Manual

PLC Lib: Tc2_Drive

TwinCAT 3

Version: 1.0
Date: 2016-05-30
# Table of contents

1 Foreword ................................................................................................................................. 4
  1.1 Notes on the documentation............................................................................................... 4
  1.2 Safety instructions ............................................................................................................... 5

2 Overview .................................................................................................................................... 6

3 ST_DriveRef for use with the library function blocks ................................................................. 9

4 Function blocks ......................................................................................................................... 11
  4.1 AX5000 specific SoE FBs ........................................................................................................ 11
    4.1.1 Conversion functions ...................................................................................................... 11
    4.1.2 FB_SoEAX5000ReadActMainVoltage_ByDriveRef .......................................................... 11
    4.1.3 FB_SoEAX5000SetMotorCtrlWord_ByDriveRef .............................................................. 13
    4.1.4 FB_SoEAX5000FirmwareUpdate_ByDriveRef .................................................................. 14
  4.2 General SoE FBs ..................................................................................................................... 17
    4.2.1 FB_SoEReset_ByDriveRef ............................................................................................... 17
    4.2.2 FB_SoEWritePassword_ByDriveRef ................................................................................ 18
    4.2.3 FB_SoEExecuteCommand_ByDriveRef .......................................................................... 19
    4.2.4 Command FBs ................................................................................................................. 20
    4.2.5 Diagnosis FBs .................................................................................................................. 23
    4.2.6 FBs for current values ....................................................................................................... 29
  4.3 IndraDriveCs FBs .................................................................................................................... 34
    4.3.1 Conversion functions ...................................................................................................... 34
  4.4 F_GetVersionTcDrive ............................................................................................................ 34

5 Data types .................................................................................................................................. 36
  5.1 General SoE DT ...................................................................................................................... 36
    5.1.1 ST_SoE_ String ............................................................................................................... 36
    5.1.2 ST_SoE_StringEx ............................................................................................................. 36
    5.1.3 List types ....................................................................................................................... 36
  5.2 AX5000 SoE DT ....................................................................................................................... 37
    5.2.1 E_FwUpdateState .......................................................................................................... 37
    5.2.2 ST_AX5000_C1D for Class 1 diagnosis .......................................................................... 38
    5.2.3 ST_AX5000DriveStatus .................................................................................................. 39
    5.2.4 E_AX5000_DriveOpMode ............................................................................................... 39
  5.3 SERCOS .................................................................................................................................. 39
    5.3.1 E_SoE_AttribLen ............................................................................................................. 39
    5.3.2 E_SoE_CmdControl ........................................................................................................ 40
    5.3.3 E_SoE_CmdState ............................................................................................................. 40
    5.3.4 E_SoE_Type ................................................................................................................... 40
  5.4 IndraDriveCs ........................................................................................................................... 41
    5.4.1 E_IndraDriveCs_DriveOpMode ....................................................................................... 41
    5.4.2 ST_IndraDriveCs_C1D for Class 1 diagnosis ................................................................. 42
    5.4.3 ST_IndraDriveCsDriveStatus .......................................................................................... 42
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 the applicable national standards.

It is essential that the following notes and explanations are followed when installing and commissioning these components.

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

The documentation has been prepared with care. The products described are, however, constantly under development.

For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics.

In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:

EP1590927, EP1789857, DE102004044764, DE102007017835 with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:

EP0851348, US6167425 with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
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!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="DANGER" /></td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="image" alt="WARNING" /></td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="image" alt="CAUTION" /></td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><img src="image" alt="Attention" /></td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
<tr>
<td><img src="image" alt="Note" /></td>
<td>Tip or pointer This symbol indicates information that contributes to better understanding.</td>
</tr>
</tbody>
</table>
Overview

This library contains functions and function blocks for SoE drives which access the drive by drive reference.

There are three libraries Tc2_Drive, Tc2_NcDrive and Tc2_MC2_Drive. Each of these libraries has its purpose. The range of functions of the libraries is practically identical. The Tc2_NcDrive and The Tc2_MC2_Drive are wrapper function blocks around the function blocks of the Tc2_Drive.

Differences in the use of the Drive libraries with AX5000 and Bosch Rexroth IndraDriveCS should be taken into consideration. See example below.

Tc2_Drive

The Tc2_Drive should be used when the drive is used completely from the SPS (that is, without NC). To this end the drive is accessed via a drive reference. See documentation Tc2_Drive. Within the library the ST_DriveRef is used with the NetID as a string. However, for linking purposes a ST_PlcDriveRef with the NetID as ByteArray is also provided. See also example with the respective function blocks.

Note: To access parameters in the drive, for which no special function block has been implemented, the FB_SoERead_ByDriveRef and FB_SoEWrite_ByDriveRef function blocks can be used.

Note: The FB_SoERead_ByDriveRef and FB_SoEWrite_ByDriveRef function blocks are, in derogation, implemented in the Tc2_EtherCAT in the SoE-Interface folder, since the general function blocks for CoE and FoE are implemented in Tc2_EtherCAT.

