# Table of contents

1 Foreword .......................................................................................................................... 5
   1.1 Notes on the documentation .................................................................................... 5
   1.2 Safety instructions ................................................................................................. 6

2 Overview ............................................................................................................................ 7

3 Function blocks .................................................................................................................. 9
   3.1 General SoE ............................................................................................................. 9
      3.1.1 FB_SoEReset .................................................................................................. 9
      3.1.2 FB_SoEWritePassword ............................................................................... 10
      3.1.3 FB_SoESetDataAccessMode ...................................................................... 11
      3.1.4 Function blocks for command ...................................................................... 12
      3.1.5 Function blocks for diagnostics .................................................................. 16
      3.1.6 Function blocks for determining current values ......................................... 21
   3.2 General CoE ........................................................................................................... 31
      3.2.1 Function blocks for determining current values ........................................... 31
      3.2.2 Function blocks for command ..................................................................... 35
   3.3 AX5000 SoE .......................................................................................................... 36
      3.3.1 FB_SoEAX5000ReadActMainVoltage ......................................................... 36
      3.3.2 FB_SoEAX5000SetMotorCtrlWord ............................................................. 37
      3.3.3 FB_SoEAX5000FirmwareUpdate ................................................................ 38
      3.3.4 FB_SoEAX5000SetPositionOffset .............................................................. 41
      3.3.5 FB_SoEAX5000DeletePositionOffset ......................................................... 42
   3.4 F_GetVersionTcMc2Drive ....................................................................................... 43
   3.5 AX8000 CoE ........................................................................................................... 44
      3.5.1 FB_CoEAX8000SetPositionOffset ............................................................. 44
      3.5.2 FB_CoEAX8000DeletePositionOffset ......................................................... 45
   3.6 EL72xx CoE ........................................................................................................... 46
      3.6.1 FB_CoEELEL72xxSetPositionOffset ......................................................... 46
      3.6.2 FB_CoEELEL72xxDeletePositionOffset .................................................... 47

4 Data types .......................................................................................................................... 49
   4.1 E_FwUpdateState ................................................................................................. 49
   4.2 E_PositionOffsetMemory .................................................................................... 50
   4.3 E_PositionOffsetFeedback .................................................................................. 50
   4.4 E_DriveErrorCodes .............................................................................................. 51

5 Global constants ................................................................................................................. 52
   5.1 SoE Parameter Access .......................................................................................... 52

6 Samples .............................................................................................................................. 53
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

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
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®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® 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:
with corresponding applications or registrations in various other countries.

EtherCAT® is a 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>
<th>Safety instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="DANGER" /></td>
<td>Serious risk of injury!</td>
<td>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!</td>
<td>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!</td>
<td>Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><img src="image" alt="NOTE" /></td>
<td>Damage to the environment or devices</td>
<td>Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer
This symbol indicates information that contributes to better understanding.
2 Overview

The Tc2_MC2_Drive library includes functions and function blocks for SoE drives that access the drive by MC2 axis structure (AXIS_REF).

Drive libraries

The three drive libraries Tc2_Drive, Tc2_NcDrive and Tc2_MC2_Drive were developed for different functional purposes, but are almost identical in their functionality. The function blocks of the libraries Tc2_NcDrive and Tc_MC2_Drive form wrapper function blocks around the function blocks of the Tc2_Drive library.

<table>
<thead>
<tr>
<th>Drive library</th>
<th>Use</th>
<th>Access to the drive</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc2_Drive</td>
<td>Use the Tc2_Drive library if you use the drive entirely from the PLC (i.e. without NC).</td>
<td>The drive is accessed via a drive reference. Within the library, the ST_DriveRef structure is used for this with the NetID as a string. For linking purposes, a structure called ST_PlcDriveRef is also provided with the NetID as a byte array. (See Drive reference ST_DriveRef)</td>
<td>If you want to access parameters in the drive for which no special function block has been implemented, use the function blocks FB_SoERead_ByDriveRef and FB_SoEWrite_ByDriveRef. These function blocks are implemented in the PLC Lib Tc2_EtherCAT in the SoE Interface folder.</td>
</tr>
<tr>
<td>Tc2_NcDrive</td>
<td>Use the Tc2_NcDrive library if you are using the drive via the NC with the Tc2_Nc libraries.</td>
<td>The drive is accessed via the NC axis structure (NC_TO_PLC). The function blocks of the Tc2_NcDrive library independently determine the access data to the drive (NetID, address and channel number) via the NC axis ID from the NC axis structure.</td>
<td>If you want to access parameters in the drive for which no special function block has been implemented, use the function blocks FB_SoERead and FB_SoEWrite.</td>
</tr>
<tr>
<td>Tc2_MC2_Drive</td>
<td>Use the Tc2_MC2_Drive library if you are using the drive via the NC with the Tc2_MC2 library.</td>
<td>The drive is accessed via the MC2 axis reference (AXIS_REF). The function blocks of the Tc2_MC2_Drive library independently determine the access data to the drive (NetID, address and channel number) via the NC axis ID from the MC2 axis reference.</td>
<td>If you want to access parameters in the drive for which no special function block has been implemented, use the function blocks FB_SoERead and FB_SoEWrite.</td>
</tr>
</tbody>
</table>

Note the differences when using the drive libraries with AX5000 and Bosch Rexroth IndraDrive CS (see Samples)

Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_GetVersionTcMc2Drive</td>
<td>Reads version information from the PLC library. The function has been replaced by the global structure stLibVersion_Tc2_MC2_Drive.</td>
</tr>
<tr>
<td>F_ConvWordToSTAX5000C1D</td>
<td>Converts the C1D word (S-0-0011) of the AX5000 to an ST_AX5000_C1D structure. See: TwinCAT3 PLC Lib:Tc2_Drive documentation.</td>
</tr>
</tbody>
</table>
### Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_SoEReset [9]</td>
<td>Resets the drive (S-0-0099).</td>
</tr>
<tr>
<td>FB_SoEWritePassword [10]</td>
<td>Sets the drive password (S-0-0267).</td>
</tr>
<tr>
<td>FB_SoEReadDiagMessage [16]</td>
<td>Reads the diagnostic message (S-0-0095).</td>
</tr>
<tr>
<td>FB_SoEReadDiagNumber [17]</td>
<td>Reads the diagnostic number (S-0-0390).</td>
</tr>
<tr>
<td>FB_SoEReadDiagNumberList [18]</td>
<td>Reads the diagnostic number list (up to 30 entries) (S-0-0375).</td>
</tr>
<tr>
<td>FB_SoEReadClassXDiag [19]</td>
<td>Reads the Class 1 diagnosis (S-0-0011) and the Class 3 diagnosis (S-0-0013).</td>
</tr>
<tr>
<td>FB_SoEReadCommandState [14]</td>
<td>Checks the command status.</td>
</tr>
<tr>
<td>FB_SoERead [21]</td>
<td>Reads a parameter.</td>
</tr>
<tr>
<td>FB_SoEReadAmplifierTemperature [27]</td>
<td>Reads the drive temperature (S-0-0384).</td>
</tr>
<tr>
<td>FB_SoEReadMotorTemperature [28]</td>
<td>Reads the motor temperature (S-0-0383).</td>
</tr>
<tr>
<td>FB_SoEReadDcBusCurrent [29]</td>
<td>Reads the DC bus current (S-0-0381).</td>
</tr>
<tr>
<td>FB_SoEReadDcBusVoltage [30]</td>
<td>Reads the DC bus voltage (S-0-0380).</td>
</tr>
<tr>
<td>FB_SoEAX5000ReadActMainVoltage [36]</td>
<td>Reads the mains voltage (P-0-0200).</td>
</tr>
<tr>
<td>FB_SoEAX5000SetMotorCtrlWord [37]</td>
<td>Sets the Motor Control Word (P-0-0096).</td>
</tr>
<tr>
<td>FB_SoEAX5000FirmwareUpdate [38]</td>
<td>Executes an automatic firmware update for the AX5000.</td>
</tr>
<tr>
<td>FB_SoEAX5000SetPositionOffset [41]</td>
<td>Saves a position offset.</td>
</tr>
<tr>
<td>FB_CoERead [31]</td>
<td>Reads a parameter.</td>
</tr>
<tr>
<td>FB_CoEWrite [33]</td>
<td>Writes a parameter.</td>
</tr>
</tbody>
</table>

### 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 Function blocks

3.1 General SoE

3.1.1 FB_SoEReset

The drive (S-0-0099) can be reset with the function block FB_SoEReset. In the case of multiple-channel devices if necessary, both channels will have to perform a reset. The timeout time must be 10 s, as the reset can take up to 10 s depending on the error. The flag "Wait For WcState is OK" must be enabled in the advanced EtherCAT settings for the AX5000.

