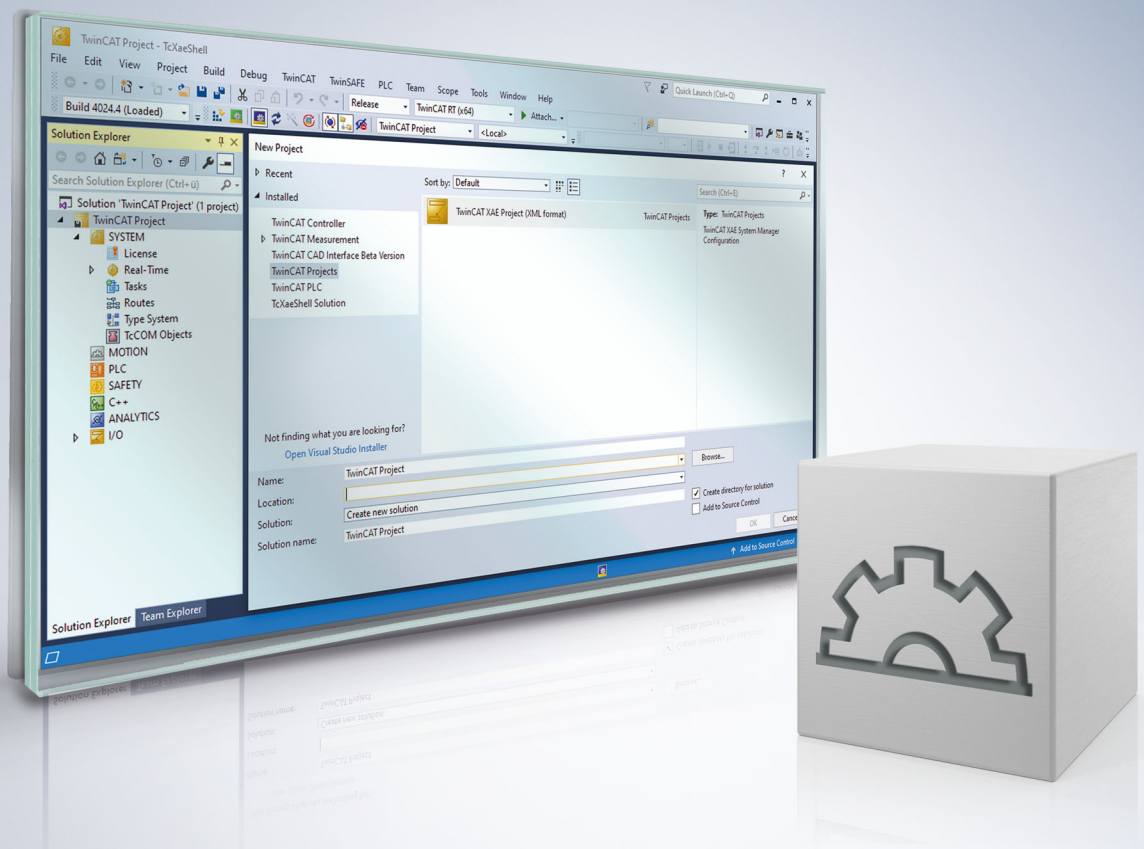
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc2_Drive



2022-10-13 | Version: 1.3.1

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

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

Copyright

1.2 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

⚠ DANGER

Serious risk of injury!

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

⚠ WARNING

Risk of injury!

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

⚠ CAUTION

Personal injuries!

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

NOTE

Damage to the environment or devices

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

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

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

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

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

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

2 Overview

The Tc2_Drive library should no longer be used in newer projects. Please use the Tc2_MC2_Drive library instead (see documentation TwinCAT 3 PLC Lib Tc2_MC2_Drive).

The Tc2_Drive library includes functions and function blocks for SoE drives that access the drive via a drive reference.

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.

Drive library	Use	Access to the drive	Note
Tc2_Drive See documentation TwinCAT 3 PLC Lib: Tc2_Drive	Use the Tc2_Drive library if you use the drive entirely from the PLC (i.e. without NC).	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 [▶ 11])	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.
Tc2_NcDrive See documentation TwinCAT 3 PLC Lib: Tc2_NcDrive	Use the Tc2_NcDrive library if you are using the drive via the NC with the Tc2_Nc libraries.	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.	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.
Tc2_MC2_Drive See: TwinCAT 3 PLC Lib Tc2_MC2_Drive documentation	Use the Tc2_MC2_Drive library if you are using the drive via the NC with the Tc2_MC2 library.	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.	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.

Note the differences when using the drive libraries with AX5000 and Bosch Rexroth IndraDrive CS (see Samples [▶ 55])

Functions

Name	Description
F_GetVersionTcDrive [▶ 40]	Reads version information from the PLC library. The function will be replaced by the global structure stLibVersion_Tc2_Drive.

Name	Description
F_ConvWordToSTAX5000C1D [▶ 31]	Converts the C1D word (S-0-0011) of the AX5000 to a structure ST_AX5000_C1D for Class 1 diagnosis [▶ 48]
F_ConvWordToSTAX5000DriveStatus [▶ 32]	Converts the Drive status word (S-0-0135) of the AX5000 into the structure ST_AX5000DriveStatus [▶ 49]
F_ConvWordToSTIndraDriveCsC1D [▶ 39]	Converts the C1D word (S-0-0011) of the IndraDrive Cs into the structure ST_IndraDriveCs_C1D for Class 1 diagnosis [▶ 50]
F_ConvWordToSTIndraDriveCsDriveStatus [▶ 39]	Converts the drive status word (S-0-0135) of the IndraDrive Cs into a structure ST_IndraDriveCsDriveStatus [▶ 50]

Function blocks

Name	Description
FB_SoEReset_ByDriveRef [▶ 13]	Resets the drive (S-0-0099).
FB_SoEWritePassword_ByDriveRef [▶ 14]	Sets the drive password (S-0-0267).
FB_SoEExecuteCommand_ByDriveRef [▶ 15]	Executes a command.
FB_SoEReadDiagMessage_ByDriveRef [▶ 20]	Reads the diagnostic message (S-0-0095).
FB_SoEReadDiagNumber_ByDriveRef [▶ 21]	Reads the diagnostic number (S-0-0390).
FB_SoEReadDiagNumberList_ByDriveRef [▶ 22]	Reads the diagnostic number list (up to 30 entries) (S-0-0375).
FB_SoEReadClassXDiag_ByDriveRef [▶ 24]	Reads Class 1 diagnosis (S-0-0011) ... Class 3 diagnosis (S-0-0013).
FB_SoEWriteCommandControl_ByDriveRef [▶ 17]	Sets the Command Control.
FB_SoEReadCommandState_ByDriveRef [▶ 18]	Checks the command status.
FB_SoERead_ByDriveRef	Reads a parameter (see PLC Lib Tc2_EtherCAT).
FB_SoEWrite_ByDriveRef	Writes a parameter (see PLC Lib Tc2_EtherCAT).
FB_SoEReadAmplifierTemperature_ByDriveRef [▶ 26]	Reads the drive temperature (S-0-0384).
FB_SoEReadMotorTemperature_ByDriveRef [▶ 27]	Reads the motor temperature (S-0-0383).
FB_SoEReadDcBusCurrent_ByDriveRef [▶ 30]	Reads the DC bus current (S-0-0381).
FB_SoEReadDcBusVoltage_ByDriveRef [▶ 28]	Reads the DC bus voltage (S-0-0380).
FB_SoEAX5000ReadActMainVoltage_ByDriveRef [▶ 32]	Reads the mains voltage (P-0-0200).
FB_SoEAX5000SetMotorCtrlWord_ByDriveRef [▶ 34]	Sets the Motor Control Word (P-0-0096).
FB_SoEAX5000FirmwareUpdate_ByDriveRef [▶ 35]	Executes an automatic firmware update for the AX5000.
FB_CoEDriveEnable [▶ 40]	Enables a CoE drive.
FB_CoEDriveMoveVelocity [▶ 41]	Generates a simple three-phase velocity profile that can be used to supply a CoE drive directly.
FB_SoEDriveEnable [▶ 43]	Enables a SoE drive.

