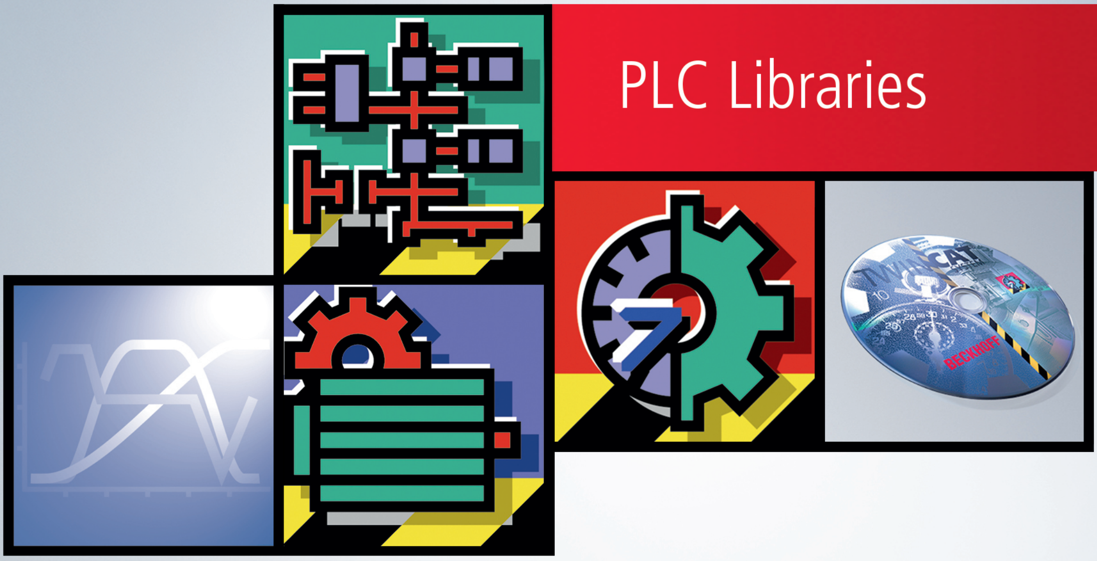
BECKHOFF New Automation Technology

Manual | EN

TX1200

TwinCAT 2 | PLC Library: TcDrive



2022-10-14 | Version: 1.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

Trademarks

Patent Pending

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 POUs of the TcDrive.lib

This library contains functions and functionblocks for SoE-drives. The access to the drive is done via Drive-Reference.

There are differences in the usage of the drive libs in combination with AX5000 and Bosch Rexroth IndraDriveCS. See sample.

The TcDrive.lib should be used, if the drive is completely used by the PLC (no NC). The FBs use a drive reference to communicate with the drive. See ST_DriveRef [▸ 10]. The FBs use internally the ST_DriveRef with the NetID as a string. In order to link a NetID to the drive there is additionally the ST_PlcDriveRef with NetID as a ByteArray. See samples in the documentation of the FBs.

Parameter access

The function blocks FB_SoERead_ByDriveRef and FB_SoEWrite_ByDriveRef can be used to access any parameter in the drive, that have no special acces FB.

Different implmentation

The function blocks FB_SoERead_ByDriveRef and FB_SoEWrite_ByDriveRef are implemented in the TcEtherCAT.lib in the folder SoE-Interface, since the TcEtherCAT.lib contains the general FBs for CoE and FoE.

Functions

Name	Description
F_GetVersionTcDrive [▸ 46]	This function can be used to read the version information of the PLC library.
F_ConvWordToSTAX5000C1D [▸ 37]	Converts the C1D-Word (S-0-0011) of an AX5000 to a structure ST_AX5000_C1D [▸ 13]
F_ConvWordToSTAX5000DriveStatus [▸ 38]	Converts the drive status word (S-0-0135) of an AX5000 to a structure ST_AX5000_DriveStatus [▸ 13]
F_ConvWordToSTIndraDriveCsC1D [▸ 44]	Converts the C1D-Word (S-0-0011) of an IndraDrive Cs to a structure ST_IndraDriveCs_C1D [▸ 19]
F_ConvWordToSTIndraDriveCsDriveStatus [▸ 45]	Converts the drive status word (S-0-0135) of an IndraDrive Cs to a structure ST_IndraDriveCsDriveStatus [▸ 19]

Functionblocks

Name	Description
FB_SoEReset_ByDriveRef [▸ 20]	Execute drive reset (S-0-0099)
FB_SoEWritePassword_ByDriveRef [▸ 21]	Set drive password (S-0-0267)
FB_SoEReadDiagMessage_ByDriveRef [▸ 26]	Read diagnostic message (S-0-0095)
FB_SoEReadDiagNumber_ByDriveRef [▸ 28]	Read diagnostic number (S-0-0390)
FB_SoEReadDiagNumberList_ByDriveRef [▸ 29]	Read diagnostic number list (up to 30 entries) (S-0-0375)
FB_SoEReadClassXDiag_ByDriveRef [▸ 30]	Read class 1 diag (S-0-0011) ... class 3 diag (S-0-0013)