Tc2_NcDrive

The Tc2_NcDrive should be used whenever the drive is used via the NC with the Tc2_Nc libraries. To this end the drive is accessed via the NC axis structure (NC_TO_PLC), which is also accessed in the function blocks of Tc2_Nc. The function blocks in the Tc2_NcDrive independently determine the access data to the drive (NetID, address and channel number) via the NC axis ID from the NC_TO_PLC structure. See example for the respective function blocks in the documentation of the Tc2_NcDrive.

Note: To access parameters in the drive, for which no special function block has been implemented, the FB_SoERead and FB_SoEWrite function blocks can be used.

Tc2_MC2_Drive

The Tc2_MC2_Drive should be used whenever the drive is used via NC with the Library Tc2_MC2. To this end the drive is accessed via the MC2 axis reference (AXIS_REF). The function blocks in the Tc2_MC2_Drive independently determine the access data to the drive (NetID, address and channel number) via the NC axis ID from the MC2-Axis reference. See example for the respective function blocks in the documentation of the Tc2_MC2_Drive.

Note: To access parameters in the drive, for which no special function block has been implemented, the FB_SoERead and FB_SoEWrite function blocks can be used.
### Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F_GetVersionTcDrive [13]</strong></td>
<td>This function can be used to read PLC library version information. The function has been replaced by the global structure stLibVersion_Tc2_Drive.</td>
</tr>
<tr>
<td><strong>F_ConvWordToSTAX5000C1D [11]</strong></td>
<td>Converts the CID word (S-0-0011) of the AX5000 to a structure ST_AX5000_C1D [38]</td>
</tr>
<tr>
<td><strong>F_ConvWordToSTAX5000DriveStatus [11]</strong></td>
<td>Converts the drive status word (S-0-0135) of the AX5000 to a structure ST_AX5000DriveStatus [39]</td>
</tr>
<tr>
<td><strong>F_ConvWordToSTIndraDriveC1D [34]</strong></td>
<td>Converts the CID word (S-0-0011) of the IndraDrive Cs to a structure ST_IndraDriveCs_C1D [42]</td>
</tr>
<tr>
<td><strong>F_ConvWordToSTIndraDriveCsDriveStatus [34]</strong></td>
<td>Converts the drive status word (S-0-0135) of the IndraDrive Cs to a structure ST_IndraDriveCsDriveStatus [42]</td>
</tr>
</tbody>
</table>

### Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FB_SoEReset_ByDriveRef [17]</strong></td>
<td>Execute drive reset (S-0-0099)</td>
</tr>
<tr>
<td><strong>FB_SoEWritePassword_ByDriveRef [18]</strong></td>
<td>Setting drive password (S-0-0267)</td>
</tr>
<tr>
<td><strong>FB_SoEExecuteCommand_ByDriveRef [19]</strong></td>
<td>Executing a command</td>
</tr>
<tr>
<td><strong>FB_SoEReadDiagMessage_ByDriveRef [23]</strong></td>
<td>Reading the diagnosis message (S-0-0095)</td>
</tr>
<tr>
<td><strong>FB_SoEReadDiagNumber_ByDriveRef [25]</strong></td>
<td>Reading the diagnosis number (S-0-0390)</td>
</tr>
<tr>
<td><strong>FB_SoEReadDiagNumberList_ByDriveRef [26]</strong></td>
<td>Reading the diagnosis number list (up to 30 entries) (S-0-0375)</td>
</tr>
<tr>
<td><strong>FB_SoEReadClassXDiag_ByDriveRef [27]</strong></td>
<td>Reading the Class 1 diagnosis (S-0-0011) ... Class 3 diagnosis (S-0-0013)</td>
</tr>
<tr>
<td><strong>FB_SoEWriteCommandControl_ByDriveRef [20]</strong></td>
<td>Setting Command Control</td>
</tr>
<tr>
<td><strong>FB_SoEReadCommandState_ByDriveRef [22]</strong></td>
<td>Checking command status</td>
</tr>
<tr>
<td><strong>FB_SoERead_ByDriveRef</strong></td>
<td>Reading a parameter, see Tc2_EtherCAT</td>
</tr>
<tr>
<td><strong>FB_SoEWrite_ByDriveRef</strong></td>
<td>Writing a parameter, see Tc2_EtherCAT</td>
</tr>
<tr>
<td><strong>FB_SoEReadAmplifierTemperature_ByDriveRef [29]</strong></td>
<td>Reading the drive temperature (S-0-0384)</td>
</tr>
<tr>
<td><strong>FB_SoEReadMotorTemperature_ByDriveRef [30]</strong></td>
<td>Reading the motor temperature (S-0-0383)</td>
</tr>
<tr>
<td><strong>FB_SoEReadDcBusCurrent_ByDriveRef [32]</strong></td>
<td>Reading the Dc-Bus current (S-0-0381)</td>
</tr>
<tr>
<td><strong>FB_SoEReadDcBusVoltage_ByDriveRef [31]</strong></td>
<td>Reading the Dc-Bus voltage (S-0-0380)</td>
</tr>
<tr>
<td><strong>FB_SoEAX5000ReadActMainVoltage_ByDriveRef [11]</strong></td>
<td>Reading the mains voltage (P-0-0200)</td>
</tr>
<tr>
<td><strong>FB_SoEAX5000SetMotorCtrlWord_ByDriveRef [13]</strong></td>
<td>Setting Motor Control Words (P-0-0096)</td>
</tr>
<tr>
<td><strong>FB_SoEAX5000FirmwareUpdate_ByDriveRef [14]</strong></td>
<td>Automatic firmware update for the AX5000</td>
</tr>
</tbody>
</table>
### Drive reference