An NC reset will not be performed. If an NC reset is necessary, it can be executed via the function block MC_Reset from the Tc2_MC2 PLC library.

**VAR_INPUT**

```plaintext
VAR_INPUT
  NetId : T_AmsNetId := '';
  Execute : BOOL;
  Timeout : TIME := T#10s;
END_VAR
```

**NetId:** String containing the AMS network ID of the PC. (type: T_AmsNetId)

**Execute:** The function block is enabled via a positive edge at this input.

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

**VAR_IN_OUT**

```plaintext
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```
**Function blocks**

**Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR OUTPUT**

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy : BOOL;</td>
</tr>
<tr>
<td>Error : BOOL;</td>
</tr>
<tr>
<td>AdsErrId : UINT;</td>
</tr>
<tr>
<td>SercosErrId : UINT;</td>
</tr>
</tbody>
</table>

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

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

**Sample**

```plaintext
fbSoEReset : FB_SoEReset;
SoEReset   : BOOL;
(* NcAxis *)
Axis       : AXIS_REF;

IF SoEReset THEN
  fbSoEReset(
    Axis := Axis,
    Execute := TRUE,
  );
IF NOT fbSoEReset.Busy THEN
  fbSoEReset(Axis := Axis, Execute := FALSE);
  SoEReset := FALSE;
END_IF
END_IF
```

### 3.1.2 **FB_SoEWritePassword**

With the FB_SoEWritePassword function block (S-0-0267) the drive password can be set.

**VAR INPUT**

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetId : T_AmsNetId := '';</td>
</tr>
<tr>
<td>Execute : BOOL;</td>
</tr>
<tr>
<td>Timeout : TIME := DEFAULT_ADS_TIMEOUT;</td>
</tr>
<tr>
<td>Password : ST_SoE_String;</td>
</tr>
</tbody>
</table>

**NetId**: String containing the AMS network ID of the PC. (type: T_AmsNetId)

**Execute**: The function block is enabled via a positive edge at this input.

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

**Password**: Password as a Sercos string.
VAR_IN_OUT

VAR_IN_OUT
Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

VAR_OUTPUT
Busy : BOOL;
Error : BOOL;
AdsErrId : UINT;
SercosErrId : UINT;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

Sample

fbWritePassword : FB_SoEWritePassword;
WritePassword : BOOL;
Password : ST_SoE_String;
(* NcAxis *)
Axis : AXIS_REF;

IF WritePassword THEN
  fbWritePassword(Axis := Axis,
                 Execute := TRUE,
                 Timeout := DEFAULT_ADS_TIMEOUT,
                 Password := Password);
ENDIF

IF NOT fbWritePassword.Busy THEN
  fbWritePassword(Axis := Axis, Execute := FALSE);
  WritePassword := FALSE;
ENDIF

3.1.3 FB_SoESetDataAccessMode

The SoE parameter access mode is set with the function block FB_SoESetDataAccessMode. Attributes & Value are accessed in parallel by default. If this access mode is not supported by the device manufacturer, sequential access can be enforced. Generally, however, parallel access is faster.

VAR_INPUT

VAR_INPUT
Execute : BOOL;
Mode : E_SoEDataAccessMode := E_SoEDataAccessMode.eSoEDataAccessMode_Parallel;
END_VAR

Execute: The function block is enabled via a positive edge at this input.

Mode: Defines the access mode.
VAR_OUTPUT

VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  ErrId : UINT;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

ErrId: In the event of a set error output returns the ADS error code.

3.1.4 Function blocks for command

3.1.4.1 FB_SoEExecuteCommand

With the FB_SoEExecuteCommand function block a command can be executed.

VAR_INPUT

VAR_INPUT
  NetId : T_AmsNetId := '';
  Idn : WORD;
  Execute : BOOL := TRUE;
  Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

NetId: String, which contains the AMS-Network ID of the PC (Type: T_AmsNetId).

Idn: Parameter number to which FB_SoEExecuteCommand refers, e.g. "P_0_IDNs + 160" for P-0-0160.

Execute: The function block is enabled via a positive edge at this input.

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

VAR_IN_OUT

VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  AdsErrId : UINT;
  SercosErrId : UINT;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.
Function blocks

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

Sample

```
fbExecuteCommand : FB_SoEExecuteCommand;
ExecuteCommand   : BOOL;
Idn              : WORD;
(* NcAxis *)     
Axis             : AXIS_REF;

IF ExecuteCommand THEN
  Idn := P_0_IDNs + 160;
  fbExecuteCommand(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    Idn := Idn,
  );
  IF NOT fbExecuteCommand.Busy THEN
    fbExecuteCommand(Axis := Axis, Execute := FALSE);
    ExecuteCommand := FALSE;
  END_IF
END_IF
```

3.1.4.2 FB_SoEWriteCommandControl

With the FB_SoEWriteCommandControl function block a command can be prepared, started or canceled.

```
VAR_INPUT
VAR_INPUT
NetId     : T_AmsNetId := '';
Idn       : WORD;
CmdControl: E_SoE_CmdControl;
Execute    : BOOL;
Timeout    : TIME := DEFAULT_ADS_TIMEOUT;
Axis       : AXIS_REF;
END_VAR

NetId: String, which contains the AMS-Network ID of the PC (Type: T_AmsNetId).
Idn: Parameter number to which FB_SoEWriteCommandControl refers, e.g. "P_0_IDNs + 160" for P-0-0160.
CmdControl: Indicates, if a command should be prepared (eSoE_CmdControl_Set := 1), executed (eSoE_CmdControl_SetAndEnable := 3) or aborted (eSoE_CmdControl_Cancel := 0).
Execute: The function block is enabled via a positive edge at this input.
Timeout: Maximum time allowed for the execution of the function block.
```

VAR_IN_OUT
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
**Axis:** The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Busy</strong> : BOOL;</td>
</tr>
<tr>
<td><strong>Error</strong> : BOOL;</td>
</tr>
<tr>
<td><strong>AdsErrId</strong> : UINT;</td>
</tr>
<tr>
<td><strong>SercosErrId</strong> : UINT;</td>
</tr>
<tr>
<td>END_VAR</td>
</tr>
</tbody>
</table>

**Busy:** This output is set when the function block is activated, and remains set until a feedback is received.

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

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

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

### Sample

```plaintext
fbWriteCommandControl : FB_SoEWriteCommandControl;
WriteCommandControl : BOOL;
Idn : WORD;
CmdControl : E_SoE_CmdControl;
(* NcAxis *)
Axis : AXIS_REF;

IF WriteCommandControl THEN
  Idn := P_0_IDNs + 160;
  fbWriteCommandControl(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    Idn := Idn,
    CmdControl := CmdControl
  );
  IF NOT fbWriteCommandControl.Busy THEN
    fbWriteCommandControl(Axis := Axis, Execute := FALSE);
    WriteCommandControl := FALSE;
  END_IF
END_IF
```

### 3.1.4.3 FB_SoEReadCommandState

With the FB_SoEReadCommandState function block the command execution can be checked.

### VAR_INPUT

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NetId</strong> : T_AmsNetId := '';</td>
</tr>
<tr>
<td><strong>Idn</strong> : WORD;</td>
</tr>
<tr>
<td><strong>Execute</strong> : BOOL;</td>
</tr>
<tr>
<td><strong>Timeout</strong> : TIME := DEFAULT_ADS_TIMEOUT;</td>
</tr>
<tr>
<td><strong>Axis</strong> : AXIS_REF;</td>
</tr>
</tbody>
</table>

**NetId:** String containing the AMS network ID of the PC. (type: T_AmsNetId)

**Idn:** Parameter number to which FB_SoEReadCommandState_ByDriveRef refers, e.g. "P_0_IDNs + 160" for P-0-0160
Execute: The function block is enabled via a positive edge at this input.

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

VAR_IN_OUT

VAR_IN_OUT
Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

VAR_OUTPUT
Busy : BOOL;
Error : BOOL;
AdsErrId : UINT;
SercosErrId : UINT;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

CmdState: Returns the command status:

- 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

Sample

fbReadCommandState : FB_SoEReadCommandState;
ReadCommandState : BOOL;
Idn : WORD;
CmdState : E_SoE_CmdState;
(* NcAxis *)
Axis : AXIS_REF;

IF ReadCommandState THEN
Idn := P_0_IDNs + 160;
fbReadCommandState(
Axis := Axis,
Execute := TRUE,
Timeout := DEFAULT_ADS_TIMEOUT,
Idn := Idn,
CmdState => CmdState
);
IF NOT fbReadCommandState.Busy THEN
 fbReadCommandState(Axis := Axis, Execute := FALSE);
3.1.5 Function blocks for diagnostics

3.1.5.1 FB_SoEReadDiagMessage

With the FB_SoEReadDiagMessage function block the diagnosis message can be read out as a Sercos string (S-0-0095).

**VAR_INPUT**

```plaintext
VAR_INPUT
  NetId   : T_AmsNetId := '';
  Execute : BOOL;
  Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

- **NetId**: String containing the AMS network ID of the PC. (Type: T_AmsNetId)
- **Execute**: The function block is enabled via a positive flank at this input.
- **Timeout**: Maximum time allowed for the execution of the function block.