Name	Description
FB_SoEExecuteCommand_ByDriveRef [▶ 22]	Execute command
FB_SoEWriteCommandControl_ByDriveRef [▶ 23]	Set command control
FB_SoEReadCommandState_ByDriveRef [▶ 25]	Read command state
FB_SoERead_ByDriveRef	Read drive parameter, see TcEtherCAT.lib (Folder SoE Interface)
FB_SoEWrite_ByDriveRef	Write drive parameter, see TcEtherCAT.lib (Folder SoE Interface)
https://infosys.beckhoff.com/content/1033/tcplclibdrive/Resources/10850019723/.htm	Read amplifier temperature (S-0-0384)
FB_SoEReadMotorTemperature_ByDriveRef [▶ 33]	Read motor temperature (S-0-0383)
FB_SoEReadDcBusCurrent_ByDriveRef [▶ 35]	Read Dc-Bus-current (S-0-0381)
https://infosys.beckhoff.com/content/1033/tcplclibdrive/Resources/10850022667/.htm	Read Dc-Bus-Voltage (S-0-0380)
FB_SoEAX5000ReadActMainVoltage_ByDriveRef [▶ 38]	Read main voltage (P-0-0200)
FB_SoEAX5000SetMotorCtrlWord_ByDriveRef [▶ 40]	Set motor control word to override brake handling (P-0-0096)
FB_SoEAX5000FirmwareUpdate_ByDriveRef [▶ 41]	Automatic firmware update of the AX5000

Drive reference

See ST_DriveRef [▶ 10].

Sample project and configuration for AX5000 drive diagnose

See https://infosys.beckhoff.com/content/1033/tcplclibdrive/Resources/10850025611/.zip,

Sample project and configuration for IndraDrive Cs drive diagnose

See https://infosys.beckhoff.com/content/1033/tcplclibdrive/Resources/10850027019/.zip, (TcDrive.lib v0.0.25 or higher)

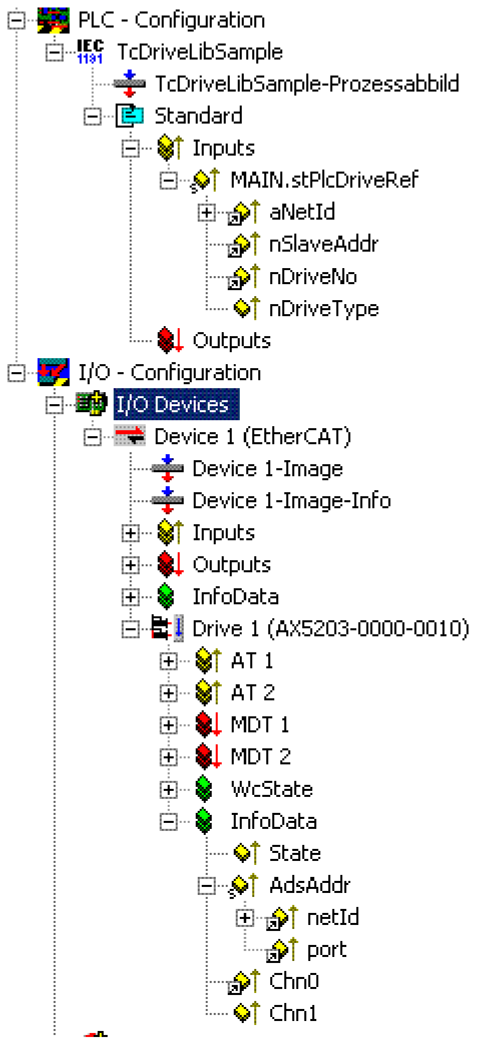
Requirements

Component	Version
TwinCAT on the development PC	2.10 Build 1335 or higher (IndraDrive Cs: 2.10 Build >1340, 2.11Build > 1541)
TwinCAT on the Windows CE-Image	2.10 Build 1333 or higher (IndraDrive Cs: 2.10 Build >1340, 2.11Build > 1541)
TwinCAT on the Windows XP-Image	2.10 Build 1333 or higher (IndraDrive Cs: 2.10 Build >1340, 2.11Build > 1541)

3 ST_DriveRef for using with the functionblocks of the library

```
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
```

The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef, which needs to be declared with "AT %I*". The 'aNetID' goes to 'netId', 'nSlaveAddr' goes to 'port' and 'nDriveNo' goes to 'Chn0' (A) or 'Chn1' (B). Drives with more than one channel share the 'netId' and the 'port'-number because it is the same EtherCAT-Slave for the channels.



```
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
```

The function blocks of the library TcDrive.lib are using an instance of the structure ST_DriveRef. The difference compared to the structure ST_PlcDriveRef is that here the NetID is of the type T_AmsNetId (STRING(23)) instead of the byte array. The function F_CreateAmsNetId() of the TcSystem.lib can be used to convert the byte array to the T_AmsNetId.

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

4 Datatypes

4.1 General SoE DTs

4.1.1 ST_SoE_String

The structure ST_SoE_StringEx describes a String, used for SoE-Access.

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

iActualSize: contains the actual length of the string (without the \0)
iMaxSize: contains the maximum length of the string (without the \0)
strData: contains the string

4.1.2 ST_SoE_StringEx

The structure ST_SoE_StringEx describes a String, used for SoE-Access, inclusive the parameter attribute.

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

dwAttribute: contains the parameter attribute
iActualSize: contains the actual length of the string (without the \0)
iMaxSize: contains the maximum length of the string (without the \0)
strData: contains the string

4.1.3 ST_SoE_DiagNumList

The structure ST_SoE_DiagNumList contains the length of the list (actual, maximum) in bytes and the history of diagnose numbers.