Name	Description
FB_SoEDriveMoveVelocity [▶ 44]	Generates a simple three-phase velocity profile that can be used to supply a SoE drive directly.

Requirements

Component	Version
TwinCAT on the development computer	3.1 Build 4016 or higher
TwinCAT on the Windows CE-Image	3.1 Build 4016 or higher
TwinCAT on the Windows XP-Image	3.1 Build 4016 or higher

3 Drive reference ST_DriveRef

The drive is accessed via a drive reference. Within the library, the ST_DriveRef structure from the Tc2_EtherCAT library is used for this with the NetID as a string. Since the NetID usually exists as a byte array at I/O level, a structure ST_PlcDriveRef, also from the Tc2_EtherCAT library, with the NetID as a byte array is additionally provided. The two structures must be transferred into each other.

Structure ST_PlcDriveRef

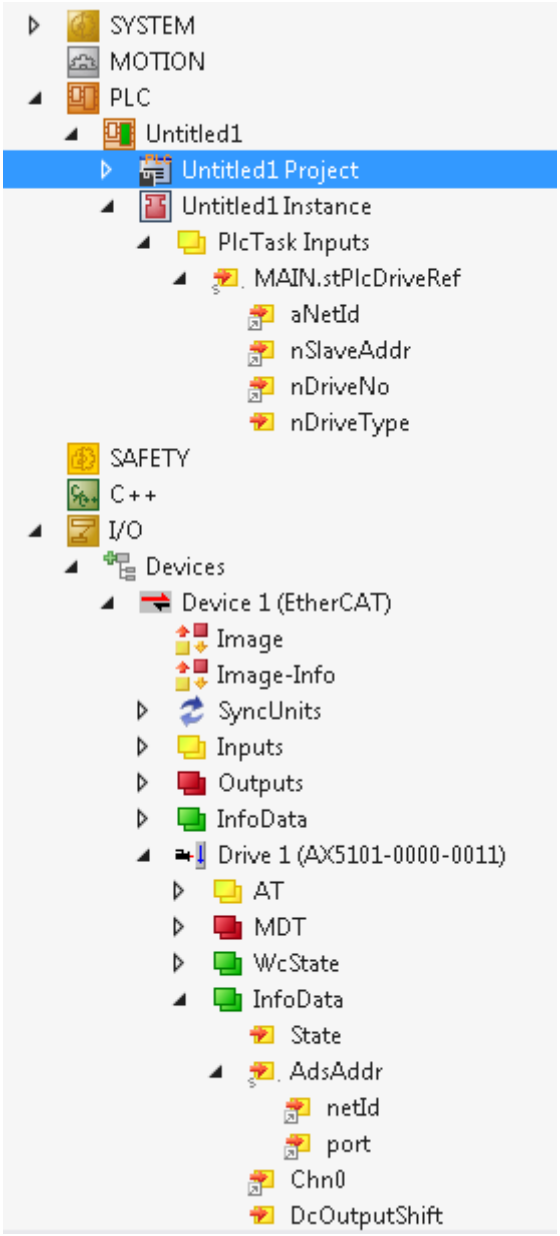
```
TYPE ST_PlcDriveRef :
STRUCT
    aNetId     : T_AmsNetIdArr; (* AmsNetId (array[0..5] of bytes) of the EtherCAT master device.*)
    nSlaveAddr : UINT; (* Address of the slave device.*)
    nDriveNo   : BYTE; (* Drive number*)
    nDriveType : BYTE; (* Drive type*)
END_STRUCT
END_TYPE
```

Structure ST_DriveRef

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

Mapping the drive reference to the PLC

The drive reference can be mapped to the PLC in the System Manager. To do this, allocate an instance of the structure ST_PlcDriveRef as AT %I*. Subsequently, link aNetID to netId, nSlaveAddr 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.



Transfer of ST_PlcDriveRef and ST_DriveRef

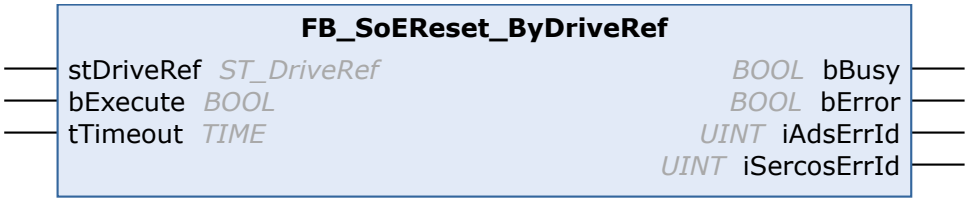
The function blocks in the library Tc2_Drive use an instance of the structure ST_DriveRef. Unlike the structure ST_PlcDriveRef, NetID is expected to be T_AmsNetId (i.e. STRING(23)). To convert the byte array, use the F_CreateAmsNetId() function of the PLC Lib Tc2_System.

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

4 Function blocks

4.1 General SoE

4.1.1 FB_SoEReset_ByDriveRef



The drive (S-0-0099) can be reset with the function block FB_SoEReset_ByDriveRef. 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. An NC reset will not be performed.

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time (10 s) allowed for the execution of the function block.

Outputs

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

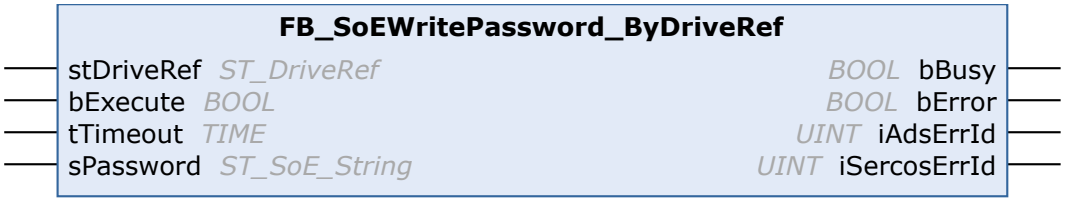
Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.