**VAR_IN_OUT**

```plaintext
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

- **Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
  Busy        : BOOL;
  Error       : BOOL;
  AdsErrId    : UINT;
  SercosErrId : UINT;
  DiagMessage : ST_SoE_String;
  Attribute   : DWORD;
END_VAR
```

- **Busy**: This output is set when the function block is activated, and remains set until a feedback is received.
- **Error**: This output is set after the Busy output has been reset when an error occurs in the transmission of the command.
- **AdsErrId**: In the case of a set Error output returns the ADS error code of the last executed command.
- **SercosErrId**: In the case of a set Error output returns the Sercos error of the last executed command.
- **Attributes**: Returns the attributes of the Sercos parameter.
- **DiagMessage**: Returns the diagnosis message.
3.1.5.2 FB_SoEReadDiagNumber

With the FB_SoEReadDiagNumber function block the current diagnostic number can be read out as UDINT (S-0-0390).

VAR_INPUT

VAR_INPUT
NetId : T_AmsNetId := '';
Execute : BOOL;
Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

NetId: String containing the AMS network ID of the PC. (Type: T_AmsNetId)

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

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

VAR_IN_OUT

VAR_IN_OUT
Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

VAR_OUTPUT
Busy : BOOL;
Error : BOOL;
AdsErrId : UINT;
SercosErrId : UINT;
DiagNumber : UDINT;
Attribute : DWORD;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.
**Error**: This output is set after the Busy output has been reset when an error occurs in the transmission of the command.

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

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

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

**DiagNumber**: Returns the current diagnostic number.

### Sample

```plaintext
fbDiagNumber : FB_SoEReadDiagNumber;
bDiagNumber : BOOL;
DiagNumber : UDINT;
(* NcAxis *)
Axis : AXIS_REF;

IF bDiagNumber THEN
    fbDiagNumber(
        Axis := Axis,
        Execute := TRUE,
        Timeout := DEFAULT_ADS_TIMEOUT,
        DiagNumber => DiagNumber
    );
    IF NOT fbDiagNumber.Busy THEN
        fbDiagNumber(Axis := Axis, Execute := FALSE);
        bDiagNumber := FALSE;
    END_IF
END_IF
```

### 3.1.5.3 FB_SoEReadDiagNumberList

With the **FB_SoEReadDiagNumberList** function block a history of the diagnosis numbers can be read out as a list (S-0-0375).

**VAR_INPUT**

```plaintext
VAR_INPUT
    NetId : T_AmsNetId := '';
    Execute : BOOL;
    Timeout : TIME := DEFAULT_ADS_TIMEOUT;
    pDiagNumber : POINTER TO ST_SoE_DiagNumList;
    Size : UDINT;
END_VAR
```

**NetId**: String containing the AMS network ID of the PC. (type: T_AmsNetId)

**Execute**: The function block is enabled via a positive edge at this input.

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

**pDiagNumber**: Pointer to the 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.

**Size**: Size of the list in bytes (as Sizeof())
VAR_IN_OUT

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>AXIS_REF</td>
<td>END_VAR</td>
<td></td>
</tr>
</tbody>
</table>

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>BOOL</td>
<td>END_VAR</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
<td>END_VAR</td>
<td></td>
</tr>
<tr>
<td>AdsErrId</td>
<td>UINT</td>
<td>END_VAR</td>
<td></td>
</tr>
<tr>
<td>SercosErrId</td>
<td>UINT</td>
<td>END_VAR</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>DWORD</td>
<td>END_VAR</td>
<td></td>
</tr>
</tbody>
</table>

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

Attributes: Returns the attributes of the Sercos parameter.

Sample

```
fbDiagNumberList : FB_SoEReadDiagNumberList;
DiagNumberList : BOOL;
stDiagNumberList : ST_SoE_DiagNumList;
{ NcAxis *)
Axis | AXIS_REF |

IF DiagNumberList THEN
  fbDiagNumberList(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    pDiagNumber := ADR(stDiagNumberList),
    Size := SIZEOF(stDiagNumberList),
  );
  IF NOT fbDiagNumberList.Busy THEN
    fbDiagNumberList(Axis := Axis, Execute := FALSE);
    DiagNumberList := FALSE;
  END_IF
END_IF
```

3.1.5.4 FB_SoEReadClassXDiag

With the function block FB_SoEReadClassXDiag, the current Class 1 diagnosis (S-0-0011) ... Class 3 diagnosis (S-0-0013) can be read out as WORD. There is the conversion function F_ConvWordToSTAX5000C1D for the evaluation of the Class 1 diagnosis as a structure ST_AX5000_C1D, (see TwinCAT 3 PLC Lib Tc2_Drive documentation).
Function blocks

VAR_INPUT

VAR_INPUT
  NetId : T_AmsNetId := '';
  Execute : BOOL;
  DiagClass : USINT:= 1; (* 1: C1D (S-0-0011) is default, 2: C2D (S-0-0012), 3: C3D (S-0-0013) *)
  Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

NetId: String containing the AMS network ID of the PC. (Type T_AmsNetId)

Execute: The function block is enabled via a positive edge at this input.

DiagClass: Specifies which diagnosis should be read. The diagnostics parameters may vary from vendor to vendor. All diagnostics parameters (C1D ... C3D) or all bits are not always implemented in them.
1: Error: Class 1 Diag (S-0-0011)
2: Warnings: Class 2 Diag (S-0-0012)
3: Information messages: Class 3 Diag (S-0-0013)

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

VAR_IN_OUT

VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  AdsErrId : UINT;
  SercosErrId : UINT;
  ClassXDiag : WORD;
  Attribute : DWORD;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

ClassXDiag: Returns the current Class X diagnosis.

Attributes: Returns the attributes of the Sercos parameter.

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample

Sample
3.1.6 Function blocks for determining current values

3.1.6.1 FB_SoERead

```plaintext
With the FB_SoERead function block a parameter can be read.

By default, attributes and values are read in parallel.

Should the call generate an ADS error on a third-party device, the lack of support for this faster parameter
access method may be the cause.

In this case, the function block FB_SoESetDataAccessMode [11] can be used to switch to the slower
sequential access method, if required.

VAR_INPUT

VAR_INPUT
NetId : T_AmsNetId := '';
Idn : WORD;
Element : BYTE;
pDstBuf : PVOID;
BufLen : UDINT;
Execute : BOOL;
Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

NetId: String containing the AMS network ID of the PC. (type: T_AmsNetId)

Idn: Parameter number to which FB_SoERead refers, e.g. "S_0_IDNs + 33" for S-0-0033

Element: Specifies which part of the parameter should be accessed, e.g. 16#40 is the value (Value) of the
parameter

EC_SOE_ELEMENT_DATASTATE : BYTE := 16#01;
EC_SOE_ELEMENT_NAME : BYTE := 16#02;
EC_SOE_ELEMENT_ATTRIBUTE : BYTE := 16#04;
EC_SOE_ELEMENT_UNIT : BYTE := 16#08;
EC_SOE_ELEMENT_MIN : BYTE := 16#10;
```
Function blocks

EC_SOE_ELEMENT_MAX : BYTE := 16#20;
EC_SOE_ELEMENT_VALUE : BYTE := 16#40;
EC_SOE_ELEMENT_DEFAULT : BYTE := 16#80;

pDstBuf: ADR() of the variables to which the value should be read.
Buflen: SIZEOF() of the variables to which the value should be read.
Execute: The function block is enabled via a positive edge at this input.
Timeout: Maximum time allowed for the execution of the function block.

VAR_IN_OUT
VAR_IN_OUT
Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT
VAR_OUTPUT
Busy : BOOL;
Error : BOOL;
AdsErrId : UINT;
SercosErrId : UINT;
Attribute : DWORD;
END_VAR

Busy: This output is set when the function block is activated, and remains set until an acknowledgement is received.
Error: This output is set after the Busy output has been reset when an error occurs in the transmission of the command.
AdsErrId: In the case of a set Error output returns the ADS error code of the last executed command.
SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.
Attributes: Returns the attributes of the Sercos parameter.

