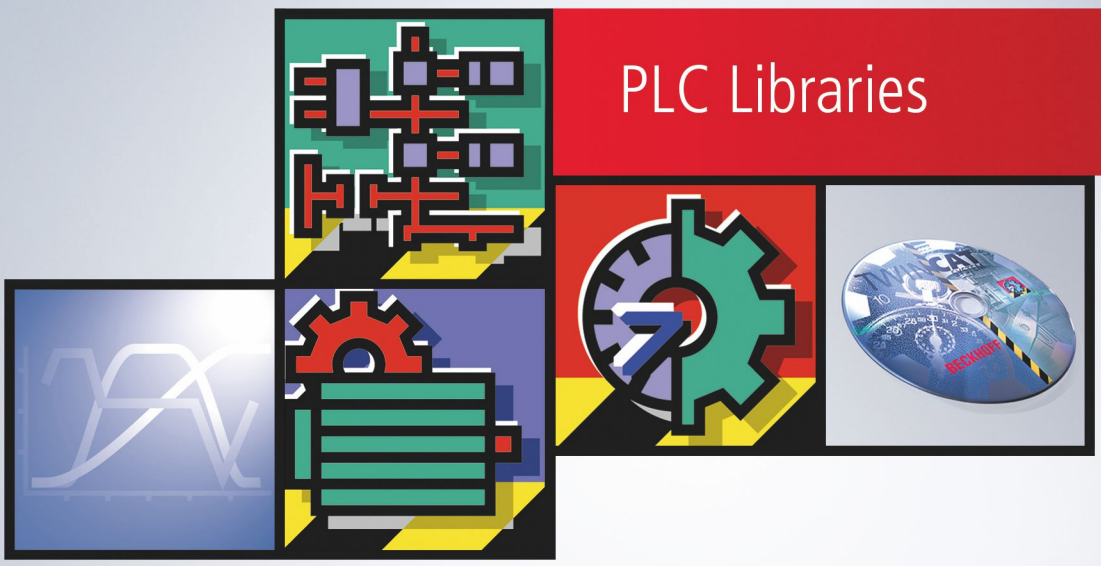
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcMC2Drive



01.07.2025 | Version: 1.3

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Kennzeichnungen führen.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie, lizenziert durch die Beckhoff Automation GmbH, Deutschland.

Copyright

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 POUs der TcMc2Drive.lib

In dieser Bibliothek sind Funktionen und Funktionsbausteine für SoE-Antriebe enthalten, die per MC2-Achsstruktur (AXIS_REF) auf den Antrieb zugreifen.

Unterschiede bei der Verwendung der Drive Libs mit AX5000 und Bosch Rexroth IndraDriveCS sind zu berücksichtigen. Siehe Beispiel unten.

Die Bibliothek TcMc2Drive.lib enthält Wrapper-Bausteine um die Bausteine der TcDrive.lib.

Die TcMc2Drive.lib sollte dann verwendet werden, wenn der Antrieb über NC mit der Bibliothek TCMC2.lib verwendet wird. Hierzu wird auf den Antrieb über die MC2-Achsreferenz (AXIS_REF) zugegriffen. Die Bausteine in der TcMc2Drive.lib ermitteln eigenständig über die NC-AchsID aus der MC2-Achsreferenz die Zugriffsdaten auf den Antrieb (NetID, Adresse und Kanalnummer). Siehe Beispiel bei den jeweiligen Funktionsbausteinen in der Dokumentation der TcMc2Drive.lib.

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

Funktionsbausteine

Name	Beschreibung
FB_SoEReset [▶ 10]	Antriebsreset ausführen (S-0-0099)
FB_SoEWritePassword [▶ 11]	Setzen des Antriebspassworts (S-0-0267)
FB_SoEReadDiagMessage [▶ 16]	Lesen der Diagnosenachricht (S-0-0095)
FB_SoEReadDiagNumber [▶ 17]	Lesen der Diagnosenummer (S-0-0390)
FB_SoEReadDiagNumberList [▶ 19]	Lesen der Diagnosenummernliste (bis zu 30 Einträge) (S-0-0375)
FB_SoEReadClassXDiag [▶ 20]	Lesen der Class 1 Diagnose (S-0-0011) ... Class 3 Diagnose (S-0-0013)
FB_SoEExecuteCommand [▶ 12]	Ausführen eines Kommandos
FB_SoEWriteCommandControl [▶ 13]	Setzen des Command Control
FB_SoEReadCommandState [▶ 15]	Prüfen des Kommandostatus
FB_SoERead [▶ 21]	Lesen eines Parameters
FB_SoEWrite [▶ 23]	Schreiben eines Parameters
FB_SoEReadAmplifierTemperature [▶ 24]	Lesen der Antriebstemperatur (S-0-0384)
FB_SoEReadMotorTemperature [▶ 26]	Lesen der Motortemperatur (S-0-0383)
FB_SoEReadDcBusCurrent [▶ 27]	Lesen des Dc-Bus-Stroms (S-0-0381)
FB_SoEReadDcBusVoltage [▶ 28]	Lesen der Dc-Bus-Spannung (S-0-0380)
FB_SoEAX5000ReadActMainVoltage [▶ 30]	Lesen der Netzspannung (P-0-0200)
FB_SoEAX5000SetMotorCtrlWord [▶ 31]	Setzen des Motor Control Words (P-0-0096)
FB_SoEAX5000FirmwareUpdate [▶ 32]	Automatischer Firmware-Update des AX5000

Beispielprojekt und Beispielkonfiguration für AX5000-Diagnose

Siehe https://infosys.beckhoff.com/content/1031/tcplclibmc2_drive/Resources/10842618635.zip,