### Sample project and sample configuration for AX5000 diagnosis

See [http://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_Drive/Resources/zip/2307584011.zip](http://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_Drive/Resources/zip/2307584011.zip)

### Sample project and sample configuration for IndraDrive Cs Diagnosis

See [http://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_Drive/Resources/zip/2307586955.zip](http://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_Drive/Resources/zip/2307586955.zip)

### Requests

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT on the development computer</td>
<td>3.1 Build 4016 or higher</td>
</tr>
<tr>
<td>TwinCAT on the Windows CE-Image</td>
<td>3.1 Build 4016 or higher</td>
</tr>
<tr>
<td>TwinCAT on the Windows XP-Image</td>
<td>3.1 Build 4016 or higher</td>
</tr>
</tbody>
</table>
3  ST_DriveRef for use with the library function blocks

The drive reference can be mapped to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be allocated as "AT %I". Subsequently 'aNetId' can be linked to 'netId', 'nSaveAddr' to 'port' and 'nDriveNo' to 'Chn0' (A) or 'Chn1' (B). In the case of multiple-channel drives both channels refer to the same 'netId' and 'port' number, since it is an EtherCAT slave.
The library function blocks of Tc2_Drive use an instance of ST_DriveRef. In contrast to ST_PlcDriveRef the NetID is expected as T_AmsNetId (that is, as a STRING(23)). The F_CreateAmsNetId() function of Tc2_System can be used to convert the Byte array.

```plaintext
stDriveRef.sNetId := F_CreateAmsNetId(stPlcDriveRef.aNetId);
stDriveRef.nSlaveAddr := stPlcDriveRef.nSlaveAddr;
stDriveRef.nDriveNo := stPlcDriveRef.nDriveNo;
stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
```
4 Function blocks

4.1 AX5000 specific SoE FBs

4.1.1 Conversion functions

4.1.1.1 F_ConvWordToSTAX5000C1D

With this function the Class 1 diagnosis FB_SoEReadClassXDiag_ByDriveRef \[27\] (S-0-0011) can be changed to an ST_AX5000_C1D \[38\] structure.

FUNCTION F_ConvWordToSTAX5000C1D : ST_AX5000_C1D
VAR_INPUT
  wClass1Diag : WORD;
END_VAR

wClass1Diag : Class 1 diagnosis Word from S-0-0011 (see FB_SoEReadClassXDiag_ByDriveRef \[27\]).

4.1.1.2 F_ConvWordToSTAX5000DriveStatus

With this function the Drive status word (S-0-0135) can be changed to a structure ST_AX5000DriveStatus \[39\].

FUNCTION F_ConvWordToSTAX5000DriveStatus: ST_AX5000DriveStatus
VAR_INPUT
  wDriveStatus : WORD;
END_VAR

wDriveStatus: Drive status word from S-0-0135 (readable by FB_SoE_Read_ByDriveRef, if necessary mappable).

4.1.2 FB_SoEAX5000ReadActMainVoltage_ByDriveRef

With the FB_SoEAX5000ReadActMainVoltage_ByDriveRef function block the current peak value of the mains voltage of the AX5000 (P-0-0200) can be read.
VAR_INPUT
VAR_INPUT
  stDriveRef : ST_DriveRef;
  bExecute   : BOOL;
  tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

stDriveRef: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

bExecute: The function block is enabled via a positive flank 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 upon the activation of the function block and remains set until feedback occurs.
bError: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId: In the case of a set bError output returns the ADS error code of the last executed command
iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command
dwAttribute: Returns the attributes of the Sercos parameter.
fActualMainVoltage: Returns the peak value of the current mains voltage of the AX5000 (e.g. 303.0V correspond to 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
4.1.3 **FB_SoEAX5000SetMotorCtrlWord_ByDriveRef**

With the FB_SoEAX5000SetMotorCtrlWord_ByDriveRef function block the ForceLock bit (Bit 0) and the ForceUnlock bit in the Motor Control Word (P-0-0096) can be set, in order to activate or release the brake. Normally the brake is automatically controlled via the Enable of the drive.

With the ForceLock-Bit, the brake can be activated independently from the Enable, with the ForceUnlock bit, the brake can be released independently from the Enable. In the case of simultaneously set ForceLock and ForceUnlock, ForceLock (Brake activated) has the higher priority.

**VAR_INPUT**

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

- **stDriveRef**: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: `ST_DriveRef [9]`)

- **bExecute**: The function block is enabled via a positive flank at this input.

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

- **bForceLock**: Activate brake separately from Enable.

- **bForceUnlock**: Release brake separately from Enable.

**VAR_OUTPUT**

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

- **bBusy**: This output is set upon the activation of the function block and remains set until feedback occurs.

- **bError**: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

- **iAdsErrId**: In the case of a set bError output returns the ADS error code of the last executed command.

- **iSercosErrId**: In the case of a set bError output returns the Sercos error of the last executed command.

