStatus

```
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 Funktionsbausteine

5.1 Allgemeine SoE FBs

5.1.1 FB_SoEReset_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReset_ByDriveRef kann ein Antriebsreset (S-0-0099) ausgeführt werden. Bei mehrkanaligen Geräten müssen ggf. beide Kanäle einen Reset ausführen. Die Timeoutzeit muss 10s betragen, da der Reset je nach Fehler bis zu 10s dauern kann.

Ein NC-Reset wird nicht ausgeführt.

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit (10 Sekunden), die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

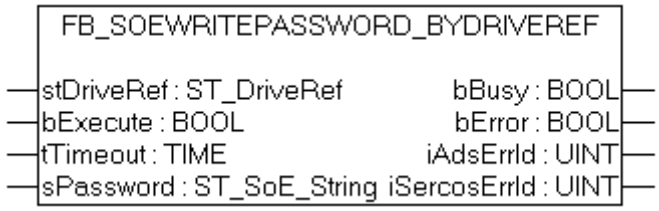
iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

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
```

5.1.2 FB_SoEWritePassword_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

sPassword: enthält das Passwort als Sercos-String

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

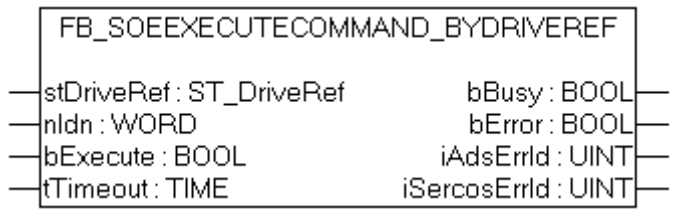
iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

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
```

5.1.3 Kommando FBs

5.1.3.1 FB_SoEExecuteCommand_ByDriveRef



Mit dem Funktionsbaustein FB_SoEExecuteCommand_ByDriveRef kann ein Kommando ausgeführt werden.

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

nIdn: Parameternummer, auf das sich das FB_SoEExecuteCommand_ByDriveRef bezieht, "P_0_IDN + 160" für P-0-0160

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

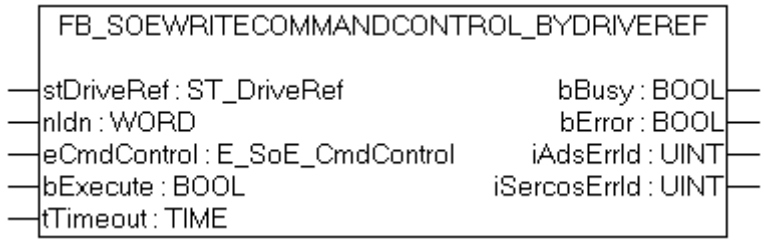
iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

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
```

5.1.3.2 FB_SoEWriteCommandControl_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

nIdn: Parameternummer, auf das sich das FB_SoEWriteCommandControl_ByDriveRef bezieht, z.B. "P_0_IDN + 160" für P-0-0160

eCmdControl: Gibt an, ob das vorbereitet (eSoE_CmdControl_Set := 1), ausgeführt (eSoE_CmdControl_SetAndEnable := 3) oder abgebrochen (eSoE_CmdControl_Cancel := 0) werden soll

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

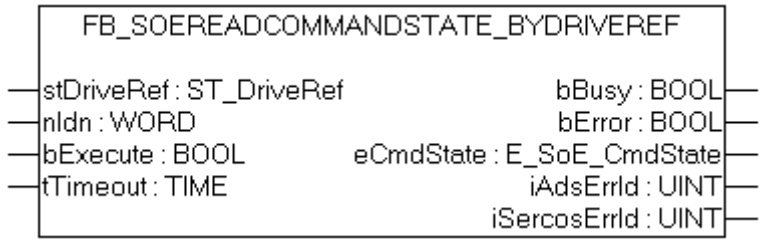
iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

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
```

5.1.3.3 FUNCTION_BLOCK FB_SoEReadCommandState_ByDriveRef



Mit dem Funktionsbaustein FB_SoEReadCommandState_ByDriveRef kann die Kommandoausführung überprüft werden.

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

Idn: Parameternummer, auf das sich das FB_SoEReadCommandState_ByDriveRef bezieht, z.B. "P_0_IDN + 160" für P-0-0160

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

dwAttribute: Liefert das Attribut des Sercos-Parameters.

eCmdState: Liefert den Kommandostatus