Sample

fbRead : FB_SoERead;
Read : BOOL;
Idn : WORD;
ReadValue : UINT;
(* NcAxis *)
Axis : AXIS_REF;

IF Read THEN
Idn := S_0_IDNs + 33;
fbRead(
  Axis := Axis,
  Idn := Idn,
  Element := 16#40,
  pDstBuf := ADR(ReadValue),
  BufLen := SIZEOF(ReadValue),
  Execute := TRUE,
  Timeout := DEFAULT_ADS_TIMEOUT,
);
IF NOT fbRead.Busy THEN
  fbRead(Axis := Axis, Execute := FALSE);
  Read := FALSE;
END_IF
END_IF
Parameter access

Note that for some parameters, the additional items of information "ActualLength" & "MaxLength" are necessary for the parameter access.

In order to determine the parameters for which this is necessary, the attributes can be displayed in the parameter list of the Drive Manager via the context menu of the header.

If the fifth place from the right contains the value 4, 5, 6 or 7, then the additional items of information are also transmitted.

The structure to read out the serial number (S-0-0432) then looks like this:

```
TYPE ST_SerialNumber:
  STRUCT
    ActualLength : UINT;
    MaxLength    : UINT;
    SerialNumber : T_MaxString;
```
3.1.6.2 FB_SoEWrite

With the FB_SoEWrite function block a parameter can be written.

VAR_INPUT

VAR_INPUT
NetId : T_AmsNetId := '';
Idn : WORD;
Element : BYTE;
SrcBuf : PVoid;
BufLen : UDINT;
Execute : BOOL;
Timeout : TIME := DEFAULT_ADS_TIMEOUT;
Password : ST_SoE_String;
END_VAR

NetId: String containing the AMS network ID of the PC. (type: T_AmsNetId)
Idn: Parameter number to which FB_SoERead refers, e.g. "S_0_IDNs + 47" for S-0-0047
Element: Specifies which part of the parameter should be accessed, e.g. 16#40 is the value (Value) of the parameter. Usually there is only write access to the value, other components of the parameter are read-only.

EC_SOE_ELEMENT_DATASTATE : BYTE := 16#01;
EC_SOE_ELEMENT_NAME : BYTE := 16#02;
EC_SOE_ELEMENT_ATTRIBUTE : BYTE := 16#04;
EC_SOE_ELEMENT_UNIT : BYTE := 16#08;
EC_SOE_ELEMENT_MIN : BYTE := 16#10;
EC_SOE_ELEMENT_MAX : BYTE := 16#20;
EC_SOE_ELEMENT_VALUE : BYTE := 16#40;
EC_SOE_ELEMENT_DEFAULT : BYTE := 16#80;

SrcBuf: ADR() of the variable containing the value to be written.
BufLen: SIZEOF() of the variable containing the value to be written
Execute: The function block is enabled via a positive edge at this input.
Timeout: Maximum time allowed for the execution of the function block.
Password: Password as a Sercos string. Not currently used. The password must be written with FB_SoEWritePassword.
VAR_IN_OUT
VAR_IN_OUT
Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system.
Among other things it contains the current status of the axis such as the position, the velocity and the error
state.

VAR_OUTPUT
VAR_OUTPUT
Busy : BOOL;
Error : BOOL;
AdsErrId : UINT;
SercosErrId : UINT;
END_VAR

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

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

AdsErrId: In the case of a set Error output returns the ADS error code of the last executed command
SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

Sample

fbWrite : FB_SoEWrite;
Idn : WORD;
Write : BOOL;
WriteValue : UINT;
Password : ST_SoE_String;
(* NcAxis *)
Axis : AXIS_REF;

IF Write THEN
  Idn := S_0_IDNs + 33;
  fbWrite(
    Axis := Axis,
    Idn := Idn,
    Element := 16#40,
    pSrcBuf := ADR(WriteValue),
    BufLen := SIZEOF(WriteValue),
    Password := Password,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
  );
  IF NOT fbWrite.Busy THEN
    fbWrite(Axis := Axis, Execute := FALSE);
    Write := FALSE;
  END_IF
END_IF

Parameter access

Note that for some parameters, the additional items of information "ActualLength" & "MaxLength" are
necessary for the parameter access.

In order to determine the parameters for which this is necessary, the attributes can be displayed in the
parameter list of the Drive Manager via the context menu of the header.
If the fifth place from the right contains the value 4, 5, 6 or 7, then the additional items of information are also transmitted.

The structure to read out the serial number (S-0-0432) then looks like this:

```plaintext
TYPE ST_SerialNumber:
  STRUCT
    ActualLength : UINT;
    MaxLength    : UINT;
    SerialNumber : T_MaxString;
  END_STRUCT
END_TYPE
```
3.1.6.3  FB_SoEReadAmplifierTemperature

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

**VAR_INPUT**

```plaintext
VAR_INPUT
    NetId   : T_AmsNetId := '';
    Execute : BOOL;
    Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

- **NetId**: String containing the AMS network ID of the PC. (Type: T_AmsNetId)
- **Execute**: The function block is enabled via a positive flank at this input.
- **Timeout**: Maximum time allowed for the execution of the function block.

**VAR_IN_OUT**

```plaintext
VAR_IN_OUT
    Axis   : AXIS_REF;
END_VAR
```

- **Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
    Busy     : BOOL;
    Error    : BOOL;
    AdsErrId : UINT;
    SercosErrId : UINT;
    AmplifierTemperature : REAL;
    Attribute : DWORD;
END_VAR
```

- **Busy**: This output is set when the function block is activated, and remains set until a feedback is received.
- **Error**: This output is set after the Busy output has been reset when an error occurs in the transmission of the command.
- **AdsErrId**: In the case of a set Error output returns the ADS error code of the last executed command.
- **SercosErrId**: In the case of a set Error output returns the Sercos error of the last executed command.
- **Attributes**: Returns the attributes of the Sercos parameter.
- **AmplifierTemperature**: Returns the drive temperature (e.g. 26.2 corresponds to 26.2 °C).

**Sample**

```plaintext
fbReadAmplifierTemp    : FB_SoEReadAmplifierTemperature;
ReadAmplifierTemp      : BOOL;
AmplifierTemperature   : REAL;
(* NcAxis *)
Axis                    : AXIS_REF;
```
IF ReadAmplifierTemp THEN
  fbReadAmplifierTemp(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    AmplifierTemperature := AmplifierTemperature
  );
END_IF

IF NOT fbReadAmplifierTemp.Busy THEN
  fbReadAmplifierTemp(Axis := Axis, Execute := FALSE);
  ReadAmplifierTemp := FALSE;
END_IF

3.1.6.4  **FB_SoEReadMotorTemperature**

With the function block **FB_SoEReadMotorTemperature** the temperature of the motor (S-0-0383) can be read. If the motor does not contain a temperature sensor, this is 0.0, i.e. 0.0 °C.

**VAR_INPUT**

VAR_INPUT
  NetId : T_AmsNetId := '';
  Execute : BOOL;
  Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

NetId: String containing the AMS network ID of the PC. (Type: T_AmsNetId)

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

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

**VAR_IN_OUT**

VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR_OUTPUT**

VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  AdsErrId : UINT;
  SercosErrId : UINT;
  MotorTemperature : REAL;
  Attribute : DWORD;
END_VAR

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

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.
Attributes: Returns the attributes of the Sercos parameter.

MotorTemperature: Returns the motor temperature (e.g. 30.5 corresponds to 30.5 °C). If the motor does not contain a temperature sensor, this is 0.0, i.e. 0.0 °C.

Sample

```plaintext
fbReadMotorTemp : FB_SoEReadMotorTemperature;
ReadMotorTemp   : BOOL;
MotorTemperature : REAL;
{" NcAxis "}
Axis            : AXIS_REF;

IF ReadMotorTemp THEN
  fbReadMotorTemp(
    Axis         := Axis,
    Execute      := TRUE,
    Timeout      := DEFAULT_ADS_TIMEOUT,
    MotorTemperature => MotorTemperature
  );
  IF NOT fbReadMotorTemp.Busy THEN
    fbReadMotorTemp(Axis := Axis, Execute := FALSE);
    ReadMotorTemp := FALSE;
  END_IF
END_IF
```

3.1.6.5 FB_SoEReadDcBusCurrent

With the FB_SoEAX5000ReadDcBusCurrent function block the DC-Bus current (S-0-0381) can be read.