**Sample**

```plaintext
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;
```

PLC Lib: Tc2_Drive  Version: 1.0
Function blocks

```plaintext
stDriveRef.nDriveType := stPlcDriveRef.nDriveType;
IF (stDriveRef.sNetId <> '') AND (stDriveRef.nSlaveAddr <> 0) THEN
  bInit := FALSE;
END_IF
END_IF
IF bSetMotorCtrlWord AND NOT bInit THEN
  fbSetMotorCtrlWord(
    stDriveRef := stDriveRef,
    bExecute := TRUE,
    tTimeout := DEFAULT_ADS_TIMEOUT,
    bForceLock := bForceLock,
    bForceUnlock := bForceUnlock
  );
  IF NOT fbSetMotorCtrlWord.bBusy THEN
    fbSetMotorCtrlWord(stDriveRef := stDriveRef, bExecute := FALSE);
    bSetMotorCtrlWord := FALSE;
  END_IF
END_IF
```

4.1.4 FB_SoEAX5000FirmwareUpdate_ByDriveRef

With the FB_SoEAX5000FirmwareUpdate_ByDriveRef function block the Firmware of the AX5000 can be checked and automatically changed to a given version (Revision and Build) or to the current Build of the configured revision.

For updating:

- the configured slave type is determined, e.g. AX5103-0000-0010
- the current slave is determined with the predefined slave address, e.g. AX5103-0000-0009
- the current slave firmware is determined, e.g. v1.05_b0009
- a comparison of the configuration and the found slave regarding number of channels, current, revision, firmware is made
- the name of the required firmware file is determined and a search for the file performed
- the firmware update (if required) is executed
- the current slave with the predefined slave address is determined again
- the slave is switched to the predefined EtherCAT state

A successful update ends with eFwUpdateState = eFwU_FwUpdateDone. If the update is not required, this is signaled via eFwUpdateState = eFwU_NoFwUpdateRequired. The firmware update is made via the specified channel (A=0 or B=1) from the stDriveRef. In the case of two-channel devices only one of the two channels can be used. The other channel signals eFwUpdateState = eFwU_UpdateViaOtherChannelActive or = eFwU_UpdateViaOtherChannel.

During the firmware update (eFwUpdateState = eFwU_FwUpdateInProgress) iLoadProgress signals the progress as a percentage.

---

CAUTION

During the update the PLC and TwinCAT must not be stopped, the EtherCAT connection must not be interrupted and der AX5000 must not be switched off!
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\Firmware-Pool" *)
sNetIdIPC : T_AmsNetId;
iReqEcState : UINT := ECDEVICE_STATE_OP;
END_VAR

stDriveRef: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

bExecute: The function block is enabled via a positive flank at this input.

tTimeout: Since the firmware update can take a long time for large EtherCAT networks, only the timeout for individual internal ADS instances is predefined here.

sFirmwareVersion: Specifies the desired firmware version in the form of vx.yy_bnnnn, e.g. "v1.05_b0009" for Version v1.05 Build 0009.

Release-Builds:

"v1.05_b0009" for a specific build, for example v1.05 Build 0009
"v1.05_b00??" most current build of a predefined version, for example v1.05
"v1.??_b00??" most current build of a predefined main version, for example v1
"v.??_b00??" most current build of the most current version

Custom Firmware-Builds:

"v1.05_b1009" for a specific build, for example v1.05 Build 0009
"v1.05_b10??" most current build of a predefined version, for example v1.05
"v1.??_b10??" most current build of a predefined main version, for example v1
"v.??_b10??" most current build of the most current version

Debug-Builds:

"v1.05_b9009" for a specific build, for example v1.05 Build 9009
"v1.05_b90??" most current build of a predefined version, for example v1.05
"v1.??_b90??" most current build of a predefined main version, for example v1
"v.??_b90??" most current build of the most current version

sFirmwarePath: Specifies the path for the firmware pool in which the firmware files are located, e.g. "C: \TwinCAT\Io\TcDriveManager\FirmwarePool".

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

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

VAR_OUTPUT

VAR_OUTPUT
bBusy : BOOL;
bError : BOOL;
iAdsErrId : UINT;
iSercosErrId : UINT;
iDiagNumber : UDINT;
eFwUpdateState : E_FwUpdateState;
iLoadProgress : INT;

PLC Lib: Tc2_Drive
Version: 1.0
Function blocks

sSelectedFirmwareFile : STRING(MAX_STRING_LENGTH); (* found firmware file, e.g. "AX5yxx_xx0000_v1_05_b0009.efw" *)
END_VAR

bBusy: This output is set upon the activation of the function block and remains set until feedback occurs.

bError: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

iAdsErrId: In the case of a set bError output returns the ADS error code of the last executed command.

iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command.

iDiagNumber: In the case of a set bError output returns the drive error of the last executed firmware update.

eFwUpdateState: Returns the status of the firmware update. See E_FwUpdateState[37].

iLoadProgress: Returns the progress of the actual firmware update as a percentage.

sSelectedFirmwareFile: Displays the name of the firmware file being searched for.