Sample

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

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

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

4.1.2 FB_SoEWritePassword_ByDriveRef



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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: Drive reference ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time (10 s) allowed for the execution of the function block.
sPassword	ST_SoE_String	Password as a Sercos string

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.

Sample

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

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

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

4.1.3 FB_SoEExecuteCommand_ByDriveRef

With the FB_SoEExecuteCommand_ByDriveRef function block a command can be executed.

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])

Name	Type	Description
nIdn	WORD	Parameter number to which FB_SoEExecuteCommand_ByDriveRef refers, e.g. "P_0_IDN + 160" for P-0-0160
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time (10 s) allowed for the execution of the function block.

Outputs

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

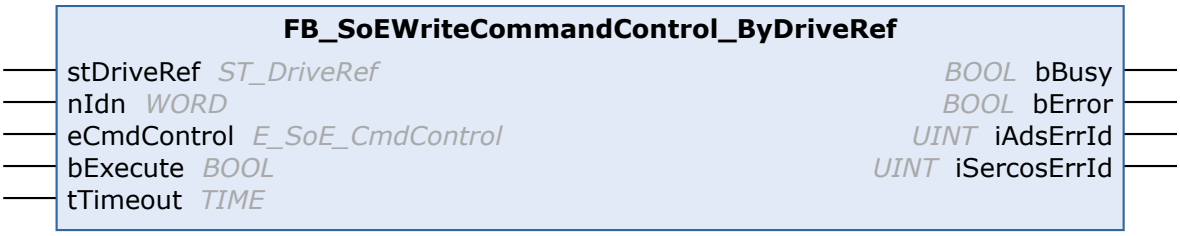
Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.

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
```

4.1.4 Function blocks for commands

4.1.4.1 FB_SoEWriteCommandControl_ByDriveRef



With the FB_SoEWriteCommandControl_ByDriveRef function block a command can either be prepared, started or aborted.

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
nIdn	WORD	Parameter number to which FB_SoEReadCommandState_ByDriveRef refers, e.g. "P_0_IDN + 23" for P-0-0023.
eCmdControl	E_SoE_CmdControl	Indicates, if a command should be prepared (eSoE_CmdControl_Set := 1), executed (eSoE_CmdControl_SetAndEnable := 3) or aborted (eSoE_CmdControl_Cancel := 0).
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time (10 s) allowed for the execution of the function block.

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	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
    );
    IF NOT fbWriteCommandControl.bBusy THEN
        fbWriteCommandControl(stDriveRef := stDriveRef, bExecute := FALSE);
        bWriteCommandControl := FALSE;
    END_IF
END_IF
```

4.1.4.2 FB_SoEReadCommandState_ByDriveRef

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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
nIdn	WORD	Parameter number to which FB_SoEReadCommandState_ByDriveRef refers, e.g. "P_0_IDN + 160" for P-0-0160
bExecute	BOOL	The function block is enabled via a positive edge at this input.

Name	Type	Description
tTimeout	TIME	Maximum time (10 s) allowed for the execution of the function block.

Outputs

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

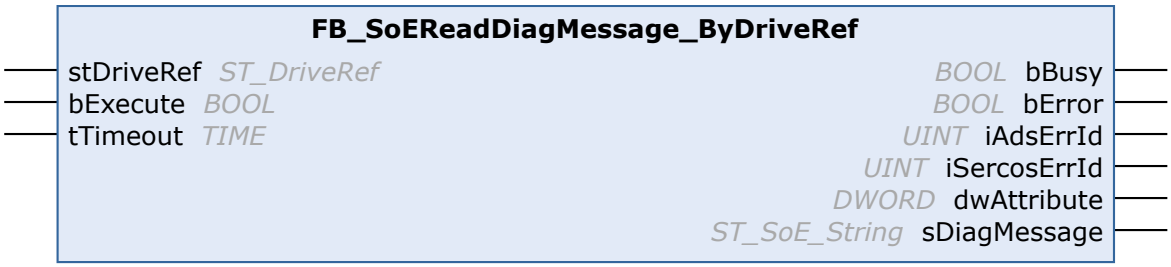
Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
eCmdState	eSoE_CmdState	Returns the command status [▶ 52].
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.

Sample

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

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

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

4.1.5 Function blocks for diagnostics

4.1.5.1 FB_SoEReadDiagMessage_ByDriveRef



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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

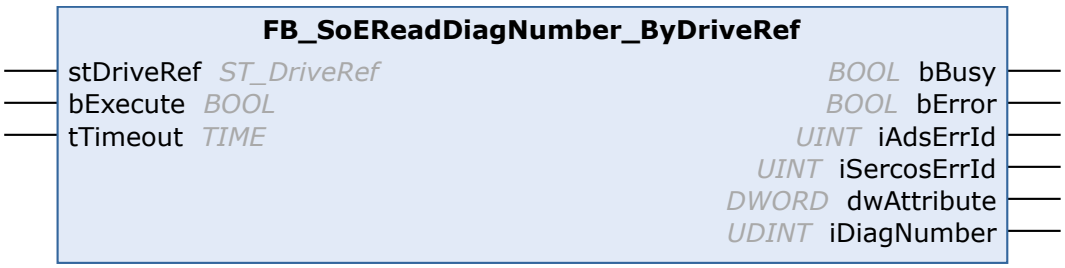
```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
    dwAttribute  : DWORD;
    sDiagMessage : ST_SoE_String;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
sDiagMessage	ST_SoE_String	Returns the diagnosis message.

Sample

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

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

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

4.1.5.2 FB_SoEReadDiagNumber_ByDriveRef



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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
iDiagNumber	UDINT	Returns the current diagnostic number.

Sample

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

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

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

4.1.5.3 FB_SoEReadDiagNumberList_ByDriveRef

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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.
piDiagNumber	POINTER TO ST_SoE_DiagNumList	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 items.
iSize	UDINT	Size of the list in bytes (as Sizeof())

Outputs

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

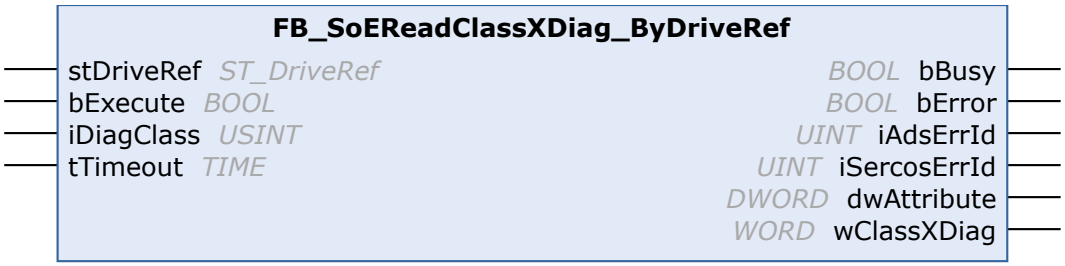
Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.

Sample

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

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

IF bDiagNumberList AND NOT bInit THEN
   fbDiagNumberList(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
```

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

4.1.5.4 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 [▶ 48] there is a conversion function F_ConvWordToSTAX5000C1D [▶ 31].