Beispielprojekt und Beispielkonfiguration für IndraDriveCS-Diagnose

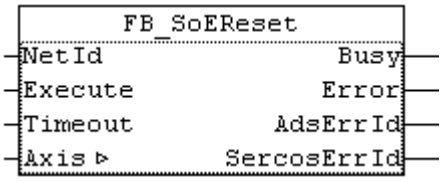
Siehe https://infosys.beckhoff.com/content/1031/tcplclibmc2_drive/Resources/10842620043.zip, https://infosys.beckhoff.com/content/1031/tcplclibmc2_drive/Resources/10842621451.zip (TcMc2Drive.lib ab v0.0.25)

Voraussetzungen

Komponente	Version
TwinCAT auf dem Entwicklungsrechner	2.10 Build 1335 oder höher
TwinCAT auf dem Windows CE-Image	2.10 Build 1333 oder höher
TwinCAT auf dem Windows XP-Image	2.10 Build 1333 oder höher

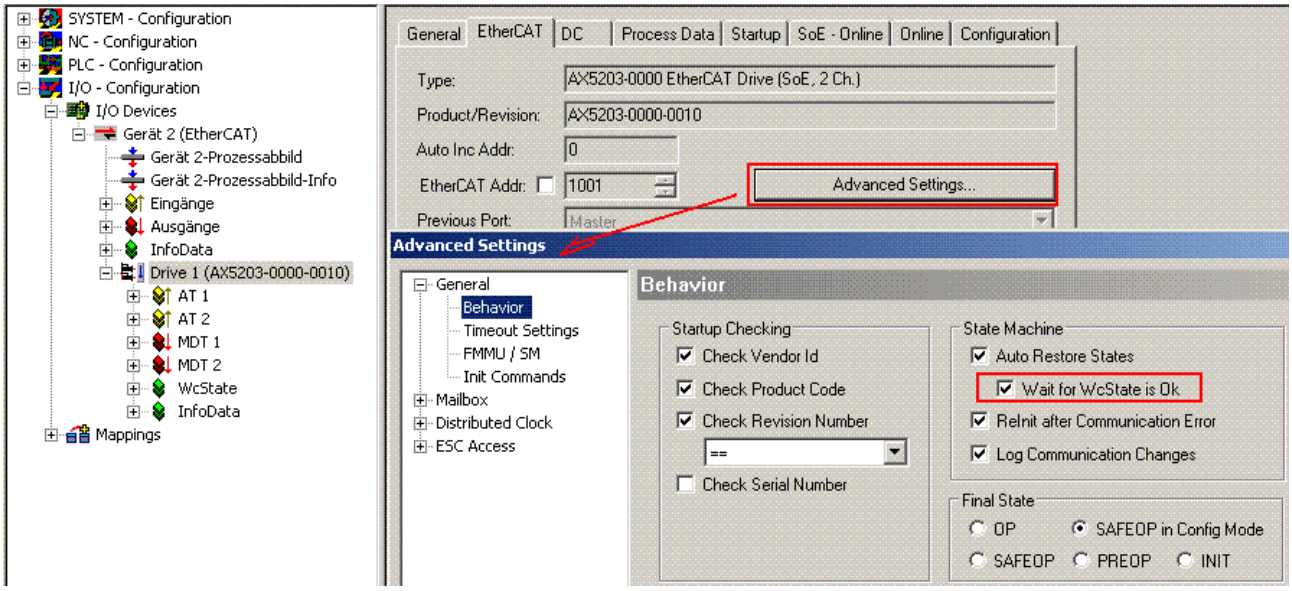
3 Allgemeine SoE FBs

3.1 FB_SoEReset



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

Für den AX5000 muss das Flag "Wait For WcState is OK" in den Advanced EtherCAT Settings aktiviert sein.



Ein NC-Reset wird nicht ausgeführt. Falls ein NC-Reset nötig ist, kann er über den MC_Reset-Baustein aus der TcMc.lib ausgeführt werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

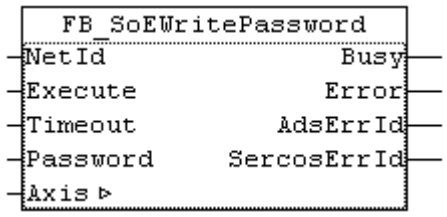
AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Beispiel

```
fbSoEReset : FB_SoEReset;
SoEReset : BOOL;

(* NcAxis *)

Axis     : AXIS_REF;
IF SoEReset THEN
    fbSoEReset(

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

3.2 FB_SoEWritePassword



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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

Password: enthält das Passwort als Sercos-String

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

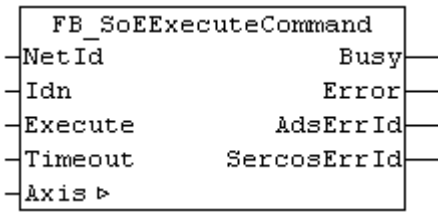
SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Beispiel

```
fbWritePassword : FB_SoEWritePassword;
WritePassword   : BOOL;
Password        :
ST_SoE_String;

(* NcAxis *)
Axis            : AXIS_REF;
IF WritePassword THEN
    fbWritePassword(
       Axis      := Axis,
       Execute   := TRUE,
       Timeout   := DEFAULT_ADS_TIMEOUT,
       Password := Password
    );
    IF NOT fbWritePassword.Busy THEN
        fbWritePassword(Axis :=Axis, Execute := FALSE);
        WritePassword :=FALSE;
    END_IF
END_IF
```

3.3 Kommando FBs

3.3.1 FB_SoEExecuteCommand



Mit dem Funktionsbaustein FB_SoEExecuteCommand kann ein Kommando ausgeführt werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

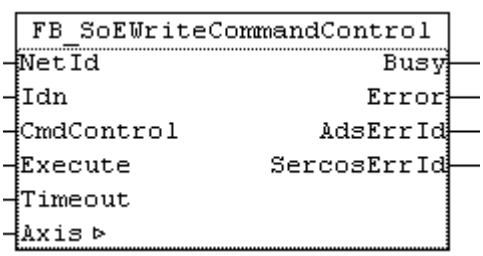
AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Beispiel