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;
  firmwareVersion : 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 := T_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,
        iRegEcState := EC_DEVICE_STATE_OP,
        eFwUpdateState => eFwUpdateState[I],
      );
    END_FOR
    iUpdateState := 2;
2:

bAnyBusy := FALSE;
bAnyError := FALSE;
FOR I := 1 TO iNumOfDrives DO
    fbFirmwareUpdate[I](
        eFwUpdateState => eFwUpdateState[I],
        sSelectedFirmwareFile => sSelectedFirmwareFile[I],
    );
    IF NOT fbFirmwareUpdate[I].bBusy THEN
        fbFirmwareUpdate[I](bExecute := FALSE);
        IF fbFirmwareUpdate[I].bError THEN
            bAnyError := TRUE;
        END_IF
    END_IF
END_FOR
END_CASE

4.2 General SoE FBs

4.2.1 FB_SoEReset_ByDriveRef

Using the FB_SoEReset_ByDriveRef function block a drive reset (S-0-0099) can be executed. In the case of multiple-channel devices if necessary, both channels will have to perform a reset. The timeout period must be 10s, since depending on the error the reset can take up to 10s.

An NC reset will not be performed.

VAR_INPUT

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

stDriveRef: The reference to the drive can be linked directly to the SPS in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

bExecute: The function block is enabled via a positive flank at this input.

tTimeout: Maximum Time (10 seconds), which must not be exceeded in 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 upon the activation of the function block and remains set until feedback occurs.

**bError**: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

**iAdsErrId**: In the case of a set bError output returns the ADS error code of the last executed command.

**iSercosErrId**: In the case of a set bError output returns the Sercos error of the last executed command.

### Sample

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

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

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

### 4.2.2 FB_SoEWritePassword_ByDriveRef

Using the FB_FB_SoEWritePassword_ByDriveRef function block the drive password (S-0-0267) can be set.

#### VAR_INPUT

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

**stDriveRef**: The reference to the drive can be linked directly to the SPS in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

**bExecute**: The function block is enabled via a positive flank at this input.

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

**sPassword**: Contains the password as a Sercos string.
VAR_OUTPUT

VAR_OUTPUT
bBusy : BOOL;
LastError : BOOL;
iAdsErrId : UINT;
iSercosErrId : UINT;
END_VAR

bBusy: This output is set upon the activation of the function block and remains set until feedback occurs.

bError: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

iAdsErrId: In the case of a set bError output returns the ADS error code of the last executed command.

iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command.

Sample

FB_ExecuteCommand_ByDriveRef

VAR_INPUT

VAR_INPUT
stDriveRef : ST_DriveRef;
Idn : WORD;
bExecute : BOOL;
tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
iAdsErrId : UINT;
iSercosErrId : UINT;
END_VAR

With the FB_ExecuteCommand_ByDriveRef function block a command can be executed.

VAR_INPUT

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

stDriveRef: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])
nIdn: Parameter number, to which the FB_SoEExecuteCommand_ByDriveRef refers, "P_0_IDN + 160" for P-0-0160

bExecute: The function block is enabled via a positive flank 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 after the bBusy output has been reset when an error occurs in the transmission of the command.

iAdsErrId: In the case of a set bError output returns the ADS error code of the last executed command.

iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command.

Sample

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

4.2.4 Command FBs

4.2.4.1 FB_SoEWriteCommandControl_ByDriveRef

FB_SoEWriteCommandControl_ByDriveRef

stDriveRef : ST_DriveRef
nIdn : WORD
eCmdControl : E_SoE_CmdControl
bExecute : BOOL
bBusy : BOOL
bError : BOOL
iAdsErrId : UINT
iSercosErrId : UINT
tTimeout : TIME
With the FB_SoEWriteCommandControl_ByDriveRef function block a command can either be prepared, started or aborted.

**VAR_INPUT**

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

*stDriveRef*: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

*nIdn*: Parameter number, to which FB_SoEReadCommandState_ByDriveRef refers, e.g. "P_0_IDN + 23" for P-0-0023

*ECmdControl*: Indicates, if a command should be prepared (eSoE_CmdControl_Set := 1), executed (eSoE_CmdControl_SetAndEnable := 3) or aborted (eSoE_CmdControl_Cancel := 0).

*bExecute*: The function block is enabled via a positive flank 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 after the bBusy output has been reset when an error occurs in the transmission of the command.

*iAdsErrId*: In the case of a set bError output returns the ADS error code of the last executed command.

*iSercosErrId*: In the case of a set bError output returns the Sercos error of the last executed command.

**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
    );
```
Function blocks

4.2.4.2  FB_SoEReadCommandState_ByDriveRef

With the FB_SoEReadCommandState_ByDriveRef function block the execution of the command can be checked.

VAR_INPUT

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

stDriveRef: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

Idn: Parameter number, to which FB_SoEReadCommandState_ByDriveRef refers, e.g. "P_0_IDN + 160" for P-0-0160

bExecute: The function block is enabled via a positive flank 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 after the bBusy output has been reset when an error occurs in the transmission of the command.

iAdsErrId: In the case of a set bError output returns the ADS error code of the last executed command.

iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command.

dwAttribute: Returns the attributes of the Sercos parameter.

eCmdState: Returns the command status

- eSoE_CmdState_NotSet = 0
  - kein Kommando aktiv

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

### 4.2.5 Diagnosis FBs

#### 4.2.5.1 FB_SoEReadDiagMessage_ByDriveRef

With the FB_SoEReadDiagMessage_ByDriveRef function block the diagnosis message can be read as a Sercos String (S-0-0095).
VAR_INPUT

stDriveRef : ST_DriveRef;
bExecute   : BOOL;
tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;

**stDriveRef**: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

**bExecute**: The function block is enabled via a positive flank 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 after the bBusy output has been reset when an error occurs in the transmission of the command.

**iAdsErrId**: In the case of a set bError output returns the ADS error code of the last executed command.

**iSercosErrId**: In the case of a set bError output returns the Sercos error of the last executed command.

**dwAttribute**: Returns the attributes of the Sercos parameter.

**sDiagMessage**: Returns the diagnosis message.

Sample