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
iDiagClass	USINT	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: Class 3 Diag (S-0-0013)
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

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

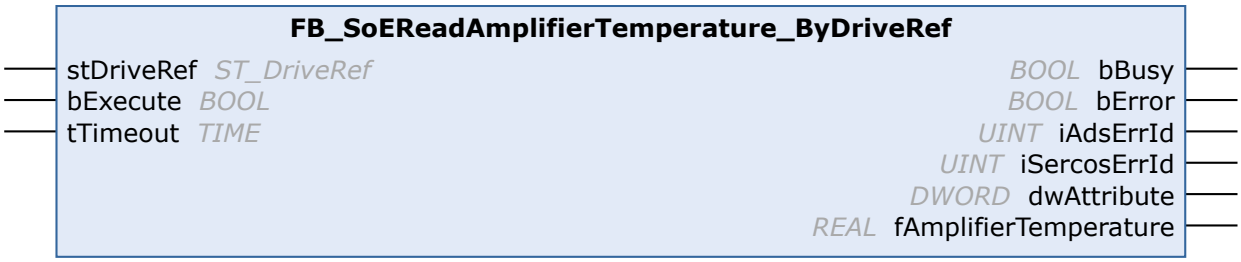
Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
wClassXDiag	WORD	Returns the current Class X diagnosis.

Sample

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

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

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

4.1.6 Function blocks for determining current values

4.1.6.1 FB_SoEReadAmplifierTemperature_ByDriveRef



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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

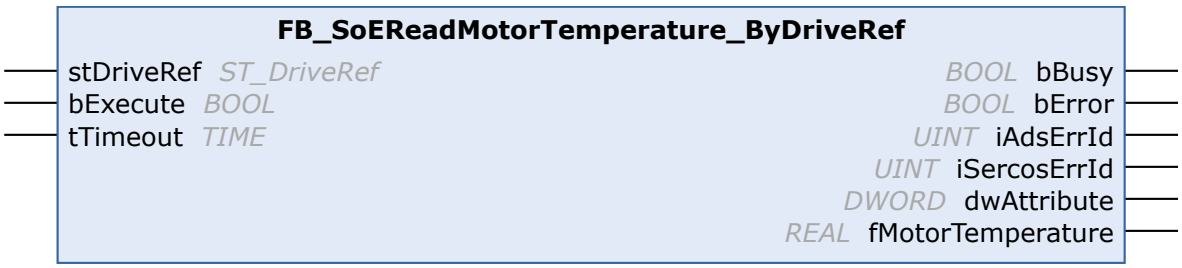
```
VAR_OUTPUT
    bBusy                : BOOL;
    bError               : BOOL;
    iAdsErrId            : UINT;
    iSercosErrId         : UINT;
    dwAttribute          : DWORD;
    fAmplifierTemperature : REAL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
fAmplifierTemperature	REAL	Returns the drive temperature (e.g. 26.2 corresponds to 26.2 °C).

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
```

4.1.6.2 FB_SoEReadMotorTemperature_ByDriveRef



With the function block FB_SoEReadMotorTemperature_ByDriveRef 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.

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

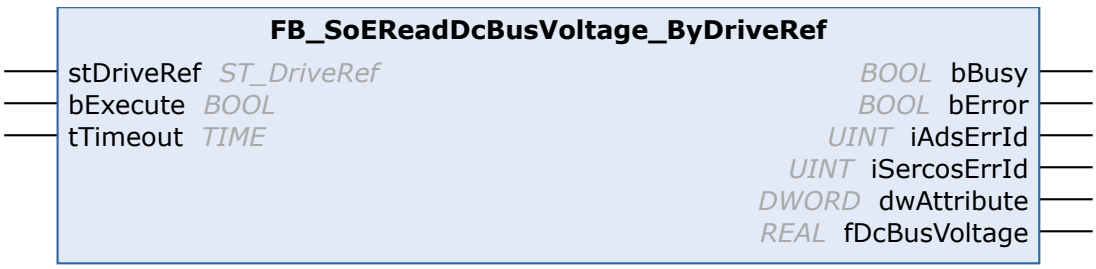
```
VAR_OUTPUT
    bBusy             : BOOL;
    bError            : BOOL;
    iAdsErrId         : UINT;
    iSercosErrId      : UINT;
    dwAttribute       : DWORD;
    fMotorTemperature : REAL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
fMotorTemperature	REAL	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

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

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

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

4.1.6.3 FB_SoEReadDcBusVoltage_ByDriveRef



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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
fDcBusVoltage	REAL	Supplies the DC bus voltage of the drive.

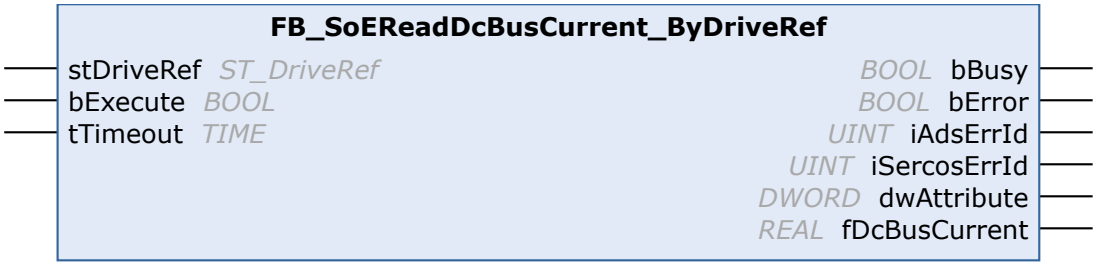
Sample

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

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

IF bReadDcBusVoltage AND NOT bInit THEN
```

```
   fbReadDcBusVoltage(stDriveRef := stDriveRef,
                      bExecute := TRUE,
                      tTimeout := DEFAULT_ADS_TIMEOUT,
                      fDcBusVoltage => fDcBusVoltage );

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

4.1.6.4 FB_SoEReadDcBusCurrent_ByDriveRef



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

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.

Name	Type	Description
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
fDcBusCurrent	REAL	Returns the DC bus current (e.g. 2,040 equals 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 bReadDcBusCurrent AND NOT bInit THEN
   fbReadDcBusCurrent(
     stDriveRef := stDriveRef,
     bExecute := TRUE,
     tTimeout := DEFAULT_ADS_TIMEOUT,
     fDcBusCurrent=>fDcBusCurrent
   );
   IF NOT fbReadDcBusCurrent.bBusy THEN
      fbReadDcBusCurrent(stDriveRef := stDriveRef, bExecute := FALSE);
      bReadDcBusCurrent := FALSE;
   END_IF
END_IF
```

4.2 AX5000 SoE

4.2.1 Conversion functions

4.2.1.1 F_ConvWordToSTAX5000C1D

With this function the Class 1 diagnosis FB_SoEReadClassXDiag_ByDriveRef [▶ 24] (S-0-0011) can be changed to an ST_AX5000_C1D [▶ 48] structure.