```
fbExecuteCommand : FB_SoEExecuteCommand;
ExecuteCommand   : BOOL;
Idn              : WORD;

(* NcAxis *)
Axis             : AXIS_REF;
IF ExecuteCommand THEN
    Idn := P_0_IDN + 160;
    fbExecuteCommand(
       Axis    := Axis,
       Execute := TRUE,
       Timeout := DEFAULT_ADS_TIMEOUT,
       Idn     := Idn,
    );
    IF NOT fbExecuteCommand.Busy THEN
        fbExecuteCommand(Axis :=Axis, Execute := FALSE);
        ExecuteCommand := FALSE;
    END_IF
END_IF
```

3.3.2 FB_SoEWriteCommandControl



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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

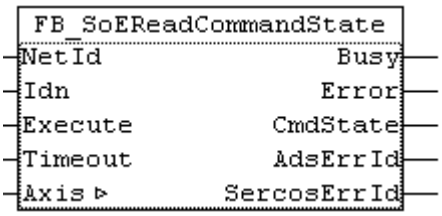
AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Beispiel

```
fbWriteCommandControl :FB_SoEWriteCommandControl;
WriteCommandControl   : BOOL;
Idn                   : WORD;
CmdControl            : E_SoE_CmdControl;

(* NcAxis *)

Axis                  : AXIS_REF;
IF WriteCommandControl THEN
    Idn := P_0_IDN + 160;
    fbWriteCommandControl(
       Axis        := Axis,
       Execute     := TRUE,
       Timeout     := DEFAULT_ADS_TIMEOUT,
       Idn         := Idn,
       CmdControl  := CmdControl
    );
    IF NOT fbWriteCommandControl.Busy THEN
       fbWriteCommandControl(Axis := Axis, Execute := FALSE);
       WriteCommandControl := FALSE;
    END_IF
END_IF
```

3.3.3 FB_SoEReadCommandState



Mit dem Funktionsbaustein FB_SoEReadCommandState kann die Kommandoausführung überprüft werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

```
VAR_OUTPUT
    Busy        : BOOL;
    Error       : BOOL;
    CmdState    : E_SoE_CmdState;
    AdsErrId    : UINT;
    SercosErrId : UINT;
END_VAR
```

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Attribute: Liefert das Attribut des Sercos-Parameters.

CmdState: Liefert den Kommandostatus

```
    eSoE_CmdState_NotSet =
0
- kein Kommando aktiv

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

    eSoE_CmdState_Executed =
2
- Kommando wurde ausgeführt
```

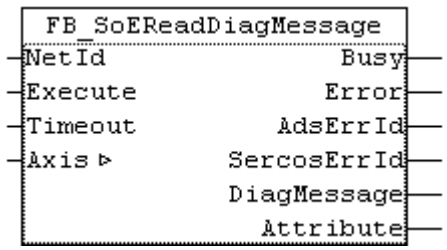
```
    eSoE_CmdState_SetEnabledExecuted =
3    - Kommando gesetzt (vorbereitet) und
ausgeführt

    eSoE_CmdState_SetAndInterrupted =
5     - Kommando wurde gesetzt aber
unterbrochen

    eSoE_CmdState_SetEnabledNotExecuted = 7 -
Kommandoausführung ist noch aktiv

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

Beispiel

```
fbReadCommandState : FB_SoEReadCommandState;
ReadCommandState   : BOOL;
Idn                : WORD;
CmdState           : E_SoE_CmdState;

(* NcAxis *)

Axis               : AXIS_REF;
IF ReadCommandState THEN
    Idn := P_0_IDN + 160;
    fbReadCommandState(
       Axis     := Axis,
       Execute  := TRUE,
       Timeout  := DEFAULT_ADS_TIMEOUT,
       Idn      := Idn,
       CmdState =>CmdState
    );
    IF NOT fbReadCommandState.Busy THEN
        fbReadCommandState(Axis:= Axis, Execute := FALSE);
        ReadCommandState := FALSE;
    END_IF
END_IF
```

3.4 Diagnose FBs

3.4.1 FB_SoEReadDiagMessage



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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

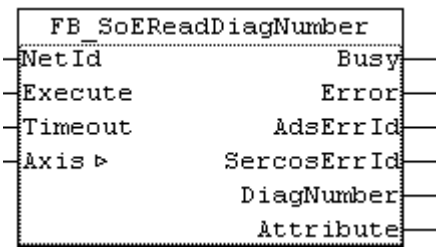
Attribute: Liefert das Attribut des Sercos-Parameters.

DiagMessage: Liefert die Diagnosenachricht.

Beispiel

```
fbDiagMessage : FB_SoEReadDiagMessage;
bDiagMessage  : BOOL;
DiagMessage   : ST_SoE_String;

(* NcAxis *)
Axis          : AXIS_REF;
IF bDiagMessage THEN
    fbDiagMessage(
       Axis     := Axis,
        Execute := TRUE,
        Timeout := DEFAULT_ADS_TIMEOUT,
        DiagMessage=> DiagMessage
    );
    IF NOT fbDiagMessage.Busy THEN
        fbDiagMessage(Axis := Axis, Execute := FALSE);
        bDiagMessage := FALSE;
    END_IF
END_IF
```

3.4.2 FB_SoEReadDiagNumber



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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

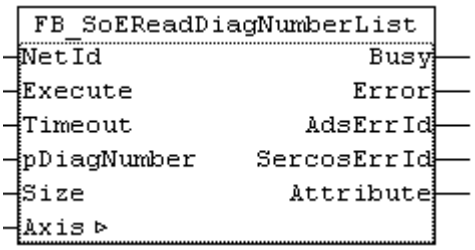
Attribute: Liefert das Attribut des Sercos-Parameters.