```pascal
fbDiagMessage : FB_NoEReadDiagMessage_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
```
With the FB_SoEReadDiagNumber_ByDriveRef function block the current diagnosis number can be read as UDINT (S-0-0390).

**VAR_INPUT**

VAR_INPUT

| stDriveRef : ST_DriveRef; |
| bExecute : BOOL; |
| tTimeout : TIME := DEFAULT_ADS_TIMEOUT; |

**stDriveRef**: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

**bExecute**: The function block is enabled via a positive flank 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; |

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

**bError**: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

**iAdsErrId**: In the case of a set bError output returns the ADS error code of the last executed command.

**iSercosErrId**: In the case of a set bError output returns the Sercos error of the last executed command.

**dwAttribute**: Returns the attributes of the Sercos parameter.

**iDiagNumber**: Returns the current diagnosis number.

**Sample**

```plaintext
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
```
Function blocks

4.2.5.3  FB_SoEReadDiagNumberList_ByDriveRef

With the FB_SoEReadDiagNumberList_ByDriveRef function block a history of the diagnosis numbers can be read as 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 reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

bExecute: The function block is enabled via a positive flank at this input.

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

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

iVar: 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 after the bBusy output has been reset when an error occurs in the transmission of the command.

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

iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command.
**Function blocks**

**dwAttribute**: Returns the attributes of the Sercos parameter.

**Sample**

```plaintext
Sample

fbDiagNumberList : FB_SoEReadDiagNumberList_ByDriveRef;
bdiaTempList     : 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 bdiaTempList 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);
    bdiaTempList := FALSE;
  END_IF
END_IF
```

**4.2.5.4  FB_SoEReadClassXDiag_ByDriveRef**

With the **FB_SoEReadClassXDiag_ByDriveRef** function block the current class 1 diagnosis (S-0-0011) ... class 3 diagnosis (S-0-0013) can be read as WORD. For the evaluation of the class 1 diagnosis as structure **ST_AX5000_C1D [11]** there is a conversion function **F_ConvWordToSTAX5000C1D [11]**.

**VAR_INPUT**

```plaintext
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 reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of **ST_PlcDriveRef** must be used and the NetID of the Byte array converted to a string. (Type: **ST_DriveRef [9]**)

**bExecute**: The function block is enabled via a positive flank at this input.

**iDiagClass**: Specifies which diagnosis should be read. The diagnosis parameters can vary from manufacturer to manufacturer. Not all diagnosis parameters (C1D ... C3D) or all bits in it are always implemented.
Function blocks

1: Errors: Class 1 Diag (S-0-0011)
2: Warnings: Class 2 Diag (S-0-0012)
3: Information: Class 3 Diag (S-0-0013)

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

### VAR_OUTPUT

```plaintext
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 after the bBusy output has been reset when an error occurs in the transmission of the command.

**iAdsErrId:** In the case of a set bError output returns the ADS error code of the last executed command.

**iSercosErrId:** In the case of a set bError output returns the Sercos error of the last executed command.

**dwAttribute:** Returns the attributes of the Sercos parameter.

**wClassXDiag:** Returns the current Class X Diagnosis.

### Sample

```plaintext
SAMPLE
  fbClassXDiag : FB_SoEReadClassXDiag_ByDriveRef;
  bClassXDiag  : BOOL;
  iDiagClass   : USINT := 1;
  wClass1Diag  : WORD;
  stAX5000C1D  := ST_AX5000_C1D;
  wClass2Diag  : 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 bClassXDiag AND NOT bInit THEN
    fbClassXDiag(
      stDriveRef := stDriveRef,
      bExecute := TRUE,
      iDiagClass := iDiagClass,
      tTimeout := DEFAULT_ADS_TIMEOUT
    );
  END_IF

  IF NOT fbClassXDiag.bBusy THEN
    fbClassXDiag(stDriveRef := stDriveRef, bExecute := FALSE);
    bClassXDiag := FALSE;
  END_CASE