**VAR_INPUT**

```plaintext
VAR_INPUT
  NetId       : T_AmsNetId := '';
  Execute     : BOOL;
  Timeout     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NetId: String containing the AMS network ID of the PC. (Type: T_AmsNetId)

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

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

**VAR_IN_OUT**

```plaintext
VAR_IN_OUT
  Axis         : AXIS_REF;
END_VAR
```

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
  Busy         : BOOL;
  Error        : BOOL;
  AdsErrId     : UINT;
  SercosErrId  : UINT;
```
Function blocks

**DcBusCurrent**: Returns the DC bus current (e.g. 2.040 equals 2.040 A).

**Sample**

```plaintext
fbReadDcBusCurrent : FB_SoEReadDcBusCurrent;
ReadDcBusCurrent : BOOL;
DcBusCurrent : REAL;
(* NcAxis *)
Axis : AXIS_REF;

IF ReadDcBusCurrent THEN
  fbReadDcBusCurrent(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    DcBusCurrent => DcBusCurrent
  );
  IF NOT fbReadDcBusCurrent.Busy THEN
    fbReadDcBusCurrent(Axis := Axis, Execute := FALSE);
    ReadDcBusCurrent := FALSE;
  END_IF
END_IF
```

**3.1.6.6 FB_SoEReadDcBusVoltage**

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

**VAR_INPUT**

- **NetId**: T_AmsNetId := '';
- **Execute** : BOOL;
- **Timeout** : TIME := DEFAULT_ADS_TIMEOUT;

- **Axis** : AXIS_REF

**NetId**: String containing the AMS network ID of the PC. (Type: T_AmsNetId)

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

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

**VAR_IN_OUT**

- **Axis** : AXIS_REF;
**Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  AdsErrId : UINT;
  SercosErrId : UINT;
  DcBusVoltage : REAL;
  Attribute : DWORD;
END_VAR
```

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

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

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

**DcBusVoltage**: Returns the DC-Bus voltage (e.g. 294.0 corresponds to 294.0 V).

**Sample**

```plaintext
fbReadDcBusVoltage : FB_SoEReadDcBusVoltage;
ReadDcBusVoltage : BOOL;
DcBusVoltage : REAL;
(* NcAxis *)
Axis : AXIS_REF;

IF ReadDcBusVoltage THEN
  fbReadDcBusVoltage(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    DcBusVoltage => DcBusVoltage
  );
ELSE fbReadDcBusVoltage(Axis := Axis, Execute := FALSE);
  ReadDcBusVoltage := FALSE;
END_IF
END_IF
```

### 3.2 General CoE

#### 3.2.1 Function blocks for determining current values

#### 3.2.1.1 **FB_CoERead**
The function block FB_CoERead allows data to be read from an object dictionary of an EtherCAT slave through an SDO (Service Data Object) access. This requires the slave to have a mailbox and to support the CoE (CANopen over EtherCAT) protocol. With the help of the SubIndex and Index parameters a selection is made as to which object should be read out. Via CompleteAccess := TRUE the parameter can be read with sub-elements.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetId</td>
<td>T_AmsNetId;(*)</td>
<td>netID of PC with NC</td>
</tr>
<tr>
<td>Index</td>
<td>WORD;(<em>CoE object index</em>)</td>
<td></td>
</tr>
<tr>
<td>SubIndex</td>
<td>BYTE;(<em>CoE sub index</em>)</td>
<td></td>
</tr>
<tr>
<td>pDstBuf</td>
<td>PVOID;(<em>Contains the address of the buffer for the received data</em>)</td>
<td></td>
</tr>
<tr>
<td>BufLen</td>
<td>UDINT;(<em>Contains the max. number of bytes to be received</em>)</td>
<td></td>
</tr>
<tr>
<td>Execute</td>
<td>BOOL;(<em>Function block execution is triggered by a rising edge at this input.</em>)</td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>TIME := DEFAULT_ADS_TIMEOUT;(<em>States the time before the function is cancelled.</em>)</td>
<td></td>
</tr>
<tr>
<td>CompleteAccess</td>
<td>BOOL;(<em>Function block reads the complete object with all sub index</em>)</td>
<td></td>
</tr>
</tbody>
</table>

**NetId**: String containing the AMS network ID of the PC. (type: T_AmsNetId)

**Index**: Index of the object that is to be read.

**SubIndex**: Subindex of the object that is to be read.

**pDstBuf**: Address (pointer) to the receive buffer

**BufLen**: Maximum available buffer size (in bytes) for the data to be read

**Execute**: The function block is enabled via a positive edge at this input.

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

**CompleteAccess**: Via Complete Access the complete object can be accessed at once.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>BOOL;</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>BOOL;</td>
<td></td>
</tr>
<tr>
<td>AdsErrId</td>
<td>UINT;</td>
<td></td>
</tr>
<tr>
<td>CANopenErrId</td>
<td>UINT;</td>
<td></td>
</tr>
</tbody>
</table>

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.

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

**AdsErrId**: In the event of a set error output returns the ADS error code.

**CANopenErrId**: In the event of a set error output returns the CANopen error code.

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>AXIS_REF;</td>
<td></td>
</tr>
</tbody>
</table>

**Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**Example of an implementation in ST:**

```plaintext
PROGRAM MAIN
VAR
   fbcCoERead : FB_CoERead;
   NetId      : T_AmsNetId := '';
   Index      : WORD := 16#1018;
   SubIndex   : BYTE := 1;
   Execute    : BOOL := TRUE;
   Timeout    : TIME := T#5S;
   CompleteAccess : BOOL := TRUE;
   Axis       : AXIS_REF;
```
3.2.1.2 **FB_CoEWrite**

With the function block FB_CoEWrite, an object from the object directory of an EtherCAT slave can be written via an SDO (Service Data Object) download. This requires the slave to have a mailbox and to support the CoE (CANopen over EtherCAT) protocol. With the help of the SubIndex and Index parameters a selection is made as to which object should be written. Via CompleteAccess := TRUE the parameter can be written with sub-elements.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetId</td>
<td>T_AmsNetId</td>
<td>String, which contains the AMS-Network ID of the PC (Type: T_AmsNetId).</td>
</tr>
<tr>
<td>Index</td>
<td>WORD</td>
<td>Index of the object that is supposed to be written.</td>
</tr>
<tr>
<td>SubIndex</td>
<td>BYTE</td>
<td>Sub-index of the object that is supposed to be written.</td>
</tr>
<tr>
<td>pSrcBuf</td>
<td>PVOID</td>
<td>Address (pointer) to the send buffer</td>
</tr>
<tr>
<td>BufLen</td>
<td>UDINT</td>
<td>Number of date to be sent in bytes</td>
</tr>
<tr>
<td>Execute</td>
<td>BOOL</td>
<td>The function block is enabled via a positive edge at this input.</td>
</tr>
<tr>
<td>Timeout</td>
<td>TIME</td>
<td>States the time before the function is cancelled.*</td>
</tr>
<tr>
<td>CompleteAccess</td>
<td>BOOL</td>
<td>Function block reads the complete object with all sub index*</td>
</tr>
</tbody>
</table>

**FB_CoEWrite**

```plaintext
 NetId := NetId,
 Index := Index,
 SubIndex := SubIndex,
 pSrcBuf := ADR(vendorId),
 BufLen := SIZEOF(vendorId),
 Execute := Execute,
 Timeout := Timeout,
 CompleteAccess := CompleteAccess,
 Axis := Axis
```

IF NOT fbCoERead.Busy THEN
  Error := fbCoERead.Error;
  AdsErrId := fbCoERead.AdsErrId;
  CANopenErrId := fbCoERead.CANopenErrId;
  Execute := FALSE;
  fbCoERead(Execute := Execute, Axis := Axis);
END_IF
CompleteAccess: Via Complete Access the complete object can be accessed at once.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>BOOL</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>AdsErrId</td>
<td>UINT</td>
</tr>
<tr>
<td>CANopenErrId</td>
<td>UINT</td>
</tr>
</tbody>
</table>

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

AdsErrId: In the event of a set error output returns the ADS error code.

CANopenErrId: In the event of a set error output returns the CANopen error code.

VAR_IN_OUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>AXIS_REF</td>
</tr>
</tbody>
</table>

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

Example of an implementation in ST:

```st
PROGRAM MAIN
VAR
  fbCoEWrite   : FB_CoEWrite;
  NetId        : T_AmsNetId := '';
  Index        : WORD := 16#1018;
  SubIndex     : BYTE := 1;
  Execute      : BOOL := TRUE;
  Timeout      : TIME := T#5S;
  CompleteAccess: BOOL := TRUE;
  Axis         : AXIS_REF;
  vendorId     : UDINT := 2;
  Error        : BOOL;
  AdsErrId     : UDINT;
  CANopenErrId : UDINT;END_VAR

fbCoEWrite(
  NetId        := NetId,
  Index        := Index,
  SubIndex     := SubIndex,
  pSrcBuf      := ADR(vendorId),
  BufLen       := SIZEOF(vendorId),
  Execute      := Execute,
  Timeout      := Timeout,
  CompleteAccess:= CompleteAccess,
  Axis         := Axis
);IF NOT fbCoEWrite.Busy THEN
  Error        := fbCoEWrite.Error;
  AdsErrId     := fbCoEWrite.AdsErrId;
  CANopenErrId := fbCoEWrite.CANopenErrId;
  Execute      := FALSE;
  fbCoEWrite(Execute := Execute, Axis := Axis);
END_IF
```
3.2.2 Function blocks for command

3.2.2.1 FB_CoEExecuteCommand

With the function block FB_CoEExecuteCommand a command can be executed.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_AmsNetId</td>
<td>String, which contains the AMS-Network ID of the PC (Type: T_AmsNetId).</td>
</tr>
<tr>
<td>WORD</td>
<td>CoE object index to which FB_CoEExecuteCommand refers.</td>
</tr>
<tr>
<td>PVOID</td>
<td>Address of the structure of the data to be sent</td>
</tr>
<tr>
<td>UDINT</td>
<td>Size in bytes of the structure of the data to be sent</td>
</tr>
<tr>
<td>PVOID</td>
<td>Address of the structure of the data to be received</td>
</tr>
<tr>
<td>UDINT</td>
<td>Size in bytes of the structure of the data to be received</td>
</tr>
<tr>
<td>BOOL</td>
<td>The function block is enabled via a positive edge at this input.</td>
</tr>
<tr>
<td>TIME</td>
<td>Maximum time allowed for the execution of the function block.</td>
</tr>
</tbody>
</table>

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIS_REF</td>
<td>The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>This output is set when the function block is activated, and remains set until a feedback is received.</td>
</tr>
<tr>
<td>BOOL</td>
<td>This output is set after the Busy output has been reset when an error occurs in the transmission of the command.</td>
</tr>
</tbody>
</table>
Function blocks

ErrorID: In the event of a set error output returns the ADS error code.

Status: Status of the command execution

3.3 AX5000 SoE

3.3.1 FB_SoEAX5000ReadActMainVoltage

With the FB_SoEAX5000ReadActMainVoltage function block the current peak value of the mains voltage of the AX5000 (P-0-0200) can be read.

VAR_INPUT

VAR_INPUT
    NetId : T_AmsNetId := '';
    Execute : BOOL;
    Timeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR

NetId: String containing the AMS network ID of the PC. (Type: T_AmsNetId)

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

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

VAR_IN_OUT

VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

VAR_OUTPUT
    Busy : BOOL;
    Error : BOOL;
    AdsErrId : UINT;
    SercosErrId : UINT;
    Attribute : DWORD;
    ActualMainVoltage : LREAL;
END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

Attributes: Returns the attributes of the Sercos parameter.
**ActualMainVoltage**: Returns the peak value of the current mains voltage of the AX5000 (e.g. 303.0 corresponds to 303.0 V).

**Sample**

```plaintext
fbReadActMainVoltage : FB_SoEAX5000ReadActMainVoltage;
ReadActMainVoltage  : BOOL;
ActualMainVoltage    : REAL;
(* NcAxis *)
Axis                 : AXIS_REF;

IF ReadActMainVoltage THEN
  fbReadActMainVoltage(
    Axis := Axis,
    Execute := TRUE,
    Timeout := DEFAULT_ADS_TIMEOUT,
    ActualMainVoltage => ActualMainVoltage
  );
  IF NOT fbReadActMainVoltage.Busy THEN
    fbReadActMainVoltage(Axis := Axis, Execute := FALSE);
    ReadActMainVoltage := FALSE;
  END_IF
END_IF
```

**3.3.2 **

**FB_SoEAX5000SetMotorCtrlWord**

With the FB_SoEAX5000SetMotorCtrlWord function block the ForceLock bit (Bit 0) or the ForceUnlock bit can be set in the Motor Control Word (P-0-0096) 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
  NetId : T_AmsNetId := '';
  Execute : BOOL;
  Timeout : TIME := DEFAULT_ADS_TIMEOUT;
  ForceLock : BOOL;
  ForceUnlock : BOOL;
END_VAR
```

**NetId**: String containing the AMS network ID of the PC. (type: T_AmsNetId)

**Execute**: The function block is enabled via a positive edge at this input.

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

**ForceLock**: Activates the brake independently of the enable.

**ForceUnlock**: Releases the brake independently of the enable.
Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR OUTPUT

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>BOOL</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>AdsErrId</td>
<td>UINT</td>
</tr>
<tr>
<td>SercosErrId</td>
<td>UINT</td>
</tr>
</tbody>
</table>

END_VAR

Busy: This output is set when the function block is activated, and remains set until a feedback is received.

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

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

SercosErrId: In the case of a set Error output returns the Sercos error of the last executed command.

Sample

```plaintext
fbSetMotorCtrlWord := FB_SoEAX5000SetMotorCtrlWord;
SetMotorCtrlWord := BOOL;
ForceLock := BOOL;
ForceUnlock := BOOL;
("NcAxis ")
Axis := AXIS_REF;
IF SetMotorCtrlWord THEN
  fbSetMotorCtrlWord(Axis := Axis,
                       Execute := TRUE,
                       Timeout := DEFAULT_ADS_TIMEOUT,
                       ForceLock := ForceLock,
                       ForceUnlock := ForceUnlock);
  IF NOT fbSetMotorCtrlWord.Busy THEN
      fbSetMotorCtrlWord(Axis := Axis, Execute := FALSE);
  SetMotorCtrlWord := FALSE;
END_IF
END_IF
```

3.3.3 FB_SoEAX5000FirmwareUpdate

With the FB_SoEAX5000FirmwareUpdate 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 the update:

- 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 and firmware is made
• the name of the required firmware file is determined and a search for the file performed
• the firmware update is executed (if necessary)
• 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 takes place via the specified channel (A=0 or B=1) from 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 eFwUpdateState = eFwU_UpdateViaOtherChannel.

During the firmware update (eFwUpdateState = eFwU_FwUpdateInProgress), iLoadProgress signals the progress in percent.

**NOTE**

Faulty update due to interruptions

Interruptions during the update may result in it not being executed or executed incorrectly. Afterwards, the terminal may no longer be usable without the appropriate firmware.

The rules during the update are:
• The PLC and TwinCAT must not be stopped.
• The EtherCAT connection must not be interrupted.
• The AX5000 must not be switched off.

**VAR_INPUT**

sNetId : T_AmsNetId;
bExecute : BOOL;
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" *)
iReqEcState : UINT := EC_DEVICE_STATE_OP;
tTimeout : TIME := DEFAULT_ADS_TIMEOUT;

sNetId: AMS-NetID of the controller (IPC).
bExecute: The function block is activated by a positive edge at this input.

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, e.g. v1.05 Build 0009
• "v1.05_b0??" latest build of a specified version, e.g. v1.05
• "v1.??.??.b0??" latest build of a specified main version, e.g. v1
• "v?.??_b0??" latest build of the latest version
• "??" latest build of the latest version

Custom Firmware-Builds:
• "v1.05_b1009" for a specific build, e.g. v1.05 Build 0009
• "v1.05_b1??" latest build of a specified version, e.g. v1.05
• "v1.??.??.b1??" latest build of a specified main version, e.g. v1
• "v?.??_b1??" latest build of the latest version

...
Function blocks

- "v1.05_b8909" for a specific build, e.g. v1.05 Build 8909
- "v1.05_b89??" latest build of a specified version, e.g. v1.05
- "v1.??_b89??" latest build of a specified main version, e.g. v1
- "v?.??_b89??" latest build of the latest version

Debug-Builds:
- "v1.05_b9009" for a specific build, e.g. v1.05 Build 9009
- "v1.05_b90??" latest build of a specified version, e.g. v1.05
- "v1.??_b90??" latest build of a specified main version, e.g. v1
- "v?.??_b90??" latest build of the latest version

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

iReqEcState: Desired EtherCAT state after the update (only if an update is actually being executed). The statuses are defined in PLC Lib Tc2_EtherCAT as global constants.

tTimeout: Since the firmware update for large EtherCAT networks can take longer, only the timeout for individual internal ADS instances is specified here.

VAR_IN_OUT

VAR IN OUT
Axis : AXIS_REF;
END_VAR

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

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 a feedback 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.

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.