DiagNumber: Liefert die aktuelle Diagnosenummer.

Beispiel

```
fbDiagNumber : FB_SoEReadDiagNumber;
bDiagNumber  : BOOL;
DiagNumber   : UDINT;

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

3.4.3 FB_SoEReadDiagNumberList



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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei einem gesetzten Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

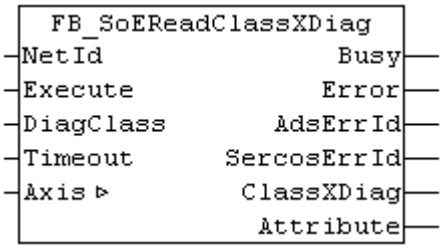
SercosErrId: Liefert bei einem gesetzten Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Attribute: Liefert das Attribut des Sercos-Parameters.

Beispiel

```
fbDiagNumberList     :
FB_SoEReadDiagNumberList;
DiagNumberList       : BOOL;
stDiagNumberList     :ST_SoE_DiagNumList;

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

3.4.4 FB_SoEReadClassXDiag



Mit dem Funktionsbaustein FB_SoEReadClassXDiag kann die aktuelle Class 1 Diagnose(S-0-0011) ... Class 3 Diagnose (S-0-0013) als WORD ausgelesen werden. Für die Auswertung der Class 1 Diagnose als Struktur ST_AX5000_C1D gibt es eine Konvertierungsfunktion F_ConvWordToSTAX5000C1D. Siehe Dokumentation TcDrive.lib.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

DiagClass: Gibt an, welche Diagnose gelesen werden soll. Die Diagnose Parameter können sich von Hersteller zu Hersteller unterscheiden. Nicht immer sind alle Diagnose Parameter (C1D ... C3D) oder alle Bits darin implementiert.
1: Fehler: Class 1 Diag (S-0-0011)
2: Warnungen: Class 2 Diag (S-0-0012)
3: Informationen: Class 3 Diag (S-0-0013)

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

ClassXDiag: Liefert die aktuelle Class X Diagnose.

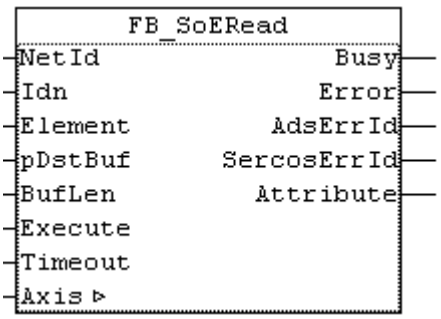
Attribute: Liefert das Attribut des Sercos-Parameters.

Beispiel

```
fbClassXDiag : FB_SoEReadClassXDiag;
bClassXDiag  : BOOL;
DiagClass    : USINT := 1;
Class1Diag   : WORD;
stAX5000C1D  : ST_AX5000_C1D;
Class2Diag   : WORD;

(* NcAxis *)
Axis : AXIS_REF;
IF bClassXDiag THEN
    fbClassXDiag(
       Axis       := Axis,
       Execute    := TRUE,
        DiagClass  := DiagClass,
        Timeout    := DEFAULT_ADS_TIMEOUT
    );
    IF NOT fbClassXDiag.Busy THEN
        fbClassXDiag(Axis := Axis, Execute := FALSE);
        bClassXDiag := FALSE;
        CASE fbClassXDiag.DiagClass OF
        1:
           Class1Diag  := fbClassXDiag.ClassXDiag;
           stAX5000C1D := F_ConvWordToSTAX5000C1D(Class1Diag);
        2:
            Class2Diag := fbClassXDiag.ClassXDiag;
        END_CASE
    END_IF
END_IF
```

3.5 FBs für aktuelle Werte

3.5.1 FB_SoERead



Mit dem Funktionsbaustein FB_SoERead kann ein Parameter eingelesen werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

Idn: Parameternummer, auf das sich das FB_SoERead bezieht, "S_0_IDN + 33" für S-0-0033

Element: Gibt an, auf welchen Teil des Parameters zugegriffen werden soll, z.B. 16#40 ist der Wert (Value) des Parameters

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

pDstBuf: ADR() der Variablen, in die der Wert gelesen werden soll.

BufLen: SIZEOF() der Variablen, in die der Wert gelesen werden soll.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

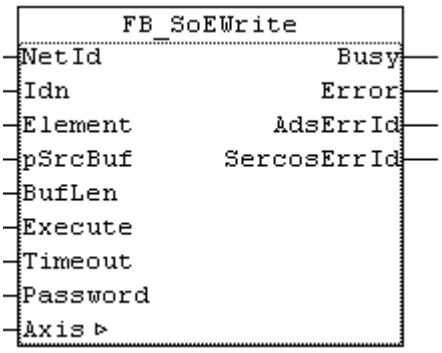
Attribute: Liefert das Attribut des Sercos-Parameters.