  CASE fbClassXDiag.iDiagClass OF
    1:
      wClass1Diag := fbClassXDiag.wClassXDiag;
      stAX5000C1D := F_ConvWordToSTAX5000C1D(wClass1Diag);
    2:
      wClass2Diag := fbClassXDiag.wClassXDiag;
  END_CASE
```

Version: 1.0
4.2.6  FBs for current values

4.2.6.1  FB_SoEReadAmplifierTemperature_ByDriveRef

With the FB_SoEReadAmplifierTemperature_ByDriveRef function block the temperature of the drive (S-0-0384) can be read.

VAR_INPUT

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

stDriveRef: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

bExecute: The function block is enabled via a positive flank at this input.

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

VAR_OUTPUT

bBusy : BOOL;
LastError : BOOL;
iAdsErrId : UINT;
iSercosErrId : UINT;
dwAttribute : DWORD;
fAmplifierTemperature : REAL;
END_VAR

bBusy: This output is set upon the activation of the function block and remains set until feedback occurs.

bLastError: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

iAdsErrId: In the case of a set bLastError output returns the ADS error code of the last executed command.

iSercosErrId: In the case of a set bLastError output returns the Sercos error of the last executed command.

dwAttribute: Returns the attributes of the Sercos parameter.

fAmplifierTemperature: Returns the drive temperature (e.g. 26.2 corresponds to 26.2 °C).

Sample

Sample: 

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

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


With the FB_SoEReadMotorTemperature_ByDriveRef function block, the temperature of the motor (S-0-0383) can be read. If the motor does not contain a temperature sensor, reads 0.0, means 0.0 °C.

**VAR_INPUT**

- **stDriveRef**: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef)
- **bExecute**: The function block is enabled via a positive flank at this input.
- **tTimeout**: Maximum time allowed for the execution of the function block.

**VAR_OUTPUT**

- **bBusy**: This output is set upon the activation of the function block and remains set until feedback occurs.
- **bError**: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
- **iAdsErrId**: In the case of a set bError output returns the ADS error code of the last executed command.
- **iSercosErrId**: In the case of a set bError output returns the Sercos error of the last executed command.
**dwAttribute**: Returns the attributes of the Sercos parameter.

**fMotorTemperature**: Returns the motor temperature (e.g. 30.5 corresponds to 30.5 °C). If the motor does not contain a temperature sensor, reads 0.0, means 0.0 °C.

**Sample**

```
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
```

**4.2.6.3 FB_SoEReadDcBusVoltage_ByDriveRef**

With the FB_SoEReadDcBusVoltage_ByDriveRef function block the DC-Bus voltage of the drive (S-0-0380) can be read.

**VAR_INPUT**

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

**stDriveRef**: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

**bExecute**: The function block is enabled via a positive flank at this input.

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

VAR_OUTPUT

- **bBusy**: This output is set upon the activation of the function block and remains set until feedback occurs.

- **bError**: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

- **iAdsErrId**: In the case of a set bError output returns the ADS error code of the last executed command.

- **iSercosErrId**: In the case of a set bError output returns the Sercos error of the last executed command.

- **dwAttribute**: Returns the attributes of the Sercos parameter.

- **fDcBusVoltage**: Returns the DC-Bus current (e.g. 2.040 corresponds to 2.040 A).

**Sample**

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

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

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

**4.2.6.4 FB_SoEReadDcBusCurrent_ByDriveRef**

With the FB_SoEAX5000ReadDcBusCurrent_ByDriveRef function block the DC-Bus current (S-0-0381) can be read.
VAR_INPUT

stDriveRef : ST_DriveRef;
bExecute : BOOL;
tTimeout : TIME := DEFAULT_ADS_TIMEOUT;

END_VAR

stDriveRef: The reference to the drive can be linked directly to the PLC in the System Manager. To this end an instance of ST_PlcDriveRef must be used and the NetID of the Byte array converted to a string. (Type: ST_DriveRef [9])

bExecute: The function block is enabled via a positive flank at this input.

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

VAR_OUTPUT

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

END_VAR

bBusy: This output is set upon the activation of the function block and remains set until feedback occurs.

bError: This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.

iAdsErrId: In the case of a set bError output returns the ADS error code of the last executed command.

iSercosErrId: In the case of a set bError output returns the Sercos error of the last executed command.

dwAttribute: Returns the attributes of the Sercos parameter.

fDcBusCurrent: Returns the DC-Bus current (e.g. 2.040 corresponds to 2.040 A).

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 fbReadDcBusCurrent 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);
    END_IF
END_IF
4.3 IndraDriveCs FBs

4.3.1 Conversion functions

4.3.1.1 F_ConvWordToSTIndraDriveCsC1D

With this function the Class 1 diagnosis FB_SoEReadClassXDiag_ByDriveRef \([\text{27}]\) (S-0-0011) can be changed to an ST_IndraDriveCs_C1D \([\text{42}]\) structure.

```plaintext
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 \([\text{27}]\)).
```

4.3.1.2 F_ConvWordToSTIndraDriveCsDriveStatus

With this function the Drive status word (S-0-0135) can be changed to a structure ST_IndraDriveCsDriveStatus \([\text{42}]\).

```plaintext
FUNCTION F_ConvWordToSTIndraDriveCsDriveStatus: ST_IndraDriveCsDriveStatus
VAR_INPUT
  wDriveStatus: WORD;
END_VAR

wDriveStatus: Drive status word from S-0-0135 (readable by FB_SoE_Read_ByDriveRef, if necessary mappable).
```

4.4 F_GetVersionTcDrive

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

```plaintext
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;
Data types

5.1 General SoE DT

5.1.1 ST_SoE_String

The ST_SoE_String structure describes a string as it can be used in SoE accesses.