```
    eSoE_CmdState_NotSet =
0
- kein Kommando aktiv

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

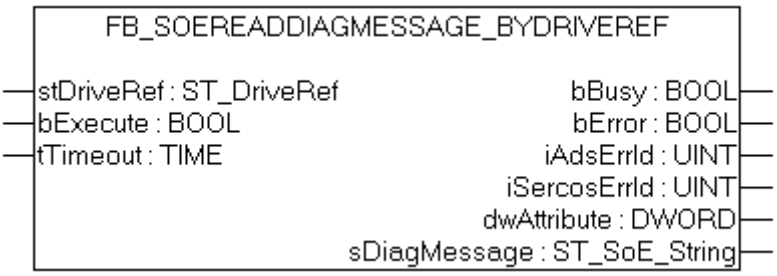
```
    eSoE_CmdState_Executed =
2
- Kommando wurde ausgeführt

    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
```

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
```

5.1.4 Diagnose FBs

5.1.4.1 FB_SoEReadDiagMessage_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

dwAttribute: Liefert das Attribut des Sercos-Parameters.

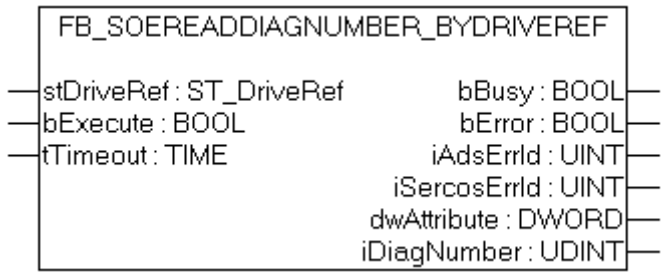
sDiagMessage: Liefert die Diagnosenachricht.

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
```

5.1.4.2 FB_SoEReadDiagNumber_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

dwAttribute: Liefert das Attribut des Sercos-Parameters.

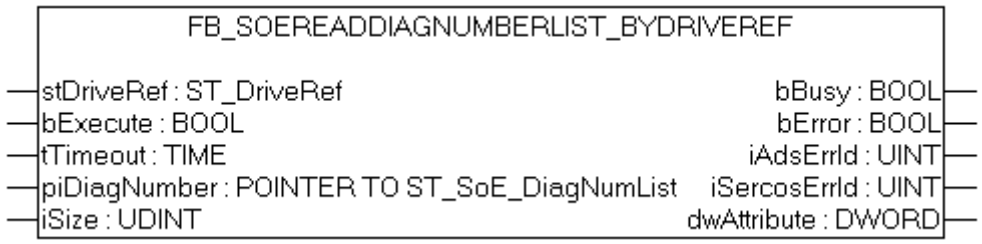
iDiagNumber: Liefert die aktuelle Diagnosenummer.

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
```

5.1.4.3 FB_SoEReadDiagNumberList_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

piDiagNumber: Zeiger auf Liste der letzten max. 30 Fehlernummern. Die Liste besteht aus aktueller und maximaler Anzahl von Bytes in der Liste, sowie den 30 Listeneinträgen

iSize: Größe der Liste in Bytes (als Sizeof())

VAR_OUTPUT

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

```
    iSercosErrId : UINT;
    dwAttribute  : DWORD;
END_VAR
```

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

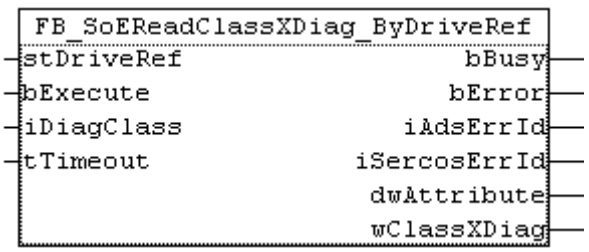
iSercosErrId: Liefert bei einem gesetzten bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

dwAttribute: 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
```

5.1.4.4 FUNCTION_BLOCK 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 [▶ 12] gibt es eine Konvertierungsfunktion F_ConvWordToSTAX5000C1D [▶ 36].

VAR_INPUT

```
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
```

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

iDiagClass: Gibt an, welche Diagnose gelesen werden soll. Die Diagnose Parameter können sich von Hersteller zu Hersteller unterscheiden. Nicht immer sind alle Diagnose Parameter (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: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

dwAttribute: Liefert das Attribut des Sercos-Parameters.

wClassXDiag: Liefert die aktuelle Class X Diagnose.