Inputs

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

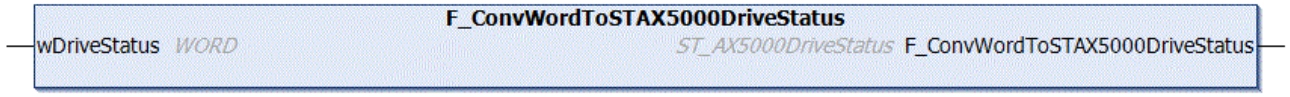
Name	Type	Description
wClass1Diag	WORD	Class 1 diagnosis Word from S-0-0011 (see FB_SoEReadClassXDiag_ByDriveRef [▶ 24])

Return value

```
FUNCTION F_ConvWordToSTAX5000C1D : ST_AX5000_C1D
```

Name	Type	Description
F_ConvWordToSTAX5000C1D	ST_AX5000_C1D [▶ 48]	Return value of the function. Class 1 diagnosis as structure.

4.2.1.2 F_ConvWordToSTAX5000DriveStatus



With this function the Drive status word (S-0-0135) can be changed to a structure ST_AX5000DriveStatus [▶ 49].

Inputs

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

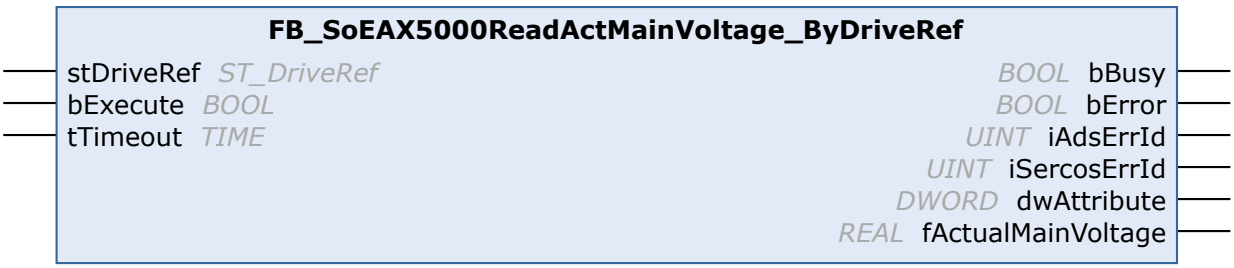
Name	Type	Description
wDriveStatus	WORD	Drive status word from S-0-0135 (can be read with FB_SoE_Read_ByDriveRef, can be mapped if necessary).

Return value

```
FUNCTION F_ConvWordToSTAX5000DriveStatus : ST_AX5000DriveStatus
```

Name	Type	Description
F_ConvWordToSTAX5000DriveStatus	ST_AX5000DriveStatus	Return value of the function. Axis state as structure.

4.2.2 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.

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.

Outputs

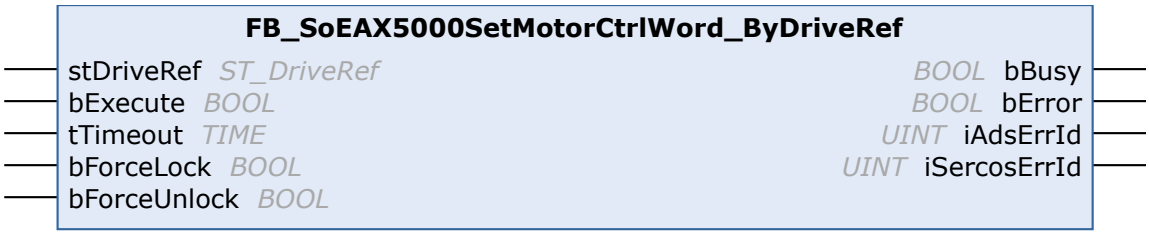
```
VAR_OUTPUT
    bBusy              : BOOL;
    bError             : BOOL;
    iAdsErrId          : UINT;
    iSercosErrId       : UINT;
    dwAttribute        : DWORD;
    fActualMainVoltage : REAL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
dwAttribute	DWORD	Returns the attributes of the Sercos parameter.
fActualMainVoltage	REAL	Returns the peak value of the current mains voltage of the AX5000 (e.g. 303.0 corresponds to 303.0 V).

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
```

4.2.3 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.

Inputs

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

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.
bForceLock	BOOL	Activates the brake independently of the enable.
bForceUnlock	BOOL	Releases the brake independently of the enable.

Outputs

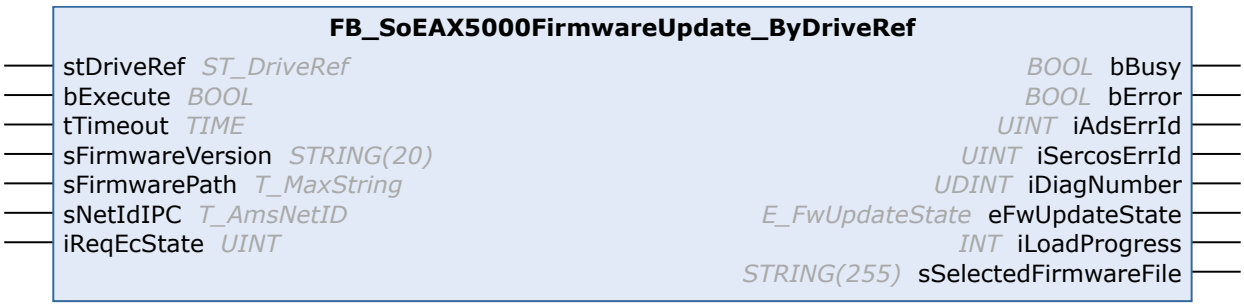
```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    iAdsErrId    : UINT;
    iSercosErrId : UINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.

Sample

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

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

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

4.2.4 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 the update:

- the configured slave type is determined, e.g. AX5103-0000-0010.
- the current slave with the specified slave address is determined, 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 specified 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 is updated 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 drive amplifier 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.

Inputs