Beispiel

```
fbRead : FB_SoERead;
Read    : BOOL;
Idn     : WORD;
ReadValue : UINT;
```

```
(* NcAxis *)
Axis            : AXIS_REF;
IF Read THEN
    Idn := S_0_IDN + 33;
    fbRead(
        Axis    := Axis,
        Idn     := Idn,
        Element := 16#40,
        pDstBuf := ADR(ReadValue),
        BufLen  := SIZEOF(ReadValue),
        Execute := TRUE,
        Timeout := DEFAULT_ADS_TIMEOUT,
    );
    IF NOT fbRead.Busy THEN
        fbRead(Axis := Axis, Execute := FALSE);
        Read := FALSE;
    END_IF
END_IF
```

3.5.2 FB_SoEWrite



Mit dem Funktionsbaustein FB_SoEWrite kann ein Parameter geschrieben werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

Idn: Parameternummer, auf das sich das FB_SoERead bezieht, "S_0_IDN + 47" für S-0-0047

Element: Gibt an, auf welchen Teil des Parameters zugegriffen werden soll, z.B. 16#40 ist der Wert (Value) des Parameters. Meist kann nur auf den Wert schreibend zugegriffen werden, andere Bestandteile des Parameters sind schreibgeschützt.

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

pSrcBuf: ADR() der Variablen, die den zu schreibenden Wert enthält.

BufLen: SIZEOF() der Variablen, die den zu schreibenden Wert enthält

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

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

Password: enthält das Passwort als Sercos-String. Wird z.Z. noch nicht verwendet. Das Passwort muss mit FB_SoEWritePassword [▶ 11] geschrieben werden.

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

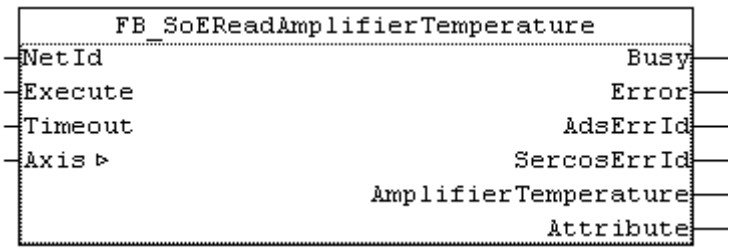
AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Beispiel

```
fbWrite : FB_SoEWrite;
Idn     : WORD;
Write   : BOOL;
WriteValue : UINT;
Password   : ST_SoE_String;

(* NcAxis *)
Axis            : AXIS_REF;
IF Write THEN
    Idn := S_0_IDN + 33;
    fbWrite(
       Axis     := Axis,
       Idn      := Idn,
        Element  := 16#40,
        pSrcBuf  := ADR(WriteValue),
        BufLen   := SIZEOF(WriteValue),
        Password := Password,
        Execute  := TRUE,
        Timeout  := DEFAULT_ADS_TIMEOUT,
    );
    IF NOT fbWrite.Busy THEN
        fbWrite(Axis := Axis, Execute := FALSE);
        Write := FALSE;
    END_IF
END_IF
```

3.5.3 FB_SoEReadAmplifierTemperature



Mit dem Funktionsbaustein FB_SoEReadAmplifierTemperature kann die Temperatur des Antriebs (S-0-0384) eingelesen werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

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

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

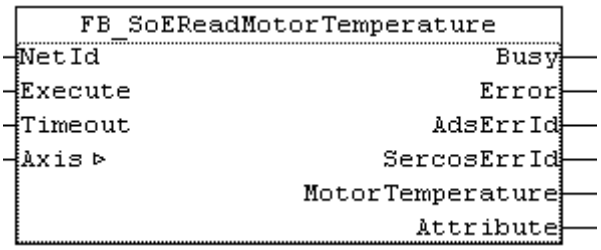
Attribute: Liefert das Attribut des Sercos-Parameters.

AmplifierTemperature: Liefert die Antriebstemperatur (z.B. 26.2 entspricht 26.2°C).

Beispiel

```
fbReadAmplifierTemp   : FB_SoEReadAmplifierTemperature;
ReadAmplifierTemp     : BOOL;
AmplifierTemperature  : REAL;

(* NcAxis *)
Axis            : AXIS_REF;
IF ReadAmplifierTemp THEN
    fbReadAmplifierTemp(
        Axis     := Axis,
        Execute  := TRUE,
        Timeout  := DEFAULT_ADS_TIMEOUT,
        AmplifierTemperature=>AmplifierTemperature
    );
    IF NOT fbReadAmplifierTemp.Busy THEN
        fbReadAmplifierTemp(Axis := Axis, Execute := FALSE);
        ReadAmplifierTemp := FALSE;
    END_IF
END_IF
```

3.5.4 FB_SoEReadMotorTemperature



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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Attribute: Liefert das Attribut des Sercos-Parameters.

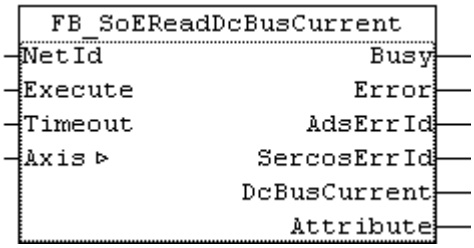
MotorTemperature: Liefert die Motortemperatur (z.B. 30.5 entspricht 30.5°C). Falls der Motor keinen Temperatursensor enthält, steht hier 0.0, heißt 0.0°C.

Beispiel

```
fbReadMotorTemp  : FB_SoEReadMotorTemperature;
ReadMotorTemp    : BOOL;
MotorTemperature : REAL;

(* NcAxis *)
Axis             : AXIS_REF;
IF ReadMotorTemp THEN
```

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

3.5.5 FB_SoEReadDcBusCurrent



Mit dem Funktionsbaustein FB_SoEAX5000ReadDcBusCurrent kann der DC-Bus-Strom (S-0-0381) eingelesen werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

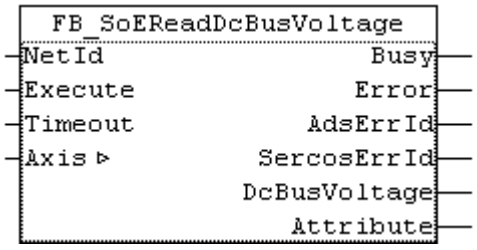
Attribute: Liefert das Attribut des Sercos-Parameters.