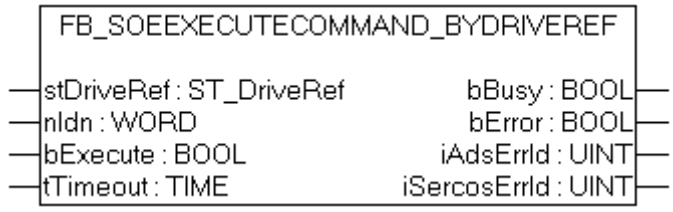
Beispiel

```
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
```

5.1.5 FBs für aktuelle Werte

5.1.5.1 FB_SoEExecuteCommand_ByDriveRef



Mit dem Funktionsbaustein FB_SoEExecuteCommand_ByDriveRef kann ein Kommando ausgeführt werden.

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

nIdn: Parameternummer, auf das sich das FB_SoEExecuteCommand_ByDriveRef bezieht, "P_0_IDN + 160" für P-0-0160

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

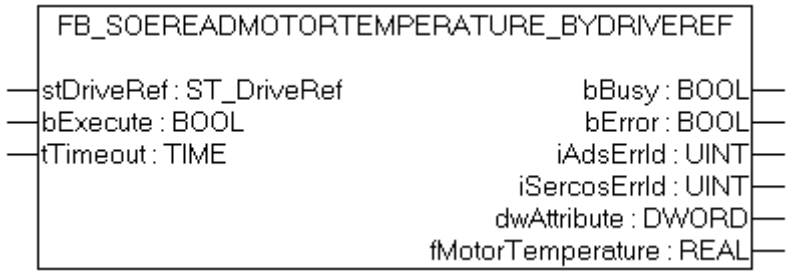
iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

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
```

5.1.5.2 FB_SoEReadMotorTemperature_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

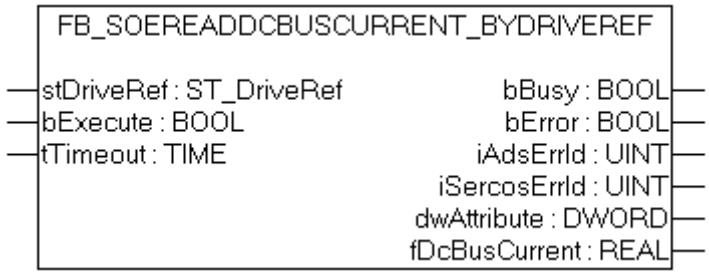
dwAttribute: Liefert das Attribut des Sercos-Parameters.

fMotorTemperature: Liefert die Motortemperatur (z.B. 30.5 entspricht 30.5°C). Falls der Motor keinen Temperatursensor enthält, steht hier 0.0, heißt 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
```

5.1.5.3 FB_SoEReadDcBusCurrent_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

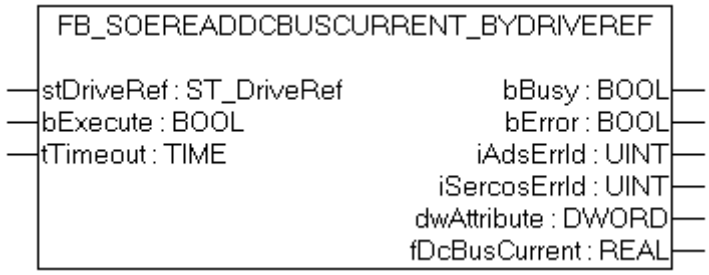
dwAttribute: Liefert das Attribut des Sercos-Parameters.

fDcBusCurrent: Liefert den DC-Bus-Strom (z.B. 2.040 entspricht 2.040A).