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

iActualSize: actual length of the list in bytes
iMaxSize: maximum length of the list in bytes
arrDiagNumbers: list of the maximum 30 last error numbers (as UDINT).

4.2 AX5000 SoE DTs

4.2.1 ST_AX5000_C1D for 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
```

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

E_FwUpdateState describes the state of the firmware update.

```
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
: initial state


eFwU_CheckCfgIdentity
: reading of the configured slave type (number of channels,
current, revision)


eFwU_CheckSlaveCount
: get configured amount of slaves


eFwU_CheckFindSlavePos
: search slave address in the master object directory


eFwU_WaitForScan
: wait for online scan


eFwU_ScanningSlaves
: online scan of slaves


eFwU_CheckScannedIdentity
: reading of scanned slave types (number of channels, current,
revision)
```

```
eFwU_CheckScannedFirmware
: get firmware version of the drive


eFwU_FindFirmwareFile
: search for firmware file


eFwU_WaitForUpdate
: wait for updates (short delay before the update)


eFwU_WaitForSlaveState
: get EtherCAT slave state


eFwU_StartFwUpdate
: Start firmware update


eFwU_FwUpdateInProgress
: firmware update active


eFwU_FwUpdateDone
: firmware update succeeded


eFwU_NoFwUpdateRequired
: no firmware update required


   eFwU_UpdateViaOtherChannelActive   : Update
via the other drive channel active


eFwU_UpdatedViaOtherChannel
: Updated via the other drive channel

Update Errors

eFwU_GetSlaveIdentityError
: reading of the configured slave type failed, see iAdsErrId


eFwU_GetSlaveCountError
: get configured amount of slaves failed, see iAdsErrId


eFwU_GetSlaveAddrError
: search slave address in the master object directory failed, see
iAdsErrId


eFwU_StartScanError
: start of online scan of slaves failed, see iAdsErrId


eFwU_ScanStateError
: online scan failed, see iAdsErrId


eFwU_ScanIdentityError
: reading of scanned slave types (number of channels, current,
revision) failed, see iAdsErrId


eFwU_GetSlaveStateError
: get EtherCAT slave state failed, see iAdsErrId


eFwU_ScanFirmwareError
: get firmware version of the drive failed, see iAdsErrId +
iSercosErrId


eFwU_FindFileError
: search for firmware file failed, see iAdsErrId
```

```
eFwU_CfgTypeInNoAX5xxx
: the configured slave is not an AX5000


eFwU_ScannedTypeInNoAX5xxx
: the scanned slave is not an AX5000


eFwU_ChannelMismatch
: The amoutn of configured and scanned channels of the AX5000 do
not match

   eFwU_ChannelMismatch_1Cfg_2Scanned : one channel
device configured but two channel device found

   eFwU_ChannelMismatch_2Cfg_1Scanned : two channel
device configured but one channel device found


eFwU_CurrentMismatch
: current of the AX5000 type does not match, i.e. AX5103 (3A)
configured but AX5106 (6A) found


eFwU_FwUpdateError
: general update error, see iAdsErrId


eFwU_ReqSlaveStateError
: switching ot requested EtherCAT state failed, see iAdsErrId
```

4.3 SERCOS

4.3.1 E_SoE_AttribLen

E_SoE_AttribLen of the parameter attribute describes, if the value of the parameter has the data type 2, 4 or 8 byte (single value) or if the parameter is a list of 1-, 2-, 4- or 8-Byte-dat types. List types (with eSoE_LEN_V...) contain the actual list length in bytes (in a 16-bit value), then the maximmum list lenght in bytes (in a 16-bit value) and then the list list in the selected data type.

Sample see ST_SoE_String [▸ 12], this is a list of the type 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
```

eSoE_LEN_2BYTE : 2-Byte data type (i.e. UINT, INT, WORD, IDN)
eSoE_LEN_4BYTE : 4-Byte data type (i.e. UDINT, DINT, DWORD, REAL)
eSoE_LEN_8BYTE : 8-Byte data type (i.e. ULINT, LINT, LREAL)
eSoE_LEN_V1BYTE : List of 1-Byte data types (i.e. String)
eSoE_LEN_V2BYTE : List of 2-Byte data types (i.e. IDN-Liste)
eSoE_LEN_V4BYTE : List of 4-Byte data types
eSoE_LEN_V8BYTE : List of 8-Byte data types

4.3.2 E_SoE_CmdControl

The E_SoECmdControl can be used to abort, set or start a command.

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

eSoE_CmdControl_Cancel : abort a command
eSoE_CmdControl_Set : set a command
eSoE_CmdControl_SetAndEnable : set and execute a command

4.3.3 E_SoE_CmdState

The E_SoE_CmdState describes the state of the SoE-Command.

```
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
- no active command

eSoE_CmdState_Set =
1
- command was set (prepared) but not (yet) executed

eSoE_CmdState_Executed =
2
- command was executed

eSoE_CmdState_SetEnabledExecuted = 3    - command
was set (prepared) and executed

eSoE_CmdState_SetAndInterrupted = 5     -
command was set but interupted

eSoE_CmdState_SetEnabledNotExecuted = 7 - command execution is
still active

eSoE_CmdState_Error =
15
- error during commandexecution, switched to error state
```

4.3.4 E_SoE_Type

The E_SoE_Type describes the representation of the parameter value in the attribute of the parameter.

```
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
```