```
VAR_INPUT
   stDriveRef       : ST_DriveRef;
   bExecute         : BOOL;
   tTimeout         : TIME := DEFAULT_ADS_TIMEOUT;
   sFirmwareVersion : STRING(20);  (* version string vx_yy_bnnnn, e.g. "v1.05_b0009" for v1.05 Build
 0009*)
   sFirmwarePath    : T_MaxString; (* drive:\path, e.g. "C:
\TwinCAT\Io\TcDriveManager\FirmwarePool" *)
   sNetIdIPC        : T_AmsNetId;
   iReqEcState      : UINT := EC_DEVICE_STATE_OP;
END_VAR
```

Name	Type	Description
stDriveRef	ST_DriveRef	Reference to the drive. The reference to the drive can be linked directly to the PLC in the System Manager. To do this, an instance of the structure ST_PlcDriveRef must be allocated and the NetID must be converted from the byte array into a string. (Type: ST_DriveRef [▶ 11])
bExecute	BOOL	The function block is enabled via a positive edge at this input.
tTimeout	TIME	Maximum time allowed for the execution of the function block.
sFirmwareVersion	STRING(20)	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_b**00**09" for a specific build, e.g. v1.05 Build 0009 • "v1.05_b**00**??" latest build of a specified version, e.g. v1.05 • "v1.??_b**00**??" latest build of a specified main version, e.g. v1 • " " latest build of the latest version **Customer-specific Firmware-Builds:** • "v1.05_b**10**09" for a specific build, e.g. v1.05 Build 0009 • "v1.05_b**10**??" latest build of a specified version, e.g. v1.05 • "v1.??_b**10**??" latest build of a specified main version, e.g. v1 • "v?.??_b**10**??" latest build of the latest version • "v1.05_b**89**09" for a specific build, e.g. v1.05 Build 8909 • "v1.05_b**89**??" latest build of a specified version, e.g. v1.05 • "v1.??_b**89**??" latest build of a specified main version, e.g. v1 • "v?.??_b**89**??" latest build of the latest version **Debug-Builds:**

Name	Type	Description
		• "v1.05_b**90**09" for a specific build, e.g. v1.05 Build 9009 • "v1.05_b**90**??" latest build of a specified version, e.g. v1.05 • "v1.??_b**90**??" latest build of a specified main version, e.g. v1 • "v?.??_b**90**??" latest build of the latest version
sFirmwarePath	T_MaxString	Specifies the path for the firmware pool in which the firmware files are located, e.g. *C:\TwinCAT\Io\TcDriveManager\FirmwarePool*.
sNetIdIPC	T_AmsNetId	AMS-NetID of the controller (IPC)
iReqEcState	UINT	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.

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set when the function block is activated, and remains set until a feedback is received.
bError	BOOL	This output is set after the bBusy output has been reset when an error occurs in the transmission of the command.
iAdsErrId	UINT	In the case of a set bError output returns the ADS error code of the last executed command.
iSercosErrId	UINT	In the case of a set bError output returns the Sercos error of the last executed command.
iDiagNumber	UDINT	Returns the drive amplifier error of the last firmware update if the bError output is set.
eFwUpdateState	E_FwUpdateState	Returns the status of the firmware update (see E_FwUpdateState [▶ 47]).
iLoadProgress	INT	Returns the progress of the actual firmware update as a percentage.
sSelectedFirmwareFile	STRING(MAX_STRING_LENGTH)	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;
    sFirmwareVersion : ARRAY[1..iNumOfDrives] OF STRING(20) := 2('v1.05_b0009');
    eFwUpdateState : ARRAY[1..iNumOfDrives] OF E_FwUpdateState;
    sSelectedFirmwareFile: ARRAY [1..iNumOfDrives] OF STRING(MAX_STRING_LENGTH);
    iUpdateState : INT;
    bExecute : BOOL;
    sNetIdIPC : T_AmsNetId := '';
    sFirmwarePath  : T_MaxString := 'C:\TwinCAT\Io\TcDriveManager\FirmwarePool';
    I : INT;
```

```
    bAnyInit : BOOL;
    bAnyBusy : BOOL;
    bAnyError : BOOL;
END_VAR

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

4.3 IndraDrive Cs

4.3.1 Conversion functions

4.3.1.1 F_ConvWordToSTIndraDriveCsC1D

With this function the Class 1 diagnosis FB_SoEReadClassXDiag_ByDriveRef [▶ 24] (S-0-0011) can be changed to an ST_IndraDriveCs_C1D [▶ 50] structure.

Inputs

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

Name	Type	Description
wClass1Diag	WORD	Class 1 diagnosis Word from S-0-0011 (see FB_SoEReadClassXDiag_ByDriveRef [▶ 24]).

Return value

```
FUNCTION F_ConvWordToSTIndraDriveCsC1D : ST_IndraDriveCs_C1D
```

Name	Type	Description
F_ConvWordToSTIndraDriveCsC1D	ST_IndraDriveCs_C1D [▶ 50]	Return value of the function Class-1-Diagnosis as ST_IndraDrivesCs_C1D structure.

4.3.1.2 F_ConvWordToSTIndraDriveCsDriveStatus

With this function the Drive status word (S-0-0135) can be changed to a structure ST_IndraDriveCsDriveStatus [▶ 50] .

Inputs

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

Name	Type	Description
wClass1Diag	WORD	Drive status word from S-0-0135 Can be read with FB_SoE_Read_ByDriveRef, can be mapped if necessary.

Return value

```
FUNCTION F_ConvWordToSTIndraDriveCsDriveStatus : ST_IndraDriveCsDriveStatus
```

Name	Type	Description
F_ConvWordToSTIndraDriveCsDriveStatus	ST_IndraDriveCsDriveStatus [▶ 50]	Return value of the function. Drive status word as ST_IndraDriveCsDriveStatus structure.

4.4 F_GetVersionTcDrive

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

Inputs

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

Name	Type	Description
nVersionElement	INT	**nVersionElement**: version element to be read. Possible parameters: • 1 : major number; • 2 : minor number; • 3 : revision number;

Return value