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
```

5.1.5.4 FB_SoEReadDcBusCurrent_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

dwAttribute: Liefert das Attribut des Sercos-Parameters.

fDcBusCurrent: Liefert den DC-Bus-Strom (z.B. 2.040 entspricht 2.040A).

Beispiel

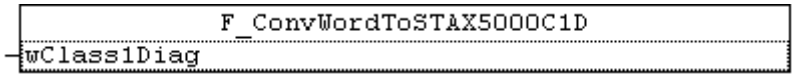
```
fbReadDcBusCurrent : FB_SoEReadDcBusCurrent_ByDriveRef;
bReadDcBusCurrent  : BOOL;
fDcBusCurrent      : REAL;
stPlcDriveRef AT %I* : ST_PlcDriveRef;
stDriveRef         : ST_DriveRef;
IF bInit THEN
    stDriveRef.sNetId     :=F_CreateAmsNetId(stPlcDriveRef.aNetId);
    stDriveRef.nSlaveAddr :=stPlcDriveRef.nSlaveAddr;
    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
```

5.2 AX5000 spezifische FBs

5.2.1 Konvertierungs-FUs

5.2.1.1 F_ConvWordToSTAX5000C1D



Mit dieser Funktion kann die Class 1 Diagnose FB_SoEReadClassXDiag_ByDriveRef [▶ 29] (S-0-0011) in eine Struktur ST_AX5000_C1D [▶ 12] gewandelt werden.

FUNCTION F_ConvWordToSTAX5000C1D : ST_AX5000_C1D

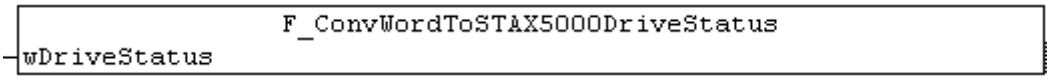
```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

wClass1Diag : Class 1 Diagnose Wort aus S-0-0011 (siehe FB_SoEReadClassXDiag_ByDriveRef [▶ 29]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1334	PC or CX (x86)	TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib
TwinCAT v2.10.0 Build > 1334	CX (ARM)	

5.2.1.2 F_ConvWordToSTAX5000DriveStatus



Mit dieser Funktion kann das Antriebsstatuswort (S-0-0135) in eine Struktur ST_AX5000DriveStatus [▶ 12] gewandelt werden.

FUNCTION F_ConvWordToSTAX5000DriveStatus : ST_AX5000DriveStatus

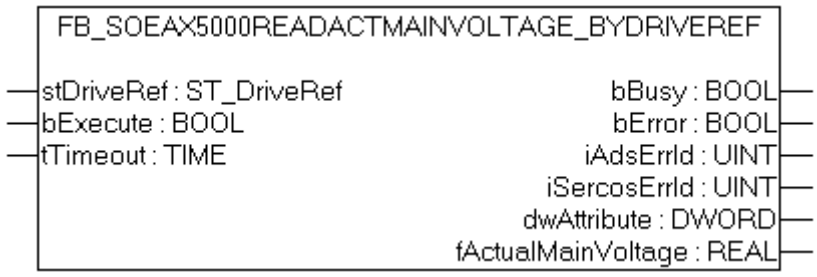
```
VAR_INPUT
    wDriveStatus : WORD;
END_VAR
```

wDriveStatus : Antriebsstatuswort aus S-0-0135 (lesbar per FB_SoE_Read_ByDriveRef, ggf. mapbar).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1334	PC or CX (x86)	TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib
TwinCAT v2.10.0 Build > 1334	CX (ARM)	

5.2.2 FB_SoEAX5000ReadActMainVoltage_ByDriveRef



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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

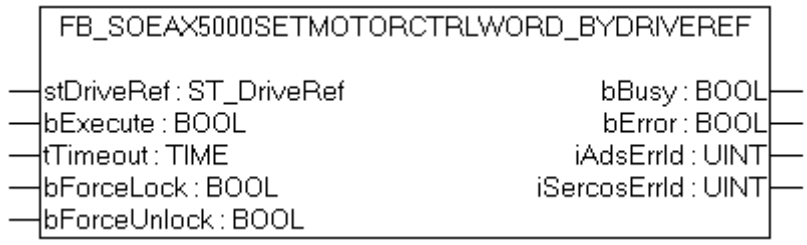
dwAttribute: Liefert das Attribut des Sercos-Parameters.

fActualMainVoltage: Liefert den Scheitelwert der aktuellen Netzspannung des AX5000 (z.B. 303.0 entspricht 303.0V).

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
```

5.2.3 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 setzen oder zu lösen. Im Normalfall wird die Bremse automatisch über das Enable des Antriebs gehandhabt.

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

VAR_INPUT

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

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf.

bForceLock: Bremse unabhängig vom Enable aktivieren.

bForceUnlock: Bremse unabhängig vom Enable lösen.

VAR_OUTPUT

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

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

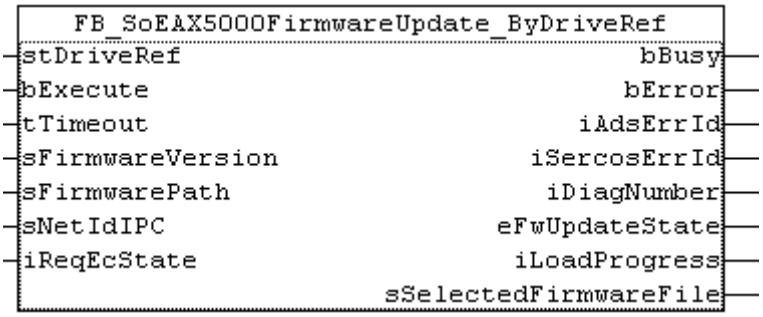
iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

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
```

5.2.4 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.