The E_SoE_Type if the value data have to be interpreted as:

eSoE_Type_BIN : binary
eSoE_Type_UNSIGNED : integer without sign
eSoE_Type_SIGNED : integer with sign
eSoE_Type_HEX : hexa decimal value
eSoE_Type_TEXT : text
eSoE_Type_IDN : parameter number
eSoE_Type_FLOAT : floating point value

4.4 IndraDriveCs DTs

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 for 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; (* C1DBit 13 *)
   bReserved_14                   : BOOL; (* C1D Bit 14 *)
   bManufacturerSpecificError     : BOOL; (* C1D Bit 15 *)
END_STRUCT
END_TYPE
```

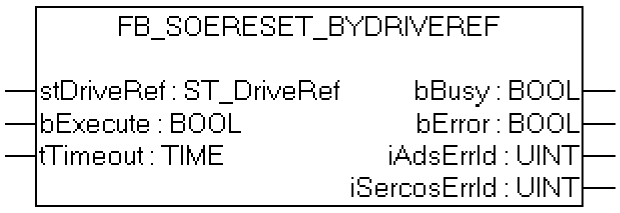
4.4.3 ST_IndraDriveCsDriveStatus

```
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 Functionblocks

5.1 GeneralSoE FB

5.1.1 FB_SoEReset_ByDriveRef



The functionblock FB_SoEReset_ByDriveRef can be used to execute a drive reset (S-0-0099). Drives with more than on channel may require a reset on all channels. The timeout time must be 10s, because the reset can take up to 10s.

An NC-Reset is not executed.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block. The reset can take up to 10s.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

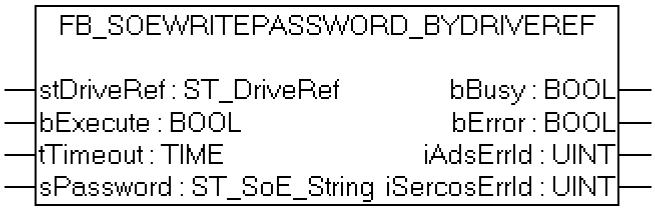
iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

Sample

```
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



The functionblock FB_SoEWritePassword_ByDriveRef can be used to set the drive password (S-0-0267).

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

sPassword: contains the password as Sercos-String

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

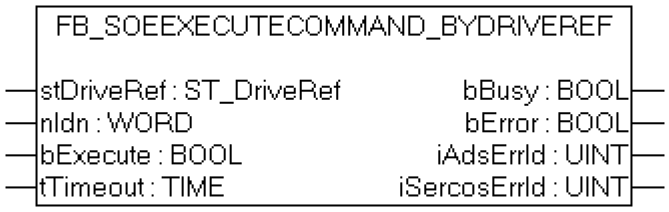
iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

Sample

```
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 Command FB

5.1.3.1 FB_SoEExecuteCommand_ByDriveRef



The functionblock FB_SoEExecuteCommand_ByDriveRef can be used to execute a command.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

nIdn: Parameter number for the command, i.e. "P_0_IDN + 160" for executing a P-0-0160 command

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

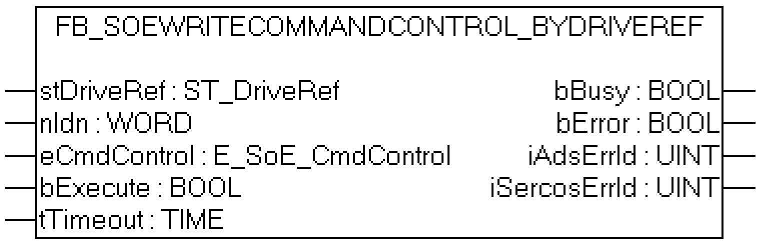
iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

Sample

```
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



The functionblock FB_SoEWriteCommandControl_ByDriveRef can be used to prepare, start or cancel a command.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

nIdn: Parameter number for the command, i.e. "P_0_IDN + 160" for setting the P-0-0160 command

eCmdControl: prepare a command with eSoE_CmdControl_Set := 1, execute a command with eSoE_CmdControl_SetAndEnable := 3 or abort a command with eSoE_CmdControl_Cancel := 0

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

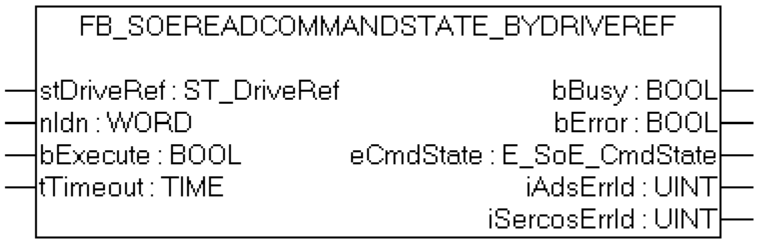
iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

Sample

```
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 FB_SoEReadCommandState_ByDriveRef



The function block FB_SoEReadCommandState_ByDriveRef can be used to read the state of a command.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

nIdn: Parameter number for the command state, i.e. "P_0_IDN + 160" for checking the state of execution of a P-0-0160 command

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

dwAttribute: Supplies the attribut of the Sercos parameter.