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

Beispiel

```
fbReadDcBusCurrent : FB_SoEReadDcBusCurrent;
ReadDcBusCurrent   : BOOL;
DcBusCurrent       : REAL;

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

3.5.6 FB_SoEReadDcBusVoltage



Mit dem Funktionsbaustein FB_SoEReadDcBusVoltage kann die DC-Bus-Spannung des Antriebs (S-0-0380) eingelesen werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Attribute: Liefert das Attribut des Sercos-Parameters.

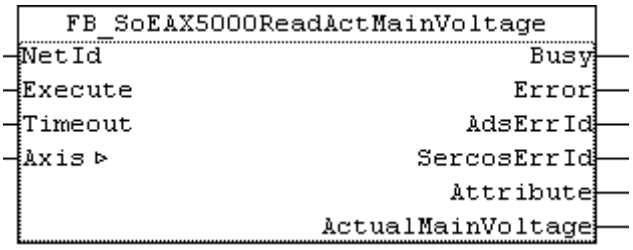
DcBusVoltage: Liefert die DC-Bus-Spannung (z.B. 294.0 entspricht 294.0V).

Beispiel

```
fbReadDcBusVoltage : FB_SoEReadDcBusVoltage;
ReadDcBusVoltage   : BOOL;
DcBusVoltage       : REAL;

(* NcAxis *)
Axis            : AXIS_REF;
IF ReadDcBusVoltage THEN
    fbReadDcBusVoltage(
        Axis     := Axis,
        Execute  := TRUE,
        Timeout  := DEFAULT_ADS_TIMEOUT,
        DcBusVoltage=>DcBusVoltage
    );
    IF NOT fbReadDcBusVoltage.Busy THEN
        fbReadDcBusVoltage(Axis := Axis, Execute := FALSE);
        ReadDcBusVoltage := FALSE;
    END_IF
END_IF
```

4 AX5000 spezifische FBs

4.1 FB_SoEAX5000ReadActMainVoltage



Mit dem Funktionsbaustein FB_SoEAX5000ReadActMainVoltage kann der aktuelle Scheitelwert der Netzspannung des AX5000 (P-0-0200) eingelesen werden.

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Attribute: Liefert das Attribut des Sercos-Parameters.

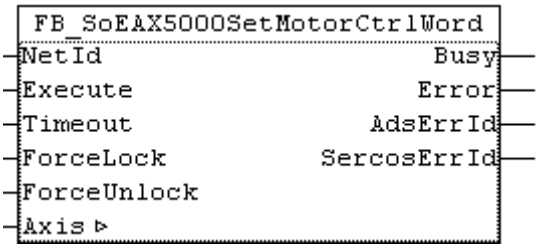
ActualMainVoltage: Liefert den Scheitelwert der aktuellen Netzspannung des AX5000 (z.B. 303.0 entspricht 303.0V).

Beispiel

```
fbReadActMainVoltage : FB_SoEAX5000ReadActMainVoltage;
ReadActMainVoltage   : BOOL;
ActualMainVoltage    : REAL;
```

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

4.2 FB_SoEAX5000SetMotorCtrlWord



Mit dem Funktionsbaustein FB_SoEAX5000SetMotorCtrlWord kann das ForceLock-Bit (Bit 0) bzw. das ForceUnlock-Bit im Motor Control Word (P-0-0096) gesetzt werden, um die Bremse zu setzen oder zu lösen. Im Normalfall wird die Bremse automatisch über das Enable des Antriebs gehandhabt.

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

VAR_INPUT

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

NetId: Ist ein String, der die AMS-Netzwerkkennung des PCs enthält.

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

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

ForceLock: Bremse unabhängig vom Enable aktivieren.

ForceUnlock: Bremse unabhängig vom Enable lösen.

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

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

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

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

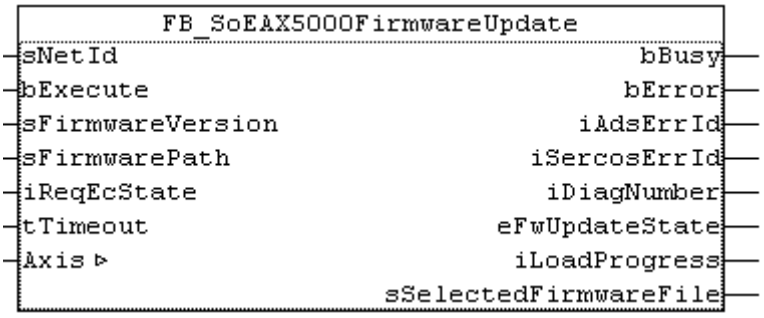
AdsErrId: Liefert bei gesetztem Error-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

SercosErrId: Liefert bei gesetztem Error-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Beispiel

```
fbSetMotorCtrlWord : FB_SoEAX5000SetMotorCtrlWord;
SetMotorCtrlWord   : BOOL;
ForceLock          : BOOL;
ForceUnlock        : BOOL;

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

4.3 FB_SoEAX5000FirmwareUpdate



Mit dem Funktionsbaustein FB_SoEAX5000FirmwareUpdate kann die Firmware des AX5000 überprüft und automatisch auf eine bestimme Version (Revision und Build) oder auf das aktuellste Build der konfigurierten Revision geändert werden.