Zum Updaten wird:
- der konfigurierte Slave-Typ ermittelt, z.B. AX5103-0000-0010
- der aktuelle Slave mit der vorgegebenen Slaveadresse ermittelt, z.B. AX5103-0000-0009
- die aktuelle Slavefirmware ermittelt, z.B. v1.05_b0009
- ein Vergleich der Konfiguration und des gefundenen Slaves, auf Anzahl der Kanäle, Strom, Revision, Firmware ausgeführt
- der Name des erforderlichen Firmware-Files ermittelt und die Datei gesucht
- der Firmwareupdate (falls erforderlich) ausfgefürt
- der aktuelle Slave mit der vorgegebenen Slaveadresse erneut ermittelt
- der Slave in den vorgegebenen EtherCAT-State geschaltet

Ein erfolgreicher Update endet mit **eFwUpdateState = eFwU_FwUpdateDone**, ist der Update nicht erforderlich, wird diese über **eFwUpdateState = eFwU_NoFwUpdateRequired** signalisiert. Der Firmwareupdate erfolgt über den angegeben Kanal (A=0 oder B=1) aus der stDriveRef. Bei zweikanaligen Geräten kann nur einer der beiden Kanäle hierfür verwendet werden. Der andere Kanal signalisiert das über den**eFwUpdateState = eFwU_UpdateViaOtherChannelActive** bzw. **= eFwU_UpdateViaOtherChannel**.

Während des Firmwareupdates (**eFwUpdateState = eFwU_FwUpdateInProgress**) signalisiert**iLoadProgress** den Fortschritt in Prozent.

Während des Updates dürfen die SPS und TwinCAT nicht gestoppt, die EtherCAT-Verbindung nicht unterbrochen und der AX5000 nicht ausgeschaltet werden!

VAR_INPUT

```
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
```

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef [▶ 10].

bExecute: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeout: Da der Firmwareupdate bei großen EtherCAT-Netzwerken länger dauern kann, wird hier nur der Timeout für einzelne interne ADS-Instanzen vorgegeben.

sFirmwareVersion: Gibt die gewünschte Firmware-Version in Form von **v**x.yy**_b**nnnn an, z.B. "**v**1.05**_b**0009" für Version v1.05 Build 0009.

Release-Builds:
- "v1.05_b**00**09" für ein spezifisches Build, zum Beispiel v1.05 Build 0009
- "v1.05_b**00**??" aktuellstes Build einer vorgegebenen Version, zum Beispiel v1.05
- "v1.??_b**00**??" aktuellstes Build einer vorgegebenen Hauptversion, zum Beispiel v1
- "v?.??_b**00**??" aktuellstes Build der aktuellsten Version
- "" aktuellstes Build der aktuellsten Version

Kundenspezifische Firmware-Builds:
- "v1.05_b**10**09" für ein spezifisches Build, zum Beispiel v1.05 Build 0009
- "v1.05_b**10**??" aktuellstes Build einer vorgegebenen Version, zum Beispiel v1.05
- "v1.??_b**10**??" aktuellstes Build einer vorgegebenen Hauptversion, zum Beispiel v1
- "v?.??_b**10**??" aktuellstes Build der aktuellsten Version

...