eCmdState: Supplies the command state

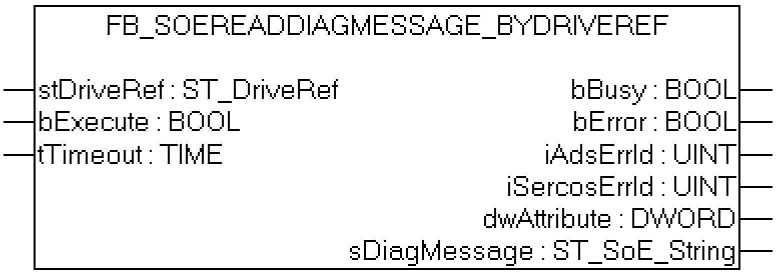
```
    eSoE_CmdState_NotSet =
0
- no command active

    eSoE_CmdState_Set =
1
- command set (prepared) but (not yet) executed

    eSoE_CmdState_Executed =
2
- command was executed

    eSoE_CmdState_SetEnabledExecuted =
3    - command set (prepared) and executed

    eSoE_CmdState_SetAndInterrupted =
5     - command was set but interupted

    eSoE_CmdState_SetEnabledNotExecuted = 7 -
command execution is stll active

    eSoE_CmdState_Error =
15
- error during command execution, switched to error state
```

Sample

```
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 Diagnosis FB

5.1.4.1 FB_SoEReadDiagMessage_ByDriveRef



The functionblock FB_SoEReadDiagMessage_ByDriveRef can be used to read the diagnose message as a Sercos-String (S-0-0095).

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

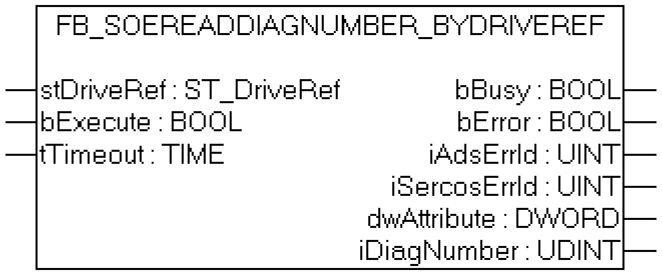
dwAttribute: Supplies the attribut of the Sercos parameter.

sDiagMessage: Supplies the diagnose message.

Sample

```
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



The functionblock FB_SoEReadDiagNumber_ByDriveRef can be used to read the actual diagnose number as UDINT (S-0-0390).

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

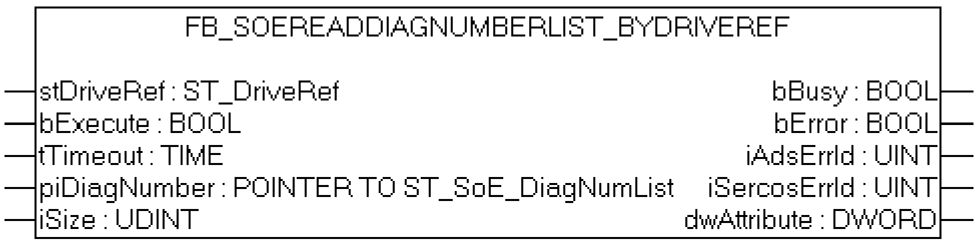
dwAttribute: Supplies the attribut of the Sercos parameter.

iDiagNumber: Supplies the diagnose number.

Sample

```
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



The functionblock FB_SoEReadDiagNumberList_ByDriveRef can be used to read the history of the diagnose numbers as a list (S-0-0375).

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: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

piDiagNumber: Pointer to the list of the last max. 30 error numbers. The list consists of the actual and maximum number of bytes in the list, and of the last 30 list entries

iSize: Size of the list in bytes (as Sizeof())

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

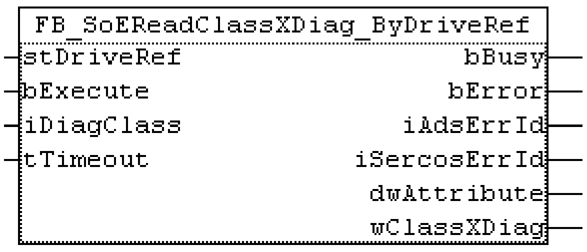
iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

dwAttribute: Supplies the attribut of the Sercos parameter.

Sample

```
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 FB_SoEReadClassXDiag_ByDriveRef



The function block FB_SoEReadClassXDiag_ByDriveRef can be used to read the actual Class 1 Diagnose (S-0-0011) ... Class 3 Diagnose (S-0-0013) as a WORD. There is a conversion function F_ConvWordToSTAX5000C1D [▶ 37]for evaluation of the Class 1 Diagnose as a structure ST_AX5000_C1D [▶ 13].

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: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

iDiagClass: Selects which diagnose should be read. The diagnose info can be different with different drive vendors. Not always all diagnose parameters (C1D ... C3D) or all bits in these parameters are implementiert.

1: Error:	Class 1 Diag (S-0-0011)
2: Warning:	Class 2 Diag (S-0-0012)
3: Informationen:	Class 3 Diag (S-0-0013)

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

dwAttribute: Supplies the attribut of the Sercos parameter.

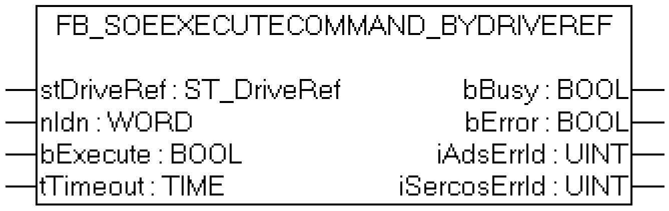
wClassXDiag: Supplies the class X diagnose.