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
  iVarNumOfDrives : INT := 2;
END_VAR

VAR
Function blocks

```plaintext
fbFirmwareUpdate : ARRAY [1..iNumOfDrives] OF FB_SoEAX5000FirmwareUpdate;
Axes : ARRAY [1..iNumOfDrives] OF AXIS_REF;
sFirmwareVersion : ARRAY [1..iNumOfDrives] OF STRING(20) (* := 2('v1.04_b0002')*);
evFwUpdateState : 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;
bAnyBusy : BOOL;
bAnyError : BOOL;
END_VAR

CASE iUpdateState OF
  0:
    IF bExecute THEN
      iUpdateState := 1;
    END_IF
  1:
    FOR I := 1 TO iNumOfDrives DO
      fbFirmwareUpdate[I] (Axis := Axes[I],
                          bExecute := TRUE,
                          tTimeout := T#15s,
                          sFirmwareVersion := sFirmwareVersion[I],
                          sFirmwarePath := sFirmwarePath,
                          sNetId := 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] (iUpdateState := eFwUpdateState[I],
                           sSelectedFirmwareFile => sSelectedFirmwareFile[I],
                          );
      IF NOT fbFirmwareUpdate[I].bBusy THEN
        fbFirmwareUpdate[I](bExecute := FALSE, Axis := Axes[I]);
      IF fbFirmwareUpdate[I].bError THEN
        bAnyError := TRUE;
      END_IF
    END_FOR
    ELSE
      bAnyBusy := TRUE;
    END_IF
    END_CASE
```

3.3.4 **FB_SoEAX5000SetPositionOffset**

With the function block **FB_SoEAX5000SetPositionOffset** a position offset can be saved in the memory of the AX5000 or of the digital name plate of the connected motor encoder.
**VAR_INPUT**

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>BOOL</td>
</tr>
<tr>
<td>Position</td>
<td>LREAL</td>
</tr>
<tr>
<td>Relative</td>
<td>BOOL</td>
</tr>
<tr>
<td>Feedback</td>
<td>E_PositionOffsetFeedback</td>
</tr>
<tr>
<td>Memory</td>
<td>E_PositionOffsetMemory</td>
</tr>
</tbody>
</table>

**Execute**: The function block is enabled via a positive edge at this input.

**Position**: New actual position of the NC axis

**Relative**: When this flag is set, the position is interpreted relatively.

**Feedback**: Enumeration of the type E_PositionOffsetFeedback. Specifies which feedback is considered.

**Memory**: Enumeration of the type E_PositionOffsetMemory. Specifies which memory the newly calculated position offset should be stored in.

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>AXIS_REF</td>
</tr>
</tbody>
</table>

**Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>BOOL</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>ErrorID</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.

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

**ErrorID**: In the event of a set error output returns the ADS error code.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT ≥4020.57</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc2_MC2_Drive ≥V3.3.16.0</td>
</tr>
<tr>
<td>TwinCAT ≥4022.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**3.3.5 FB_SoEAX5000DeletePositionOffset**

With the function block FB_SoEAX5000DeletePositionOffset, a position offset can be deleted from the memory of the AX5000 or from the digital name plate of the connected motor encoder.
**VAR_INPUT**

Execute : BOOL;
Feedback : E_PositionOffsetFeedback;
Memory : E_PositionOffsetMemory;

**VAR_IN_OUT**

Axis : AXIS_REF;

**VAR_OUTPUT**

Busy : BOOL;
Error : BOOL;
ErrorID : UDINT;

**Execute**: The function block is enabled via a positive edge at this input.

**Feedback**: Enumeration of the type E_PositionOffsetFeedback. Specifies which feedback is considered.

**Memory**: Enumeration of the type E_PositionOffsetMemory. Specifies the memory from which the position offset is to be deleted.

**Axis**: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.

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

**ErrorID**: In the event of a set error output returns the ADS error code.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT ≥4020.57</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc2_MC2_Drive ≥V3.3.16.0</td>
</tr>
<tr>
<td>TwinCAT ≥4022.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.4 F_GetVersionTcMc2Drive

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

**FUNCTION F_GetVersionTcMc2Drive**: UINT

nVersionElement : INT;

nVersionElement: Version element to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
3.5 AX8000 CoE

3.5.1 FB_CoEAX8000SetPositionOffset

With the function block FB_CoEAX8000SetPositionOffset a position offset can be saved in the memory of the AX8000 or of the digital name plate of the connected motor encoder.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>The function block is enabled via a positive edge at this input.</td>
</tr>
<tr>
<td>Position</td>
<td>New actual position of the NC axis</td>
</tr>
<tr>
<td>Relative</td>
<td>When this flag is set, the position is interpreted relatively.</td>
</tr>
<tr>
<td>Feedback</td>
<td>Enumeration of the type [E_PositionOffsetFeedback] [50]. Specifies which feedback is considered.</td>
</tr>
<tr>
<td>Memory</td>
<td>Enumeration of the type [E_PositionOffsetMemory] [50]. Specifies which memory the newly calculated position offset should be stored in.</td>
</tr>
</tbody>
</table>

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>This output is set when the function block is activated, and remains set until a feedback is received.</td>
</tr>
<tr>
<td>Error</td>
<td>This output is set after the Busy output has been reset when an error occurs in the transmission of the command.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>In the event of a set error output returns the ADS error code.</td>
</tr>
</tbody>
</table>
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT ≥4020.57</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc2_MC2_Drive ≥V3.3.16.0</td>
</tr>
<tr>
<td>TwinCAT ≥4022.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.5.2 FB_CoEAX8000DeletePositionOffset

With the function block FB_CoEAX8000DeletePositionOffset, a position offset can be deleted from the memory of the AX8000 or from the digital name plate of the connected motor encoder.

#### VAR_INPUT

```plaintext
VAR_INPUT
  Execute : BOOL;
  Position : LREAL;
  Relative : BOOL;
  Feedback : E_PositionOffsetFeedback;
  Memory : E_PositionOffsetMemory;
END_VAR
```

**Execute**: The function block is enabled via a positive edge at this input.

**Position**: New position of the NC axis is displayed.

**Relative**: When this flag is set, the position is interpreted relatively.

**Feedback**: Enumeration of the type `E_PositionOffsetFeedback`[^50]. Specifies which feedback is considered.

**Memory**: Enumeration of the type `E_PositionOffsetMemory`[^50]. Specifies the memory from which the position offset is to be deleted.

#### VAR_IN_OUT

```plaintext
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

**Axis**: The axis data structure of type `AXIS_REF` addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

#### VAR_OUTPUT

```plaintext
VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  ErrorID : UDINT;
END_VAR
```

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.

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

**ErrorID**: In the event of a set error output returns the ADS error code.

[^50]: Please refer to the documentation for the specific values and meanings of `E_PositionOffsetFeedback` and `E_PositionOffsetMemory`.
# Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT ≥4020.57</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc2_MC2_Drive ≥V3.3.16.0</td>
</tr>
<tr>
<td>TwinCAT ≥4022.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 3.6 EL72xx CoE

### 3.6.1 FB_CoEEL72xxSetPositionOffset

With the function block FB_CoEEL72xxSetPositionOffset, a position offset can be stored in the memory of the EL72xx. Saving in the digital name plate of the connected motor encoder is not currently provided for.

#### VAR_INPUT

```plaintext
VAR_INPUT
  Execute : BOOL;
  Position : LREAL;
  Relative : BOOL;
  Feedback : E_PositionOffsetFeedback;
  Memory : E_PositionOffsetMemory;
END_VAR
```

**Execute**: The function block is enabled via a positive edge at this input.

**Position**: New actual position of the NC axis

**Relative**: When this flag is set, the position is interpreted relatively.

**Feedback**: Enumeration of the type `E_PositionOffsetFeedback` [50]. Specifies which feedback is considered.

**Memory**: Enumeration of the type `E_PositionOffsetMemory` [50]. Specifies which memory the newly calculated position offset should be stored in.

#### VAR_IN_OUT

```plaintext
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

**Axis**: The axis data structure of type `AXIS_REF` addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

#### VAR_OUTPUT

```plaintext
VAR_OUTPUT
  Busy : BOOL;
  Error : BOOL;
  ErrorID : UDINT;
END_VAR
```

**Busy**: This output is set when the function block is activated, and remains set until a feedback is received.
Error: This output is set after the Busy output has been reset when an error occurs in the transmission of the command.

ErrorID: In the event of a set error output returns the ADS error code.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform type</th>
<th>PLC libraries to include</th>
<th>Firmwareversion</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT &gt;4020.57</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc2_MC2_Drive ≥V3.3.17.0</td>
<td>≥ 19</td>
<td>≥0030</td>
</tr>
<tr>
<td>TwinCAT &gt;4022.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TwinCAT ≥4024.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.6.2 FB_CoEEL72xxDeletePositionOffset

With the function block FB_CoEEL72xxDeletePositionOffset, a position offset can be deleted from the memory of the EL72xx.