"v1.05_b**89**09"	für ein spezifisches Build, zum Beispiel v1.05 Build 8909
"v1.05_b**89**??"	aktuellstes Build einer vorgegebenen Version, zum Beispiel v1.05
"v1.??_b**89**??"	aktuellstes Build einer vorgegebenen Hauptversion, zum Beispiel v1
"v?.??_b**89**??"	aktuellstes Build der aktuellsten Version

Debug-Builds:

"v1.05_b**90**09"	für ein spezifisches Build, zum Beispiel v1.05 Build 9009
"v1.05_b**90**??"	aktuellstes Build einer vorgegebenen Version, zum Beispiel v1.05
"v1.??_b**90**??"	aktuellstes Build einer vorgegebenen Hauptversion, zum Beispiel v1
"v?.??_b**90**??"	aktuellstes Build der aktuellsten Version

sFirmwarePath: Gibt den Pfad für den Firmwarepool an, in dem sich die Firmware-Dateien befinden, z.B. "C:\TwinCAT\Io\TcDriveManager\FirmwarePool".

sNetIdIPC: AMS-NetID der Steuerung (IPC).

iReqEcState: Gewünschter EtherCAT-State nach dem Update (nur wenn tatsächlich ein Update ausgeführt wird). Die States sind in der TcEtherCAT.lib als globale Konstanten definiert.

VAR_OUTPUT

```
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
```

bBusy: Dieser Ausgang wird bei der Aktivierung des Funktionsbausteins gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

iDiagNumber: Liefert bei gesetztem bError-Ausgang den Antriebsfehler des letzten Firmware-Updates

eFwUpdateState: Liefert den Status der Firmware-Updates. Siehe E_FwUpdateState [▶ 13].

iLoadProgress: Liefert den Fortschritt des eigentlichen Firmware-Update in Prozent.

sSelectedFirmwareFile: Zeigt den Namen der gesuchten Firmware-Datei an.

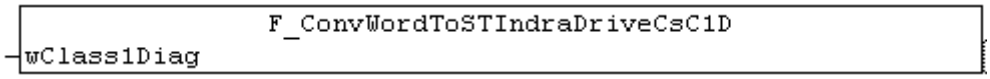
Beispiel

```
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](
           Axis := stNcToPlc[I],
           eFwUpdateState => eFwUpdateState[I],
           sSelectedFirmwareFile => sSelectedFirmwareFile[I],
        );
        IF NOT fbFirmwareUpdate[I].bBusy THEN
           fbFirmwareUpdate[I](bExecute := FALSE, Axis := stNcToPlc[I]);
           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 := 3; (* Error *)
        END_IF
    END_IF
3:
    (* Error handling *)
    iUpdateState := 0;

END_CASE
```

5.3 IndraDriveCs POUs

5.3.1 F_ConvWordToSTIndraDriveCsC1D



Mit dieser Funktion kann die Class 1 Diagnose FB_SoEReadClassXDiag_ByDriveRef [▶ 29] (S-0-0011) in eine Struktur ST_IndraDriveCs_C1D [▶ 18] gewandelt werden.

FUNCTION F_ConvWordToSTIndraDriveCsC1D : ST_IndraDriveCs_C1D

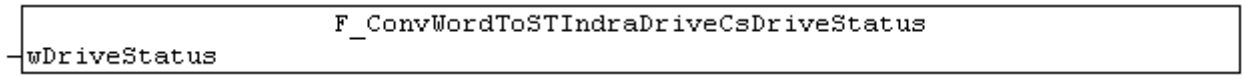
```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

wClass1Diag : Class 1 Diagnose Wort aus S-0-0011 (siehe FB_SoEReadClassXDiag_ByDriveRef [▶ 29]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	PC or CX (x86)	TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	CX (ARM)	

5.3.2 F_ConvWordToSTIndraDriveCsDriveStatus



Mit dieser Funktion kann das Antriebsstatuswort (S-0-0135) in eine Struktur ST_IndraDriveCsDriveStatus [▶ 18] gewandelt werden.

FUNCTION F_ConvWordToSTIndraDriveCsDriveStatus : ST_IndraDriveCsDriveStatus

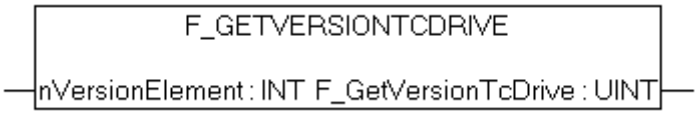
```
VAR_INPUT
    wDriveStatus : WORD;
END_VAR
```

wDriveStatus : Antriebsstatuswort aus S-0-0135 (lesbar per FB_SoE_Read_ByDriveRef, ggf. zu mappen).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	PC or CX (x86)	TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	CX (ARM)	

6 F_GetVersionTcDrive



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

FUNCTION F_GetVersionTcDrive : UINT

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

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1329	PC or CX (x86)	TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib
TwinCAT v2.10.0 Build >= 1329	CX (ARM)	

Mehr Informationen:
www.beckhoff.de/tx1200

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