Sample

```
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 FB for current values

5.1.5.1 FB_SoEExecuteCommand_ByDriveRef



The functionblock FB_SoEExecuteCommand_ByDriveRef can be used to execute a command.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

nIdn: Parameter number for the command, i.e. "P_0_IDN + 160" for executing a P-0-0160 command

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

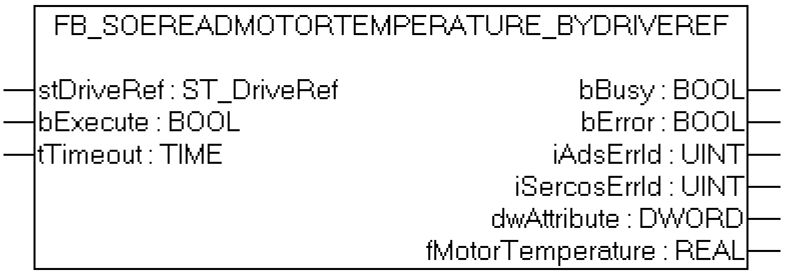
iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

Sample

```
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



The functionblock FB_SoEReadMotorTemperature_ByDriveRef canbe used to read the temperatur of the motor (S-0-0383). If the motor does not contain a temperature sensor then the FB reports 0.0, means 0.0°C.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

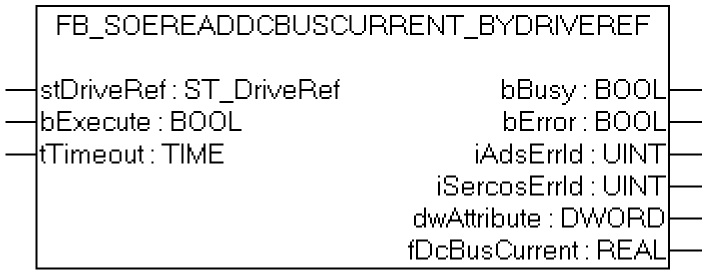
dwAttribute: Supplies the attribut of the Sercos parameter.

fMotorTemperature: Supplies the motor temperature (i.e. 30.5 means 30.5°C). If the motor does not contain a temperature sensor thenthe value is 0.0, means 0.0°C.

Sample

```
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



The functionblock FB_SoEAX5000ReadDcBusCurrent_ByDriveRef can be used to read the DC-Bus-Current (S-0-0381).

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

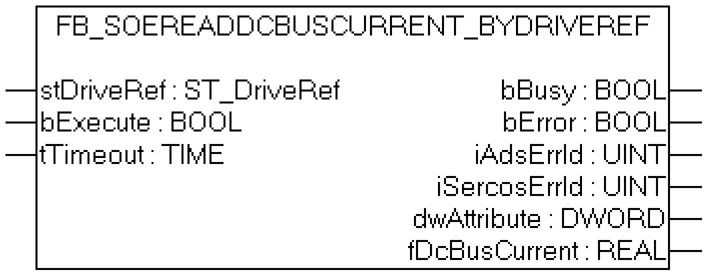
dwAttribute: Supplies the attribut of the Sercos parameter.

fDcBusCurrent: Supplies the DC-Bus-Current (i.e. 2.040 means 2.040A).

Sample

```
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



The functionblock FB_SoEAX5000ReadDcBusCurrent_ByDriveRef can be used to read the DC-Bus-Current (S-0-0381).

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

dwAttribute: Supplies the attribut of the Sercos parameter.

fDcBusCurrent: Supplies the DC-Bus-Current (i.e. 2.040 means 2.040A).

Sample

```
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 specific FB

5.2.1 Conversion FUs

5.2.1.1 F_ConvWordToAX5000C1D

F_ConvWordToSTAX5000C1D
wClass1Diag

This function can be used to convert the Class 1 Diag word FB_SoEReadClassXDiag_ByDriveRef [▸ 30] (S-0-0011) to a structure ST_AX5000_C1D [▸ 13].

FUNCTION F_ConvWordToSTAX5000C1D : ST_AX5000_C1D

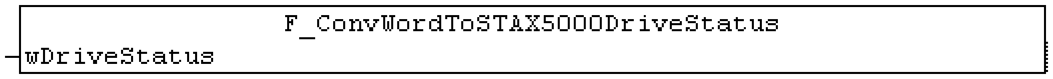
```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

wClass1Diag : Class 1 Diagnose Word from S-0-0011 (see FB_SoEReadClassXDiag_ByDriveRef [▸ 30]).

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1335	PC or CX (x86)	TcDrive.lib (TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib, Standard.Lib, TcBase.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1335	CX (ARM)	

5.2.1.2 F_ConvWordToAX5000DriveStatus



This function can be used to convert the drive status word (S-0-0135, readable via FB_SoERead_ByDriveRef) to a structure ST_AX5000DriveStatus [▶ 13].