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

- **iActualSize**: contains the current length of the string (without terminating \0)
- **iMaxSize**: contains the maximum length of the string (without terminating \0)
- **strData**: contains the string

5.1.2 ST_SoE_StringEx

The ST_SoE_StringEx structure describes a string, as it can be used in SoE accesses, including prefixed parameter attribute.

```plaintext
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 current length of the string (without terminating \0)
- **iMaxSize**: contains the maximum length of the string (without terminating \0)
- **strData**: contains the string

5.1.3 List types

5.1.3.1 ST_SoE_DiagNumList

The ST_SoE_DiagNumList structure contains the list length (Minimum, Maximum) in bytes as well as the history of the diagnosis numbers.

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

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

## 5.2 AX5000 SoE DT

### 5.2.1 E_FwUpdateState

The E_FwUpdateState describes the state of a firmware update.

```plaintext
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_WaitForSlaveState,
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_ScannedFirmwareError := -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
Data types

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

5.2.2 ST_AX5000_C1D for Class 1 diagnosis

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 *)
5.2.3 ST_AX5000DriveStatus

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

5.2.4 E_AX5000_DriveOpMode

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

5.3 SERCOS

5.3.1 E_SoE_AttribLen

The E_SoE_AttribLen in the attribute of a parameter specifies whether the value of the parameter is a 2-, 4-
or 8-byte data type (individual value) or it is a list consisting of 1-, 2-, 4- or 8-Byte data types. List types (with 
eSoE_LEN_V...) first have the current list length in bytes (in a 16 bit value), then the maximum list length in 
bytes (in a 16 bit value) and the actual list in the specified data type.

Example. see ST_SoE_String [36] of the eSoE_LEN_V1BYTE type

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

eSoE_LEN_2BYTE : 2-Byte data type (e.g. UINT, INT, WORD, IDN)
eSoE_LEN_4BYTE : 4-Byte data type (e.g. UDINT, DINT, DWORD, REAL)
eSoE_LEN_8BYTE : 8-Byte data type (e.g. ULINT, LINT, LREAL)
eSoE_LEN_V1BYTE : List of 1-Byte data types (e.g. String)
5.3.2 **E_SoE_CmdControl**

E_SoE_CmdControl determines whether the command should be canceled, set or started.

```plaintext
type E_SoE_CmdControl := (
    eSoE_CmdControl_Cancel := 0,
    eSoE_CmdControl_Set := 1,
    eSoE_CmdControl_SetAndEnable := 3
); end_type
```

- **eSoE_CmdControl_Cancel**: Cancel command
- **eSoE_CmdControl_Set**: Set command
- **eSoE_CmdControl_SetAndEnable**: Set command and execute

5.3.3 **E_SoE_CmdState**

E_SoE_CmdState describes the state of an SoE command.

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

- **eSoE_CmdState_NotSet**: 0
  - kein Kommando aktiv
- **eSoE_CmdState_Set**: 1
  - Kommando gesetzt (vorbereitet) aber (noch) nicht ausgeführt
- **eSoE_CmdState_Executed**: 2
  - Kommando wurde ausgeführt
- **eSoE_CmdState_SetEnabledExecuted**: 3
  - Kommando gesetzt (vorbereitet) und ausgeführt
- **eSoE_CmdState_SetAndInterrupted**: 5
  - Kommando wurde gesetzt aber unterbrochen
- **eSoE_CmdState_SetEnabledNotExecuted**: 7
  - Kommandoausführung ist noch aktiv
- **eSoE_CmdState_Error**: 15
  - Fehler bei der Kommandoausführung, es wurde in den Fehlerstate gewechselt

5.3.4 **E_SoE_Type**

E_SoE_Type describes the presentation of the parameter value in the attribute of the parameter.

```plaintext
type E_SoE_Type := (
    eSoE_Type_BIN := 0,
    eSoE_Type_UNSIGNED := 1,
    eSoE_Type_SIGNED := 2,
    eSoE_Type_HEX := 3,
    eSoE_Type_TEXT := 4
); end_type
```
E_SoE_Type specifies how the data can be interpreted:

eSoE_Type_BIN : binary

eSoE_Type UNSIGNED : Integer without sign

eSoE_Type_SIGNED : Integer without sign

eSoE_Type_HEX : Hexadecimal number

eSoE_Type_TEXT : Text

eSoE_Type_IDN : Parameter number

eSoE_Type_FLOAT : Floating point number

5.4 IndraDriveCs

5.4.1 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_PosCtrlDrvCtrlFbk1 := 259,
   eIDC_PosCtrlDrvCtrlFbk2 := 260,
   eIDC_PosCtrlDrvCtrlFbk1LagLess := 267,
   eIDC_PosCtrlDrvCtrlFbk2LagLess := 268,
   eIDC_DrvCtrlldPositioningFbk1 := 531,
   eIDC_DrvCtrlldPositioningFbk2 := 532,
   eIDC_DrvCtrlldPositioningFbk1LagLess := 539,
   eIDC_DrvCtrlldPositioningFbk2LagLess := 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,
5.4.2 ST_IndraDriveCs_C1D for Class 1 diagnosis

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 *)
  bOvertravelLimitExceeded : BOOL; (* C1D Bit 13 *)
  bReserved_14 : BOOLEAN; (* C1D Bit 14 *)
  bManufacturerSpecificError : BOOL; (* C1D Bit 15 *)
  END_STRUCT
END_TYPE

5.4.3 ST_IndraDriveCsDriveStatus

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