VAR_INPUT

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
</tr>
<tr>
<td>Position</td>
</tr>
<tr>
<td>Relative</td>
</tr>
<tr>
<td>Feedback</td>
</tr>
<tr>
<td>Memory</td>
</tr>
</tbody>
</table>

Execute: The function block is enabled via a positive edge at this input.

Position: New position of the NC axis is displayed.

Relative: When this flag is set, the position is interpreted relatively.

Feedback: Enumeration of the type E_PositionOffsetFeedback [50]. Specifies which feedback is considered.

Memory: Enumeration of the type E_PositionOffsetMemory [50]. Specifies the memory from which the position offset is to be deleted.

VAR_IN_OUT

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
</tr>
</tbody>
</table>

Axis: The axis data structure of type AXIS_REF addresses an axis unambiguously within the system. Among other things it contains the current status of the axis such as the position, the velocity and the error state.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>ErrorID</td>
</tr>
</tbody>
</table>

Busy: This output is set when the function block is activated, and remains set until a feedback is received.
Error: This output is set after the Busy output has been reset when an error occurs in the transmission of the command.

ErrorID: In the event of a set error output returns the ADS error code.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform type</th>
<th>PLC libraries to include</th>
<th>Firmwareversion</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT &gt;4020.57</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc2_MC2_Drive ≥V3.3.17.0</td>
<td>≥ 19</td>
<td>≥0030</td>
</tr>
<tr>
<td>TwinCAT &gt;4022.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TwinCAT ≥4024.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4 Data types

4.1 E_FwUpdateState

The enumeration E_FwUpdateState describes the state of a 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_ScanFirmwareError := -7,
    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

<table>
<thead>
<tr>
<th>E_FwU_NoError</th>
<th>Initial state</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_FwU_CheckCfgIdentity</td>
<td>Read the configured slave types (number of channels, current, revision).</td>
</tr>
<tr>
<td>E_FwU_CheckSlaveCount</td>
<td>Determine the configured number of slaves.</td>
</tr>
<tr>
<td>E_FwU_CheckFindSlavePos</td>
<td>Search for the slave address in the master object directory.</td>
</tr>
<tr>
<td>E_FwU_WaitForScan</td>
<td>Wait for online scan.</td>
</tr>
<tr>
<td>E_FwU_ScanningSlaves</td>
<td>Online scan of the slaves.</td>
</tr>
<tr>
<td>E_FwU_CheckScannedIdentity</td>
<td>Read the scanned slave types (number of channels, current, revision).</td>
</tr>
<tr>
<td>E_FwU_CheckScannedFirmware</td>
<td>Read the firmware version.</td>
</tr>
<tr>
<td>E_FwU_FindFirmwareFile</td>
<td>Search for the selected firmware file.</td>
</tr>
<tr>
<td>E_FwU_WaitForUpdate</td>
<td>Wait for status of the update.</td>
</tr>
<tr>
<td>E_FwU_WaitForSlaveState</td>
<td>Determine the EtherCAT slave status.</td>
</tr>
<tr>
<td>E_FwU_StartFwUpdate</td>
<td>Start the firmware update.</td>
</tr>
<tr>
<td>E_FwU_FwUpdateInProgress</td>
<td>Firmware update active.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>eFwU_FwUpdateDone</td>
<td>Firmware update successfully completed.</td>
</tr>
<tr>
<td>eFwU_NoFwUpdateRequired</td>
<td>No firmware update required.</td>
</tr>
<tr>
<td>eFwU_UpdateViaOtherChannelActive</td>
<td>Update takes place via the other axis channel.</td>
</tr>
<tr>
<td>eFwU_UpdatedViaOtherChannel</td>
<td>Update took place via the other axis channel.</td>
</tr>
</tbody>
</table>

### Update error

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eFwU_GetSlaveIdentityError</td>
<td>Reading of the configured slave type failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_GetSlaveCountError</td>
<td>Determination of the configured number of slaves failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_GetSlaveAddrError</td>
<td>Search for the slave address in the master object directory failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_StartScanError</td>
<td>Start of the online scan failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_ScanStateError</td>
<td>Online scan failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_GetSlaveStateError</td>
<td>Reading of the scanned slave types (number of channels, current, revision) failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_StartScanError</td>
<td>Determination of the EtherCAT slave status failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_ScanFirmwareError</td>
<td>Reading of the firmware version failed (see iAdsErrId + iSercosErrId).</td>
</tr>
<tr>
<td>eFwU_FindFileError</td>
<td>Search for the selected firmware file failed (see iAdsErrId).</td>
</tr>
<tr>
<td>eFwU_CfgTypeInNoAX5xxx</td>
<td>The configured slave is not an AX5000.</td>
</tr>
<tr>
<td>eFwU_ScannedTypeInNoAX5xxx</td>
<td>The scanned slave is not an AX5000.</td>
</tr>
<tr>
<td>eFwU_ChannelMismatch</td>
<td>The number of configured and found channels of the AX5000 do not match.</td>
</tr>
<tr>
<td>eFwU_ChannelMismatch_1Cfg_2Scanned</td>
<td>Single-channel device configured, but two-channel device found.</td>
</tr>
<tr>
<td>eFwU_ChannelMismatch_2Cfg_1Scanned</td>
<td>Two-channel device configured, but single-channel device found.</td>
</tr>
<tr>
<td>eFwU_CurrentMismatch</td>
<td>AX5000 type does not match in terms of current, e.g. AX5103 (3 A) configured, but AX5106 (6 A) found.</td>
</tr>
<tr>
<td>eFwU_FwUpdateError</td>
<td>General update error (see iAdsErrId)</td>
</tr>
<tr>
<td>eFwU_ReqSlaveStateError</td>
<td>Switching to the desired EtherCAT status failed.</td>
</tr>
</tbody>
</table>

#### 4.2 E_PositionOffsetMemory

```plaintext
TYPE E_PositionOffsetMemory :=
  ePositionOffsetMemory_Encoder := 0
  ePositionOffsetMemory_Drive := 1
) BYTE;
END_TYPE
```

ePositionOffsetMemory_Encoder: As memory for the new position offset the motor name plate of the encoder is selected.

ePositionOffsetMemory_Drive: As memory for the new position offset the internal memory of the drive is selected.

#### 4.3 E_PositionOffsetFeedback

```plaintext
TYPE E_PositionOffsetFeedback :=
  ePositionOffsetFeedback1 := 0
  ePositionOffsetFeedback2 := 1
) BYTE;
END_TYPE
```

ePositionOffsetFeedback1: Reference for the recalculation of the position offset is Feedback System 1.

ePositionOffsetFeedback2: Reference for the recalculation of the position offset is Feedback System 2.
4.4 E_DriveErrorCodes

TYPE E_DriveErrorCodes :
  MC_Error_MaxNcPositionOverrun := 16#4BB0
  MC_Error_MaxPositionOffsetOverrun := 16#4BB1
  MC_Error_MinNcPositionUnderrun := 16#4BB2
  MC_Error_MinPositionOffsetUnderrun := 16#4BB3
  MC_Error_WrongTargetForFeedbackOrMemory := 16#4BB4
  MC_Error_PositionReinitializationFailed := 16#4BB5
  MC_Error_CommandRejectedNoResponse := 16#4BB6
  MC_Error_CommandRejectedWithResponse := 16#4BB7
) UDINT;
END_TYPE
5 Global constants

5.1 SoE Parameter Access

The parameters of the individual parameter sets can be accessed with the help of these constants.

Global_Variables

```plaintext
VAR_GLOBAL CONSTANT
  S_0_IDNs : WORD := 16#0000;
  S_1_IDNs : WORD := 16#1000;
  S_2_IDNs : WORD := 16#2000;
  S_3_IDNs : WORD := 16#3000;
  S_4_IDNs : WORD := 16#4000;
  S_5_IDNs : WORD := 16#5000;
  S_6_IDNs : WORD := 16#6000;
  S_7_IDNs : WORD := 16#7000;
  P_0_IDNs : WORD := 16#8000;
  P_1_IDNs : WORD := 16#9000;
  P_2_IDNs : WORD := 16#A000;
  P_3_IDNs : WORD := 16#B000;
  P_4_IDNs : WORD := 16#C000;
  P_5_IDNs : WORD := 16#D000;
  P_6_IDNs : WORD := 16#E000;
  P_7_IDNs : WORD := 16#F000;
END_VAR
```
6 Samples

Sample project and sample configuration for AX5000 diagnostics

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2_Drive/Resources/zip/2327326731.zip

Sample project and sample configuration for IndraDrive Cs diagnostics

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2_Drive/Resources/zip/2327323403.zip