FUNCTION F_ConvWordToSTAX5000DriveStatus : ST_AX5000DriveStatus

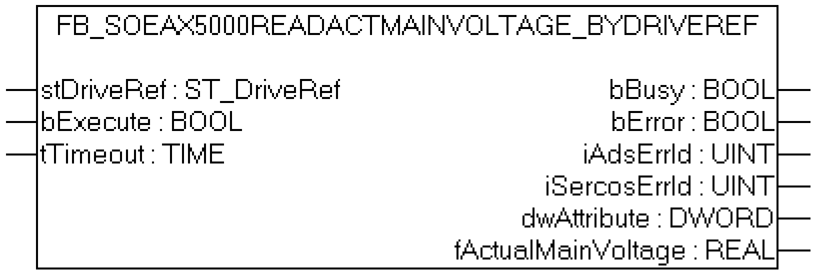
```
VAR_INPUT
    wDriveStatus : WORD;
END_VAR
```

wDriveStatus : Drive status word from S-0-0135 (see FB_SoERead_ByDriveRef).

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1335	PC or CX (x86)	TcDrive.lib (TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib, Standard.Lib, TcBase.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1335	CX (ARM)	

5.2.2 FB_SoEAX5000ReadActMainVoltage_ByDriveRef



The functionblock FB_SoEAX5000ReadActMainVoltage_ByDriveRef can be used to read the main voltage (P-0-0200) of the AX5000.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▸ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated, and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

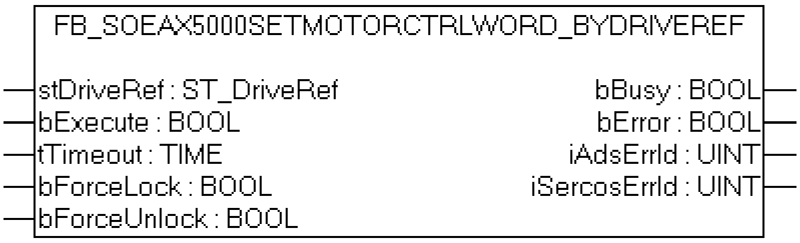
dwAttribute: Supplies the attribut of the Sercos parameter.

fActualMainVoltage: Supplies the main voltage of the AX5000 (i.e. 303.0 means 303.0V).

Sample

```
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



The functionblock FB_SoEAX5000SetMotorCtrlWord_ByDriveRef can be used to set the ForceLock-Bit (Bit 0) or the ForceUnlock-Bit in the motor control word (P-0-0096), in order to set or release the brake. The brake is set and released automatically via the enable of the drive.

TheForceLock-Bit can be used to set the brake independent of the enable, the ForceUnlock-Bit can be used to release the brake independent of the enabl. If ForceLock and ForceUnlock are set simultaneously then the ForceLock (brake locked) has the higher priority.

VAR_INPUT

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

stDriveRef: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

bForceLock: Lock the brake independent of the enable.

bForceUnlock: Release (unlock) the brake independent of the enable.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated, and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

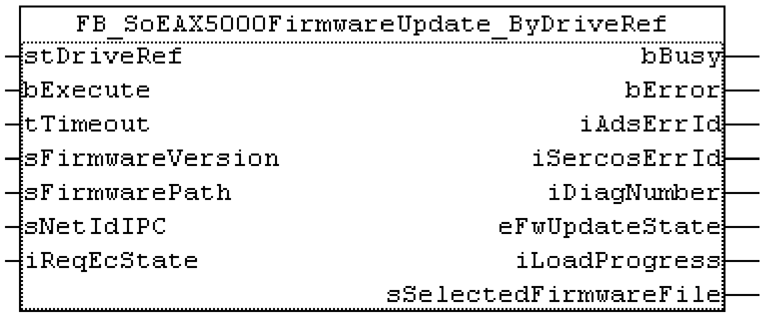
dwAttribute: Supplies the attribut of the Sercos parameter.

Sample

```
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



The functionblock FB_SoEAX5000FirmwareUpdate_ByDriveRef can be used to check and update the firmware of the AX5000 to a requested version (revision and build) or to the newest build of the configured revision.

In order to update the following sequence is executed:
- get configured slave type, i.e. AX5103-0000-0010
- get scanned slave type for the slave address, i.e. AX5103-0000-0009
- get current slave firmware, i.e. v1.05_b0009
- compare configured and scanned slave (number of channels, current, revision, firmware)
- create firmware files name and search for the file
- update firmware (if required)
- rescan slave
- switch the slave to the requested EtherCAT state

A successful updatefinishes with**eFwUpdateState = eFwU_FwUpdateDone**. If the update is not required, then the state returns **eFwUpdateState = eFwU_NoFwUpdateRequired**. The firmware update is executed via the channel of the drive (A=0 or B=1) set in stDriveRef. With two channel devices, the firmware update can only be executed via one of the channels. The other channel signals**eFwUpdateState = eFwU_UpdateViaOtherChannelActive** or **= eFwU_UpdateViaOtherChannel**.

During the firmware update (**eFwUpdateState = eFwU_FwUpdateInProgress**) the update progress is reported via**iLoadProgress in percent**.

During the update the PLC and TwinCAT have to stay in RUN mode, the EtherCAT connection must be maintained and the AX5000 must stay powered up!

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: The drive reference can be linked in the System Manager between PLC and drive. The link can be done to an instance of the ST_PlcDriveRef. The structure ST_PlcDriveRef contains the NetID as byte array. The byte array can be converted to a string. See ST_DriveRef [▶ 10].

bExecute: The block is activated by a rising edge at this input.

tTimeout: The firmware update can take a few minutes, the timeout here defines the timeout for internal ADS instances.

sFirmwareVersion: The required firmware version in form of **v**x.yy_**b**nnnn, i.e. "**v**1.05_**b**0009" for version v1.05 build 0009.