```
FUNCTION F_GetVersionTcDrive : UINT
```

Name	Type	Description
F_GetVersionTcDrive	UINT	Return value of the function. Version element as UINT.

4.5 SimplePlcMotion

Simple PLC Motion function blocks enable easy operation of a drive directly from the PLC.

4.5.1 FB_CoEDriveEnable

The FB_CoEDriveEnable function block enables a CoE drive in order to be able to subsequently supply it with setpoints using the function block FB_CoEDriveMoveVelocity [▶ 41].

Inputs

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

Name	Type	Description
bEnable	BOOL	Activates the CoE drive.
bReset	BOOL	Performs a drive reset in the event of a fault. "Bit 7" is set in the drive control word.

/ Inputs/Outputs

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

Name	Type	Description
stCoeDriveIoInterface	ST_CoeDriveIoInterface	Data structure to which the process image of the CoE drive must be linked.

Outputs

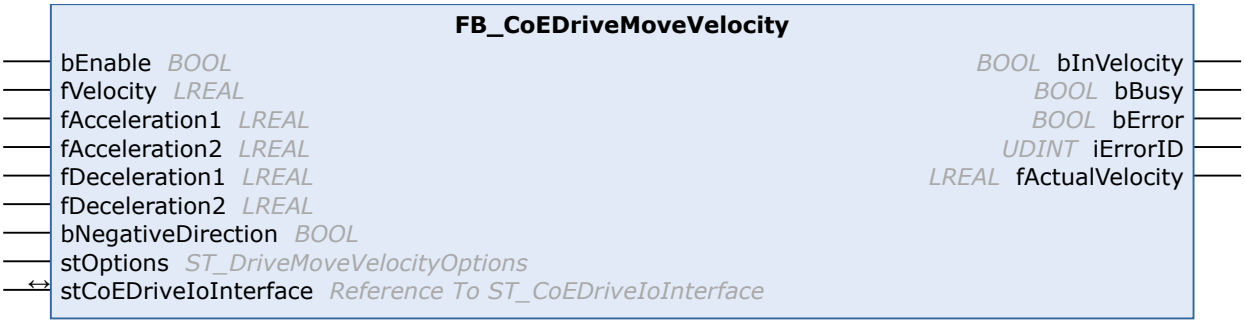
```
VAR_OUTPUT
    bStatus     : BOOL;
    bDriveError : BOOL;
END_VAR
```

Name	Type	Description
bStatus	BOOL	If bStatus=TRUE, the drive is ready for operation and follows the setpoints.
bDriveError	BOOL	The drive is in the error state.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

4.5.2 FB_CoEDriveMoveVelocity



The FB_CoEDriveMoveVelocity function block generates a simple three-phase velocity profile (without jerk limitation) that can be used to supply a CoE drive directly. Different accelerations or decelerations can be used below and above a parameterizable velocity threshold. The target velocity can be changed during operation.

The CoE drive must first be enabled via the function block FB_CoEDriveEnable [▶ 40].

Inputs

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

Name	Type	Description
bEnable	BOOL	Activates the setpoint generation.
fVelocity	LREAL	Target velocity. fVelocity can be changed during operation.
fAcceleration1	LREAL	Acceleration 1 is used below the parameterized velocity threshold stOptions. fVelocityThreshold.
fAcceleration2	LREAL	Acceleration 2 is used above the parameterized velocity threshold stOptions. fVelocityThreshold.
fDeceleration1	LREAL	Deceleration 1 is used below the parameterized velocity threshold stOptions. fVelocityThreshold.
fDeceleration2	LREAL	Deceleration 2 is used above the parameterized velocity threshold stOptions. fVelocityThreshold.
bNegativeDirection	BOOL	bNegativeDirection reverses the direction of travel.
stOptions	ST_DriveMoveVelocityOptions	Data structure with additional parameters.

/ Inputs/Outputs

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

Name	Type	Description
stCoEDriveIoInterface	ST_CoeDriveIoInterface	Process image of the CoE drive

Outputs

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

Name	Type	Description
bInVelocity	BOOL	Target velocity is reached.
bBusy	BOOL	bBusy is TRUE as long as the function block is active and a setpoint profile is being calculated.
bError	BOOL	bError becomes TRUE if an error occurs.
iErrorID	UDINT	Error number
fActualVelocity	LREAL	Currently reached velocity of the drive.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

4.5.3 FB_SoEDriveEnable

The FB_SoEDriveEnable function block enables a SoE drive in order to be able to subsequently supply it with setpoints using the function block FB_SoEDriveMoveVelocity [▶ 44].

Inputs

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

Name	Type	Description
bEnable	BOOL	Activates the SoE drive.

/ Inputs/Outputs

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

Name	Type	Description
stSoeDriveIoInterface	ST_SoEDriveIoInterface	Data structure to which the process image of the SoE drive must be linked.

Outputs

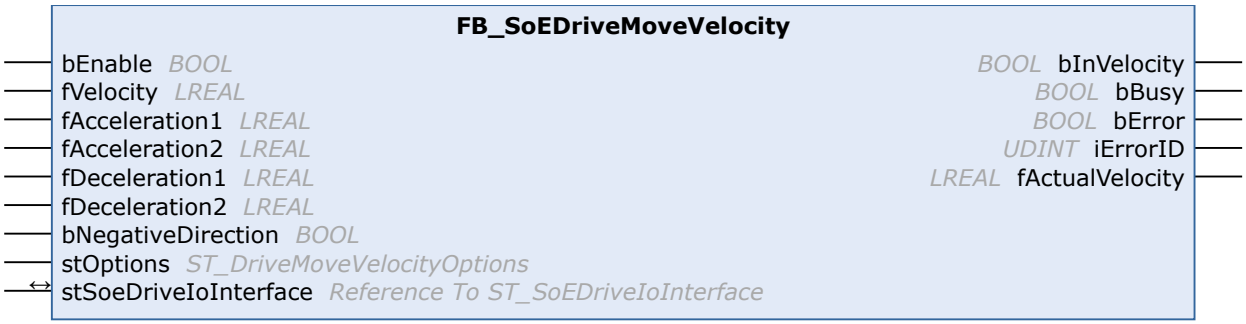
```
VAR_OUTPUT
    bStatus     : BOOL;
    bDriveError : BOOL;
END_VAR
```

Name	Type	Description
bStatus	BOOL	If bStatus=TRUE, the drive is ready for operation and follows the setpoints.
bDriveError	BOOL	The drive is in the error state.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

4.5.4 FB_SoEDriveMoveVelocity



The FB_SoEDriveMoveVelocity function block generates a simple three-phase velocity profile (without jerk limitation) that can be used to supply an SoE drive directly. Different accelerations or decelerations can be used below and above a parameterizable velocity threshold. The target velocity can be changed during operation.

The SoE drive must first be enabled via the function block FB_SoEDriveEnable [▶ 43].

Inputs

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

Name	Type	Description
bEnable	BOOL	Activates the setpoint generation.
fVelocity	LREAL	Target velocity. fVelocity can be changed during operation.
fAcceleration1	LREAL	Acceleration 1 is used below the parameterized velocity threshold stOptions. fVelocityThreshold.
fAcceleration2	LREAL	Acceleration 2 is used above the parameterized velocity threshold stOptions. fVelocityThreshold.
fDeceleration1	LREAL	Deceleration 1 is used below the parameterized velocity threshold stOptions. fVelocityThreshold.
fDeceleration2	LREAL	Deceleration 2 is used above the parameterized velocity threshold stOptions. fVelocityThreshold.
bNegativeDirection	BOOL	bNegativeDirection reverses the direction of travel.
stOptions	ST_DriveMoveVelocityOptions	Data structure with additional parameters.

/ Inputs/Outputs

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

Name	Type	Description
stSoEDriveIoInterface	ST_SoeDriveIoInterface	Process image of the CoE drive

Outputs

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

Name	Type	Description
bInVelocity	BOOL	Target velocity is reached.
bBusy	BOOL	bBusy is TRUE as long as the function block is active and a setpoint profile is being calculated.
bError	BOOL	bError becomes TRUE if an error occurs.
iErrorID	UDINT	Error number
fActualVelocity	LREAL	Currently reached velocity of the drive.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4024.22	PC or CX (x86 or x64)	Tc2_Drive

5 Data types

5.1 General SoE

5.1.1 ST_SoE_String

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

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

Name	Type	Description
iActualSize	UINT	Current length of the string (without closing \0)
iMaxSize	UINT	Maximum length of the string (without closing \0)
strData	STRING(MAX_STRING_LENGTH)	String

5.1.2 ST_SoE_StringEx

The structure ST_SoE_StringEx describes a string such as can be used with SoE accesses, including the preset parameter attribute.

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

Name	Type	Description
dwAttribute	DWORD	Parameter attribute
iActualSize	UINT	Current length of the string (without closing \0)
iMaxSize	UINT	Maximum length of the string (without closing \0)
strData	STRING(MAX_STRING_LENGTH)	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.

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

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

Name	Type	Description
iActualSize	UINT	Current length of the string (without closing \0)
iMaxSize	UINT	Maximum length of the string (without closing \0)
arrDiagNumbers	ARRAY [0..29] OF UDINT	List of the maximum 30 last error numbers (as UDINT)

5.2 AX5000 SoE

5.2.1 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_GetSlaveStateError            := -7,
    eFwU_ScanFirmwareError             := -8,
    eFwU_FindFileError                 := -9,
    eFwU_CfgTypeInNoAX5xxx             := -10,
    eFwU_ScannedTypeInNoAX5xxx         := -11,
    eFwU_ChannelMismatch               := -12,
    eFwU_ChannelMismatch_1Cfg_2Scanned := -13,
    eFwU_ChannelMismatch_2Cfg_1Scanned := -14,
    eFwU_CurrentMismatch               := -15,
    eFwU_FwUpdateError                 := -16,
    eFwU_ReqSlaveStateError            := -17
);
END_TYPE
```