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

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

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

HINWEIS

Fehlerhaftes Update durch Unterbrechungen

Unterbrechungen während des Updates können dazu führen, dass dieses nicht oder fehlerhaft ausgeführt wird. Die Klemme kann danach ohne die passende Firmware möglicherweise nicht mehr verwendet werden.

Während des Updates gilt:

- Die SPS und TwinCAT dürfen nicht gestoppt werden.
- Die EtherCAT-Verbindung darf nicht unterbrochen werden.
- Der AX5000 darf nicht ausgeschaltet werden.

VAR_INPUT

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

sNetId: AMS-NetID der Steuerung (IPC).

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

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

Release-Builds:

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

Kundenspezifische Firmware-Builds:

"v1.05_b**10**09"	für ein spezifisches Build, zum Beispiel v1.05 Build 0009
"v1.05_b**10**??"	aktuellstes Build einer vorgegebenen Version, zum Beispiel v1.05
"v1.??_b**10**??"	aktuellstes Build einer vorgegebenen Hauptversion, zum Beispiel v1
"v?.??_b**10**??"	aktuellstes Build der aktuellsten Version
...	
"v1.05_b**89**09"	für ein spezifisches Build, zum Beispiel v1.05 Build 8909
"v1.05_b**89**??"	aktuellstes Build einer vorgegebenen Version, zum Beispiel v1.05
"v1.??_b**89**??"	aktuellstes Build einer vorgegebenen Hauptversion, zum Beispiel v1
"v?.??_b**89**??"	aktuellstes Build der aktuellsten Version

Debug-Builds:

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

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

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

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

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : AXIS_REF;(* reference to NC axis *)
END_VAR
```

Axis: Achsstruktur (siehe TcMc2.lib).

VAR_OUTPUT

```
VAR_OUTPUT
   bBusy                : BOOL;
   bError               : BOOL;
   iAdsErrId            : UINT;
   iSercosErrId         : UINT;
   iDiagNumber          : UDINT;
   eFwUpdateState       : E_FwUpdateState;
   iLoadProgress        : INT;
   sSelectedFirmwareFile : STRING(MAX_STRING_LENGTH); (* found firmware file, e.g.
"AX5yxx_xxxx_0010_v1_05_b0009.efw" *)
END_VAR
```

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

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

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

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

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

eFwUpdateState: Liefert den Status der Firmware-Updates. Siehe E_FwUpdateState [▶ 37].

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

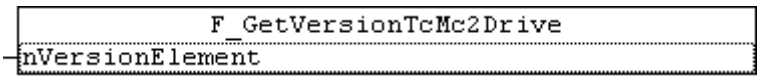
sSelectedFirmwareFile: Zeigt den Namen der gesuchten Firmware-Datei an.

Beispiel

```
VAR CONSTANT
   iNumOfDrives         : INT := 2;
END_VAR

VAR
    fbFirmwareUpdate     : ARRAY[1..iNumOfDrives] OF FB_SoEAX5000FirmwareUpdate;
    Axes                 : ARRAY[1..iNumOfDrives] OF AXIS_REF;
    sFirmwareVersion     : ARRAY
[1..iNumOfDrives] OF STRING(20)(* := 2('v1.04_b0002')*);
   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;
   bAnyBusy              : BOOL;
   bAnyError             : BOOL;
END_VAR
CASE iUpdateState OF
0:
    IF bExecute THEN
        iUpdateState := 1;
    END_IF
1:
    FOR I := 1 TO iNumOfDrives DO
       fbFirmwareUpdate[I](
           Axis := Axes[I],
           bExecute := TRUE,
```

```
            tTimeout := T#15s,
            sFirmwareVersion := sFirmwareVersion[I],
            sFirmwarePath := sFirmwarePath,
            sNetId := sNetIdIPC,
            iReqEcState := EC_DEVICE_STATE_OP,
            eFwUpdateState => eFwUpdateState[I],
        );
    END_FOR
    iUpdateState := 2;

2:
    bAnyBusy := FALSE;
    bAnyError:= FALSE;
    FOR I := 1 TO iNumOfDrives DO
       fbFirmwareUpdate[I](
            Axis := Axes[I],
            eFwUpdateState => eFwUpdateState[I],
            sSelectedFirmwareFile => sSelectedFirmwareFile[I],
        );

        IF NOT
fbFirmwareUpdate[I].bBusy THEN
            fbFirmwareUpdate[I](bExecute := FALSE, Axis := Axes[I]);
            IF fbFirmwareUpdate[I].bError THEN
                bAnyError := TRUE;
            END_IF
        ELSE
            bAnyBusy := TRUE;
        END_IF
    END_FOR

    IF NOT bAnyBusy THEN
        bExecute := FALSE;
        IF NOT bAnyError THEN
            iUpdateState := 0; (* OK *)
        ELSE
            iUpdateState := 3; (* Error *)
        END_IF
    END_IF
3:
    (* Error handling *)
    iUpdateState := 0;

END_CASE
```

5 F_GetVersionTcMc2Drive



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

FUNCTION F_GetVersionTcMc2Drive : UINT

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

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

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1329	PC or CX (x86)	TcMc2.lib, TcMath.lib, TcMathBase.lib,TcDrive.lib, TcEtherCAT.lib, TcUtilities.Lib, TcSystem.lib, TcBase.lib
TwinCAT v2.10.0 Build >= 1329	CX (ARM)	

6 Datentypen

6.1 E_FwUpdateState

Der E_FwUpdateState beschreibt den Zustand eines Firmware-Updates.