Release builds:

"v1.05_b**00**09"	for specific build, i.e. v1.05 Build 0009
"v1.05_b**00**??"	newstest build of a version, i.e. v1.05
"v1.??_b**00**??"	newstest build of a major version, i.e. v1
"v?.??_b**00**??"	newstest version and newstest build
""	newstest version and newstest build

Customer specific firmware builds:

"v1.05_b**10**09"	for specific build, i.e. v1.05 Build 1009
"v1.05_b**10**??"	newstest build of a version, i.e. v1.05
"v1.??_b**10**??"	newstest build of a major version, i.e. v1
"v?.??_b**10**??"	newstest version and newstest build
...	
"v1.05_b**89**09"	for specific build, i.e. v1.05 Build 8909
"v1.05_b**89**??"	newstest build of a version, i.e. v1.05
"v1.??_b**89**??"	newstest build of a major version, i.e. v1
"v?.??_b**89**??"	newstest version and newstest build

Debug builds:

"v1.05_b**90**09"	for specific build, i.e. v1.05 Build 9009
"v1.05_b**90**??"	newstest build of a version, i.e. v1.05
"v1.??_b**90**??"	newstest build of a major version, i.e. v1
"v?.??_b**90**??"	newstest version and newstest build

sFirmwarePath: The path for the firmware pool, where the firmware files are located, i.e. "C:\TwinCAT\Io\TcDriveManager\FirmwarePool".

sNetIdIPC: AMS-NetID of the controller (IPC).

iReqEcState: Requested EtherCAT state after the update (only if an update is executed). The states are defined in the TcEtherCAT.lib as globale constants.

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: This output is set when the function block is activated, and remains set until an acknowledgement is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

iAdsErrId: Supplies the ADS error code associated with the most recently executed command if the bError output is set.

iSercosErrId: Supplies the Sercos error code associated with the most recently executed command if the bError output is set.

iDiagNumber: Supplies the drive error code associated with the most recently firmware update if the bError output is set.

eFwUpdateState: Supplies the status of the firmware update. See E_FwUpdateState [▶ 14].

iLoadProgress: Supplies the progress of the firmware load in percent.

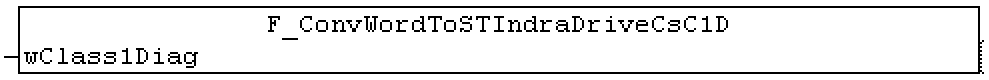
sSelectedFirmwareFile: Supplies the name of the searched firmware file.

Sample

```
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_ConvWordToIndraDriveCsC1D



This function can be used to convert the Class 1 Diag word FB_SoEReadClassXDiag_ByDriveRef [▸ 30] (S-0-0011) to a structure ST_IndraDriveCs_C1D [▸ 19].

FUNCTION F_ConvWordToSTIndraDriveCsC1D : ST_IndraDriveCs_C1D

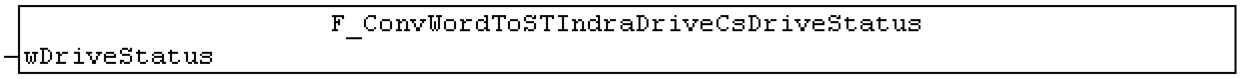
```
VAR_INPUT
    wClass1Diag : WORD;
END_VAR
```

wClass1Diag : Class 1 Diagnosis Word from S-0-0011 (see FB_SoEReadClassXDiag_ByDriveRef [▸ 30]).

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	PC or CX (x86)	TcDrive.lib (TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib, Standard.Lib, TcBase.Lib are included automati-cally)
TwinCAT v2.10.0 Build > 1340 TwinCAT v2.11.0 Build > 1541	CX (ARM)	

5.3.2 F_ConvWordToIndraDriveCsDriveStatus



This function can be used to convert the drive status word (S-0-0135, readable via FB_SoERead_ByDriveRef) to a structure ST_IndraDriveCsDriveStatus [▶ 19].

FUNCTION F_ConvWordToSTIndraDriveCsDriveStatus : ST_IndraDriveCsDriveStatus

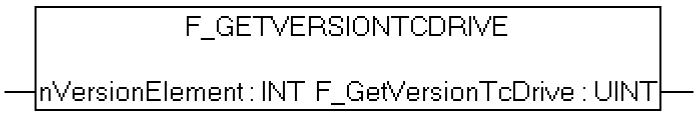
```
VAR_INPUT
    wDriveStatus : WORD;
END_VAR
```

wDriveStatus : Drive status word from S-0-0135 (see FB_SoERead_ByDriveRef).

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	PC or CX (x86)	TcDrive.lib (TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib, Standard.Lib, TcBase.Lib are included automati-cally)
TwinCAT v2.10.0 Build > 1340, TwinCAT v2.11.0 Build > 1541	CX (ARM)	

6 F_GetVersionTcDrive



This function can be used to read PLC library version information.

FUNCTION F_GetVersionTcDrive : UINT

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

nVersionElement : Version element to be read. Possible parameters:

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

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1329	PC or CX (x86)	TcDrive.lib (TcEtherCAT.lib, TcUtilities.Lib, Tc-System.lib, Standard.Lib, TcBase.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1329	CX (ARM)	

More Information:
www.beckhoff.com/tx1200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com