Update Status

eFwU_NoError	Initial state
eFwU_CheckCfgIdentity	Read the configured slave types (number of channels, current, revision)
eFwU_CheckSlaveCount	Determine the configured number of slaves
eFwU_CheckFindSlavePos	Search for the slave address in the master object directory
eFwU_WaitForScan	Wait for online scan

eFwU_ScanningSlaves	Online scan of the slaves
eFwU_CheckScannedIdentity	Read the scanned slave types (number of channels, current, revision)
eFwU_CheckScannedFirmware	Read the firmware version
eFwU_FindFirmwareFile	Search for the selected firmware file
eFwU_WaitForUpdate	Wait for status of the update
eFwU_WaitForSlaveState	Determine the EtherCAT slave status
eFwU_StartFwUpdate	Start the firmware update
eFwU_FwUpdateInProgress	Firmware update active
eFwU_FwUpdateDone	Firmware update successfully completed
eFwU_NoFwUpdateRequired	No firmware update required
eFwU_UpdateViaOtherChannelActive	Update takes place via the other axis channel
eFwU_UpdatedViaOtherChannel	Update took place via the other axis channel

Update error

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

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

5.2.3 ST_AX5000DriveStatus

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

5.2.4 E_AX5000_DriveOpMode

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

5.3 IndraDrive Cs

5.3.1 E_IndraDriveCs_DriveOpMode

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

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

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

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

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

    eIDC_DrvCtrldPositioningFbk1        := 531,
    eIDC_DrvCtrldPositioningFbk2        := 532,
```

```
    eIDC_DrvCtrldPositioningFbk1LagLess := 539,
    eIDC_DrvCtrldPositioningFbk2LagLess := 540,

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

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

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

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

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

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

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

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

    eIDC_PosSynchronization := 581

);

END_TYPE
```

5.3.2 ST_IndraDriveCs_C1D 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 *)
    bOvertravelLimitExceeeded   : BOOL; (* C1D Bit 13 *)
    bReserved_14                : BOOL; (* C1D Bit 14 *)
    bManufacturerSpecificError  : BOOL; (* C1D Bit 15 *)
END_STRUCT
END_TYPE
```

5.3.3 ST_IndraDriveCsDriveStatus

```
TYPE ST_IndraDriveCsDriveStatus :
STRUCT
    bStatusCmdValProcessing : BOOL;
    bRealTimeStatusBit1     : BOOL;
```

```
    bRealTimeStatusBit2     : BOOL;
    bDrvShutdownBitC1D      : BOOL;
    bChangeBitC2D           : BOOL;
    bChangeBitC3D           : BOOL;
    bNotReadyToPowerUp      : BOOL;
    bReadyForPower          : BOOL;
    bReadyForEnable         : BOOL;
    bEnabled                : BOOL;
    iActOpModeParNum        : UINT;
    eActOpMode              : E_IndraDriveCs_DriveOpMode;
    iReserved               : UINT;
END_STRUCT
END_TYPE
```

5.4 SERCOS

5.4.1 E_SoE_AttribLen

The enumeration 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 (single value), or whether 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.

Sample: see ST_SoE_String [▶ 46] of the eSoE_LEN_V1BYTE type.

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

Name	Description
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)
eSoE_LEN_V2BYTE	List of 2-byte data types (e.g. IDN list)
eSoE_LEN_V4BYTE	List of 4-Byte data types
eSoE_LEN_V8BYTE	List of 8-Byte data types

5.4.2 E_SoE_CmdControl

The enumeration E_SoECmdControl determines whether the command is to be canceled, set or started.

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

Name	Description
eSoE_CmdControl_Cancel	Cancel command.
eSoE_CmdControl_Set	Set command.
eSoE_CmdControl_SetAndEnable	Set command and execute.

5.4.3 E_SoE_CmdState

The enumeration E_SoE_CmdState describes the state of an SoE command.

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


eSoE_CmdState_NotSet = 0
- kein Kommando aktiv

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

eSoE_CmdState_Executed = 2
- Kommando wurde ausgeführt

eSoE_CmdState_SetEnabledExecuted = 3
- Kommando gesetzt (vorbereitet) und ausgeführt

eSoE_CmdState_SetAndInterrupted = 5
- Kommando wurde gesetzt aber unterbrochen

eSoE_CmdState_SetEnabledNotExecuted = 7
- Kommandoausführung ist noch aktiv

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

5.4.4 E_SoE_Type

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

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

The enumeration E_SoE_Type is used to determine how the data can be interpreted:

Name	Description
eSoE_Type_BIN	Binary
eSoE_Type_UNSIGNED	Integer without sign
eSoE_Type_SIGNED	Integer with sign
eSoE_Type_HEX	Hexadecimal number
eSoE_Type_TEXT	Text
eSoE_Type_IDN	Parameter number
eSoE_Type_FLOAT	Floating point number

5.5 SimplePlcMotion

5.5.1 E_CoEDriveEnableState

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

5.5.2 E_DriveMoveVelocityError

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

5.5.3 ST_CoEDriveIoInterface

Data structure for mapping the process image of a CoE drive to use the function blocks FB_CoEDriveEnable [▶ 40] and FB_CoEDriveMoveVelocity [▶ 41].

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

5.5.4 ST_DriveMoveVelocityOptions

The structure ST_DriveMoveVelocityOptions describes additional parameters of the function blocks FB_CoEDriveMoveVelocity [▶ 41] and FB_SoEDriveMoveVelocity [▶ 44].

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

5.5.5 ST_SoEDriveIoInterface

Data structure for mapping the process image of a SoE drive to use the function blocks FB_SoEDriveEnable [▶ 43] and FB_SoEDriveMoveVelocity [▶ 44].

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

6 Samples

Sample project and sample configuration for AX5000 diagnostics

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_Drive/Resources/2307584011/.zip

Sample project and sample configuration for IndraDrive Cs diagnostics

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_Drive/Resources/2307586955/.zip

More Information:
www.beckhoff.com/te1000

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