```
TYPE E_SoE_CmdState :(
   (* update states *)
   eFwU_NoError := 0,
   eFwU_CheckCfgIdentity,
   eFwU_CheckSlaveCount,
   eFwU_CheckFindSlavePos,
   eFwU_WaitForScan,
   eFwU_ScanningSlaves,
   eFwU_CheckScannedIdentity,
   eFwU_CheckScannedFirmware,
   eFwU_FindFirmwareFile,
   eFwU_WaitForUpdate,
   eFwU_WaitForSlaveState,
   eFwU_StartFwUpdate,
   eFwU_FwUpdateInProgress,
   eFwU_FwUpdateDone,
   eFwU_NoFwUpdateRequired,

   (* not updating via this channel *)
   eFwU_UpdateViaOtherChannelActive,
   eFwU_UpdatedViaOtherChannel,

   (* error states *)
   eFwU_GetSlaveIdentityError         := -1,
   eFwU_GetSlaveCountError            := -2,
   eFwU_GetSlaveAddrError             := -3,
   eFwU_StartScanError                := -4,
   eFwU_ScanStateError                := -5,
   eFwU_ScanIdentityError             := -6,
   eFwU_GetSlaveStateError            := -7,
   eFwU_ScanFirmwareError             := -8,
   eFwU_FindFileError                 := -9,
   eFwU_CfgTypeInNoAX5xxx             := -10,
   eFwU_ScannedTypeInNoAX5xxx         := -11,
   eFwU_ChannelMismatch               := -12,
   eFwU_ChannelMismatch_1Cfg_2Scanned := -13,
   eFwU_ChannelMismatch_2Cfg_1Scanned := -14,
   eFwU_CurrentMismatch               := -15,
   eFwU_FwUpdateError                 := -16,
   eFwU_ReqSlaveStateError            := -17
);
END_TYPE
```

```
Update Status

eFwU_NoError
: Initialzustand


eFwU_CheckCfgIdentity
: Einlesen des konfigurierten Slavetypen (Anzahl Kanäle, Strom,
Revision)


eFwU_CheckSlaveCount
: Ermitteln der konfigurierten Slaveanzahl


eFwU_CheckFindSlavePos
: Suchen der Slave-Adresse im Master-Objektverzeichnis


eFwU_WaitForScan
: Warten auf Online-Scan


eFwU_ScanningSlaves
: Online-Scan der Slaves
```

```
eFwU_CheckScannedIdentity
: Einlesen des gescannten Slavetypen (Anzahl Kanäle, Strom,
Revision)


eFwU_CheckScannedFirmware
: Einlesen der Firmware-Version


eFwU_FindFirmwareFile
: Suchen nach der gewählten Firmware-Datei


eFwU_WaitForUpdate
: Warten auf State des Updates


eFwU_WaitForSlaveState
: Ermitteln des EtherCAT Slave-States


eFwU_StartFwUpdate
: Starten des Firmware-Updates


eFwU_FwUpdateInProgress
: Firmwareupdate aktiv


eFwU_FwUpdateDone
: Firmwareupdate erfolgreich beendet


eFwU_NoFwUpdateRequired
: Kein Firmwareupdate erforderlich


   eFwU_UpdateViaOtherChannelActive   : Update
erfolgt über den anderen Achskanal


eFwU_UpdatedViaOtherChannel
: Update erfolgte über den anderen Achskanal

Update Fehler

eFwU_GetSlaveIdentityError
: Einlesen des konfigurierten Slavetypen schlug fehl, siehe
iAdsErrId


eFwU_GetSlaveCountError
: Ermitteln der konfigurierten Slaveanzahl schlug fehl, siehe
iAdsErrId


eFwU_GetSlaveAddrError
: Suchen der Slave-Adresse im Master-Objektverzeichnis schlug fehl,
siehe iAdsErrId


eFwU_StartScanError
: Starten des Online-Scan schlug fehl, siehe iAdsErrId


eFwU_ScanStateError
: Online-Scan schlug fehl, siehe iAdsErrId


eFwU_ScanIdentityError
: Einlesen des gescannten Slavetypen (Anzahl Kanäle, Strom,
Revision) schlug fehl, siehe iAdsErrId


eFwU_GetSlaveStateError
: Ermitteln des EtherCAT Slave-States schlug fehl, siehe
iAdsErrId
```

```
eFwU_ScanFirmwareError
: Einlesen der Firmware-Version schlug fehl, siehe iAdsErrId +
iSercosErrId


eFwU_FindFileError
: Suchen nach der gewählten Firmware-Datei schlug fehl, siehe
iAdsErrId


eFwU_CfgTypeInNoAX5xxx
: Der konfigurierte Slave ist kein AX5000


eFwU_ScannedTypeInNoAX5xxx
: Der gescannte Slave ist kein AX5000


eFwU_ChannelMismatch
: Anzahl der konfigurierten bzw. gefundenen Kanäle des AX5000
passen nicht zusammen

   eFwU_ChannelMismatch_1Cfg_2Scanned : Einkanaliges
Gerät konfiguriert aber zweikanaliges Gerät gefunden

   eFwU_ChannelMismatch_2Cfg_1Scanned : Zweikanaliges
Gerät konfiguriert aber einkanaliges Gerät gefunden


eFwU_CurrentMismatch
: AX5000-Type paßt vom Strom her nicht, z.B. AX5103 (3A)
konfiguriert aber AX5106 (6A) gefunden


eFwU_FwUpdateError
: Allgemeiner Updatefehler, siehe iAdsErrId


eFwU_ReqSlaveStateError
: Umschalten in den gewünschten EtherCAT-State schlug fehl
```

Trademark statements

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.

Third-party trademark statements

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Mehr Informationen:
www.beckhoff.com/tx1200

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